Higher Level Techniques for the Artistic Rendering of Images and Video

submitted by

John Philip Collomosse

for the degree of Doctor of Philosophy

of the

University of Bath

2004

Higher Level Techniques for the Artistic Rendering of Images and Video

John Philip Collomosse

SUMMARY

This thesis investigates the problem of producing non-photorealistic renderings for the purpose of aesthetics; so called Artistic Rendering (AR). Specifically, we address the problem of image-space AR, proposing novel algorithms for the artistic rendering of real images and post-production video.

Image analysis is a necessary component of image-space AR; information must be extracted from two-dimensional content prior to its re-presentation in some artistic style. Existing image-space AR algorithms perform this analysis at a "low" spatiotemporal level of abstraction. In the case of static AR, the strokes that comprise a rendering are placed independently, and their visual attributes set as a function of only a small image region local to each stroke. In the case of AR animation, video footage is also rendered on a temporally local basis; each frame of animation is rendered taking account of only the current and preceding frame in the video. We argue that this low-level processing paradigm is a limiting factor in the development of image-space AR. The process of deriving artwork from a photograph or video demands visual interpretation, rather than localised filtering, of that source content — a goal challenging enough to warrant application of higher level image analysis techniques, implying interesting new application areas for Computer Vision (and motivating new Computer Vision research as a result).

Throughout this thesis we develop a number of novel AR algorithms, the results of which demonstrate a higher spatiotemporal level of analysis to benefit AR in terms of broadening range of potential rendering styles, enhancing temporal coherence in animations, and improving the aesthetic quality of renderings. We introduce the use of global salience measures to image-space AR, and propose novel static AR algorithms which seek to emphasise salient detail, and abstract away unimportant detail within a painting. We also introduce novel animation techniques, describing a "Video Paintbox" capable of creating AR animations directly from video clips. Not only do these animations exhibit a wide gamut of potential styles (such as cartoon-style motion cues and a variety of artistic shading effects), but also exhibit a significant improvement in temporal coherence over the state of the art. We also demonstrate that consideration of the AR process at a higher spatiotemporal level enables the diversification of AR styles to include Cubist-styled compositions and cartoon motion emphasis in animation.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Peter Hall, for taking me on, and wish to express my sincere gratitude for all his hard work, patience and encouragement. Thanks are also due to members of the Media Technology Research Centre at the University of Bath: Phil Willis, Dan Su, Emmanuel Tanguy, and in particular to David Duke, whose advice and feedback over the last three years has been invaluable. Many thanks also go to "our animator" David Rowntree (and others at Nanomation Ltd.) who, on numerous occasions, supplied useful advice and assisted with edits of both conference papers and the Video Paintbox show-reel. Thanks also to Catriona Price, our ballerina in the show-reel, and the artists who have commented on our work over the course of this project. I am also grateful to the numerous conference attendees and anonymous referees, in both the Computer Graphics and Vision communities, who have provided encouragement and suggestions regarding this work.

Thanks to all the postgraduate students and research officers, with whom I have shared the lab, for the numerous coffees and distractions from work: Adam B., Andy H., Dan, Emma, Marc, James, Owen, and Vas. Thanks also to Ben for teaching me backgammon, and occasionally allowing me to win. Special mentions go those who "star" in the videos in this thesis: Emmanuel, Adam D., Siraphat, and of course Martin who was coerced into a bear costume to co-star in the sequences at Sham castle. I am also indebted to our lab support staff Mark and Jim for turning a blind eye to the copious amount of disk storage occupied by this work.

Finally, I would like to thank my family for their emotional and financial support throughout this project. Thanks also to Becky for her love and support.

I gratefully acknowledge the financial support of the EPSRC who funded this research under grant `GR/M99279`.

John P. Collomosse,
May 2004.

PUBLICATIONS

Portions of the work described in this thesis have appeared in the following papers:

Chapter 3

[23] Collomosse J. P. and Hall P. M. (October 2003). Cubist style Rendering from Photographs. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 4(9)*, pp. 443–453.

[22] Collomosse J. P. and Hall P. M. (July 2002). Painterly Rendering using Image Salience. In *Proc.* 20th *Eurographics UK Conference*, pp. 122–128. (awarded Terry Hewitt Prize for Best Student Paper, 2002)

Chapter 4

[24] Collomosse J. P. and Hall P. M. (October 2003). Genetic Painting: A Salience Adaptive Relaxation Technique for Painterly Rendering. *Technical Report, University of Bath. Report No. CSBU-2003-02.*

[66] Hall P. M. and Owen M. J. and Collomosse J. P (2004). A Trainable Low-level Feature Detector. *Proc. Intl. Conference on Pattern Recognition (ICPR)*, to appear.

Chapter 6

[27] Collomosse J. P., Rowntree D. and Hall P. M. (September 2003). Video Analysis for Cartoon-like Special Effects. In *Proc.* 14th *British Machine Vision Conference (BMVC)*, pp. 749–758. (awarded BMVA Industry Prize, 2003)

[25] Collomosse J. P., Rowntree D. and Hall P. M. (July 2003). Cartoon-style Rendering of Motion from video. In *Proc.* 1st *Intl. Conference on Video, Vision and Graphics (VVG)*, pp. 117–124.

[28] Collomosse J. P. and Hall P. M. (2004). Automatic Rendering of Cartoon-style Motion Cues in Post-production Video. submitted to *Journal on Graphical Models and Image Processing (CVGIP).*

Chapter 8

[26] Collomosse J. P., Rowntree D. and Hall P. M. (June 2003). Stroke Surfaces: A Spatio-temporal Framework for Temporally Coherent Non-photorealistic Animations. *Technical Report, University of Bath. Report No. CSBU-2003-01.*

NON-PUBLICATIONS

Collomosse J. P. and Hall P. M. (March 2004). Stroke Surfaces: A Spatio-temporal Framework for Temporally Coherent Non-photorealistic Animations. *Presented at BMVA Symposium on Spatiotemporal Processing.*

Collomosse J. P. and Hall P. M. (July 2002). Applications of Computer Vision to Non-photorealistic Rendering. *Poster at* 8^{th} *EPSRC/BMVA Summer School on Computer Vision.*

In some cases, these papers describe work in an earlier stage of development than presented in this thesis. Electronic versions are available on the DVD-ROM in Appendix C and at `http://www.cs.bath.ac.uk/~jpc/research.htm`.

Contents

Part I

Introduction

Chapter 1

Introduction

Research in the field of Computer Graphics has traditionally been dominated by attempts to achieve photorealism; modelling physical phenomena such as light reflection and refraction to produce scenes lit, ostensibly, in a natural manner. Over the past decade the development of novel rendering styles outside the bounds of photorealism has gathered momentum — so called *non-photorealistic rendering* or *NPR*. Given that they are defined by what they are not, it is perhaps unsurprising that NPR techniques are broad in classification. NPR has been successfully applied by scientific visualisation researchers to improve the clarity and quantity of information conveyed by an image [133]; exploded diagrams in maintenance manuals and false colour satellite imagery are two common examples of such visualisations. NPR algorithms have also been produced that are capable of emulating a broad range of artistic media from pastel to paint [20, 62], and to mimic many artistic techniques such as hatching and shading [65, 97]. This latter subset of NPR algorithms concerns the production of renderings solely for the benefit of aesthetic value, and is the field to which this thesis contributes. We collectively term these techniques "Artistic Rendering" to draw distinction from the rather general definition of NPR.

Illustrations offer many advantages over photorealism, including their ability to stylise presentation, clarify shape, abstract away detail and focus attention. Contemporary artists typically draw not only to convey knowledge of a scene, but also to convey a sense of how that scene is to be perceived. Strothotte *et al* [151] formalised this distinction by writing of the *transmitted* versus the *transputed* (perceived) image. For example, it is common practice that architects will often trace over draft designs with a pencil to produce a "sketchy" look, so helping to convey to clients the incomplete nature of their design. By contrast, photorealism carries with it an inherent sense of accuracy and completeness, which offers little potential for manipulation of the transputed image. Psychophysical experiments have shown that eighty percent of the sensory input we process is visual [165], and as such Computer Graphics provides one of the most natural

means of communicating information — a principle that underpins much of scientific visualisation, and was first posited in the 1960s by Sutherland with his *SKETCHPAD* systems [153]. Nowadays computer generated imagery is all pervasive in our society, and common applications for Computer Graphics include cinema, art, television and advertising. The ability to present such imagery in styles other than photorealism is of great value both commercially and in terms of aesthetics. As such the study of non-photorealism, and specifically of Artistic Rendering, is arguably of considerable relevance to modern society.

1.1 Contribution of this Thesis

Artistic Rendering (AR) techniques may be segregated into two broad categories; those producing artistic renderings from object-space (3D) scenes and those operating from image-space (2D) data. This thesis is concerned solely with the latter category, and specifically addresses the problem of automatically rendering both real images and post-production video sequences in artistic styles.

The problem of image-space AR is a particularly challenging one. Valuable geometric information such as silhouettes and relative depth are easily obtained from an object space representation. By contrast, much of the structure and information required to produce quality renderings is difficult to recover from a two dimensional scene. Image analysis therefore forms part of the rendering pipeline in image-space AR methods. We will show that existing automatic image-space AR algorithms perform this analysis at a "low" spatiotemporal level. In the case of static rendering (for example, transforming photographs into paintings), each brush stroke is painted independently and according to decisions based upon only a small local pixel neighbourhood surrounding that stroke's location. In the case of animations, video footage is also rendered on a temporally local basis; each frame of animation is rendered taking account of only the current and preceding frame in the video.

This thesis argues for the interpretation of image and video content at a higher spatiotemporal level, contending that this low-level processing paradigm is a limiting factor in the development of image-space AR. Throughout this thesis we develop a number of novel image-space AR algorithms, the results of which demonstrate a higher spatiotemporal level of analysis to be beneficial in terms of broadening range of potential rendering styles, enhancing temporal coherence in animations, and improving the aesthetic quality of renderings. The enabling factor for our work is the application of both novel and existing contemporary Computer Vision techniques to the problem of AR, and as such our work falls within the newly developing convergence area between both

Computer Graphics and Computer Vision.

1.2 Motivation for a Higher level of Analysis

We now give a brief overview of image-space AR, highlighting the difficulties arising from the low-level signal processing operations which drive the current state of the art. We use these observations to motivate our argument for a higher level of spatiotemporal analysis in image-space AR.

1.2.1 The Low-level Nature of Image-space AR

Early image-space AR techniques were highly interactive, and the vast majority concentrated upon the simulation of traditional artistic media within interactive painting environments [20, 150, 142]. The development of automated AR algorithms arguably began to gain momentum with Haeberli's semi-automatic paint systems [62]. These allowed users to interactively generate impressionist style "paintings" from photographs, by creating brush strokes on a virtual canvas. The colour and orientation of these strokes were determined by point-sampling the reference photograph. In such systems the onus was on the user to guide the rendering process, supplementing information lost by the (camera) projection to 2D by means of their natural ability to interpret image content. As applications increasingly demanded automation, such rendering decisions shifted necessarily away from the user and towards increasingly automated processes. In Haeberli's system a user might choose a finer brush to render detailed areas in an image, but to mimic this behaviour automatically is more difficult. Early automated systems were based, for the most part, upon pseudo-randomness [63]. However data dependent approaches were later presented, driven by heuristics based upon local image processing operations which automatically estimated stroke attributes such as scale or orientation [71, 103, 159].

We present a comprehensive review of such techniques in Chapter 2, but briefly summarise that the heuristics of current automatic algorithms make an important but implicit assumption that all fine detail in an image is salient (visually important). The vast majority of automated techniques paint to conserve high frequency detail in a scene. For example, in fully automatic coarse-to-fine multi-scale painterly renderers [71, 140], strokes depicting the finest scale artifacts are painted last. Additionally, we observe that all current algorithms are guided by local information, for example intensity gradient, colour, or statistical measures such as variance within a window. Consequently, when placing a stroke, only the pixels within a small region surrounding that stroke's location influence the decisions which determine that stroke's attributes (Figure 1-1). Such algorithms therefore operate at a *spatially low-level*, typically as

non-linear image filters which seek to texture content to convey the impression of some artistic style (for example, stipples or oil paint), whilst preserving all fine scale image content within the artistic rendering.

Researchers have found the extension of automatic AR algorithms to image sequences to be non-trivial. Current AR algorithms can not be applied independently to individual video frames without introducing aesthetically poor temporal incoherence into the resulting animation (manifested as distracting motion and rapid flickering, termed *"swimming"*). Attempts have been made to control swimming, for example by translating virtual paint strokes between frames, rather than painting each frame independently. Inter-frame motion is estimated on a per pixel basis, using either optical flow [96, 103], or frame differencing [75] operations. Although such approaches continue to operate at a spatially low-level, we observe that they also operate at a *temporally low-level*; analysing and rendering video on a per frame sequential basis, considering only content in the previous frame when rendering the next.

1.2.2 Limitations of a Low-Level Approach to AR

We argue that the ubiquitous trend to process images and video sequences at a low spatiotemporal level is a limiting factor in current AR. There are a number of disadvantages of low-level approaches, which we summarise here, and attempt to resolve through higher level spatiotemporal analysis in later chapters (3–8).

1. Aesthetic Quality of Rendering suffers

1a. Control over level of detail (emphasis) in renderings

Drawings and paintings are abstractions of photorealistic scenes in which salient elements are emphasised. Artists commonly paint to capture the structure and elements of the scene which they consider to be important; the remaining detail is abstracted away in some differential style or possibly omitted. Similarly, fully automatic image-space AR algorithms modulate stroke placement to preserve high frequency detail in the rendered output. However the assumption that high frequency artifacts unequivocally correlate with regions of importance does not hold true in the general case. The typical result is a painting in which all fine detail is emphasised, rather than only the *salient* detail. Arguably this disparity contributes to the undesirable impression that such paintings are of machine (AR) rather than natural origin.

Figure 1-1a gives an example which demonstrates that not all fine detail should be regarded as salient. In this figure, salient and non-salient artifacts are of similar scale, that is, we require windows of comparable size to detect them reliably. Such examples

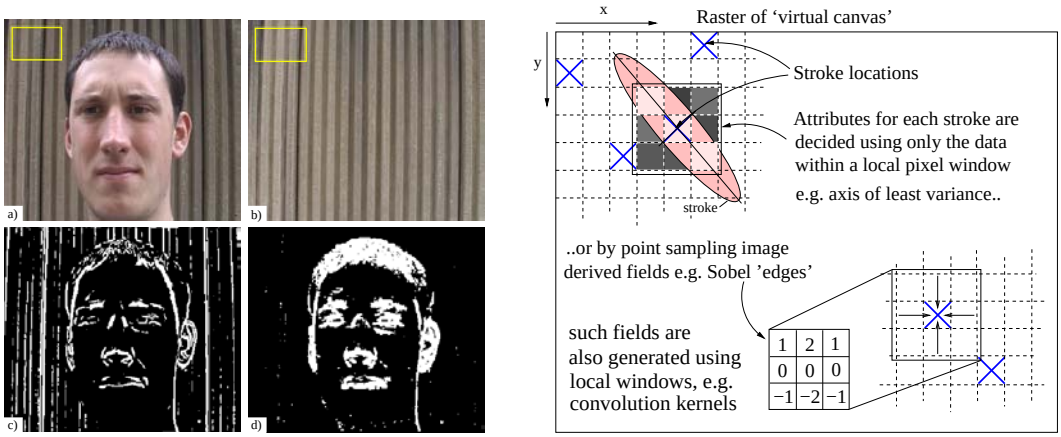


Figure 1-1 Left: By examining a window locally (yellow) in an image, we can not determine the importance of that region relative to the whole image; in (b) the stripes would be considered salient, while in (a) the face would. It is generally incorrect to correlate the presence of fine detail with image salience. In photo (a), salient edges and non-salient texture edges are of similar scale and magnitude, prohibiting isolation of salient artifacts through application of local measures (c). By considering all pixels using a global statistical measure of salience (proposed in Chapter 3) we approach a more intuitive result (d). Right: Illustrating the spatially local nature of typical image-space AR techniques.

make the case for some other measure of salience incontrovertible. When one speaks of the salience of image regions, one implicitly speaks of the importance of those regions relative to the image as a whole. It follows that global image analysis is a prerequisite to salience determination. Restricting attention to local image properties, by independently examining small pixel neighbourhoods, can give no real indication of salience in an image. The implication is that low-level image analysis prohibits the use of image salience to drive the level of emphasis in a rendering, and thereby imposes limitations upon the quality of renderings that may be produced.

Some recent image-space AR techniques have begun to re-examine collaborative (semi-automatic) approaches to rendering — allowing the user to interactively control the level of detail in certain regions of the image. Rather than emphasise high frequency artifacts, the image is broken into small segments which may be interactively merged to reduce detail in desired areas. This is achieved by specifying masks over image regions, either manually [3, 71] or using more exotic devices such as eye trackers [38]. We observe that such systems appeal to the human visual system to perceive the image, and interactively correct the level of emphasis attributed to regions by the AR heuristics. In doing so they are tacitly supporting our argument that a global image analysis is required to introduce some notion of relative importance within the image. In Chapters 3 and 4 we propose an automatic solution to this problem, introducing novel painterly rendering algorithms which regulate the level of emphasis over the rendering using a globally derived measure of image salience.

Figure 1-2 Previewing some of the results of our algorithms, which perform higher spatiotemporal analysis upon image and video sequence content to produce artistic renderings. We demonstrate benefit of higher level analysis to static AR: improving quality of rendering using salience adaptive stroke placement (left, Chapter 4) and improving diversity of style, through use of high level features to create Cubist compositions (middle, Chapter 3). We are also able to improve the temporal coherence of AR animations, and extend the gamut of video driven AR to include both cartoon shading, and motion emphasis (right, Chapters 6–8).

1b. Temporal Coherence of Animations

Rendering video sequences on a temporally local (per frame, sequential) basis causes rapid deterioration of temporal coherence after only a few frames of animation. Errors present in the estimated inter-frame motion fields quickly accumulate and propagate to subsequent frames, manifesting as swimming within the rendered animation. This can only be mitigated by exhaustive manual correction of motion fields. For example, the film "What Dreams May Come" [Universal Studios, 1997], featured a short painterly video sequence (rendered using [103]) requiring more than one thousand man-hours of motion field correction before being deemed aesthetically acceptable [61].

The problem of processing a video sequence for artistic effect is complex, and basing decisions upon a local, per frame algorithm is unlikely to result in an optimal (in this context, temporally coherent) solution. A global analysis over all frames seems intuitively more likely to produce coherent renderings. In Chapter 8 we develop such a rendering framework, which operates upon video at a higher spatiotemporal level to produce animations exhibiting superior temporal coherence than the current state of the art.

2. Diversity of style suffers

By restricting ourselves to a low-level analysis of the source image or video sequence, we also restrict the range of potential rendering styles to those of a low level. For example, the significant majority of existing artistic rendering algorithms follow the ordered brush stroke paradigm first proposed by Haeberli [62], in which atomic rendering elements (strokes) are individually arranged to form artwork (subsection 2.3.1). It

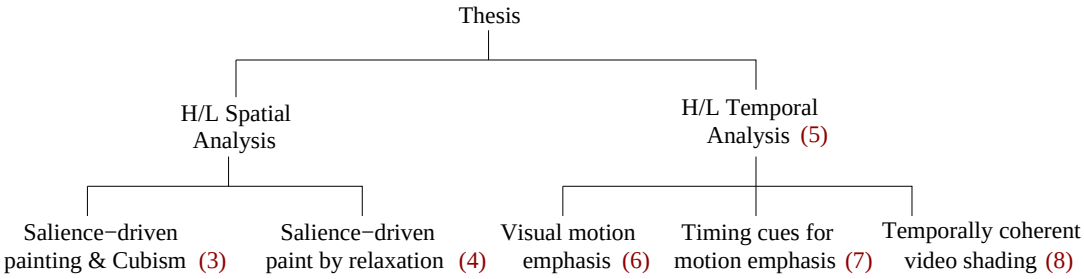


Figure 1-3 Structure and principal contributions of this thesis. The argument for higher level analysis in image-space AR is presented in terms of spatial and temporal processing, in parts II and III respectively. Chapter numbers are in parentheses.

is arguable that any artistic rendering would consist of multiple strokes of some media type (much as any digital image is comprised of pixels). However, we argue that identification and manipulation of conceptually higher level features in images and video can broaden the range of potential rendering styles in AR. Spatially, we may identify high level features within an image (such as eyes, or mouths) for the purposes of producing compositions, for example those of abstract art (see the Cubist composition algorithm of Chapter 3), which could not be achieved through image analysis at the level of abstraction of pixel neighbourhoods. Temporally, we may track the trajectories of features over extended periods of time, in order to characterise the essence of their movement using cartoon motion cues (see Chapters 6 and 7). If we restrict ourselves to local spatiotemporal content analysis, then these novel rendering styles are not possible.

1.3 Structure of the Thesis

This thesis therefore argues that performing analysis of the image or video sequence at a higher level proves beneficial in terms of both improving quality of output (controlling level of emphasis, and enhancing temporal coherence), and broadening the gamut of potential AR styles. The argument may be broadly summarised by the statement "to draw well, one must be able to see"; that is, in order to reap these benefits, one must approach AR from the stand-point of interpreting or "perceiving" content, rather than simply applying non-linear transformations to local regions of that content independently.

As evidence for our argument we propose several novel AR algorithms which operate at a higher spatial (Part II: Chapters 3 and 4) and temporal (Part III: Chapters 5, 6, 7 and 8) level to render images and video sequences, highlighting the benefits over low-level approaches in each case. We now outline the structure of the thesis, summarising the principal contributions made in each chapter and how they contribute to our central argument for higher level analysis in AR (Figure 1-3).

Part I — Introduction

Chapter 1 — Introduction

In which we describe the contribution of the thesis by outlining our case for the higher level analysis of images and video sequences for automated AR. We give a brief summary of the algorithms proposed in the thesis, their contributions to AR, and the evidence they provide to support our thesis.

Chapter 2 — State of the "Art"

In which we present a comprehensive literature survey of related AR techniques, forming observations on trends and identifying gaps in the literature. In particular we identify the local spatiotemporal nature of existing image-space AR algorithms, and that the extension of static AR techniques to video is an under-researched problem.

Part II — Salience and Art

Chapter 3 — Painterly and Cubist-style Rendering using Image Salience

In which we introduce the use of perceptual "salience" measures from Computer Vision to AR. We argue that existing AR methods should more closely model the practice of real artists, who typically emphasise only the salient regions in paintings. To this end, we propose a novel single-pass painting algorithm which paints to conserve salient detail and abstracts away non-salient detail in the final rendering. Image salience is determined by global analysis of the image, rather than on a local pixel-neighbourhood basis (as with current AR). We further apply this measure to propose the use of high level, salient features (connected groups of salient pixels, for example an eye or ear in a portrait) as a novel alternative to the stroke as the atomic element in artistic renderings. We describe a novel rendering algorithm capable of producing compositions in a Cubist style, using salient features identified across an image set. Control of the AR process is specified at the compositional, rather than the stroke based level. We also demonstrate how preferential rendering with respect to salience can emphasise detail in important areas of the painting, for example the eyes in a portrait. The painterly and Cubist algorithms demonstrate the benefit of higher level spatial analysis to image-space AR in terms of enhancing quality of rendering and diversity of style, respectively.

Chapter 4 — Genetic Painting: A Salience Adaptive Relaxation Technique for Painterly Rendering

In which we build on the success of our single-pass salience based painterly technique (Chapter 3) to propose a novel, relaxation based iterative process which uses curved spline brush strokes to generate paintings. We build upon our observations relating artwork and salience to define the degree of optimality for a painting to be measured

by the correlation between the salience map of the original image and the level of detail in the corresponding painting. We describe a novel genetic algorithm based relaxation approach to search the space of possible paintings and so locate the optimal painting for a given photograph, subject to our criterion. In this work we make use of a more subjective, user trained measure of salience. This work serves to reinforce our argument that AR quality can benefit from higher level spatial analysis, especially when combined with a relaxation based painting process. Furthermore, differential rendering styles are also possible by varying stroke style according to the classification of salient artifacts encountered, for example edges or ridges. We also compensate for noise, present in any real image. The context-dependent adaptation of style and noise compensation are further novel contributions to AR.

Part III — The Video Paintbox

Chapter 5 — Foreword to the Video Paintbox

In which we give a brief introduction to the "Video Paintbox"; a novel system for generating AR animations from video. This system is developed throughout Part III (Chapters 6, 7, and 8). We recall our observations in Chapter 2 which highlighted the limited scope of existing video driven AR techniques. We use these observations to provide motivation to the Video Paintbox, and give a brief overview of its core subsystems and their principal contributions.

Chapter 6 — Cartoon-style Visual Motion Emphasis from Video

In which we propose a framework capable of rendering motion within a video sequence in artistic styles, specifically emulating the visual motion cues commonly employed by traditional cartoonists. All existing video driven AR methods mitigate against the presence of motion in the video for the purpose of maintaining temporal coherence. By contrast our method is unique in emphasising motion within the video sequence. We are able to synthesise a wide range of augmentation cues (streak-lines, ghosting lines, motion blur) and deformation cues (squash and stretch, exaggerated drag and inertia). The system specifically addresses problematic issues such as camera motion and occlusion. Effects are generated by analysing the trajectories of tracked features over large temporal windows within the source video — in some cases (for example squash and stretch) object collisions must also be automatically detected, as they influence the style in which the motion cue is rendered. Breaking from the per frame sequential processing paradigm of current AR enables us to introduce a diverse range of motion cues into video footage.

Chapter 7 — Time and Pose Cues for Motion Emphasis

In which we extend the framework proposed in Chapter 6 to include the animation

timing cues employed by traditional cartoonists, again by analysing the trajectories of features over the course of the video sequence. We begin by deriving and comparing a number of algorithms to recover the articulation parameters of a tracked subject. We apply our chosen algorithm to automatically locate pivot point locations, and extract trajectories in a pose space for a subject moving in the plane. We then perform local and global distortions to this pose space to create novel animation timing effects in the video; cartoon "anticipation" effects and motion exaggeration respectively. We demonstrate that these timing cues may be combined with the cues of Chapter 6 in a single framework. The high level of temporal analysis required to analyse and emphasise a subject's pose over time is an enabling factor in the production of this class of motion cue.

Chapter 8 — Stroke Surfaces: Temporally Coherent AR Animations from Video

In which we describe a novel spatiotemporal approach to processing video sequences for artistic effect. We demonstrate that by analysing the video sequence at a high spatiotemporal level, as a video volume (rather than on a per frame, per pixel basis as with current methods) we are able to generate artistically shaded video exhibiting a high degree of temporal coherence. Video frames are segmented into homogeneous regions, and heuristic associations between regions formed over time to produce a collection of conceptually high level spatiotemporal objects. These objects carve sub-volumes through the video volume delimited by continuous isosurface "Stroke Surface" patches. By manipulating objects in this representation we are able to synthesise a wide gamut of artistic effects, which we allow the user to stylise and influence through a parameterised framework. In addition to novel temporal effects unique to our method we demonstrate the extension of "traditional" static AR styles to video including painterly, sketchy and cartoon shading effects. An application to rotoscoping is also identified, as well as potential future applications arising from the compact nature of the Stroke Surface representation. We demonstrate how this coherent shading framework may be combined with earlier motion cue work (Chapters 6 and 7) to produce complete cartoon-styled animations from video clips using our Video Paintbox.

Part IV — Conclusion

Chapter 9 — Conclusions and Further Work

In which we summarise the contributions of the thesis, and discuss how the results of the algorithms we have developed support our central argument for higher level spatiotemporal analysis in image-space AR. We suggest possible avenues for the future development of our work.

Part V — Appendices
Appendix A — Details of Miscellaneous Algorithms
Appendix B — Points of Definition
Appendix C — Electronic Supplementary Material (images, videos and papers.)

1.4 Application Areas

When the first cameras were developed there was great concern amongst artists that their profession might become obsolete. This of course did not happen, for the most-part because the skill to abstract and emphasise salient elements (the interpretation of the scene) could not, and may never be, fully automated. Likewise, modern AR algorithms come far from obsoleting the skills of artists. Our work does not aim to replace the human artist or animator. Rather we wish to produce automated tools and frameworks that allow the artist to express their creativity but without the associated tedium that often accompanies the process.

Our motivation is to produce a series of encompassing frameworks, capable of stylising images and video sequences with a high degree of automation. Users express their artistic influence through a series of high level controls that translate to parameters used to drive our rendering frameworks; control is typically exerted through specification of artistic style, rather than by manipulating pixel masks or individual strokes. Applications of this work lie most clearly within the entertainment industry, for example film special effects, animation and games. For example, consider the Video Paintbox developed throughout Part III (Chapters 5, 6, 7 and 8). Animation is a very costly process and a facility to shoot footage which could be automatically rendered into artistic styles could see commercial application. More domestic applications can be envisaged, for example the facility for a child to turn a home movie of their toys into a cartoon. In this latter case, the child would not have the skills (or likely the patience) to create a production quality animation. However they would often have an idea of what the result should look like. The "Video Paintbox" technology thus becomes an enabling tool, as well as tool for improving productivity. We hope that, in time, the algorithms and frameworks presented in this thesis may allow experimentation in new forms of creativity; for example the "Video Paintbox" might be used for experimentation in new forms of dynamic art.

1.5 Measuring Success in Artistic Rendering

Non-photorealistic rendering is a new but rapidly growing field within Computer Graphics, and it is now common for at least two or three AR papers to appear per major annual conference. A characteristic of the field's youth is that many proposed techniques break new, imaginative ground. However as the field matures, a difficult issue likely to emerge is that of comparison and evaluation of novel contributions which seek to improve on existing techniques. Although it is possible to design objective performance measures for some aspects of an AR algorithm (for example to evaluate temporal coherence of an animation, Section 8.6), comparative assessment of a rendering created solely for the purposes of aesthetics is clearly a subjective matter. Throughout our work we have worked closely with artists and animators who judge our output to be of high aesthetic value; we believe this is important given the absence of any specific ground truth. The aesthetics of our output are further evidenced by non-academic publication of our work; a Cubist-styled portrait of Charles Clark MP recently took the front page of the Times Higher Educational Supplement [3rd January 2003, see Figure 3-18], and we have received interest from a producer wishing to incorporate our cartoon rendering work into a pilot MTV production.

Novel AR developments are often inventive, yet also tend to be extremely specific; for example, describing a novel brush-type. We believe that as the field matures, less value will be gleaned from proposing specific novel, or unusual, rendering styles. Rather, greater value will be attributed to the development of encompassing frameworks and representations, capable of producing a diverse range of stylised renderings. In this thesis we present novel rendering algorithms. However, we do not wish to be novel for novelty's sake, but rather to illustrate the benefits that higher level image analysis confers to AR. For example, in Chapter 3 we make use of conceptually high level salient features (for example, eyes, ears, mouths) to synthesise abstract compositions reminiscent of Cubism. Although this work certainly extends the gamut of styles encompassed by AR it is important to note that such compositions arguably could not be produced without the consideration of high level salient features. Likewise the temporal coherence afforded by the Stroke Surface framework (Chapter 8) and the ability to render motion cues (Chapters 6 and 7) could not have been achieved without performing a high level spatiotemporal analysis of the source video sequence. We believe that our success should therefore be judged by the versatility of our proposed frameworks, and through the advantages demonstrated as a consequence our higher level spatiotemporal approach to processing images and video for artistic effect.

Chapter 2

The State of the "Art"

In this chapter we present a comprehensive survey of relevant Artistic Rendering techniques, forming observations on trends and identifying gaps in the literature. We explain the relevance of our research within the context of the reviewed literature.

2.1 Introduction

There have been a number of published AR surveys, including Lansdown and Schofield ([97], 1995), a comprehensive SIGGRAPH course edited by Green ([61], 1999), and a recent text ([59], 2000), which have documented progress in the field. However development has gathered considerable momentum, and these surveys are no longer representative of current rendering techniques. We now present a comprehensive survey of artificial drawing techniques, which for the purposes of this review have been segmented in to the following broad categories:

1. physical simulations and models of artistic materials (brushes, substrate, etc.)
2. algorithms and strategies for rendering static artwork
   - 2a. interactive and semi-automatic AR systems
   - 2b. fully automatic AR systems
3. techniques for producing non-photorealistic animations

Although research continues to be published in each of these areas, we indicate a rough chronological ordering in selecting these categories. Arguably the field of AR grew from research modelling various forms of physical artistic media in the late eighties and early nineties. These models were quickly incorporated into interactive painting systems, and were later enhanced to provide semi-automatic (yet still highly interactive) painting environments, for example assisting the user in simple tasks such as colour selection [62]. An increased demand for automation drove the development of these semi-automatic

systems towards fully automatic AR systems in the late nineties. More recently, the high level of automation achieved by these techniques has facilitated the development of techniques for production of non-photorealistic animations.

We now review each of these AR categories in turn, forming observations and drawing conclusions at the end of the Chapter (Section 2.6).

2.2 Simulation and Modelling of Artistic Materials

The vast majority of early AR techniques addressed the digital simulation of physical materials used by artists. Such research largely addresses the modelling of artists' brushes, but also extends to the modelling of both the medium (e.g. sticky paint) and the substrate. Such models vary from complete, complex physical simulations — for example, of brush bristles — to simulations which do not directly model the physics of artists' materials, but approximate their appearance to a sufficient level to be aesthetically acceptable for the purposes of Computer Graphics.

2.2.1 Brush models and simulations

Among the earliest physical simulations of brushes was Strassman's hairy brush model [150]. As with most subsequently published brush models, Strassman assumes knowledge of the stroke trajectory in the form of a 2D spline curve. To render a stroke, a one-dimensional array is swept along this trajectory, orientated perpendicular to the curve. Elements in the array correspond to individual brush bristles; the scalar value of each element simulates the amount of paint remaining on that bristle. As a bristle passes over a pixel, that pixel is darkened and the bristle's scalar value decremented to simulate a uni-directional transfer of paint. Furthermore, pressure values may be input to augment the curve data, which vary the quantity of paint transferred. Strassman was able to produce convincing greyscale, sumi-e[1] artwork using his brushes (Figure 2-1a), although hardware at the time prevented his LISP implementation from running at interactive speeds (brush pressure and position data were manually input via the keyboard). The principal contribution of this work was the use of simulated brush bristles; a significant improvement over previous paint systems which modelled brushes as patterned "stamps", which were swept over the image to create brush strokes [166].

A further sumi-e brush model was later proposed by Pham [121]. Similar to Strassman's technique, the system interpolates values (for example, pressure value or colour)

[1]Sumi-e: a form of traditional Japanese art. The popularity of sumi-e in brush modelling may be attributed to the fact that such artwork consists of few elegant, long strokes on a light background, which often serves as an excellent basis for clearly demonstrating brush models.

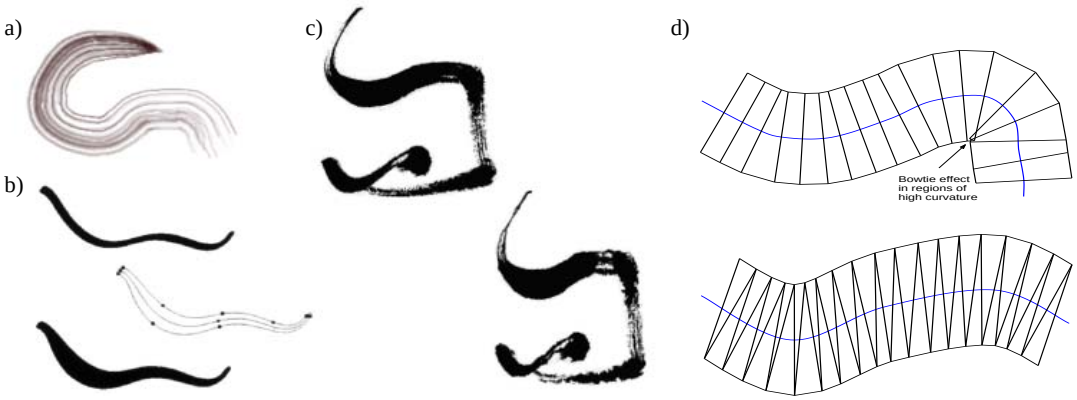


Figure 2-1 Demonstrating various brush models developed over the years: (a) Implementation of Strassman's hairy brush [150]; (b) Pudet's rigid brush (top) and dynamic brush (bottom) model with dynamic model skeleton (middle) — reproduced from [124]; (c) Artwork produced with a real brush (top) and Xu's calligraphic volume based brush (bottom) — reproduced from [177]; (d) Typical quadrilateral (top) and triangular (bottom) methods of texture mapping splines, demonstrating the problematic bow-tie effect in areas of high curvature (mitigated by the approach of Hsu *et al* [80]).

between knots on the curve. Each bristle's course is plotted parallel to the predefined stroke trajectory, and the entire stroke is then scan-converted in a single pass (rather than by sweeping a one-dimensional array over the image).

Pudet [124] proposed incremental improvements over previous systems [121, 150, 166]. Pudet's system makes use of a cordless stylus device similar to those used with modern day graphics tablets. This stylus is analogous to a brush, and rendering takes into account not only pressure and position, but also the angle of the stylus. Most notably this work achieves frame rates sufficient to allow interactive painting, and introduces the concept of the rigid and dynamic brushes. Rigid brushes change their width according to angle of the stylus, whilst dynamic brushes also vary in width according to stylus pressure (Figure 2-1b).

The first attempt to use a physical model to simulate the bristles of a brush was proposed by Lee [99]. Lee was dissatisfied with the limited physical simulation of then current models, and made a thorough study of the physics of brushes during painting, with particular regard to the elasticity of brush bristles. He proposes a 3D brush model, in which the friction and inertia of bristles are modelled, although as with previous models, paint transfer is only modelled uni-directionally (flowing from brush to paper). It was not until the simulated water-colour work of Curtis *et al* [33] that bi-directional transfer of paint was considered (discussion of this system is deferred to Section 2.2.2). Baxter *et al* recently presented "DAB", the first painting environment to provide haptic feedback (via a Phantom device) on several types of physically mod-

elled brushes. Painting takes place in a 3D environment, similar to that of Disney's *Deep Canvas* [35] (described in Section 2.3), and also features bi-directional transfer of paint between brush and surface. Recent advances in volume based brush design have been published incrementally by Xu *et al* in [178] and [177]. These systems feature full simulations of brush bristles and their interaction with the substrate entirely in object-space. Although computationally very expensive to render, such systems produce highly realistic brush strokes (Figure 2-1c) and are the current state of the art in physical brush simulation.

As mentioned in Section 2.2, many practical techniques do not seek to perfectly simulate the physics of the brush, but emulate the appearance of real brush strokes on the digital canvas to an aesthetically acceptable level. To this end, a large number of automatic AR algorithms use texture mapping techniques to copy texture sampled from real brush strokes onto digital brush strokes. Often an intensity displacement map is used to simulate the characteristic pattern of the brush or media being emulated; for example crayon, pastel and paint can be emulated satisfactorily in this manner. Recently bump mapping has been used in place of texture mapping to good effect [73].

Hsu *et al* [80] presented "skeletal strokes" which are, in essence, a means of smoothly mapping textures on to the curved trajectories of strokes. Standard texture mapping techniques, demonstrated in Figure 2-1d, harbour the disadvantage that high curvature regions of stroke trajectories often cause the vertices of polygons used to create texture correspondences to cross (the bow-tie effect). Hsu *et al* mitigate this behaviour by proposing a new deformable material which is used to map a wide variety of textures to a skeleton trajectory. Anchor points may also be set on the skeleton which, when placed judiciously, can create the aesthetically pleasing variations in brush width demonstrated by Pudet's dynamic brushes [124].

The brush techniques reviewed so far are intended to be rendered once, at a single scale. Subsequent large-scale magnification of the canvas will naturally cause pixelisation of the image. This can be a problem for certain applications, where successive magnifications should ideally yield an incremental addition of stroke detail in the image without pixelisation. Perlin and Velho [120] present a multi-resolution painting system which is capable of such behaviour. Strokes may be stored in one of two ways; either as a procedural texture (allowing stroke synthesis at any scale) or as a band-pass pyramid[2]. The latter approach is ultimately limited in its range of potential rendering scales. However the advantage of the band-pass approach is that textures are precomputed prior to

[2]A band-pass pyramid consists of a coarse scale image and a series of deltas from which finer scale images may be reconstructed. By contrast, the more common low-pass pyramid stores multiple complete images which have been low-pass filtered at various scales.

display, and procedural knowledge of the brush textures is therefore not required by the display algorithm (allowing for the simpler, faster display processes often required by interactive systems). Since the pyramidal approach does not require specification of a generative function for the texture, it may also be viewed as a more general solution.

2.2.2 Substrate and Media models

Models of artistic media and substrate are often presented together as inter-dependent, tightly coupled systems and accordingly we review both categories of model simultaneously in this section.

The first attempt to simulate the fluid nature of paint was the cellular automata based system of Small [142]. Small modelled the canvas substrate as a two-dimensional array of automata, each cell corresponding uniquely to a pixel, with attributes such as paint quantity, and colour. Paint could mix and flow between adjacent automata (i.e. pixels), to create the illusion of paint flowing on the substrate's surface. Furthermore Small proposed a novel model of a paper substrate, comprising intertwined paper fibres and binder which determined the substrate's absorbency. Rendering proceeded in two stages. First, the movement of fluids on the substrate surface was computed. This movement of fluid was simulated using a diffusion-like process which encouraged communication of paint between adjacent automata. Second, the transfer of fluid from the paper to a "paper interior" layer was computed, which fixed the pigment. Small describes no brush model for use in his system, but suggests application of Strassman's model [150].

Cockshott *et al* [20] also propose a two-dimensional cellular automata system addressing the modelling wet and sticky media on canvas. Their system allows for a much wider range of parameters than Small's, and although this causes a corresponding increase in complexity of control, the wet and sticky model is capable of emulating a wider range of media. Although the wet and sticky system seeks only to emulate the apparent behaviour of paint (rather than describing a physical model) the results are highly realistic and the simulated paint may be seen to bleed, run, drip and mix in a manner typical of wet and sticky substances when applied to a canvas. Their model is divided into three components: paint particles, an "intelligent" canvas, and a painting engine responsible for movement of paint particles on the canvas. The paint particles and the cellular automata may be seen to be analogous to elements in Small's system. However the novel paint engine they propose also simulates the drying of paint over time, adjusting the liquid content of each substrate cell. This affects the paint's response to environmental conditions such as humidity and gravity; simulation of the latter allows the authors to create novel running effects when paint is applied to a tilted

canvas. The principal contribution however, is in the use of a height field on the canvas to model the build up of paint. This height field can then be rendered as a relief surface (via application of bump mapping), which produces a distinctive painterly appearance. The authors acknowledge the computational expense of their system and appeal to parallel processing techniques to make their algorithm run interactively, although to the best of our knowledge, no such interactive systems have yet been implemented.

The first physically based simulation of water colour painting was proposed by Curtis *et al* [33]. The authors propose a more complex model of pigment-substrate diffusion consisting of multiple substrate layers. In descending order of depth in the substrate these are: the shallow water layer, where pigment may mix on the surface; the pigment deposition layer, where pigment is deposited or lifted from the paper; and the capillary layer, where pigment is absorbed into the substrate via capillary action. Aside from this novel physical model, a further novel contribution is that the authors model *bi-directional transfer* of painting medium between the brush and canvas. Thus a light stroke which passes through wet, dark pigment causes pollution of the brush, leading to realistic smearing of the pigment on paper (wet-on-wet effects). The results of this system are very realistic, and are often indistinguishable from real water colour strokes.

Although the majority of media emulation work concentrates on paint, there have also been attempts by Sousa and Buchanan to model the graphite pencil [146, 147, 148]. Their proposed systems incorporate physically based models which are based upon observations of the pencil medium through an electron microscope. In particular pencils are modelled by the hardness and the shape of the tip; the latter varies over time as the pencil tip wears away depositing graphite on to discrete cells in the virtual paper (adapted from the substrates of Cockshott *et al* and Small). A volume based model for coloured pencil drawing was proposed by Takagi *et al* [156]. The authors identify that graphite particles may be deposited not only in convex regions of the paper volume due to friction, but also may adhere to the paper surface. Pencils may deposit their particles in either manner, however the authors also provide a means to simulate the brushing of water over the image in which only the latter class of deposited particles are shifted. A further model of the graphite pencil is presented by Elber [44] as part of an object-space sketching environment.

2.3 Interactive and Semi-automatic AR Systems

The use of the computer as an interactive digital canvas for artwork was first pioneered by Sutherland in his doctoral thesis of 1963. During this period the Cathode Ray Tube (CRT) was a novelty, and Sutherland's *SKETCHPAD* system [153] allowed users to

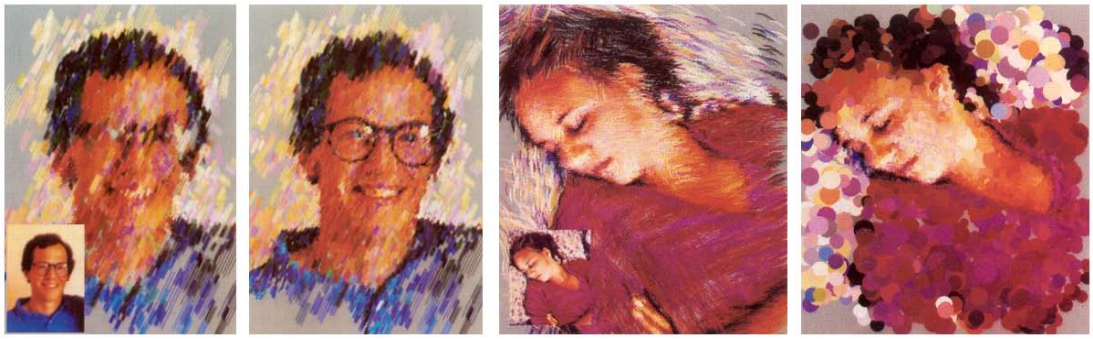


Figure 2-2 Paintings produced interactively using Haeberli's impressionist system — images reproduced from [62]. The user clicks on the canvas to create strokes, the colour and orientation of which are point sampled from a reference photograph (inset).

draw points, lines and arcs on the CRT using a light-pen. However it was not until the early seventies that the first digital painting systems (SuperPaint, and later BigPaint) were implemented by Shoup and Smith at Xerox Parc, Palo Alto. These systems pioneered many of the features still present in interactive painting software today, as well as making hardware contributions such as the 8-bit framebuffer used as the digital canvas. Numerous commercial interactive paint systems followed in the early eighties, and many interactive systems exist on the today's software market.

The use of interactive systems in AR initially grew out of the desire to create environments for the use and demonstration of novel artistic material models (Section 2.2). However, these later evolved to semi-automatic systems which assisted the user in creating artwork from images. Many modern systems extend the concept of painting on a two-dimensional canvas to a three-dimensional environment. These object-space painting environments have been used extensively in commercial digital animation.

2.3.1 User assisted digital painting

Haeberli [62] observed that artwork produced in digital paint systems often lacked colour depth, and attributed this behaviour to a prohibitively long "time to palette"; the time taken to select new colours in contemporary paint systems. In his influential paper entitled "Paint by numbers", Haeberli proposes a novel, semi-automatic approach to painting which permits a user to rapidly generate impressionist style "paintings" by creating brush strokes, the colour and orientation of which are determined by point-sampling a reference image. This reference image is of identical geometry to the digital canvas. Users are able to choose various brush shapes and sizes for painting; for example, choosing fine brushes to render detailed areas of the source image. The process of point-sampling the reference image, combined with image noise and the presence of stochastic variation in stroke parameters, causes variation in stroke colour which gives

an impressionistic look to the painting. The system thus enables even the unskilled user to create impressionist paintings from photographs via a computer assisted interactive process (Figure 2-2).

In the same paper, Haeberli formalises the concept of a painting as an ordered list of brush strokes — each with associated attributes, which in his system are:

- Location: the positions of the brush stroke
- Colour: the RGB and alpha colour of the stroke
- Size: the scale of the stroke
- Direction: angle of the stroke in the painting
- Shape: the form of the stroke, e.g. circle or square

Virtually all modern AR techniques make use of this paradigm in their generation and representation of paintings and drawings. Furthermore many painterly rendering systems use a variation upon Haeberli's point-sampling technique to create their output from reference images. For example, Curtis *et al* use a similar approach to demonstrate application of their water colour simulations [33].

Systems which allow the user to apply paint and texture to modelled 3D surfaces have also been proposed. Arguably the first was described by Hanrahan and Haeberli [67], who implement a WYSIWYG painting system in object-space. Their system facilitates not only application of pigment to the surface but also models the build up of paint on surfaces, which are subsequently lit and rendered using bump mapping. Disney's *Deep Canvas* is a descendant of this system in which brush stroke parameters update dynamically according to viewing angle, and has been used in many feature length animations such as "Tarzan" [Disney, 1999]. The Piransei system, described by Lansdown and Schofield [97], is another example of a system which permits drawing and painting directly on to object-space surfaces.

An interesting approach to applying artistic texture to object-space surfaces was presented by Markosian *et al* [108], who adapt the concept of "Graftals" [143] for use in AR. Graftal textures are specified procedurally at a number of discrete levels of scale. The area of view-plane occupied by a Graftal after camera projection determines the level of detail it will be rendered at — though, unlike pyramidal texture approaches such as MIP-maps [168], graftals are not interpolated between levels of scale (the texture corresponding to the closest scale representation is rendered). The authors demonstrate application of their system with a Dr Seuss style animation, in

which close-up tree graftals exhibit leaves, yet distant views of the same graftals indicate only an approximation of the tree using a couple of leaves. The system prevents superfluous stroke texture being applied to distant objects, and so gives the impression of abstracting away distant detail. However the approach is expensive in terms of interaction to set up (due to the procedural specification of graftals), and flickering may be observed when animating Graftal textures at scales halfway between any two consecutive discrete scale boundaries.

Recently, semi-automatic image segmentation approaches to painting have been proposed [3, 38]. Both techniques operate by segmenting the source image at various scales (ranging very coarse to very fine), to form a nested hierarchical representation of regions in the image similar to that of a low-pass pyramid. An image region in the output may then be rendered at scales, proportional to the depth to which the "scale-space" hierarchy is traversed. This tree depth is specified interactively by the user. In [38] control of scale is varied using interactive gaze trackers, while in [3] a cross shaped mask is placed at a user specified location in the image — hierarchy depth is proportional to distance from this mask.

2.3.2 User assisted sketching and stippling

Salisbury *et al* [134] presented a suite of interactive tools to assist in the creation of digital pen-and-ink illustrations, using a library of stroke textures. Textures were interactively placed and clipped to edges in the scene; these edges could either be drawn or estimated automatically using an intensity based Sobel filter. This system was later enhanced, using local image processing operators (intensity gradient direction), to increase automation in [135]. In this enhanced system pen-and-ink renderings could be quickly constructed using a user assisted system, similar in spirit to Haeberli. Users specify texture types and orientations for regions of an image. Line-art textures comprised of β-splines are then composited on to the canvas, adapting automatically to tone present in the image. A further interactive pen-and-ink system is described by Elber [45] which makes use of graphite pencils to sketch on three-dimensional surfaces. Techniques to interactively vary the weight, thickness, and accuracy of contours in a CAD system's output were outlined by Lansdown and Schofield [97]. Draft CAD designs rendered in a sketchy form by this system were found to convey a lesser degree of finality to users, and the qualitative studies presented were among the first to show that variation in rendering style can strongly influence perception of an image [97, 138].

Saito and Takahashi [133] described a "comprehensible rendering" system, in which operations such as Sobel edge detection and depth mapping were performed to produce a bank of filtered image from an object-space model, for example a machine part.

Users were then able to interactively combine various filtered images to generate more "comprehensible" renderings of the model, for example with thick holding lines. This research formed the basis of later work by Gooch *et al* [57] who proposed a suite of novel lighting models for non-photorealistic rendering of CAD output. Notably this work also improves on Saito and Takahashi's approach to silhouette edge detection by computing such edges directly using the object-space scene (rather than performing Sobel filtering on the 2D rendered image).

Various AR techniques for interactive stippling have been presented, for example [9, 43]. Stipples are small marks used to depict the shape of objects; typically dense regions of stipples are used to depict visually dark regions and are so used to convey shape through shading cues. In this sense, stippling, dithering and half-toning algorithms have much in common, and a growing trend among researchers has been to draw upon the wealth of dithering research (originally developed for use on low colour depth displays) for the purposes of AR. Indeed it could be argued that artistic styles such as pointillism unconsciously anticipated the modern day dithering processes such as offset lithography and half-toning [48]. Conversely novel dithering algorithms now often cite AR as a potential end-user application. One such example is Buchanan's work [9], in which controlled artifacts such as high frequency noise may be used to model AR textures, such as coarse paper.

Recent developments in interactive systems tend to concentrate upon human-computer interaction issues, for example transfer of skill from the domain of brush and canvas to that of the digital computer. This was the aim of Baxter *et al*'s [4] haptic feedback brush, and partial motivation for DeCarlo and Santella's painting system [38] which was driven by eye tracking data. Interactive tools have also been developed to facilitate virtual sculpture using more exotic media such as wood [117], and the engraving of copper-plate [100].

Finally, various user interfaces for the creation of 3D shapes *from* sketches have been presented. The SKETCH system proposed by Zeleznik *et al* [180] (and the later Teddy system [82]) permit the rapid creation of 3D shapes using gestural sketches.

2.4 Fully Automatic AR Systems

The modelling of artistic materials, and their successful application to interactive and semi-automatic painting environments, provided a means by which the user and computer could collaborate to create impressive digital artwork. However this process was often labour intensive due to the high level of interaction required (usually at the

level of individual stroke placement), and increasing demands for automation lead researchers to propose novel, fully automatic AR systems. The subsequent evolution of automated AR from interactive painting environments (notably from Haeberli's "Paint by numbers" system [62]) has driven a trend among AR systems to create artwork by compositing discrete atomic rendering elements termed "strokes" on a digital canvas. These "strokes" may take the form of simulated brush strokes or some other medium such as stipples. This process is generically termed "stroke based rendering" and is the paradigm adopted by the vast majority of automated AR algorithms operating in both object and image space. The automatic nature of such systems dictates that the attributes of strokes are no longer directly influenced by the user, but are necessarily decided by procedural, automated heuristics specific to the particular AR algorithm used. These systems aspire to provide a "black-box" which will accept a source model or image, and output an artistic rendering with a bare minimum of user interaction.

2.4.1 Painterly Rendering Techniques

The earliest automated painterly rendering algorithm is described by Haggerty [63], and is a commercial adaptation of Haeberli's "Paint by numbers" system. Many stroke attributes set interactively in Haeberli's system, such as stroke scale and location, as well as some data-dependent attributes such as orientation, are replaced by simple pseudo-random processes in this automated system. Although such use of non-determinism serves to disguise the machine origins of the rendering, salient image detail is often lost in the final painting and results are difficult to reliably reproduce. Loss of salient detail can be mitigated by aligning strokes tangential to the direction of intensity gradient in the image [62, 71, 103]. Many commercial software packages such as the Linux GIMP and Adobe Photoshop contain painterly filters which operate in this manner.

Meier extended Haeberli's point-sampled particle system into three-dimensions [111], addressing both the problem of rendering object-space scenes to produce paintings, and in particular of producing temporally coherent painterly animations. The principal contribution of the paper is the stochastic, but uniform, scattering of paint particles uniformly on to surfaces in object-space (in contrast to [62, 63]); this is achieved by triangulating surfaces and distributing strokes over each triangle in proportion to its area. However brush strokes are not rendered upon the triangulated surfaces themselves — instead the stroke locations are projected to image-space during scene rendering, where they are rendered and textured entirely in 2D. The order of rendering is determined by scene depth, with standard hidden surface removal algorithms determining which strokes are visible for a given viewpoint. The resulting animations exhibit good temporal coherence in the majority of cases. Problems arise when certain projections, for example perspective, cause gaps to appear between strokes. Also as objects rotate in

axes non-parallel to the viewer's optical axis, the varying depths of stroke particles can cause swimming in the animation. A similar technique for rendering static object-space scenes in water-colour was implemented by Curtis *et al* as a means of demonstrating their water-colour media simulation system. A real-time implementation of Meier's system is described in [107], where faster execution speeds are possible due to a novel probabilistic speed-up of Appel's hidden surface removal algorithm.

Litwinowicz [103] proposed the first automatic, data-dependent painting algorithm to operate entirely in image-space (that is, the first technique to place strokes according to image content, rather than in a pseudo-random fashion). The strategy employed by Litwinowicz is to "paint inside the lines", so preventing strokes from crossing strong edges in the source image. This clipping strategy may be seen to be partly motivated by Salisbury *et al*'s earlier clipping of textures in [134]. By preserving edge detail the algorithm generates paintings of subjectively higher quality than previous pseudo-random methods. Processing begins by computing a Sobel edge field $(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ (see Figure 1-1) for a source image I, from which gradient direction $\theta(I)$ and gradient magnitude $E(I)$ fields are derived:

$$\begin{aligned} \theta(I) &= \arctan\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right) \\ E(I) &= \left[\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right]^{\frac{1}{2}} \end{aligned} \tag{2.1}$$

The system paints with short, rectangular strokes of fixed size, which are aligned tangential to the local gradient direction field. The edge magnitude field is globally thresholded at a user-selected limit to create a binary mask of edges. Any strokes which cross an edge pixel are clipped, thus retaining the integrity of edges in the final painting. Strokes are placed in uniformly sampled positions, but drawn in a random order (again, to help convey the impression that the painting was not created by a deterministic machine). The colour of each stroke is set by point sampling a reference image as with [62]. Notably, Litwinowicz was the first to extend his technique to produce painted animations from video, and we defer discussion of this aspect of his algorithm to Section 2.5.2.

Treavett and Chen propose an automatic image-space painting method in [159], which used statistical measures to drive placement of strokes. Strokes are initially placed stochastically, and for each stroke location a small, local pixel window is selected. The standard deviation and variance of pixel intensities within this window are calculated, and strokes aligned along the axis of least variance (i.e. the principal eigen-axis). Shiraishi and Yamaguchi propose a very similar painterly algorithm in [140], using

Figure 2-3 Incremental stages in Hertzmann's coarse to fine approach — images reproduced from [71]. The source image is shown on the top-left, the coarse sketch is top-right. The bottom row of images contain the final two layers of painting, from left to right. Observe that fine non-salient texture on the shirt, and salient detail on the hands, is afforded equal emphasis.

chromatic variance rather than intensity variance, and using windows of varying size.

Hertzmann presented the first automatic painterly rendering technique to paint with curved (β-spline) strokes rather than dabs of paint [71]. Hertzmann's technique operates in image-space and employs a multi-scale approach to painting. First, a low-pass image pyramid is created at various predefined scales. Sobel edge magnitude and direction fields are computed for each pyramid level (after Litwinowicz [103]). A "coarse sketch" is produced by painting curved strokes using data from the coarse pyramid layer. Successively finer pyramid layers are painted over this sketch using progressively smaller strokes; however stroke are only overlaid in areas where the difference between the previous and current canvas layers is above a user-tuned threshold. Regions containing the finest scale detail are therefore painted last. Control points for each spline stroke are selected in an iterative manner, by hopping a predefined distance between pixels in the direction of their Sobel edge gradient and keeping a history of visited pixels.

Gooch *et al* propose a further image-space painterly technique in [58] which paints using curved strokes. The system approaches painting in a novel way, by identifying regions of homogeneous intensity value in the image, which are subsequently skeletonised (i.e.

their medial axis is computed). Curved strokes are then fitted to this medial axis. We observe that this approach is extremely intolerant to noise or texture in an image which prevents the formation of regions of homogeneous intensity, and so causes the system to tend toward photorealism in most real images.

Hertzmann *et al* proposed a novel *example based* approach to painting in [74]. Given a image A and a painting of that image A', the system is able to take a new image B and render it in a similar style to generate B'. Hertzmann *et al* dub this process "paint by analogy". For each point in the novel image B a search is performed to determine the best matching pair of pixels in A and A'. The best match for a particular point is determined by examining its immediate pixel neighbourhood, and also by seeking to maintain the coherence of that point with its neighbours. Various low-level filters such as blurring and sharpening can be learnt, as well as approximations to AR painterly filters.

Finally, relaxation based approaches to painting have briefly been considered. Although Haeberli was the first to openly propose such approaches [62], the first algorithmic solution was described by Hertzmann in [72]. In his system, β-spline strokes double up as active contours (snakes) and are used to seek a painting which minimises a predetermined objective function. Hertzmann selects a function which favours preservation of the maximum amount of high frequency content between the original and painted images. His relaxation algorithm is iterative in nature, and proceeds as follows:

1. Compute a "score" for the paintings by evaluating the objective function.
2. Select a stroke at random, or possibly add a new stroke.
3. Modify the position of the selected stroke's control points (via active contour relaxation), or possibly delete it.
4. Repeat from (1) until the painting's score is acceptably close to the goal state.

As with all active contour implementations success is sensitive to initial contour location, due to the inherent susceptibility of snakes to local minima [91]. Unfortunately Hertzmann does not give details on how his snakes are initialised, though implies that the initial locations might be established via some stochastic process. A further relaxation process employing stochastic search is proposed in [155] in which strokes are iteratively added to a painting at random; after each iteration an objective function is evaluated. After a predetermined amount of time, the painting which has evaluated to the lowest score is output. The objective function is again based upon conservation of all high frequency information in the rendering.

2.4.2 Sketchy and Line-art Techniques

The first automated sketch rendering was reportedly produced by a plotter with a wobbly pen attached [161]. Many improvements toward automated pen-and-ink and sketchy rendering have since been presented.

Hall describes Q-mapping (a corruption of "cue mapping") [65], and is able to produce a wide variety of stylised pen-and-ink renderings from both 2D and 3D sources. In the simplest case, Q-maps are generated by creating multiple logical slabs aligned to each principal axis in the source. Logical rules are then specified with respect to these slabs; for example, marks may be made where slabs intersect (binary AND). The width and spacing of slabs are set as data-dependent power functions of image intensity (2D) or surface irradiance (3D). Convincing cross-hatching effects are easily created using this system.

The vast majority of pen-and-ink renderers, however, operate exclusively in object-space. A number of algorithms have been presented which use the principal curvature direction of a surface to orientate hatches [44, 170, 171]. One such system, proposed by Winkenbach and Salesin, used pen-and-ink textures defined at multiple resolutions. Users could interactively specify levels of detail for particular surfaces, and thus vary the resolution at which texture on that surface was rendered. This permitted a wall, for example, to be indicated by the rendering of just a couple of bricks, rather than rendering every single brick. This produces impressive interactively driven abstraction of detail, though the authors acknowledge that automation of such composition would be difficult.

Other pen-and-ink approaches make use of 3D surface parameterisation to guide hatch placement [44, 171], though the surface parameterisation itself ultimately determines the quality of the hatching (if one exists at all). Veryovka and Buchanan [162] employ modified half-toning, and real-time implementations of hatching are described in [107] and [122]. An unusual approach to sketch rendering is presented by Mohr *et al* [112], in which API calls to the OpenGL graphics library are intercepted and modified to produce a sketch-like output.

2.5 Non-photorealistic Animation

The success and availability of automatic AR techniques has encouraged research in to the problem of automatically producing non-photorealistic animations. The principal goal in producing such animations is that of generating animations exhibiting good temporal coherence. Without careful rendering, the discrete strokes that comprise a

scene tend to flicker or move in a counter-intuitive manner which distracts from the content of the animation. Consequently it is important that the motion of strokes is smooth, and matches the motion of semantic objects within the animation. Although computing object motion is trivial given an object-space scene, motion recovery is very difficult from 2D monocular sources such as video. It is predominantly for this reason that object-space AR techniques have been more successfully applied to produce artistically styled animations than their image-space cousins.

2.5.1 Animations from Object-space (3D)

Many modern animations are produced on computer and it is now unusual to encounter a hand-drawn feature length animated film, or animated television series. This is arguably due to improved productivity digital systems offer animation houses. Most modern graphical modelling packages (3D Studio MAX!, Maya, XSI SoftImage) support plug-ins which offer the option of rendering object-space scenes to give a flat shaded, cartoon-like appearance rather than strict photorealism. Because scene geometry is known at the time of rendering, these systems are able to generate temporally coherent animations with relatively fewer additional complications than when rendering a single frame. Typically cartoon shaders operate using specially adapted lighting models for shading, and compute silhouette edges with respect to the camera perspective, which are rendered as thick "holding lines". Though not without their technical intricacies, such functions are easily performed with an object-space representation of the scene to hand.

Meier was the first to produce painterly animations from object-space scenes [111] (see Section 2.4.1). Since this initial work, a number of object-space AR environments have been presented. Hall's Q-maps [65] (Section 2.4.2) may be applied to create coherent pen-and-ink shaded animations from object-space geometries. A system capable of rendering object-space geometries in a sketchy style was outlined by Curtis [32], and operates by tracing the paths of particles travelling stochastically around isocontours of a depth image, generated from the 3D object.

2.5.2 Artistic rendering from video (2D)

A few post-production effects have been developed to create animations in artistic styles directly from video (2D) — in particular the problem of producing temporally coherent painterly animations has received some attention. The problem central to painterly animation is that painting individual frames of video on an independent basis produces an unappealing swimming, often due to noise or motion in the image, and sometimes due to the presence of stochastic decisions in the algorithm itself. This appears to be

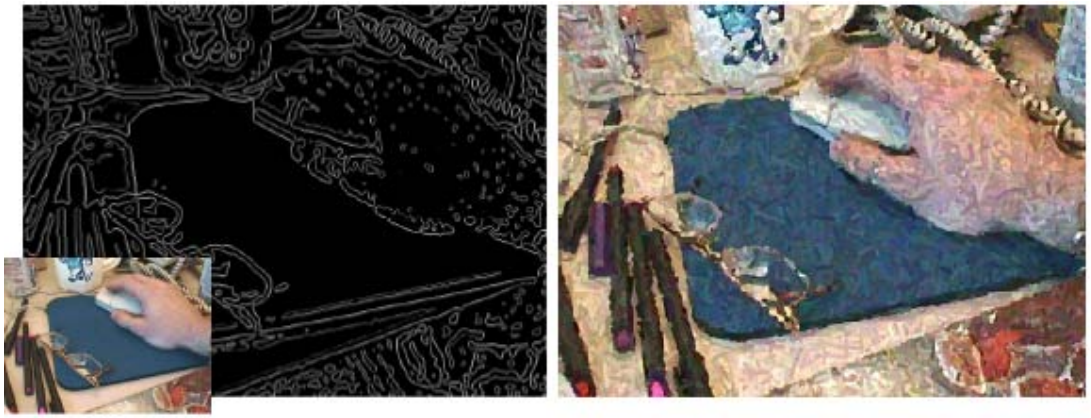


Figure 2-4 Illustrating Litwinowicz's painterly video technique — images reproduced from [103], original inset. Sobel edges (left) are computed for each frame; strokes are placed and clipped to these edges (right). Strokes are translated over the course of the animation, according to the optical flow vectors computed between frames. This algorithm represents the state of the art in creating automated painterly animations from video.

an in-principal difficulty when extending any static, stroke based AR algorithm to video.

A naive solution is to fix the position of strokes on the view-plane, however this gives the illusion of motion behind a "shower door"; strokes adhere to the image plane rather than to moving objects within the scene. However some simplistic post-production texturing effects have been created using this approach. A texture is typically blended via per-pixel RGB multiplication with each frame in the video, to give the impression of a canvas or cross-hatched texture within the image content. Examples are the Inland Revenue advertisements currently appearing on British television, where actors seemingly move within a bank note — the texture of the note is simply multiplied over the video frame image. Of course such textures are entirely independent of the video itself and are static relative to image content, giving the impression of video moving behind a textured filter rather that the appearance of authentic animated footage. Hall's Q-maps may be used in a similar manner to generate textures in video, and though they do exhibit adaptation to the intensity of footage moving behind them, the textures themselves are static with respect to video content.

Ideally strokes should adhere to and move with the objects to which they are semantically attached (as with Meier's system [111]), but since video is only a 2D camera projection of object space rather than an object-space representation itself, this goal is difficult to achieve. There have been two solutions proposed in the literature which address this problem.

The first solution was proposed as an extension to Litwinowicz's static painterly tech-

nique (see Section 2.4.1), and makes use of optical flow [7] to estimate a motion vector field from frame to frame [103]. Brush strokes are painted upon the first frame, and translated from frame to frame in accordance with the optical flow vectors estimated from the video (Figure 2-4). This lends a degree of coherence to stroke motion in the resulting animation. Further steps are introduced to prevent strokes bunching together in the view-plane. Prior to rendering each frame a Delauney triangulation of stroke centroids is performed as a means of measuring the local stroke distribution density. In regions where strokes have densely bunched together (as a consequence of stroke motion) strokes are deleted at random until the density falls to an acceptable level. Conversely, in sparsely populated regions where strokes have moved too far apart, strokes are added. The ordering of strokes is decided stochastically in the first frame, but that ordering is then maintained through all subsequent frames. Any newly added strokes are inserted into the ordered stroke list at random locations.

In practice Litwinowicz's algorithm achieves satisfactory coherence only after painstaking hand correction of these optical flow fields; errors quickly accumulate due to the per frame sequential manner in which the video is processed. The quality of the method is therefore bound to the quality of optical flow algorithm implemented. No robust optical flow algorithm is known, and these cumulative errors tend to creep in due to simplifying assumptions, for example assuming well textured surfaces, constant scene illumination, or no occlusion, which do not hold true in real-world video sequences [54]. Optical flow methods have also been employed by Kovacs and Sziranyi [96] to produce painted video in a similar way; their algorithm paints using rectangular strokes, extending their "Random Paintbrush Transformation" [155] to video. Again, this technique operates on per frame sequential manner; translating strokes based on optical flow computed from frame to frame. The small contribution over Litwinowicz's method [103] is the ability to set "key-frame" instants in the video in which strokes are painted as if the key-frame were the initial frame. This can mitigate the creeping error problem present due to the accumulation of inaccuracies in optical flow estimates, but can not remove swimming in the sequence (observe that in the limit each frame is painted individually, using a stochastic algorithm [155]).

A second, simpler solution is proposed by Hertzmann [75], who differences consecutive frames of video, re-painting only those areas which have changed above some global (user-defined) threshold. Swimming is therefore localised to those areas of the image are detected as having moved; he dubs this approach "paint-over". However slow moving, intensity homogeneous, or low intensity regions tend to be missed as a consequence of the RGB frame differencing operation. Hertzmann tries to mitigate this by using cumulative differencing approaches, though results do not improve significantly. Fur-

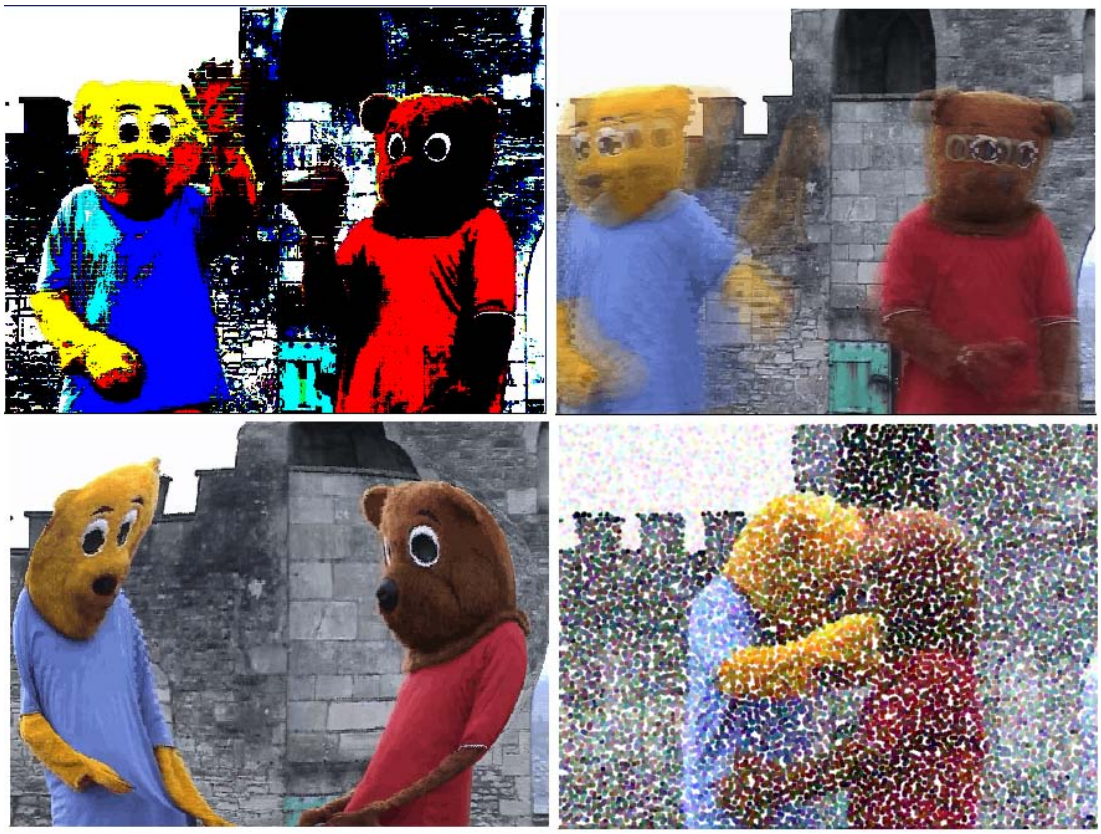


Figure 2-5 Representative examples of the classes of low-level video effects available in commercial software — rendered with Adobe Premier 6.5 [Adobe, 2003]. (a) Posterisation (palette colour quantisation), (b) Motion blur (temporal low-pass filter), (c) Spatial warps (spherical warp), (d) Single scale, per-frame implementation of Haeberli's paint system [62], exhibiting heavy swimming.

thermore, the method only mitigates swimming in static regions — in a fast moving video the entire animation will swim unhindered.

We observe that various image-space tools have also been presented which, while highly interactive, assist in the process of creating digital non-photorealistic animations. Fekete *et al* describe one such system in [46] to assist in the creation of line art cartoons. An interactive system for cartoon rendering was described in [1], whereby a user hand-segmented an image, and active contours (snakes [91]) were used to track the segmentation boundaries from frame to frame (again in a per frame sequential manner). The system fails if the segmented topology of the frame changes i.e. under occlusion. Animations also flicker due to the instability of snakes which are relaxed at each frame (this instability is a consequence of the susceptibility of snakes to local minima).

The film "Waking Life" [Fox Searchlight, 2001] made use of an automated rotoscoping system to give a cartoon appearance to video. The technique of rotoscoping was

pioneered by animators of the 1930s, who captured real-world footage on film, and subsequently traced over frames to rapidly produce cels of animation. Rotoscoping thus reduced the workload for these animators, and enabled Disney to produce their first animated feature-length film "Snow White" [Disney, 1937]. In the automated system used for "Waking Life", artists drew a shape in a single frame, and another ten or twenty frames away. The system then interpolated the shape over the interval between key-frames — a process referred to as "in-betweening" by animators. This automated blend of rotoscoping and in-betweening is sometimes referred to as "advanced rotoscoping" by the Computer Graphics community, and although highly labour intensive, alleviates some of the tedium of the animation process since not every animation frame must be keyed. The resulting animations still tend to swim, due to the motion discontinuities produced by repeatedly keying the interpolation process. Critics have suggested this technical issue to be the motivation behind the scintillating "dream world" setting of the "Waking Life" film.

A number of post-production video effects are available in commercial programs, such as Adobe Premier (Figure 2-5). These provide a suite of low-level effects, for example slow-motion, spatial warping, and motion blur (achieved by temporally averaging, i.e. low-pass filtering, video). Filters are available which modify the RGB or HSV attributes of the video sequence globally; these may be used to produce monochrome and sepia toned video, or vibrant colours by boosting saturation. A posterisation function is sometimes used to give a surreal non-photorealistic effect to video; this effect is produced by quantising colours in the scene. Since palette colour quantisation does not consider spatial relationships between pixels in a scene, the boundaries between coloured regions are often very noisy and do not correspond well to the boundaries of semantic objects within the video itself (Figure 2-5a).

2.5.3 Rendering Motion in Image Sequences

Very little literature exists regarding the rendering of motion in image sequences, since the majority of relevant image-space techniques attempt to mitigate the presence of motion in the video for the purposes of temporal coherence, rather than emphasise it. In an early study [98], Lasseter highlights many of the motion emphasis techniques commonly used by animators for the benefit of the computer graphics community, though presents no algorithmic solutions. Streak-lines and deformation for motion emphasis (Chapter 6) and anticipation (Chapter 7) are discussed. Strothotte *et al* [151] and Hsu [80] glance upon the issue by interactively adding streak-lines and similar marks to images. In the former case, the lines drawn form the basis for a psychological study into perception of motion; in the latter case they are only a means of demonstrating a novel brush model. Artificial squash-and-stretch effects have been applied to em-

phasise motion of cylindrical and spherical objects prior to ray-tracing [18]. A further object-space technique was recently presented by Li *et al* [102], in which users may adjust trajectories of objects to stylise their motion via an interactive process. This work is aligned strongly with motion retargetting (see also [8]), and differs from our work (Chapters 6, 7) in both scope and level of interaction. We automatically analyse trajectories in image-space, visually characterising the essence of that motion using cartoon style motion cues — rather than requiring users to interactively specify novel trajectories in object-space prior to rendering.

The portrayal of motion in an abstract manner is currently under-studied. An empirical study [16] suggests that superimposing several frames of animation at specific intervals can affect perception of time. A set of interactive tools [93] recently proposed by Klein *et al* allows a user to interactively manipulate video via a point and click process; this interactive system can be used to produce output ostensibly similar to earlier stages of our Cubist work which allows motion to be depicted in an abstract manner (Chapter 3), but differs greatly in both level of interaction and abstraction (we construct compositions using identified high level features rather that allowing a user to interactively subdivide blocks of video). The literature remains sparse concerning techniques for two-dimensional rendering of motion, and the synthesis of abstract artistic styles.

2.6 Observations and Summary

We observe that whilst earlier work in artistic rendering concentrated upon the emulation of media and the production of interactive paint systems, there has been a strong trend in recent work toward producing automated rendering systems (see the distribution of publications in Figure 2-7). The overwhelming majority of these interactive and automated systems synthesise artwork by placing discrete "strokes" either upon surfaces within object-space or directly upon an image-space canvas. However, in such systems, little consideration is given to the rendering and appearance of the individual strokes placed. Interestingly this abstraction of stroke placement strategy from media type is a simplification that has crept into the AR literature with seemingly no "real-world" parallel; neither can any discussion or justification of this decoupling be found in any AR literature. Certainly artists would strongly disagree with such an abstraction; brush and pen technique, for example, varies greatly depending on the nature of the subject being drawn [69].

In this thesis we also subscribe to this convenience of abstraction. The majority of algorithms we propose in the following chapters are concerned only with issues of stroke

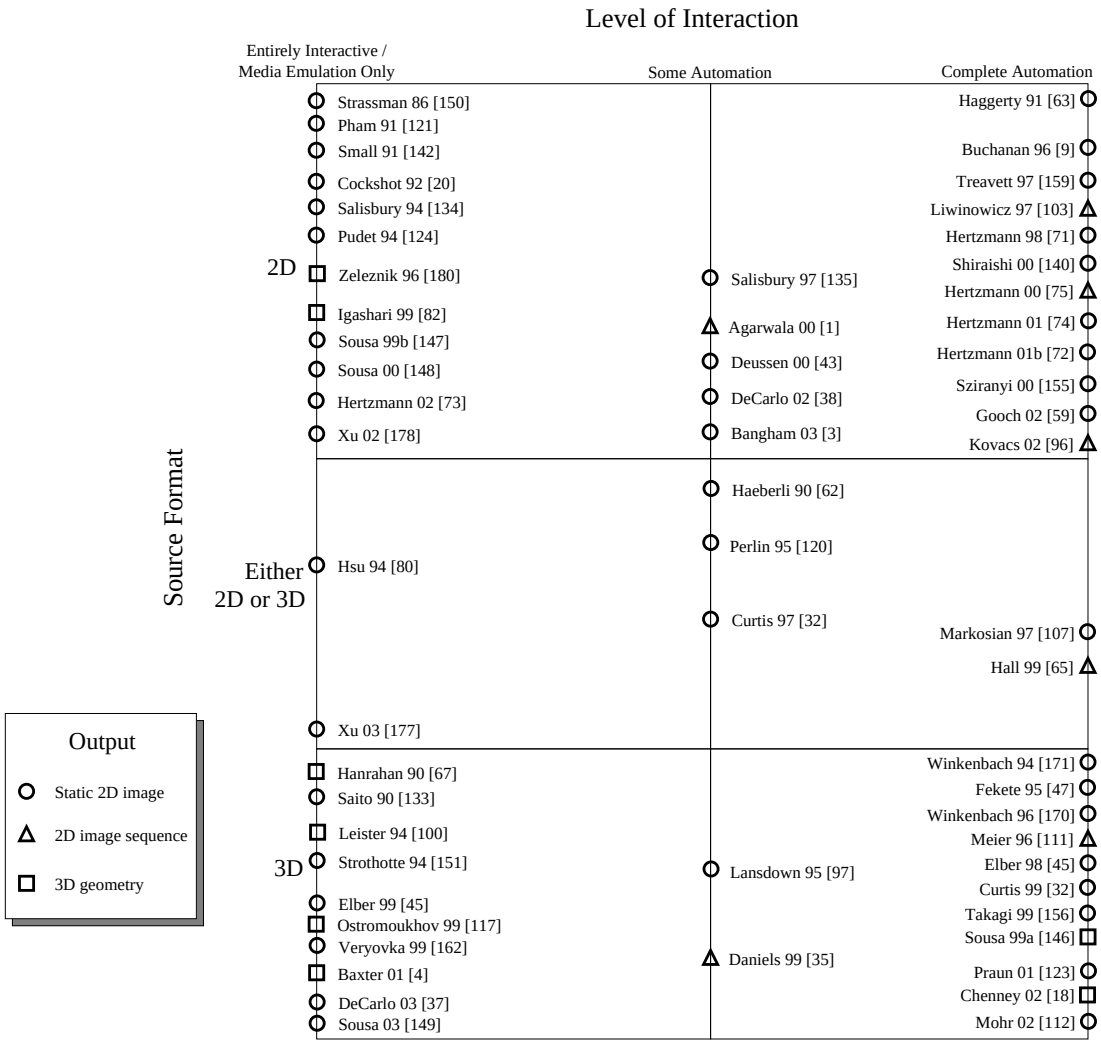


Figure 2-6 Reviewed artistic rendering systems classified according to dimensionality of source and rendered content, and according to level of user interaction. Systems are denoted by first author, year of publication, and thesis citation. Entries applicable to more than one category have been placed according to their principal contribution area.

placement. Simple texture mapped strokes are almost always sufficient to demonstrate such algorithms, and the substitution for more extravagant strokes is straightforward due to the decoupled nature of AR. However in Chapter 4 we glance upon this issue, describing a system that interpolates between preset media styles according to the artifact (for example, edge or ridge) encountered in the source image. We suggest that an area open for future research may involve a reconsideration of the decoupled nature of AR, and the stylistic limitations that such an approach imposes on potential development of the field.

Haeberli formalised the representation of a painting as an ordered list of strokes over a decade ago [62]. Researchers have since concentrated upon either novel stroke place-

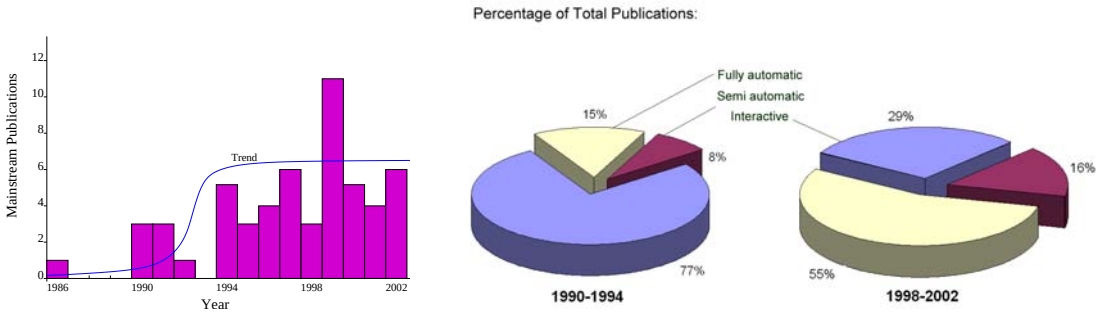


Figure 2-7 Illustrating trends in AR. Left: demonstrating the increasing rate of mainstream publication of AR techniques (major international conferences and journals, trend in blue). Right: The focus of AR development has shifted from interactive to automatic systems, which make use of local (low-level) signal processing operators to guide rendering.

ment strategies, or upon the realistic emulation of artistic media within individual placed strokes. The gamut of artistic rendering styles has therefore been constrained to traditional stroke based forms of art; for example sketching, hatching and painting. No consideration has been given to the automated synthesis of artistic styles at a higher level, for example to the development abstract artistic styles or composition technique.

It is perhaps the simple, local nature of these virtual strokes that has led researchers to employ local image processing operators to guide stroke placement in automatic image-space AR systems. In our review we have documented the use of three classes of local window operator to drive automatic stroke placement; intensity gradient (Sobel operator), intensity or chromatic variance, and segmentation.

Intensity gradient is by far the most popular measure for driving stroke placement heuristics, and is used by the majority of algorithms [62, 71, 72, 75, 103, 134, 135]. Intensity gradient is measured, in all cases, using the Sobel operator; a simple edge detection technique developed in the seventies, which computes the first order derivative of 2D scalar fields (in the case of AR, greyscale images) to which it is applied. This is achieved by convolving a small kernel (pixel window) of particular configuration (see Figure 1-1) with the image, which simultaneously smooths the image signal along one principal axis whilst differentiating the image signal along the orthogonal axis. The Sobel operator is a high-pass filter, and as such its response over a window is directly proportional to the magnitude of high frequency content within that window. The frequency at which the response peaks can be reduced by low-pass filtering (smoothing) the Sobel kernel prior to application, or equivalent, low-pass filtering the image itself.

The majority of automatic AR algorithms scatter strokes over the image, and give preferential treatment to strokes falling in regions exhibiting high Sobel response (regions

containing high frequency artifacts, such as edges or fine texture). The nature of this preferential treatment depends on the heuristics used in the AR algorithm — we recall two representative examples:

1. In Litwinowicz's method [103], the Sobel edge magnitude field (equation 2.1) is globally thresholded; this serves as a simple means to determine whether a given pixel forms part of a high frequency artifact, such as an edge. Strokes are then clipped during painting to ensure that they do not cross the thresholded edges. The painting process is intentionally biased toward conservation of fine scale, high frequency content in the final rendering; thus there is an tacit assumption that high frequency content correlates with salient detail in the image. In practice Litwinowicz advocates blurring of the original image to remove extraneous fine detail (effectively shifting the peak response of the Sobel filter to lower frequencies). The scale of the low-pass filter and the aforementioned Sobel threshold value are user-defined, and constant over the entire image; the user tweaks these constants to create an aesthetically pleasing rendering. It is assumed that simple blurring, i.e. filtering on scale, can be used to distinguish salient and non-salient areas of the image. If non-salient detail is of greater or equivalent scale to salient detail, we can not set a frequency threshold which removes all non-salient detail without removing some salient detail.

2. In Hertzmann's methods [71, 74, 75], strokes are laid down in successive layers using a coarse to fine pyramid based approach. The initial layer of strokes is taken from the coarsest scale layer of the low-pass pyramid. Successive layers of strokes correspond to increasingly finer scale artifacts, and thus strokes corresponding to the finest scale artifacts are painted last, on top of coarser strokes. Again we see there has been an implicit assumption correlating fine detail with salience, since strokes from fine scale regions are painted over coarser-scale regions of the image. The finest scale at which artifacts are painted can be used as a threshold, as with the previous example [103], raising identical objections.

This assumption is also made in algorithms [140, 159] which use statistical variance measures, rather than the Sobel operator, to automatically decide stroke parameters within a window local to that stroke. The measure of variance magnitude over a window has no spatial component; one can determine if signal variation exists within a window, but can extract no information regarding the frequencies or spatial relationships within that signal. Consequently, algorithms vary the size of the window over which variance is measured in order to determine the spatial scale of the signal, and so decide stroke size in the painting. Yamaguchi and Shiraishi [140] use windows of several sizes to implement their variance based painting approach, closely emulating the coarse to fine approach of Hertzmann [71]. In doing so they again correlate fine

scale, high frequency details with image salience. Moments have been employed to orientate paint strokes; just as Sobel-driven algorithms orientate strokes tangential to intensity gradient direction. Treavett and Chen [159] compute moments within a local window of constant size, and align strokes along the axis of least variance (a technique also adopted by the multi-scale window approach of [140]). There does not appear to be any advantage in orientating strokes using moments rather than gradient direction.

Finally, we have reviewed a fully automatic, segmentation driven technique [58] which locally segments the source image regions of homogeneous luminance (using a modified flood fill). These regions are used to form brush strokes for painterly rendering. In regions containing detail such as edges or texture, the size of regions generated by the segmentation process decreases and the rendering tends back toward photorealism. Again, the painting process is biased toward conservation of fine scale content.

Examining this chapter retrospectively, all fully automatic image-space AR algorithms we have reviewed may be seen to make an important but implicit assumption that high frequency image artifacts are salient, and that all such artifacts should therefore be conserved in the final rendering. In fact this goal was explicitly stated both in the objective function of Hertzmann's active contour based relaxation algorithm [74], and in [155]. Given this observation, one may consider automatic AR algorithms as local, non-linear filtering processes which operate at the pixel neighbourhood level to generate "artistic" renderings. Current painterly renderers may be thought of as performing an anisotropic blurring of the image, the effect of which is attenuated in areas of the image populated by high frequency artifacts. This observation is novel to us, but well illustrated by Hertzmann *et al*'s paint-by-analogy systems [72]. These learn "painterly styles" by training on which pixel neighbourhoods in an original map onto which pixel neighbourhoods in a painting. By attempting this, Hertzmann *et al* subscribe to this spatially local, low-level approach to artistic rendering, which characterises current AR. In the following chapter we discuss the difficulties which arise with this low-level approach; the most significant of which is an unappealingly consistent conservation of fine detail within the final rendering regardless of its perceptual importance.

We have reviewed some semi-automatic AR algorithms which are driven by image segmentation operations [3, 38]. As with [58], these algorithms paint by breaking the image up into small regions in a manner conducive to the preservation of detail. Rather than relying on image processing driven heuristics to identify salient regions of the image, the user is instead permitted to specify these regions (using binary masks, or interactive eye trackers). As we discussed in Chapter 1, this appeal to the human visual system to guide rendering is in line with the form of higher level image analysis that this thesis

advocates; although such an appeal is ultimately limiting, for example in the context of producing AR animations. In Chapters 3 and 4 we develop *fully automatic* painterly renderers which use globally computed salience measures to drive emphasis in rendering.

We observe from the distribution of publications in Figure 2-6 that the automated creation of AR animations from video clips is a very sparsely researched problem. Although it is often possible to produce temporally coherent animations from techniques which operate in object-space [18, 35, 111], comparatively few methods [75, 96, 103] are capable of producing stylised animations from image-space (for example post-production video, see Figure 2-7). Techniques that aim to produce temporally coherent animations from 2D image sequences use local motion approximations (optical flow, inter-frame differencing) to correlate stroke motion with video content, which often yields imprecise or erroneous motion fields (Section 2.5.2). Additionally, these techniques process video on a per frame sequential basis, translating brush strokes from frame to frame using the estimated motion field. This is another manifestation of local processing though in a temporal rather than spatial sense. The result is that processing errors from a single frame propagate forward to subsequent frames, causing a cumulative error that manifests as uncontrolled swimming in the animation. We address this problem in Chapter 8. Furthermore all automatic video driven AR algorithms seek to mitigate against motion for the purposes of coherence; none attempt to emphasise motion, visually (Chapter 6) or by affecting temporal properties of the video (Chapter 7). We argue that this is further manifestation of the disadvantages of processing at temporally low level. By analysing video at a temporally higher level we later show that we are able to extend the range of video driven AR to include stylisation of motion with the sequence itself, an important artistic device for modern day animators, but one that has so far been neglected in AR.

We conclude that current AR techniques for rendering images and video sequences concentrate predominantly upon stroke based artistic styles, employing spatiotemporal local processing methods to guide stroke placement. There is clear scope for development of artistic rendering algorithms which operate at a higher level, both spatially (for example, performing a global analysis to identify salient features within images) and temporally (examining past and future events rather than only the subsequent frame during video processing).

Part II

Salience and Art: the benefits of Higher level spatial analysis

Chapter 3

Painterly and Cubist-style Rendering using Image Salience

In this chapter we make observations on the manner in which artists draw and paint, and contrast this with the spatially local manner in which existing automatic AR techniques operate. To address this discrepancy we introduce the use of a globally computed perceptual salience measure to AR, which we apply to propose a novel algorithm for automatically controlling emphasis within painterly renderings generated from images[1]. We also propose a further algorithm which uses conceptually high level, salient features (such as eyes or ears) identified across set of images as a basis for producing compositions in styles reminiscent of Cubism[2]. These algorithms support our claim that higher level spatial analysis benefits image-space AR — specifically, enhancing aesthetic quality of output (though controlled emphasis of detail) and broadening the gamut of automated image-space AR to include artistic styles beyond stroke-based rendering.

3.1 Introduction

A drawing or painting is an artist's impression of a scene encapsulated on a two-dimensional canvas. Traditional artists often build up their renderings in layers, from coarse to fine. Coarse structure in a scene is closely reproduced, either using a trained eye, or by "squaring up"; overlaying a grid to create correspondence between image and canvas. By contrast, details within the scene are not transcribed faithfully, but are individually stylised by the artist who can direct the viewer's focus to areas of interest through judicious abstraction of the scene. Typically an artist will paint fine strokes to

[1] The salience adaptive painterly rendering technique was published in [22], and was awarded the Terry Hewitt prize for best student conference paper.

[2] An earlier version of our Cubist rendering work was published in [23].

Figure 3-1 Two examples of paintings illustrating the artist's importance-driven stylisation of detail. Non-salient texture in the background has been abstracted away, yet salient details are emphasised on the figures (left) and portrait (right) using fine strokes.

emphasise detail deemed to be important (salient) and will abstract the remaining detail away. By suppressing non-salient fine detail, yet implying its presence with coarser strokes or washes, the artist allows the viewer to share in the experience of interpreting the scene (see, for example, the paintings in Figure 3-1). Tonal variation may also be used to influence the focus, or centre of interest, within a piece [69].

The omission of extraneous, unimportant detail has been used to improve the clarity of figures in medical texts, such as Gray's anatomy [60] which consists primarily of sketches. Such sketches remain common practice in technical drawing, and were also used heavily by naturalists in the 19th century. In cinematography too, camera focus is often used to blur regions of the scene, and so redirect the viewer's attention. Both adults and children can be observed to create quick sketches and drawings of a scene by jotting down the salient lines; effectively leaving the viewer the task of interpreting the remainder of the scene. Picasso is known to have commented on what he regarded as the pain-staking nature of Matisse's art [64]; suggesting that Matisse worked by tracing the lines of a subject, then tracing the lines of the resulting drawing, and so on, each time stripping down the figure further toward its essence — "... He is convinced that the last, the most stripped down, is the best, the purest, the definitive one". In effect Matisse is iteratively refining his sketches to the few lines and strokes he deems to be salient.

We have observed (Section 2.6) that the heuristics of fully automatic AR techniques modulate the visual attributes of strokes to preserve all fine detail present in the source image. Specifically the emphasis, through level of stroke detail, afforded to a region within an artistic rendering is strongly correlated with the magnitude of high frequency

content local to that region. Such behaviour can be shown to differ from artistic practice in the general case. First, consider that fine detail, such as a fine background texture, is often less salient than larger scale objects in the foreground. An example might be a signpost set against a textured background of leaves on a tree, or a flagstone set in a gravel driveway. Second, consider that artifacts of similar scale may hold differing levels of importance for the viewer. We refer the reader back to the example of the portrait against a striped background (Figure 3-3, middle-left), in which the face and background are of similar scale but of greatly differing importance in the scene. An artist would abstract away high frequency non-salient texture, say, on a background, but retain salient detail of similar frequency characteristic in a portrait's facial features. This behaviour is not possible with automatic AR algorithms which seek to conserve all fine detail in a rendering, irrespective of its importance in the scene.

The level of detail rendered by existing automatic AR techniques is determined by the user. Constants of proportionality must be manually set which relate stroke size (and so, detail in the final rendering) with high frequency magnitude. The values of these parameters are constant over the entire image and, in practice, setting these values is a tricky, iterative process, which often requires several runs of the algorithm to produce an aesthetically acceptable rendering [58, 71, 103]. Keeping too little of the high frequency information causes salient detail to be lost, and the painting to appear blurry; keeping too much causes retention of non-salient texture (too many details in the "wrong places") which cause the output to tend toward photorealism. Moreover, if salient detail is of lower frequency magnitude than non-salient detail, then there is no acceptable solution obtainable by varying these constants — either some non-salient detail will be erroneously emphasised to keep the salient detail sharp, or some salient detail will be abstracted away in an effort to prevent emphasis of non-salient detail. This, of course, points to a flaw in the original premise; that all fine detail is salient. We thus observe that although for a given image the set of salient artifacts may intersect the set of fine scale artifacts, there may remain many salient artifacts that are not fine scale, and many fine scale artifacts that are not salient. We conclude that many images exist for which current AR methods do not emphasise some or all of the salient elements in the scene. The behaviour of current AR is at odds with that of the artist, and it is arguably this discrepancy that contributes to the undesirable impression that AR synthesised renderings are of machine, rather than true human origin.

In our opening paragraphs (Chapter 1), we argued that the notion of importance, or salience, is a relative term. When one speaks of the salience of regions in an image, one speaks of the perceptual importance of those regions *relative to that image as a whole*. Global analysis is therefore a prerequisite to salience determination; the independent

examination of local pixel neighbourhoods can give no real indication of salience in an image. The aesthetic quality of output synthesised by automatic AR would benefit from higher level (global) spatial analysis of images to drive the decision processes governing emphasis during rendering.

A further observation in Chapter 2 notes the ubiquitous trend in AR to process images at the conceptually low-level of the stroke (stroke based rendering). There are certain advantages to processing artwork at this low-level; algorithms are not only fast and simple to implement, but very little modelling of the image content is required — we have already highlighted the frequency based decision model used to guide rendering heuristics. The simplicity of this modelling admits a large potential range of images for processing. However this low-level, stroke based approach to rendering also restricts current AR to the synthesis of traditional artistic stroke based styles (such as hatching [135], stippling [43] or painterly impressionism [103]). We argue that a higher level of abstraction is necessary to extend the gamut of automated AR to encompass compositional forms of art, including abstract artistic styles such as Cubism. The successful production of such compositions is again predicated upon the development of a global image salience measure, which may be applied to identify high level salient features (for example, eyes or ears in a portrait). Such features may then be used as a novel alternative to the stroke as the atomic element in artistic renderings. In this case, the model we choose must be sufficiently general to envelope a large range of input imagery, but sufficiently high level to allow the extraction of these conceptually higher level salient features.

In the next Section, we propose a rarity based measure of salience which performs a global statistical analysis of the image to determine the relative importance of pixels. We apply this measure to develop two novel algorithms, each respectively addressing one of the two deficiencies in AR identified in the preceding paragraphs:

1. Limited ability to control emphasis, through level of detail, in renderings.

2. Limited diversity of style.

First, we propose a novel single-pass painting algorithm which paints to conserve salient detail and abstract away non-salient detail in the final rendering. Arguably this approach is more in line with traditional artistic practice, and we demonstrate the improved results (with respect to level of emphasis in the rendering) of our salience based rendering in comparison to existing AR. This algorithm serves as a pilot for salience driven painterly rendering, which we build upon to propose a salience-adaptive, relaxation based painting technique in Chapter 4, and extend to process video footage into painterly animations in Chapter 8. Second, we propose a novel rendering algorithm

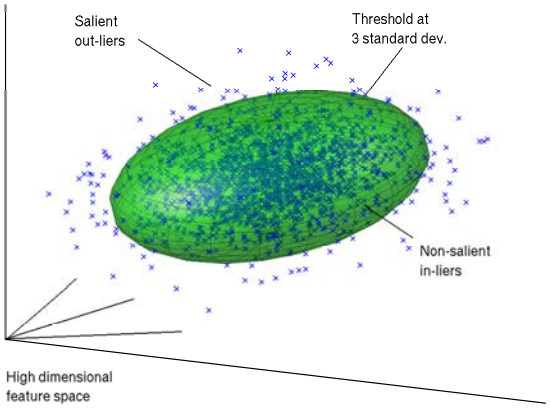


Figure 3-2 Our rarity based salience measure computes a series of derivatives for each image location, which form feature vectors in a high dimensional space. By fitting an eigenmodel to this distribution and isolating the outliers (the green hyper-ellipse boundary indicates a threshold), we may identify salient image artifacts.

capable of producing compositions in a Cubist-style, using salient features (such as eyes, ears etc.) identified across an image set. Control of the AR process is specified at the compositional, rather than the stroke based level; a further novel contribution to AR. We incorporate both these algorithms into a single system, where we apply our salience based painting algorithm to the output of the Cubist rendering algorithm to give our compositions a painterly appearance. Furthermore, we show how preferential rendering with respect to salience can emphasise detail in important areas of the composition (for example, to bring out the eyes in a portrait using tonal variation). This salience adaptation is a novel contribution to automatic image-space AR that could not be achieved without a spatially higher level, global analysis of the source image.

3.2 A Global Measure of Image Salience

We locate salient features within a single image by modifying a technique due to Walker *et al* [163], who observe that salient pixels are uncommon in an image. The basic technique is to model the statistical distribution of a set of measures associated with each pixel, and to isolate the outliers of this distribution. The pixels corresponding to these outliers are regarded as salient (Figure 3-2).

To compute these measures, $\underline{x}$, over each pixel we convolve each RGB channel of the image with a set of origin-centred 2D Gaussian derivative filters. Specifically we use 5 first and second order directional derivatives: $\partial G(x,y;\sigma)/\partial x$, $\partial G(x,y;\sigma)/\partial y$, $\partial^2 G(x,y;\sigma)/\partial x^2$, $\partial^2 G(x,y;\sigma)/\partial y^2$, and $\partial^2 G(x,y;\sigma)/\partial x \partial y$. These filters smooth the image before computing the derivative; they respond well to edge and other signals of

characteristic scale σ, but as Figure 3-3 shows, our method is more general than edge detection. We filter using octave intervals of σ, as such intervals contain approximately equal spectral power. In our implementation we use σ values of 1, 2, 4 and 8; thus with each pixel we associate a vector $\underline{x}$ of $20 = 5 \times 4 \times 3$ components.

For an image of M pixels we will have M vectors $\underline{x} \in \Re^n$, where for us $n = 60$. We assume these points are Gaussian distributed, which we represent using an eigenmodel; a simple and convenient model that works acceptably well in practice. The eigenmodel provides a sample mean $\underline{\mu}$; a set of eigenvectors each a column in orthonormal matrix $\underline{\underline{U}}$; each eigenvector has a corresponding eigenvalue along the diagonal of $\underline{\underline{\Lambda}}$. An eigenmodel allows us to compute the squared Mahalanobis distance of any point $\underline{x} \in \Re^n$:

$$d^2(\underline{x}) = (\underline{x} - \underline{\mu})^T \underline{\underline{U\Lambda U}}^T (\underline{x} - \underline{\mu}) \tag{3.1}$$

The Mahalanobis distance measures the distance between a point and the sample mean, and does so using the standard deviation (in the direction $\underline{x} - \underline{\mu}$). This provides a convenient way of deciding which sample points are salient; we use a threshold, $d^2(\underline{x}) > 9$, since 97% of normally distributed points are known to fall within 3 standard deviations of the mean. This threshold has also been shown empirically to produce reasonable results (Figure 3-3). Notice that because we look for statistical outliers we can record pixels in flat regions as being salient, if such regions are rare; a more general method than using high frequency magnitude.

Figure 3-3 demonstrates some of the results obtained when applying our salience measure to examples of real and synthetic data, and compares these results to "ground truth" importance (salience) maps which are representative of those generated by independent human observers. The global measure can be seen to out-perform local high-frequency detectors in the task of "picking out" salient artifacts, even against a textured background. We use the Sobel edge detector for comparison, as it is used by the majority of image-space AR techniques. Our measure is also shown to be sensitive to chromatic variations, whereas the Sobel edge detector used in existing AR techniques is concerned only with luminance. We observe that our global salience measure produces salience maps qualitatively closer to the ground truth that the local (Sobel) edge measure.

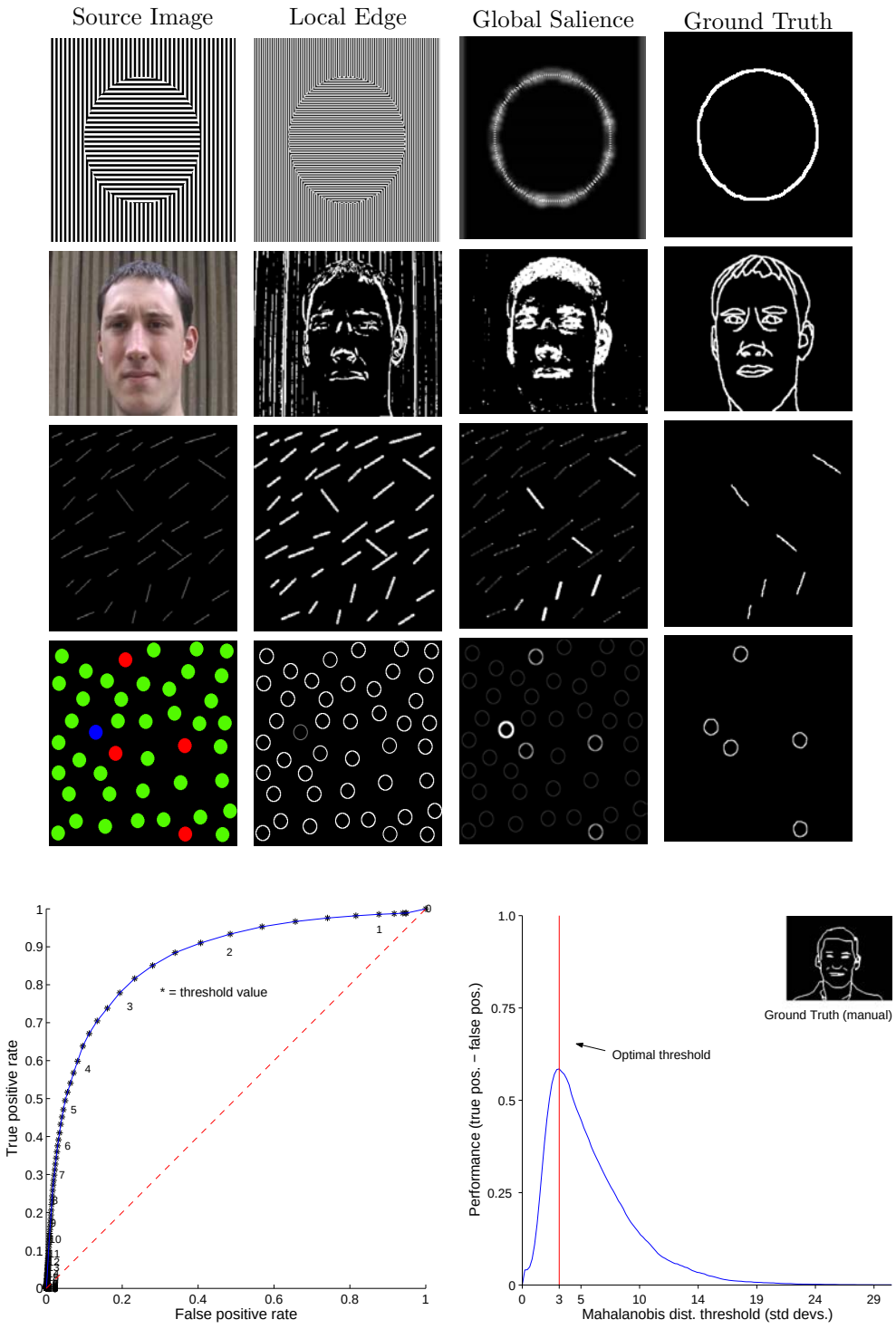


Figure 3-3 Top: Examples of real and synthetic images, processed by a local edge filter (Sobel) and our global, rarity based salience map. We observe that our approach produces maps qualitatively closer to a manually specified ground truth for image salience; we can "pick out" the circle and face where edge detection fails. Chromatic variations, scale and orientation are encapsulated in the rarity measure. Bottom: ROC curve representing sensitivity (true positive rate) vs. specificity (one minus false positive rate), as the Mahalanobis distance threshold is varied. The source image for these results is given in Figure 3-11 (middle-left). The pure chance response is plotted in dotted red. Right: Performance of the measure with various thresholds, derived from the ROC. The manually specified ground truth segmentation for this comparison is inset.

3.3 Painterly Rendering using Image Salience

We now describe a novel single-pass painterly rendering algorithm which applies our global salience measure to generate pointillist-style painterly renderings from photographs. By automatically controlling the level of emphasis in the painting (adapting the level of detail according to the salience of the region being painted), we address the first issue — aesthetic quality of rendering — raised in Section 3.1.

Our algorithm accepts a 2D image as input, and outputs a single 2D painting generated from that image. Paintings are formed by sampling a reference image at regular spatial intervals to generate a series of three-dimensional brush strokes; inverted cones with superquadric cross-section. The superquadric class of functions can be represented by the equation:

$$\left(\frac{x}{a}\right)^{\frac{2}{\alpha}} + \left(\frac{y}{b}\right)^{\frac{2}{\alpha}} = r^{\frac{2}{\alpha}} \tag{3.2}$$

where a and b are normalised constants ($a + b = 1$; $a, b > 0$) which influence the horizontal and vertical extent of the superquadric respectively, and r is an overall scaling factor. We observe that equation 3.2 reduces to the general equation for a closed elliptic curve when $\alpha = 1$, tends toward a rectangular form as $\alpha = 0$, and toward a four-pointed star as $\alpha \to \infty$. Thus the superquadrics can express a wide variety of geometric forms, using a single parameter.

Each generated conic stroke is z-buffered and the result is projected orthogonally onto the (2D) image plane to generate the final painting (Figure 3-4). There are seven parameters to each stroke; a, b, r, α (from equation 3.2), RGB colour $\underline{j}(\underline{c})$, orientation angle θ and height h. Parameter α determines the form of the stroke, and is preset by the user. Low values (< 1) of α create cross-sections of a rectangular form, giving the image a chiselled effect, whilst higher values of α produce jagged brush styles. Strokes are shaded according to the colour $\underline{c}$ of the original image at the point of sampling. Function $\underline{j}(\underline{c})$ transforms, or "jitters", the hue component of stroke colour $\underline{c}$ by some small uniformly distributed random quantity, limited by a user defined amplitude ϵ. By increasing ϵ, impressionist results similar to those of Haeberli's interactive systems [62] can be automatically produced. Further brush styles can also be generated by texturing the base of each cone with an intensity displacement map, cut at a random position from a sheet of texture; we find that this process greatly enhances the natural, "hand-painted" look of the resulting image. The remaining five stroke parameters (a, b, r, θ, and h) are calculated by an automated process which we now describe.

Stroke height h, is set proportional to image salience at the point of sampling. Higher

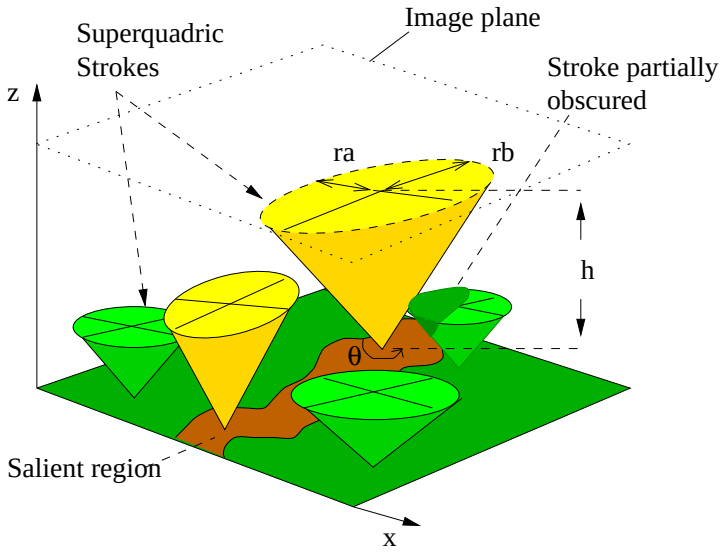


Figure 3-4 Strokes take the form of inverted cones with superquadric cross-section, and are z-buffered to produce the final painting.

salience image pixels tend to correspond to the features and detail within the image, and so produce strokes of greater height to protrude over the lower salience strokes in the z-buffer. The scale of the base of the cone, r, is set inversely proportional to salience magnitude. This causes small, definite strokes to be painted in the vicinity of artifacts corresponding to salient detail in the image. Larger strokes are used to shade non-salient areas, mimicking the behaviour of the artist. Hence our method tends to draw low, fat cones in regions of low salience, and tall, narrow cones in regions of high salience.

We also derive gradient information from the reference image, by convolving the intensity image with a Gaussian derivative of first order. Stroke orientation θ is derived from gradient orientation; the larger axis of the superquadric is aligned tangential to the edge direction. In areas where gradient magnitude is low, orientation derived in this manner becomes less reliable. We therefore vary the eccentricity of the superquadric (a, b) in relation to the magnitude of the image gradient at the position sampled. If the gradient is low, then $a \approx b$, and orientation becomes less important as the superquadric is not greatly expressed in either horizontal or vertical directions. Where image gradient is high, then $a > b$ and the superquadric stretches out. One emergent property of this approach is that strokes typically stretch along salient edges tending to merge, often causing edge highlights to appear as though produced by fewer, longer strokes. This is typical of the manner in which an artist might manually render such highlights, and adds aesthetic quality to the image.

Although we have used a global salience measure to drive emphasis in the rendering

Figure 3-5 Comparison of our salience based method. Left: Results of an impressionist algorithm (due to Litwinowicz [103]). Middle: Our algorithm, but driven using Sobel response rather than global salience. Right: Our proposed salience-adaptive algorithm; non-salient background detail is abstracted away, whilst salient detail is emphasised on the face. Observe that the Sobel driven algorithms (left, middle) emphasise all high frequency detail to a similar degree, detracting from the emphasis given to the facial features. See Figure 3-3 for corresponding Sobel and salience maps.

Figure 3-6 Left: Three images of identical subject; the original image (top), painterly rendering with salience (middle), and painterly rendering without salience (bottom). The right hand column holds the salience map of the original image (top), and the edge maps of the two paintings (middle, right). Right: Strokes applied with (top) and without (bottom) salience adaptation. We make the qualitative observation that salient detail is conserved using our painterly technique.

Figure 3-7 Illustrating the application of our rendering algorithm. A section of the phone-box rendering has been magnified, demonstrating the alignment of strokes tangential to the salient window frames. Portions of the pickup truck painting with (inset, left) and without (inset, right) salience are shown. The source image and salience map are also inset below.

Figure 3-8 A "salient sketch" (right) produced by adapting our painterly technique to draw along the principal axis of each superquadric stroke — demonstrating both the close relationship between an artist's sketch and a salience map, and the potential of salience measures in driving alternative artistic rendering styles. In descending order, the original image, our automatically derived rarity based salience map, and a ground salience map, are shown on the left.

process, certain local measures have also been used to set attributes such as stroke orientation. These are inherently local properties, and we are justified in setting them as such; by contrast the concept of importance demands global analysis for computation.

3.3.1 Results and Qualitative Comparison

We present the results of applying our painting algorithm to a variety of images in Figures 3-5, 3-6 and 3-7, and demonstrate two advantages of our salience adaptive approach to painting.

Figure 3-5 contains three paintings of identical subject, painted using automatic, single-pass painterly algorithms. The right-hand painting was created using our global salience adaptive painting scheme, and demonstrates how non-salient detail (in this case, repetitive background texture) is abstracted away with coarse strokes. Salient detail has been emphasised with fine strokes, and the contrast produced against the coarser background serves to further emphasise this detail. The middle painting was generated using our algorithm, but for the purposes of illustration we have replaced our global salience measure with Sobel intensity gradient magnitude; the measure used by virtually all automatic image-space AR algorithms. Observe that the non-salient back-

ground, and salient foreground are emphasised equally, since they exhibit similar high frequency characteristics. The left-hand painting was generated using a recent daub based, painterly algorithm from the literature [103]. This algorithm is also driven by local, high frequency based heuristics and so also erroneously emphasises the non-salient background texture. A sketchy rendering of the portrait has been generated in Figure 3-8 by plotting the principal axis of each superquadric. This serves to illustrate both the alignment and placement of individual strokes, and the possibility of alternative salience driven rendering styles.

Our algorithm causes the least salient strokes to be laid down first, much as an artist might use a wash to generate wide expanses of colour in an image, and fill in the details later. Without this sensitivity to salience, the rendering procedure can obscure regions of high salience with strokes of lower salience, demonstrated by Figure 3-6. By setting the conic height h proportional to salience, salient detail is conserved within the painting — this is especially clear around the eyes and nose in Figure 3-6, left-middle. Ignoring the implicit ordering of strokes can still produce a painterly effect, but without the adaptive sensitivity to salient detail that our method provides (Figure 3-6, left-bottom). By inspection we make the qualitative observation that the majority of salient pixels in the original, and edge pixels (corresponding to detail) in the salience-painted images correspond; this is not true for the non-salience adaptive paintings.

A salience adaptive approach to painting therefore benefits aesthetic quality in two respects. Not only are salient regions painted with improved clarity (strokes from non-salient regions do not encroach upon and distort regions of greater salience — Figure 3-5), but renderings also exhibit a sense of focus around salient regions due to the abstraction of non-salient detail (Figure 3-6).

Figure 3-7 contains a gallery of paintings generated by our algorithm. The image of the pickup truck was rendered with superquadric shape parameter $\alpha = 1$. Portions of the painting rendered with and without salience adaptation are shown inset, as well as with the source image. The phone-box has been rendered with $\alpha = 0.5$; in particular we draw attention to the detail retained in the window frames (inset). Strokes have been aligned tangential to the edges of each frame, merging to create sweeping brush strokes. The strokes rendering the window glass do not encroach upon the window frames, which are more salient, and for the most-part, salient detail is conserved within the painting. A clear exception where salient detail has been lost, is within the plaque containing the words “Telephone”. Our conditioned ability to immediately recognise and read such text causes us to attribute greater salience to this region. The degradation in aesthetic quality is therefore not due to the argument for salience adaptive painting,

but rather due to the salience map of our measure diverging from the ground truth (the expectation of the viewer). This highlights the simplistic nature of our rarity driven salience measure, and suggests that one's experiences cause certain classes of artifact to be regarded as more salient than others. We return to this point later in Chapter 4, when we extend our single-pass salience adaptive painting algorithm in a number of ways — one of which is to make use of a trainable salience measure, capable of learning the classes of artifact the user typically deems to be salient.

3.4 Cubist-style Rendering from Photographs

We now describe a novel AR algorithm which addresses the second deficiency in AR identified in Section 3.1; the limited diversity of styles available by approaching AR through the low-level paradigm of stroke-based rendering. Our aim was to investigate whether aesthetically pleasing art, reminiscent of the Cubist style, could be artificially synthesised. We are influenced by artists such as Picasso and Braque, who produced art work by composing elements of a scene taken from multiple points of view. Paradoxically the Cubist style conveys a sense of motion in a scene without assuming temporal dependence between views. The problem of synthesising renderings in abstract styles such as Cubism is especially interesting to our work, since it requires a higher level of spatial analysis than currently exists in AR, in order to identify the salient features used to form stylised compositions. By *salient feature* we refer to an image region containing an object of interest, such as an eye or nose; a composition made from elements of low salience would tend to be uninteresting. We considered the following specific questions:

- How is salience to be defined so that it operates over a wide class of input images?
- How should salient features be selected from amongst many images, and how should the selected features be composed into a single image?
- How should the angular geometry common in Cubist art be reproduced?
- How should the final composition be rendered to produce a painted appearance?

Resolution of the first two questions provides the basic mechanism by which a Cubist-like image can be formed; resolution of latter two questions enhances aesthetic quality.

Our algorithm accepts one or more 2D images taken from different viewpoints as input (see Figure 3-17a), and produces a single 2D image rendered in the Cubist style. Salient artifacts within each image are first identified using our global salience measure (Section 3.2). This can produce disconnected features, which requires correction; in our case by minimal user interaction. These features are geometrically distorted. A subset

is then selected and composited, ensuring that non-selected features do not inadvertently appear in the final composition – naïve composition allows this to happen. The composition process is stochastic, and a new image may be produced on each new run of the method. An element of control may also be exerted over the composition process, affecting the balance and distribution of salient features in the final painting. The ability to influence rendering at a compositional level, rather than setting parameters of individual strokes, is a novel contribution of our method. In rendering a painting from the final composition we make use of our previously described salience based painting algorithm (Section 3.3) which treats brush strokes in a novel way, ensuring that salient features are not obscured.

We begin by registering all source images so that objects of interest, such as faces, fall upon one another; this assists the composition process in subsection 3.4.3. We threshold upon colour to partition foreground from background, and translate images so that first moments of foreground are coincident. Finally we clip the images to a uniform width and height. This step creates spatial correspondence between source images on a one-to-one basis: pixels at the same location $(x, y)^T$ in any image correspond. The remaining algorithm stages are of greater interest, and we describe each of them in turn in the following subsections.

3.4.1 Identification of Salient Features

We wish to find a set of salient features amongst the registered images. These images should be unrestricted in terms of their subject (for example, a face or guitar). In addition, we want our salient features to be relatively "high level", that is they correspond to recognisable objects, such as noses or eyes. This implies we need a definition of salience that is both general and powerful; such a definition does not currently exist in the computer vision literature, or elsewhere. However, we can make progress by choosing a definition of salience that is sufficiently general for our needs, and allow user interaction to provide power where it is needed.

We begin by applying our global salience measure to the set of source images (Section 3.2). In practice salient pixels within these images form spatially coherent clusters, which tend to be associated with interesting objects in the image, including conceptually high level features such as eyes (Figure 3-3). However, our method is general purpose, and therefore has no specific model of eyes, or indeed of any other high level feature. It is therefore not surprising that what a human regards as a salient feature may be represented by a set of disconnected salient clusters.

Given that the general segmentation problem, including perceptual grouping, remains

unsolved we have two choices: either to specialise the detection of salient regions to specific classes of source images, such as faces (see the fully automatic case study described later in Section 3.5); or to allow the user to group the clusters into features. We adopt the latter approach for its power and simplicity: powerful because we rely on human vision, and simple not only to implement but also to use. We allow the user to draw loose bounding contours on the image to interactively group clusters. This mode of interaction is much simpler for the user than having to identify salient features from images *ab initio*; that is with no computer assistance. Feature grouping is also likely to be consistent between source images, because our salience measure provides an objective foundation to the grouping. The contour specified by the user forms the initial location for an active contour (or "snake"), which is then iteratively relaxed to fit around the group of salient clusters. Active contours are parametric splines characterised by an energy function E_{snake}; the sum of internal and external forces [91]. Internal forces are determined by the shape of the contour at a particular instant, and external forces are determined by the image upon which the contour lies. Here the spline is defined by a parametric function $\underline{v}(s)$:

$$E_{snake} = \int_0^1 E_{snake}(\underline{v}(s))ds \tag{3.3}$$

$$E_{snake} = \int_0^1 E_{internal}(\underline{v}(s)) + E_{external}(\underline{v}(s))ds \tag{3.4}$$

During relaxation we seek to minimise the contour's energy by iteratively adjusting the position of the spline control points, and thus tend toward an optimal contour fit to the salient feature. In our case energy is defined as:

$$E_{internal} = \alpha\left|\frac{d\underline{v}}{ds}\right|^2 + \beta\left|\frac{d^2\underline{v}}{ds^2}\right|^2 \tag{3.5}$$

$$E_{external} = \gamma f(\underline{v}(s)) \tag{3.6}$$

where the two terms of the internal energy constrain the spacing of control points and curvature of the spline respectively. The external energy function is simply the sum of salience map pixels bounded by the spline $\underline{v}(s)$, normalised by the number of those pixels. Constants α, β and γ weight the importance of the internal and external constraints and have been determined empirically to be 0.5, 0.25 and 1. In our application we fit an interpolating (Catmull-Rom) spline [51] through control points to form the parametric contour $\underline{v}(s)$. We assume that the user has drawn a contour of approximate correct shape around the feature clusters; the weighting constants have been chosen

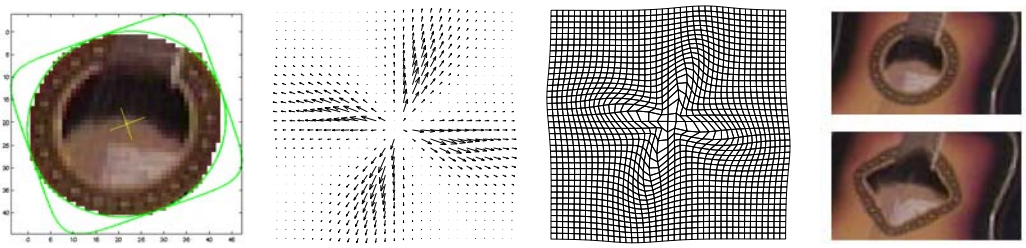


Figure 3-9 Four stages of a geometric warp where $\alpha' = 0.3$. From left to right: (a) the source and target superquadrics, fitted about a salient feature; (b) the continuous forward vector field; (c) the mesh of quadrilaterals (mapped pixel areas); (d) the final distorted image.

to promote retention of initial shape in the final contour. Relaxation of the snake proceeds via an algorithm adapted from Williams [167], in which we blur the salience map heavily in early iterations and proportionately less on subsequent iterations. This helps prevent the snake snagging on local minima early in the relaxation process. Our initial approach [23] made use of a convex hull based grouping technique to form salient clusters, however this precluded the possibly of accurately extracting concave features. Further advantages of the snake segmentation method are a tighter, more accurate fit to features, and greater robustness to noise. There is also a lesser degree of sensitivity upon the initial positioning of the contour, since the snake shrinks to fit the exterior boundary of the salient pixel cluster.

The union of the salient features identified in each source image forms the set of salient features we require, which we call $\mathcal{F}$. In addition to grouping clusters into features, the user may also label the features. These labels partition the set of all salient features $\mathcal{F}$ into equivalence classes, such as "eyes", providing a useful degree of high level information (these classes represent a simple model of the object in the picture). We make use of $\mathcal{F}$, and associated equivalence classes, throughout the remaining three stages of our algorithm.

3.4.2 Geometric Distortion

We now wish to distort the identified features, in $\mathcal{F}$, to produce the more angular forms common in Cubist art. Our approach is to construct a continuous vector field $\mathcal{V}$ over each source image, which is a sum of the contributions made by distorting the set of all features $f \in \mathcal{F}$ belonging to that image. That is, we define a vector-valued distortion function $\underline{g} : \Re^2 \mapsto \Re^2$, so that for every point $\underline{u} \in \Re^2$, we have $\underline{g}(\underline{u}) = \underline{u} + \mathcal{V}(\underline{u})$ where

$$\mathcal{V}(\underline{u}) = \sum_{\phi \in f} \underline{d}_\phi(\underline{u}) \tag{3.7}$$

To define a particular distortion function $\underline{d}_\phi(.)$ we fit a superquadric about the perime-

ter of feature ϕ, then transform that fitted superquadric to another of differing order; thus specifying a distortion vector field $\underline{d}_\phi(\Re^2)$. We now describe the details of this process.

Recall equation 3.2 in which the superquadric class of functions may be represented in Cartesian form by:

$$\left(\frac{x}{a}\right)^{\frac{2}{\alpha}} + \left(\frac{y}{b}\right)^{\frac{2}{\alpha}} = r^{\frac{2}{\alpha}} \tag{3.8}$$

We use a parametric form of equation 3.2 determined by an angle θ about the origin, by which we correlate points on the perimeter of one superquadric with those on another.

$$x = \frac{r\cos(\theta)}{\left(|\cos(\theta)/a|^{\frac{2}{\alpha}} + |\sin(\theta)/b|^{\frac{2}{\alpha}}\right)^{\frac{\alpha}{2}}} \tag{3.9}$$

$$y = \frac{r\sin(\theta)}{\left(|\cos(\theta)/a|^{\frac{2}{\alpha}} + |\sin(\theta)/b|^{\frac{2}{\alpha}}\right)^{\frac{\alpha}{2}}} \tag{3.10}$$

We calculate the distortion for a given feature by fitting a general superquadric of order α, and warping it to a target superquadric of new order α'. Features whose forms differ from this target superquadric are therefore distorted to a greater degree than features that already approximate its shape; thus each feature boundary converges toward the geometric form specified by α'. Typically we choose $\alpha' < 1$ to accentuate curves into sharp angles. The initial superquadric is fitted about the bounding pixels of the feature using a 6-dimensional Hough transform based search technique described in Appendix A.1. We find this global fitting method suitable for our purpose due to its relatively high tolerance to noise.

Recall the distortion function $\underline{d}_\phi(.)$; we wish to produce a displacement vector $\underline{v}$ for a

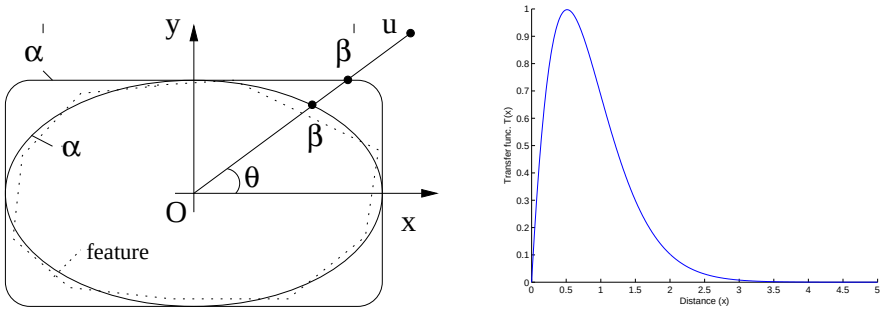


Figure 3-10 Left: The fitted and target superquadrics, described by α and α' respectively. Intersection with line $\vec{O\underline{u}}$ is calculated using angle θ. Right: The decay function (equation 3.12) used to dampen the vector field magnitude.

given point $\underline{u} = (u_x, u_y)$. We first calculate the points of intersection of line $\underline{\vec{Ou}}$ and the two superquadric curves specified by α and α', where $\underline{O}$ is the origin of both superquadrics (these origins are coincident). We derive the intersections by substituting a value for $\theta = arctan(u_y/u_x)$ into equations 3.9) and (3.10. We denote these intersection points by $\underline{\beta}$ and $\underline{\beta'}$ respectively (see Figure 3-10, left). The vector $\underline{\beta'} - \underline{\beta}$ describes the maximum distortion in direction θ. We scale this vector by passing the distance (in superquadric radii) of point $\underline{u}$ from the origin, through a non-linear transfer function $\mathcal{T}(.)$. So, for a single feature ϕ:

$$\underline{d}_\phi(\underline{u}) = \mathcal{T}\left(\frac{|\underline{u} - \underline{O}|}{|\underline{\beta} - \underline{O}|}\right)(\underline{\beta'} - \underline{\beta}) \tag{3.11}$$

The ideal characteristics of $\mathcal{T}(x)$ are a rapid approach to unity as $x \to 1$, and a slow convergence to zero as $x \to \infty$. The rise from zero at the origin to unity at the superquadric boundary maintains internal continuity, ensuring a topologically smooth mapping within the superquadric (Figure 3-10, right). Convergence to zero beyond unit radius mitigates against noticeable distortion to surrounding areas of the image that do not constitute part of the feature. The Poisson distribution function (equation 3.12) is a suitable $\mathcal{T}$, where $\Gamma(.)$ is the gamma function [123] and λ is a scaling constant.

$$\mathcal{T}(x) = \frac{\lambda^{x\mathrm{e}^\lambda}}{\Gamma(x)} \tag{3.12}$$

Recall from equation 3.7 that we sum the individual vector fields of each feature belonging to a specific source image, to construct the overall vector field for that image. With this field defined, we sample those points corresponding to the corners of every pixel in the source image, and so generate their new locations in a target image. This results in a mesh of quadrilaterals, such as that in Figure 3-9*c*. Mapping each pixel area from the original bounded quadrilateral to the target bounded quadrilateral yields the distorted image.

The distortion process is repeated for each source image, to produce a set of distorted images. At this stage we also warp the bounding polygon vertices of each feature, so that we can identify the distorted salient features $\mathcal{F}'$. For reasons of artistic preference, we may wish to exercise control to use different values of α' for each equivalence class; for example, to make eyes appear more angular, but leave ears to be rather more rounded.

We draw attention to issues relating to the implementation of our method; specifically that the feature distortion stage can be relatively expensive to compute. This bottleneck can be reduced by: (a) precomputing the transfer function $\mathcal{T}(.)$ at suitably small

discrete intervals, and interpolating between these at run-time; (b) using a fast but less accurate method of integrating distorted pixel areas such as bilinear interpolation. In both cases we observed that the spatial quantisation induced later by the painterly rendering stage mitigates against any artifacts that may result.

3.4.3 Generation of Composition

We now describe the process by which the distorted salient features are selected from $\mathcal{F}'$ and composited into a target image. Specifically we wish to produce a composition in which:

- The distribution and balance of salient features composition may be influenced by the user.
- Features do not overlap each other.
- The space between selected salient features is filled with some suitable non-salient texture.
- Non-salient regions are "broken up" adding interest to the composition, but without imposing structure that might divert the viewer's gaze from salient regions.

A subset of the distorted salient features $\mathcal{F}'$ are first selected via a stochastic process. These chosen features are then composited, and an intermediary composition produced by colouring uncomposited pixels with some suitable non-salient texture. Large, non-salient regions are then fragmented to produce the final composition.

Selection and Composition

We first describe the process by which a subset of distorted salient features in $\mathcal{F}'$ are selected. We begin by associating a scalar $s(f)$ with every feature $f \in \mathcal{F}$:

$$s(f) = A(f) \cdot T(\mathcal{E}(f)) \tag{3.13}$$

in effect the area of the feature $A(f)$ weighted by a function $T(.)$ of the fractional size of the equivalence class to which it belongs (which we write as $\mathcal{E}(f)$). By varying the transfer function $T(.)$, the user may exercise control over the balance of the composition. We use:

$$T(x) = x^{\beta} \tag{3.14}$$

Figure 3-11 Left: Three source images used to create a Cubist portrait. Right: Features selected from the set $\mathcal{F}'$ via the stochastic process of Section 3.4.3 with balance parameter $\beta = 1$. Notice that the number of facial parts has a natural balance despite our method having no specific model of faces; yet we still allow two mouths to be included. Pixels not yet composited are later coloured with some suitable non-salient texture.

β is a continuous user parameter controlling visual "balance" in the composition. If set to unity, features are distributed evenly in similar proportion to the equivalence classes of the original image set. By contrast $\beta = 0$ introduces no such bias into the system, and a setting of $\beta = -1$ will cause highly unbalanced compositions, in which rarer classes of feature are more likely to be picked.

We treat each scalar $s(f)$ as an interval, and concatenate intervals to form a range. This range is then normalised to span the unit interval. We choose a random number from a uniform distribution over $[0, 1]$, which falls in a particular interval, and hence identifies the corresponding feature. Features of larger area with large associated scalar values ($s(f)$) tend to be selected in preference to others, which is a desirable bias in our stochastic process. The selected feature is removed from further consideration, and included in a set $\mathcal{C}$, which is initially empty.

This selected feature may intersect features in other images, by which we mean at a least one pixel (i, j) in the selected feature may also be in some other feature in some other image (recall the registration process aligns pixels to correspond on a one-to-one basis). Any features that intersect the selected feature are also removed from further consideration, but are not placed in the set $\mathcal{C}$.

We have found the choice of $\beta = 1$ to produce aesthetically pleasing renderings. In this case the process is biased toward producing a set $\mathcal{C}$ containing features whose equivalence classes are similar in proportion to the original source images. For example, if the original subject has two eyes and a nose, the algorithm will be biased toward

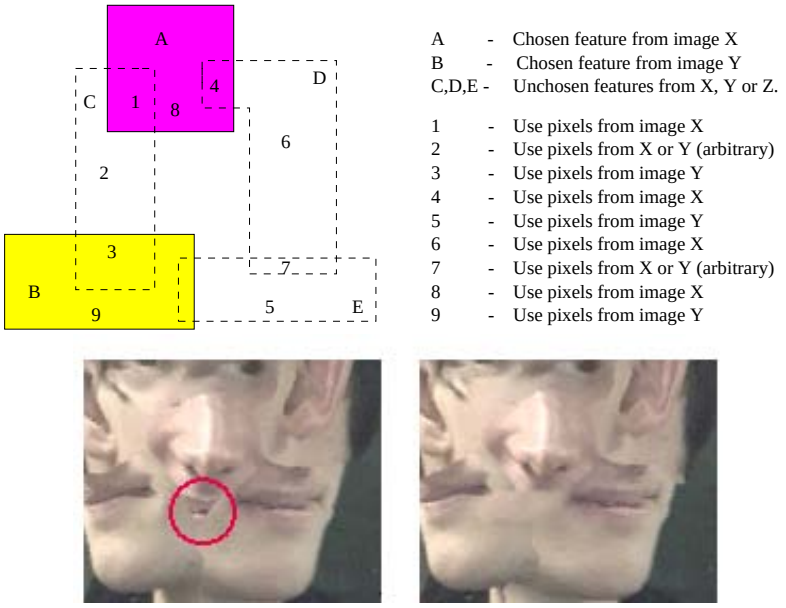


Figure 3-12 (a) potential intersections between features (top); (b) final compositions without (left) and with (right) the second stage of processing.

producing a composition also containing two eyes and a nose, but deviation is possible, see Figure 3-11.

The second step of our process is concerned with the composition of the chosen features in $\mathcal{C}$ to produce the final image. We begin this step by copying all chosen features into a target plane, producing a result such as Figure 3-11. In order to complete the composition we must determine which image pixels have not yet been composited, and colour them with some suitable non-salient texture.

An initial approach might be to compute a distance transform [145] for each non-composited pixel, which determines its distance to the nearest feature. The corresponding pixel in the distorted source image containing this nearest feature is used to colour the uncomposited pixel. This produces similar results to a Voronoi diagram, except that we seed each Voronoi segment with a region rather than a point. Unfortunately this initial approach is unsatisfactory: under some circumstances regions may be partially textured by unchosen salient features, and images such as Figure 3-12*b* (left) may result. To mitigate against partial mouths and similar unappealing artifacts requires greater sophistication, which we introduce by performing a second set of intersection tests.

We copy each of the unchosen features onto the image plane, and test for intersection with each of the chosen features $\mathcal{C}$. If an unchosen feature u intersects with a cho-

Figure 3-13 Illustrating composition from a collection of warped salient features in the guitar image set. Composition balance parameter $\beta = 1$.

sen feature c, we say that *'c holds influence over u'*. Unchosen features can not hold influence over other features. By examining all features, we build a matrix detailing which features hold influence over each other. If an unchosen feature u is influenced by exactly one chosen feature c, we extend feature c to cover that influenced area. We fill this area by copying pixels from corresponding positions in the distorted image from which c originates. Where an unchosen feature is influenced by several chosen features, we arbitrarily choose one of these chosen features to extend over the unchosen one (Figure 3-12a, region 2). However, we do not encroach upon other chosen regions to do this – and it may be necessary to subdivide unchosen feature areas (Figure 3-12a, regions 1, 3 and 4). Only one case remains: when two unchosen features intersect, which are influenced by features from two or more differing source images (Figure 3-12a, region 7). In this case we arbitrarily choose between those features, and copy pixels from the corresponding distorted source image in the manner discussed.

We now perform the previously described distance transform procedure on those pixels not yet assigned, to produce our abstract composition.

Fragmentation of Non-salient Areas

The composition produced at this stage (Figure 3-14a, left) is often composed of pieces larger than those typically found in the Cubist paintings. We wish to further segment non-salient regions to visually "break up" uninteresting parts of the image, whilst avoiding the imposition of a structure upon those areas.

We initially form a binary mask of each non-salient segment using information from the previous distance transform stage of Section 3.4.3, and calculate the area of each segment. We then average the area of the chosen salient features $\mathcal{C}$, to produce a desired "segment size" for the composition. Each non-salient segment is fragmented into n pieces, where n is the integer rounded ratio of that segment's area to the desired segment size of the composition. To perform the segmentation we produce a dense point cloud of random samples within the binary mask of each non-salient segment. Expectation maximisation [41] is used to fit n Gaussians to this point cloud. We then

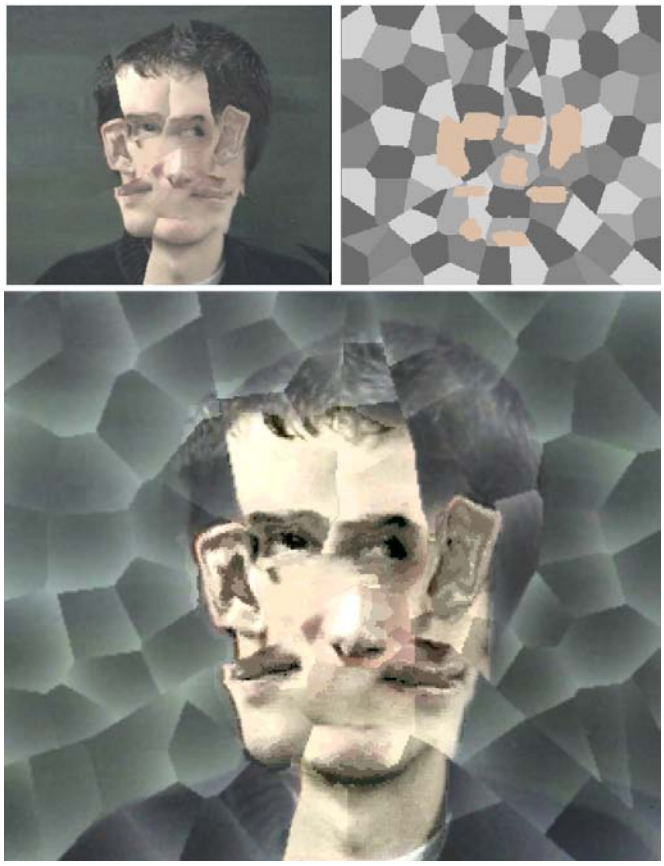


Figure 3-14 (a) Composition after application of steps in Section 3.4.3 exhibiting large non-salient segments (left) and a uniquely coloured finer segmentation (right) (b) Results of finer segmentation and shading of non-salient areas in the composition.

calculate the Gaussian centre to which each pixel within a given mask is closest; a Voronoi diagram is thereby constructed, the boundaries of which subdivide the non-salient segment being processed into multiple non-salient *fragments*.

Each of the non-salient fragments must now be shaded to break up the composition. We choose a point, or "epicentre" along each fragment's boundary, and decrease the luminosity of pixels within that fragment proportional to their distance from the epicentre (see Figure 3-15). The result is a modified intensity gradient across each fragment, simulating light cast over a fragment's surface. In practice it is desirable that no two adjacent fragments have an intensity gradient of similar direction imposed upon them; doing so induces a noticeable regular structure in non-salient areas, which can divert the viewer's attention from the more interesting *salient* features elsewhere in the composition. Placement of the epicentre at a random location upon the boundary produces too broad a range of possible gradient directions, causing shading to appear as noise. We therefore restrict shading to a minimal set of directions, calculated in the following manner.

A region adjacency graph is constructed over the entire composition; each non-salient fragment corresponds to a node in the graph with vertices connecting segments adjacent in the composition. We then assign a code or "colour" to each node in the graph, such that two directly connected nodes do not share the same colour. Graph colouring is well-studied problem in computer science, and an minimal colour solution is known to be NP-hard to compute. We therefore use a heuristic based approximation which is guaranteed to return a colouring in P-time, but which may not be minimal in the number of colours used (see Appendix A.2 for details). The result is that each fragment is assigned an integer coding in the interval $[1, t]$, where t is the total number of colours used by our approximating algorithm to encode the graph.

The result of one such colouring is visualised in Figure 3-14*a*. The epicentre of each fragment is placed at the intersection of the fragment's boundary and a ray projected from the centroid of the fragment at angle θ from vertical (Figure 3-15), where θ is determined by:

$$\theta = 2\pi \left(\frac{segment\ code}{t} \right) \tag{3.15}$$

This expression guarantees placement of the epicentre at one of t finite radial positions about the boundary of the segment, as the segment coding is an integer value.

The introduction of additional segmentation, and therefore edge artifacts, into non-salient areas of the composition can have the undesired effect of diverting a viewer's gaze from salient features present in the picture. We mitigate against this effect in two ways. First, we convolve the non-salient regions of the image with a low-pass filter kernel such as a Gaussian. This has the effect of smoothing sharp edges between fragments, but conserving the more gradual intensity differential over each non-salient fragment's

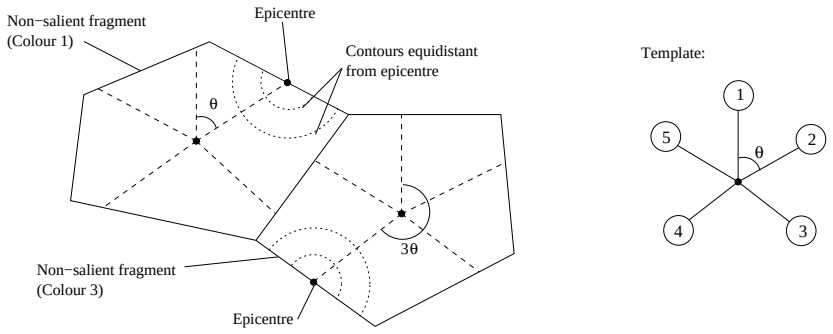


Figure 3-15 The geometry of two adjacent non-salient fragments, and a single template determining the location of the epicentre within a fragment. The epicentre of a fragment of colour i lies at the intersection of that fragment's boundary and the ith template spoke.

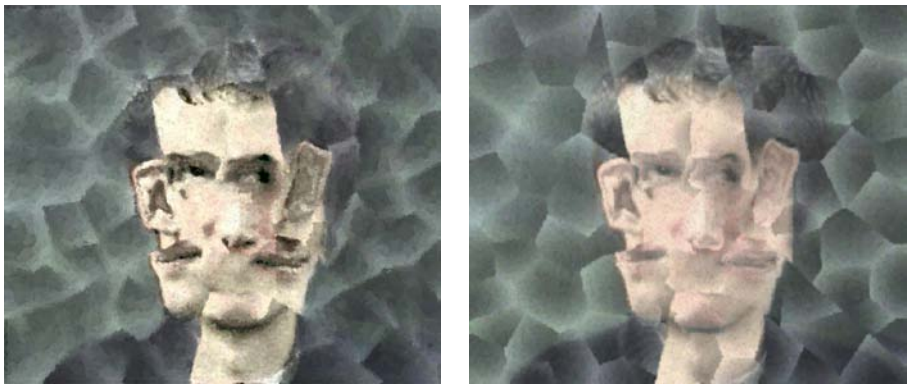


Figure 3-16 Comparison of two painted compositions with (left) and without (right) preferential shading treatment to salient regions. Observe how salient features such as the eyes are brought out within the salience adaptive composition; tonal variation has been used to emphasise the salient regions via histogram equalisation.

surface. This also proves advantageous in that the jagged edges of lines partitioning fragments are smoothed. Second, we use a variation upon histogram equalisation [145] to boost contrast within the foreground of the composition (determined during image registration), causing features such as eyes or noses to "stand out" from the softened segmentation boundaries. Specifically, we calculate the transfer function between the luminosities of the source and equalised compositions. For each pixel in the composition we then interpolate between these luminosities proportional to that pixel's salience; thus contrast is boosted in more salient areas of the composition, greatly improving the aesthetics of the painting (Figure 3-16).

This produces a final composition such as that of Figure 3-14*b*. We draw attention to the fact that major segmentation lines (produced by the steps of Section 3.4.3) and salient features remain unaffected by this final segmentation of the composition.

3.4.4 Applying a Painterly Finish

The final stage of our algorithm is concerned with creating a painterly effect on the generated composition, to which there are two sub-stages: colour quantising, and brush stroke generation.

The colour quantising step should be performed prior to composition, but is described here for the sake of clarity. We use variance minimisation quantisation [174], to reduce the colour depth of three independent areas within the image: the distorted salient features ($\mathcal{F}'$); the foreground of each distorted image; and the background of each distorted image. Distinction between foreground and background is made by thresholding upon a simple characteristic property of the image, such as hue or intensity (as was performed during image registration). Our motivation to quantise follows the observation

that an artist typically paints with a restricted palette, and often approximates colours as a feature of the Cubist style [81]. We allow a level of control over this effect by differentiating the level of quantisation over the various image components, and have found that heavy quantisation of the features and foreground, contrasted by a lesser degree of background quantisation can produce aesthetically pleasing effects.

At this stage we optionally introduce false colour to the image. Artists such as Braque and Gris often painted in this manner, contrasting shades of brown or grey with yellows or blues to pick out image highlights. We use a look-up table based upon a transfer function, which generates a hue and saturation for a given intensity, calculated from the original input colour. Typically we define this function by specifying several hue and saturation values at various intensities, and interpolate between these values to produce a spectrum of false colour to populate the look-up table.

The second step of the rendering process concerns the generation of "painted" brush strokes, using our previously described salience adaptive painterly technique (Section 3.3). This ensures that strokes from non-salient regions to do encroach upon salient features during this final rendering step.

3.4.5 Results of Cubist Rendering

We present the results of applying our Cubist algorithm to three image sets; a portrait, a guitar, and a nude. These subjects were popular choices for artists of the Cubist period, and we use them to demonstrate the processes of composition, distortion, and painting respectively.

The original source image sets were captured using a digital video camera, and are given in Figure 3-17a. Figure 3-17b presents the results of processing the portrait images; salient features were the ears, eyes, nose and mouth. Figures 3-17b_1 and 3-17b_2 were created by successive runs of the algorithm, using identical distortion parameters; the stochastic nature of feature selection produces varying compositions in the same visual style. Figure 3-17b_3 demonstrates the consequence of relaxing the constraints which maintain proportion between equivalence classes during composition; equivalence classes are no longer proportionally represented; in this case parameter $\beta = 0$.

The nude has been rendered with minimal distortion; salient features were the eyes, nose, mouth, chest and arm. False colour has been introduced to Figure 3-17c_2, using the complementary colours of blue and orange to contrast highlight and shadow. Many abstract artists make use of complementary colour pairs in a similar manner.

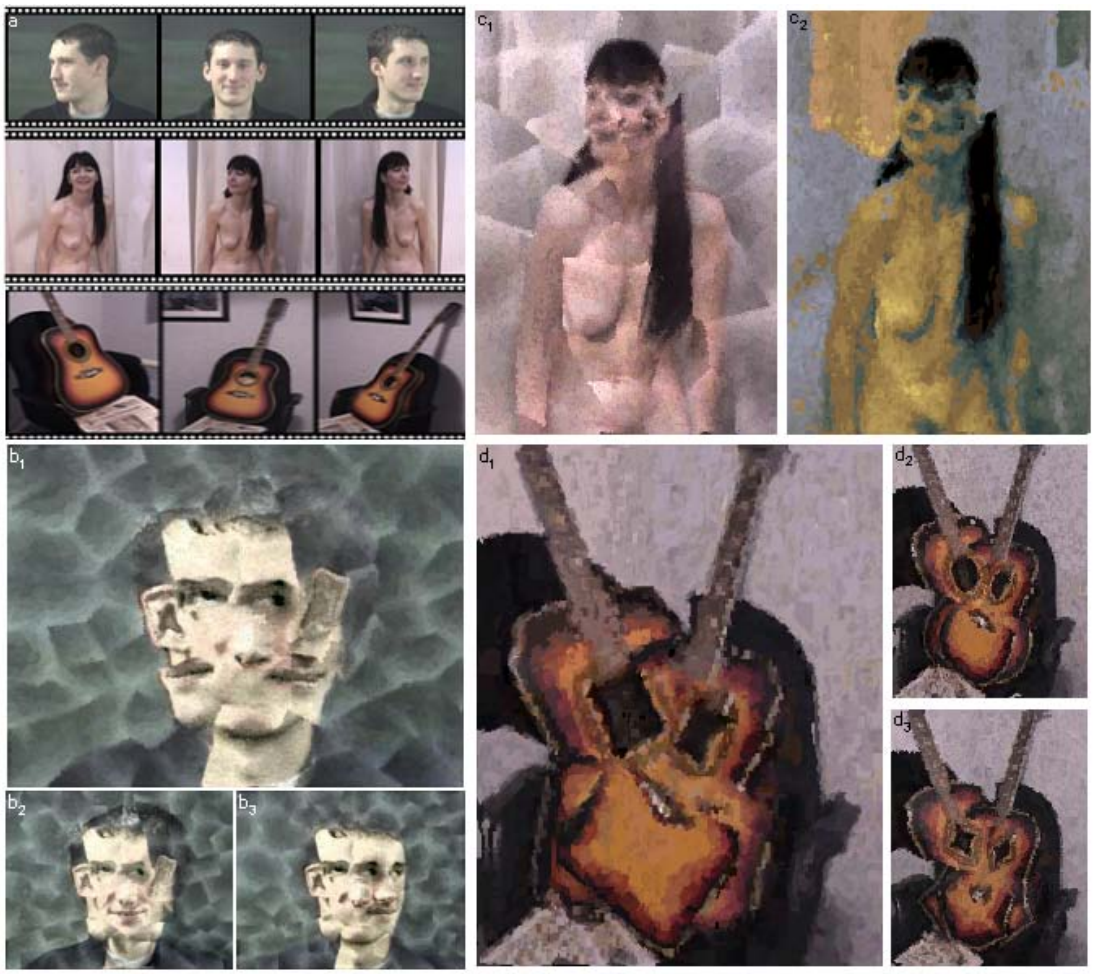


Figure 3-17 A gallery of images illustrating the application of our rendering algorithm.

Paintings produced from the guitar images are presented in Figure 3-17*d*; salient features were the hole, neck, bridge and chair arms. Figures 3-17d_1, 3-17d_2, and 3-17d_3 are identical compositions rendered with different distortion and painting parameters. The values of distortion parameter α' for each of the renderings is 0.5, 1, and 2 respectively. Notice how the hole in the guitar changes shape, from rectangular to star-like. By changing only these parameters, a varied range of styles are produced. The finer segmentation of non-salient regions was not performed on these renderings, to allow clear demonstration of distortion effects.

New year honours
James Lovelock is among academe's acclaimed 9

Briefing notes
What should the government address in its strategy paper? 6

Rising stars
Which big name is destined to find mainstream fame? 18

THE TIMES
HIGHER
EDUCATION SUPPLEMENT

www.thes.co.uk
JANUARY 3 2003 No.1,570 £1.40

Surrey seeks escape from state control

Caroline Davis

Surrey University could become Britain's first public university to opt out of government control and rely on independent funding.

In 2000-01, the university was among the institutions that received the lowest proportion of their income from the Higher Education Funding Council for England.

New figures released by Hefce show that, on average, institutions received 41 per cent of their income from the funding council. Only 17 universities now derive the majority of their funding from the council. Just 25 per cent of Surrey's income came from Hefce, down from 28 per cent three years earlier.

Vice-chancellor Patrick Dowling said that the university listed government higher education policy as one of the highest risk factors in its strategy. "Something has to be done. It could reach the stage where we feel strongly enough [to become independent]. I would like to be able to keep our options open," he said.

He told *The THES*: "It is not easy going private overnight. But if we were given a major sum or endowment, we would be one of the first ones to say let's have a go."

Until recently, Surrey's undergraduate degree in dance was privately funded by student fees and was never short of applicants. But Professor Dowling said tuition fees should not rise as this could deter students.

He said the university could operate totally independently from government, being research-led with high-quality teaching at both undergraduate and postgraduate levels.

Since suffering huge budget cuts in the early 1980s, Surrey has worked to distance itself from government funding. It increased the proportion of postgraduate and overseas students (who pay fees) and set up a "land bank", profiting from sales, most recently for the building of a hospital. Surrey was also the first UK university to own a science research park, from which it nets about £4 million a year in rent.

In 2000-01, the university received £29 million of its £116 million income from Hefce grants. The rest came from academic fees and support grants, research grants and contracts, endowments, rental income from its research park, and its ten subsidiary companies.

Other institutions, however, said they would be unable to turn down millions of pounds in funding council support.

Imperial College London received 29 per cent of its income from Hefce last year. But despite raising the prospect of tuition fees of up to £15,000, the college said it could not afford to give up government support.

Rodney Eastwood, Imperial's director of planning and information, said: "We could not be independent from the £60 million Hefce research grant." He said that including grants from the research councils, Imperial was 50 per cent funded by the public.

Cambridge University derived 32 per cent of its income — £140 million — from Hefce. Treasurer Joanna Womack said that the endowment required to generate a similar income was unrealistic.

"At 4 per cent, that represents a capital sum of £3.5 billion," she said. "We could not raise that, and even if we could, such a sum would just leave us where we are at present, whereas, like all universities, we need to grow our income in order to cover increased costs. What we really need is increased Hefce funding for both research and teaching, guaranteed for several years so that we can make sensible plans."

Michael Sterling, vice-chancellor of Birmingham University, suggested that he would like to move towards more independent funding. With one-third of its income from Hefce, he said there was no possibility of the university opting out. "That's still too much on our

continued on page 2

Universities that receive least and most Hefce funding as percentage of income

Institution	00-01	07-08	Change
London Business School	6	9	-33%
LSE	19	23	-17%
City	24	27	-11%
Surrey	25	28	-11%
Cranfield	16	17	-6%
North London	57	58	-2%
London Guildhall	60	61	-2%
Manchester Met	59	58	2%
Ulster	58	55	5%
Lincoln	61	55	11%

Source: Hefce

Time to work on the image, Mr Clarke?

Steve Farrar

Culture minister and self-appointed art critic Kim Howells will have to curb his tongue this time. Just weeks before his colleague Charles Clarke makes clear his vision for the future of higher education, the education secretary's features have been distorted in the cubist style (pictured) to demonstrate new computer technology at Bath University.

The technique, developed by postgraduate student John Collomosse and lecturer Peter Hall, involves minimal human intervention to turn photographic images into the sort of artworks once created by Picasso.

The researchers used a selection of photographs of Mr Clarke, taken from different viewpoints and supplied by *The THES*.

Dr Hall said: "In order to draw, you have to be able to see." The software, he explained, picks out individual elements, which it geometrically distorts before reassembling them into a coherent composition. It then turns a selection of dots into brush strokes that work around important aspects of the picture, giving a painterly effect.

Professional artists have judged the project's output to be of a high aesthetic quality, and the Bath team has entered work in computer art competitions.

The research was supported by the Engineering and Physical Sciences Research Council and will be published in the journal *Transactions*.

***THES* diary, page 13**

Figure 3-18 A Cubist-style image of Charles Clark MP appeared in the Times Higher Educational Supplement [3rd January 2003], and was rendered from a set of photographs supplied by Times reporter Steve Farrar (below).

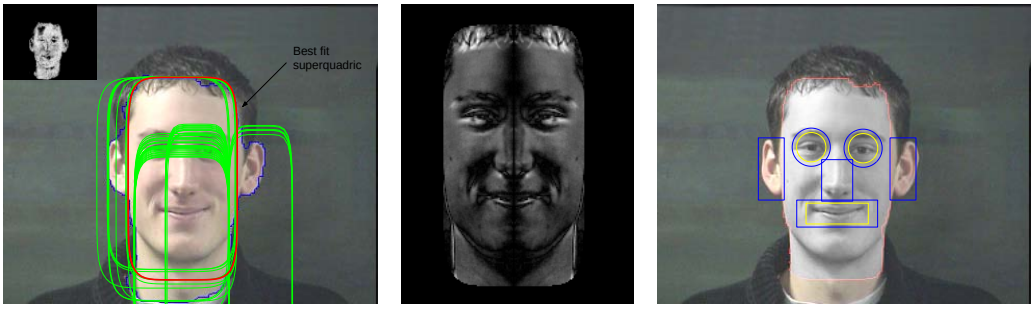


Figure 3-19 Illustrating our ad-hoc facial registration technique. Left: Images are processed using a hue/saturation based eigen-model to identify flesh coloured areas (inset). Candidate faces (green) are found using our Hough based search strategy (Appendix A.1), which operates on edge pixels identified in the image (blue) following eigen-analysis. Middle: Difference map used to quantify the symmetry expressed by image data along the principal axis of a candidate superquadric. Right: A rigid template containing eyes and mouth (yellow) is fitted to the image data by translation and scaling within the ortho-normal basis of the best fit superquadric. Snake initial contour locations are also transformed with the template (blue), and subsequently relaxed to identify salient features (Section 3.4.1).

3.5 Personal Picasso: Fully Automating the Cubist Rendering System

The general segmentation problem unfortunately prohibits the fully automatic extraction of salient features from a general image. However certain classes of image are well studied in Computer Vision, and can be automatically segmented into such features. In this section we describe an implementation of our Cubist rendering system which adapts techniques from the Vision literature to locate the features of a face within a single video frame. These features are then tracked through subsequent video frames, and so substitute the interactive elements of our Cubist algorithm to create a fully automatic rendering process. The motivation for this case study is to produce a "Personal Picasso" system capable of rendering Cubist portraits from video. Potential applications for this system might include installation as a feature in commercial photo booths. We describe a proof of concept implementation of such a system in the following subsections.

3.5.1 An Algorithm for Isolating Salient Facial Features

We begin by describing an ad-hoc technique for locating salient facial features within a single image. The first stage of processing involves the location of the face within the image; we assume that a full frontal image of a single face will always be present. The second stage localises the various facial features, e.g. eyes, mouth within the identified face.

Locating the face

Faces are located on the basis of their distinctive colour signature and shape, using a ad-hoc process which draws upon both previous colour-blob based face location strategies (for example, [125]) and the Hough transform based, geometric approach of [106].

It is well known that, despite natural variations in skin tone and colour, skin pigments tend to form a tight cluster in 2D hue/saturation space [49]. Prior to processing, we perform a one-time training process which fits an eigenmodel to various samples of skin colour taken from photographs; this has empirically proven to be a satisfactory model of the unimodal distribution of pigments. The eigenmodel is specified by a mean $\underline{\mu}$, eigenvectors $\underline{\underline{U}}$, and eigenvalues $\underline{\underline{\Lambda}}$. When processing a novel source image for face location, we compute the Mahalanobis distance of each novel pixel's colour with respect to the trained eigenmodel. The Mahalanobis distance $L(\underline{c})$ of a particular pixel with colour $\underline{c} = (c_h, c_s)^T$ (where c_h is the colour hue component, and c_s the saturation — both components normalised to range [0,1]) may therefore be written as:

$$L(\underline{c}) = \left((\underline{c} - \underline{\mu})^T \underline{\underline{U\Lambda U}}^T (\underline{c} - \underline{\mu})\right)^{\frac{1}{2}} \tag{3.16}$$

More precisely, taking into account $c_h = c_h \ \mathbf{mod}\ 1$ we use:

$$L'(\underline{c}) = \min(L(\underline{c}), L(\underline{c} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix})) \tag{3.17}$$

This produces a modified Mahalanobis field such as that of Figure 3-19, left (inset). Canny edges [12] are detected within this map to produce a set of binary edge pixels.

Faces vary in their shape; some tend toward elliptical forms whilst others are described as being more rectangular. In line with this observation we have chosen to model the face as a superquadric (equation 3.2) which encompasses all of these forms in a single parameterised framework. Superquadrics are fitted to the edge pixels using a Hough based search technique, which incorporates a novel implementation strategy to handle the large parameter space defining potential superquadrics. The reader is referred to Appendix A.1 for a full discussion of this fitting process. The fitting process results in a ranked list of 6-tuples, each corresponding to a potential location for a superquadric (Figure 3-19, left). Each 6-tuple contains a set of parameters: $[C_x,\ C_y,\ r,\ a,\ \theta,\ \alpha]$, which correspond to the 2D centroid location, scale, eccentricity, orientation and form factor respectively.

We take advantage of the natural symmetry of faces to assist in selecting the best fit superquadric from those returned. We quantify the symmetry (about the principal axis)

of the image region bounded by each candidate superquadric. For a given superquadric, the line of symmetry passes through point $(C_x, C_y)^T$ at angle θ degrees from the vertical. The reflected image $\underline{p}'$ of point $\underline{p}$ in homogeneous form may be obtained using the following affine transformation:

$$\begin{aligned} \underline{p}' &= \underline{\underline{T}}^{-1}\underline{\underline{R}}^{-1}\underline{\underline{F}}\underline{\underline{R}}\underline{\underline{T}}\underline{p} \qquad (3.18) \\ \underline{\underline{T}} &= \begin{bmatrix} 1 & 0 & -C_x \\ 0 & 1 & -C_y \\ 0 & 0 & 1 \end{bmatrix}, \ \underline{\underline{R}} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \underline{\underline{F}} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

We perform the reflection on all pixels within the candidate superquadric, and compute the mean squared error (MSE) of the reflected and original images (Figure 3-19, middle). We score each candidate by multiplying the integer "evidence" for that superquadric (returned by the Hough search, see Appendix A.1), with unity minus its MSE. The highest scoring candidate is deemed to correspond to the facial region.

Locating features within the face

Facial features are located using a simple template based correlation scheme. Prior to processing, several eyes and mouths are manually segmented and registered upon one another to produce "average" feature templates. We used sixteen samples from the Olivetti face database [136] for this task; these faces are available only as greyscale images, and the template matching process thus correlates using only luminance data. The pixel data for each template is stored, along with the average triangle formed by the centroids of the two eyes and the mouth. This results in a rigid, planar template of the three features, which we attempt to register on to the facial region identified at run-time.

Registration is a two step process. First, the basis of the template is rotated to align with the ortho-normal basis of the superquadric bounding the facial region. Second, the template is subjected to translation $(T_x, T_y)^T$ and uniform scaling s in order to minimise the MSE between the template and the image data. Minimisation is performed using a Nelder-Mead search [114] to locate the optimal triple (T_x, T_y, s). Coarsely fitting shapes (circles for the eyes, rectangles for other features) are also drawn onto the template prior to processing. The vertices of these shapes form the initial control points for the active contours (snakes) which are relaxed on to salient features as per Section 3.4.1 (Figure 3-19, right), once the template has been fitted. The template also assigns preset equivalence class categories to the localised features, for example "eye", "nose" etc.

Note that we assume that this initial image contains a full frontal image of the face free from occlusion; though this image may be subjected to affine variations. This seems a reasonable constraint to impose given the photo booth application that motivates this case study.

3.5.2 Tracking the Isolated Salient Features

We can further automate the Cubist system by tracking identified salient features over consecutive frames of video. This implies that a temporal dependency must exist between source images, which has not been a constraint on the process until this point. However since the majority of source imagery is likely to be captured in the form of video, and this may be acceptable in the majority of cases.

The tracking process commences directly after the snake relaxation step for the initial frame (see previous subsection). We write $\underline{p}_i$ to represent the i^{th} of n inhomogeneous control points describing the spline fitted about a salient feature. We may write these points as a column vector to obtain a point $\underline{\phi}$:

$$\underline{\phi} = \left(\underline{p}_1^T, \underline{p}_2^T, ..., \underline{p}_n^T\right)^T \in \Re^{2n} \tag{3.19}$$

The problem of tracking a salient feature from one frame to the next is now reformulated to that of determining the mapping of the feature's bounding spline $\underline{\phi}$ to a new point $\underline{\phi}' = M(\underline{\phi})$ in the high dimensional space $\Re^{2n}$. If we assume all points on the feature boundary to be co-planar, this mapping $M(.)$ decomposes into a homography $\underline{\underline{H}}$ plus some additive term representing spatial deformation $\underline{s}$. Writing $\underline{\underline{P}}$ as a homogeneous representation of the points encoded in $\underline{\phi}$:

$$\underline{\underline{P}} = \begin{bmatrix} \underline{p}_1, \underline{p}_2, ..., \underline{p}_n \\ 1 \end{bmatrix} \tag{3.20}$$

we write the mapping as:

$$\underline{\underline{P}}' = \underline{\underline{HP}} + \underline{s} \tag{3.21}$$

If we assume that the object being tracked deforms very little from frame to frame, then all image-space deformation is due to viewpoint change. Under these conditions, homography $\underline{\underline{H}}$ well describes the mapping $M(.)$:

$$\underline{\underline{H}} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{3.22}$$

where h_j is the j^{th} component of the homography $\underline{\underline{H}}$ (by convention, $\underline{\underline{H}}$ is normalised such that $h_9 = 1$; the transformation has eight degrees of freedom). We can therefore consider an eight dimensional subspace in $\Re^{2n}$, within which points corresponding to valid splines (and so tracked features) will approximately lie[3]. Deviation from this space corresponds to feature shape deformation during tracking. We assume the manifold of valid splines to be locally linear and so well approximated by a hyper-plane (this assumption is justified momentarily). The bases of this plane are obtained by computing the Jacobian of $M(.)$. Applying a Taylor expansion of 1^{st} order to $M(.)$ we obtain:

$$M(\underline{\phi} + \mathrm{d}\underline{\phi}) = M(\underline{\phi}) + \underline{\nabla}^T_{M(\underline{\phi})}\mathrm{d}\underline{\phi} \tag{3.23}$$

where $\underline{\nabla}^T_{M(\underline{\phi})}$ is the gradient of $M(\underline{\phi})$ at $\underline{\phi}$. Under our assumption that $M(.)$ varies only by homography, then $\underline{\nabla}_{M(\underline{\phi})}$ may be written as:

$$\underline{\nabla}_{M(\underline{\phi})} = \begin{bmatrix} \frac{\partial \underline{\phi}_1}{\partial h_1} & \frac{\partial \underline{\phi}_1}{\partial h_2} & \cdots & \frac{\partial \underline{\phi}_1}{\partial h_8} \\ \frac{\partial \underline{\phi}_2}{\partial h_1} & \frac{\partial \underline{\phi}_2}{\partial h_2} & \cdots & \frac{\partial \underline{\phi}_2}{\partial h_8} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \underline{\phi}_{2n}}{\partial h_1} & \frac{\partial \underline{\phi}_{2n}}{\partial h_2} & \cdots & \frac{\partial \underline{\phi}_{2n}}{\partial h_8} \end{bmatrix} \tag{3.24}$$

where $\frac{\partial \underline{\phi}_i}{\partial h_j}$ denotes the shift of the i^{th} control point under a small change of the j^{th} component of the homography.

The basis of the valid subspace corresponds to the eight columns of $\underline{\nabla}_{M(\underline{\phi})}$, scaled by the reciprocal of the square of their respective L_2 norms. This process compensates for the greater influence over motion that some homography components (e.g. projections) hold over others (e.g. translations) given similar numerical variation. The remaining null space $\Re^{2n-8}$ accounts for arbitrary shape deformations of the tracked spline. Generating small normal variate offsets from $\underline{\phi}$ within the homography space basis set generates "similar" splines, related by homography to the spline specified by $\underline{\phi}$ (see Figure 3-20). Notice that as projections are cast further from the original contour they tend away from homographies and begin to exhibit shape deformations. This is because our linear approximation is only local to $\underline{\phi}$, and deteriorates as we move further from $\underline{\phi}$ so digressing from the space of valid contours into the shape deformation (null) space.

A two stage process is employed to track features. First we determine the homography

[3] In fact, our $\Re^{2n}$ parameter space was chosen specifically because of its support for shape change due to homography. Such support is not generally present for any parameter space; consider, for example, the 6D parameter space of the superquadric (Appendix A.1).

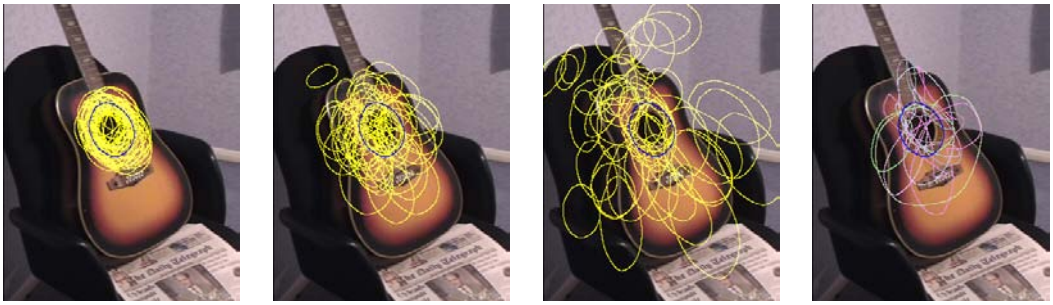


Figure 3-20 The bounding spline of a salient feature (the guitar hole) after snake relaxation (dark blue). The three leftmost images show variation in homography space using normal variates of standard deviation 20, 50, and 100 (from left to right). Shape deformation becomes apparent with greater standard deviation as the local linear approximation deteriorates. By contrast the rightmost image shows variation in the null (arbitrary shape deformation) space.

that best maps the feature from one frame to the next. In this context the "best" mapping corresponds to a minimisation of the mean squared error (MSE) $E(.)$ between the RGB colour values of pixels bounded by the spline in the current frame $I_t(.)$ and those bounded by the putative spline location in the next frame $I_{t+1}(.)$:

$$E(M_i(.); \underline{\phi}, I) = \frac{1}{N} \left| I_{t+1}(M_i(\underline{\phi})) - I_t(\underline{\phi}) \right|^2 \tag{3.25}$$

where N is the number of pixels bounded by spline $M_i(\underline{\phi})$. $M(.)_i$ is a putative mapping (homography) obtained by local stochastic sampling in the valid subspace as previously described — we use a random variate centred at $\underline{\phi}$, with a preset standard deviation σ. Choice of σ effectively determines the maximum inter-frame distance an object can move, such that we may continue to track it; note that σ can not be too large, or the locally linear approximation to the homography breaks down. Values up to around 50 are reasonable for video frame size images; lower values are possible for tracking slower movements. This process yields an approximate grouping contour which is used as the initial position for a snake which is iteratively relaxed to account for any shape deformation in the feature (as per the method of Section 3.4.1). We perform this relaxation to take into account shape deformations, which we do not wish to track explicitly. This is due to their high dimensionality and unconstrained nature, which has been shown to cause unstable tracking in the absence of a predetermined, specific motion model [84] (which we did not wish to introduce for reasons of generality).

This approach does not yet take into account the possibility of occlusion — however features can become occluded when subjected to large camera viewpoint changes. We handle this problem in two ways. First, we may choose to track individual features (as in Figure 3-20), or the entire set of features simultaneously under the assumption that all features are co-planar in the world. This latter approach holds the advantage

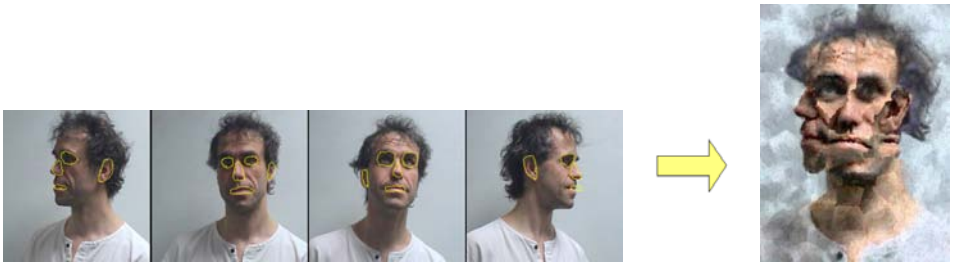


Figure 3-21 Four frames of a source video after image registration. Features have been tracked as a planar group, and each active contour subsequently relaxed to yield a salient feature boundary (shown in yellow). The corresponding Cubist painting is on the right.

that if one salient feature, say an eye, rotates away from the view of the camera, the other features may still be tracked. This yields a robust common homography under which all features are assumed to have moved. We have found this approach to work acceptably well for faces (Figure 3-21). Second, if the MSE for a feature (equation 3.25) rises above a preset upper threshold, then that feature is deemed to be occluded. In such cases, the feature is not to be sampled from the current frame, to prevent garbled image fragments being forwarded to the composition engine in place of the occluded feature. The equivalence classes attached to a feature persist over time during tracking.

We stated earlier that a locally linear approximation to the homography space was satisfactory for our application. We have shown this to be true by examining the 2^{nd} order Taylor expansion of $M(.)$. This results in eight second order Jacobians, which we observe have determinants close to zero in all but the cases in which h_7 and h_8 vary. Thus the valid space is approximately linear except for the projective components of the homography. If we make the intuitively valid assumption that that change of viewpoint is small (relative to other homography components) from one frame to another, then a linear approximation to the space is justified.

The decomposition approach allows us to track robustly since we restrict the possible deformations of the contour to those caused by a change of viewpoint, which we assume accounts for the majority of deformation in the 2D scene projection. However, there are further advantages gained through decomposition of motion into a homography plus some general shape deformation. As each transform $\underline{\phi}' = M(\underline{\phi})$ is computed, we calculate a motion vector $\underline{\phi}' - \underline{\phi}$ in the valid subspace $\Re^8$. Based on the line integral of the path traced in this space, we are able to quantify the magnitude of change of viewpoint. Since we approximate the space to be locally linear we are justified in using a Euclidean distance to compute this inter-frame distance; however the path integral approach is necessary over larger time periods, since we have shown the space to be globally non-linear. When this accumulated distance rises above a certain threshold,

we sample the current frame from the video for use in the composition. Thus the selection of frames for composition is also automated (subject to the user setting a suitable threshold for the integral function).

Future work might attempt to further decompose motion into the various components of the homography to estimate how the feature has translated, rotated etc. Currently the Cubist algorithm performs a translational image registration using distinct colour signatures within the image. Using the homography to determine an inverse translation may allow us to register images regardless of their colour characteristics, and we predict that this may improve the generality of the algorithm.

3.6 Summary and Discussion

We have argued that the paradigm of spatial low-level processing limits AR in two ways. First, that quality of rendering suffers since magnitude of high frequency content, rather than the perceptual importance, of artifacts governs emphasis during rendering. We have shown that addressing this limitation demands global image analysis, rather than the spatially local approach so far adopted by AR. Second, that the spatially local nature of processing limits AR to low level, stroke based styles. We argued that the synthesis of compositional forms of art, such as Cubism, can not be achieved without processing images at a spatially higher level than that of local pixel neighbourhood operations.

In this chapter we introduced a global salience measure to AR, to determine the relative importance of image regions. We applied this measure to propose two novel AR algorithms, which respectively addressed each of the AR limitations identified in the previous paragraph. The first was a single-pass AR algorithm capable of rendering photographs in a painterly style reminiscent of pointillism. This algorithm adaptively varies the emphasis in a painting to abstract away non-salient detail, and emphasise salient detail. The second was an algorithm capable of producing compositions in a style reminiscent of Cubism. Uniquely, this algorithm made use of salient features (eyes, ears, etc.) as the atomic element in the painting, rather than the low-level stroke. The two algorithms respectively demonstrate how a spatially higher level of image analysis can improve the aesthetic quality of renderings (more closely mimicking the practice of human artists), and extend the gamut of AR beyond stroke based rendering to encompass compositional artistic styles such Cubism.

There are a number of directions in which this work might be developed further. We have shown the introduction of a global salience measure can remove limitations im-

posed by spatially local nature of current AR. Although this rarity based salience measure is new to computer graphics, it is quite simplistic. The definition of image salience is highly subjective and context sensitive. Consider a snap-shot of a crowd: in one scenario a particular face might be salient (for example, searching for a friend); in another scenario (for example, crowd control), each face might hold equivalent salience. The development of image salience measures is an area of considerable interest in Computer Vision, and no doubt other global salience measures might be substituted for our own — the loose coupling between the salience measure and rendering steps facilitates this. Indeed, in Chapter 4 we make use of a more subjective, user trained measure of salience to drive a more sophisticated salience adaptive painterly rendering process, which follows on from this work.

The Cubist rendering algorithm is an illustration of the potential expansion of AR's gamut of artistic styles that can be achieved by considering spatially higher level features within a scene. However to achieve this higher level of segmentation we must impose a restrictive model upon the scene, removing the need for interaction, at the cost of generality. It is unfortunate that contemporary Computer Vision techniques limit full automation to only a few well studied cases. However, interactive grouping typically takes less than one minute of user time, and so we are content with our method as a compromise between a general system and automation. As regards the "Personal Picasso" proof of concept system, although the face localisation algorithm is reasonably robust, the location of facial features themselves leaves something to be desired. Likewise, the tracker is adequate but could be improved to be more robust to occlusion. The implementation of a more sophisticated facial feature location and tracking system for the Cubist renderer is a live BSc. project at Bath.

Although Chapter 4 serves as a continuation of the pilot painterly rendering algorithm presented in this Chapter, there are a number of interesting directions the Cubist rendering work might take. The depiction of movement within a static image is a unique contribution to AR, but may hold further applications. For example, the production of static "thumb-nail" images to help summarise and index video content. We might consider undertaking a compositional analysis in order to more aesthetically place our high level features, and progress yet further toward emulating Cubism; however we believe such an analysis is a considerable challenge that is not necessary to demonstrate the synthesis of Cubist-*like* renderings are made possible through higher level spatial analysis. We might revisit the way in which we apply paint, so that it appears more in the tradition of a particular artist, but there is no compelling reason to focus our work in such a way at this stage: the manipulation of high level features is the only necessary step to producing images that can be classified as "Cubist" or, at least, "Cubist

influenced".

A higher level of analysis still, might be applied to extract salient features from image; perhaps a full 3D scene reconstruction and re-projection from novel perspectives to generate alternative Cubist-like styles. Perhaps alternative global analyses of the image might generate aesthetically pleasing abstract artwork. For example, one might investigate use of the Hough transform to identify target shapes within an image, taken from a user defined "shape library". The subsequent rendering of those shapes may provide a basis for synthesising alternative abstract artistic styles. Such possibilities lend further credence to our argument that higher level spatial analysis opens the doorway to a wide range of otherwise unobtainable artistic rendering styles.

Chapter 4

Genetic Painting: A Salience Adaptive Relaxation Technique for Painterly Rendering

In this chapter we build on the single-pass salience adaptive painterly technique of the previous chapter to propose a novel, relaxation based iterative process which uses curved spline brush strokes to generate paintings[1]. We draw upon our previous observations of artistic practice to define the degree of optimality for a painting to be measured by the correlation between the salience map of the original image and the level of detail present in the corresponding painting. We describe a novel genetic algorithm based relaxation approach to search the space of possible paintings and so locate the optimal painting for a given photograph, subject to this criterion. In this work we make use of a more subjective, user trained measure of salience[2]. The quality of the rendering is further enhanced through the use of context dependent stroke rendering styles, and compensation for image noise; both are additional novel contributions to image-space AR.

4.1 Introduction

In the previous chapter we observed a relationship between the importance that artists assign to artifacts in a scene, and the level of detail and emphasis afforded to such artifacts in a piece of artwork. We also observed that automatic image-space AR techniques are at odds with this behaviour, emphasising all detail regardless of its salience.

[1]This work has previously appeared as [24].

[2]The trainable salience measure we use is described in Section 4.3 and will appear as [66]. The primary investigator of the measure was Peter Hall (University of Bath), with whom the author collaborated. For the purposes of examination, the areas of the measure to which the author has contributed are clearly indicated *in situ* within Section 4.3.

We rectified this discrepancy by developing a novel, single-pass painterly rendering process which modelled the artist's salience adaptive behaviour. The resulting paintings exhibited improved aesthetics; not only were salient regions painted precisely (strokes from non-salient regions did not encroach upon and distort regions of greater salience), but the rendering also exhibited a sense of focus around salient regions due to the abstraction of non-salient detail. We compared the edge maps of the resulting paintings with the salience maps of their original images, and observed the two to exhibit qualitatively closer correspondence than when using local frequency response filters to drive the rendering process (Figure 3-6).

We now build upon the success of this pilot, single-pass salience adaptive rendering technique (Section 3.3) to propose a novel relaxation based approach to salience adaptive painting. We build upon our previous observations to define the degree of optimality for a painting to be measured by the correlation between the salience map of the original image and level of detail within the corresponding painting. We describe novel salience based approach to painting which uses a genetic algorithm (GA) relaxation technique to search the space of possible paintings, and so locate the optimal painting for a given photograph. In doing so we use a more subjective definition of salience that can be trained to select features interesting to an individual user, and which performs global analysis to simultaneously filter and classify low-level features, for example to detect edges, ridges and corners. A further contribution of our method is that differential rendering styles are possible by varying stroke style according to the classification of salient artifacts encountered. Uniquely, we also compensate for noise; a component of any real image.

There are several advantages that this novel technique holds over our previous single-pass technique of Section 3.3.

1. Single-pass rendering techniques (such as our pointillist-style algorithm and many existing image-space AR methods [58, 71, 103, 140, 159]) are highly susceptible to image noise. Some stroke attributes, such as orientation, are determined by local sampling of fields obtained through differentiation of the source image; for example Sobel intensity gradient fields. This differentiation process often serves to exaggerate noise present in the image, which manifests as numerical inaccuracies in the field leading to poor stroke placement or poor setting of visual attributes. In single-pass techniques, a stroke's location and visual attributes are determined once only, after which they remain fixed. One has no guarantee that the compositing of multiple, placed strokes will lead to an aesthetically optimal painting; strokes do not take into account the effect of other strokes in their vicinity. By contrast, a goal directed iterative painting strategy can approach a more optimal

Figure 4-1 Section of a painting generated using our salience based relaxation technique, taken from the fittest individual within the 80th generation of paintings. High resolution versions of this and other paintings in this chapter are included in the electronic supplementary material in Appendix C.

solution. We describe a GA based relaxation process which iteratively varies the attributes of strokes, assessing the "fitness" of the entire painting at each iteration (using our salience based optimality criterion), with the aim of producing an optimal, target painting.

2. We make use of a user-trained salience measure, which users may teach to recognise artifacts which they deem to be perceptually important. As we mention in Chapter 3, salience is a task specific, subjective concept which can only be addressed by prescriptive measures at an early visual level. This trainable method provides a more subjective basis to the problem of determining the salience of artifacts within an image.

3. Our painterly technique not only drives emphasis in the painting via salience magnitude, but can also vary stroke rendering style according to the classification of salience artifact encountered (for example, edge or ridge).

4. Our algorithm composits multiple curved spline strokes to create paintings, rather than simple daubs of paint. This allows synthesis of paintings exhibiting elegant, flowing brush strokes, for example in the style of Van Gogh (Figure 4-17, right). This approach presents a more general painting solution, since the rendering parameters controlling maximum stroke length can be reduced, causing the output of the system to degenerate back toward daub based pointillist styles. Thus the proposed salience-driven system is capable of rendering photographs in a wider gamut of artistic styles than our previous salience-driven approach.

Our paintings are formed by compositing curved Catmull-Rom [14] spline brush strokes via an adaptation of the multi-scale curved stroke painterly technique proposed by

Hertzmann [71]. We build upon this work in two ways. First, we modify the technique to accommodate preferential rendering with regard to salience. Strokes are more densely placed in salient regions, then ordered and modulated to prevent strokes from non-salient areas encroaching upon more salient ones. Differential rendering styles are also possible by varying stroke style according to the classification of salient artifacts, for example edges or ridges. This context-dependent adaptation of stroke style is a novel contribution to AR. Second, we use our novel relaxation scheme to iteratively converge the rendering toward the "optimal" painting for a given image. We adapt Hertzmann's contour tracing algorithm to account for the influence of noise, present in any real image. As a consequence, post-relaxation strokes tightly match the contours of salient objects whilst non-salient high frequency detail (emphasised with other painterly methods) is attenuated. We demonstrate the results of our painterly technique on a wide range of images, illustrating the benefits of rendering with regard to salience and the improvements gained by subsequent relaxation of the painting using our GA based technique.

4.2 Background in Evolutionary Computing

Evolutionary Algorithms (EAs) seek to model the evolutionary processes found in biology, such as natural selection or mutation, to search for an optimal solution to a problem under a specific set of constraints. The early development of EAs dates back to the sixties, when the research of such algorithms fell under the encompassing title of "Evolutionary Computing". However the independent development of similar ideas by separate research groups ensured that, by the early eighties, EAs had diversified into three subtly distinct categories [70]: Genetic Algorithms (GAs) [36, 56, 77], Evolutionary Programming (EP) [50] and Evolutionary Strategies (ES) [131].

GAs are generally considered to originate from the cellular automata work of Holland *et al* [77]. The GA operates upon a population of individuals; each individual represents a point in the problem space and is uniquely characterised by its associated genome containing a "genetic code"; this code often takes the form of a binary string but this is not strictly necessary. Each individual can be evaluated via a "fitness function" to yield a scalar value corresponding to the optimality of the solution it represents. The GA operates by breeding successive generations of these individuals. Selection of individuals for breeding is via a stochastic process biased toward selecting fitter individuals; so exhibiting a Darwinian "survival of the fittest" behaviour. The breeding process itself involves the swapping of genetic code (genome fragments) between parents, to produce a novel individual. This exchange of code is termed the "cross-over" process. In addition, each element of the genome may be perturbed by some random amount

— however the probability of large scale perturbation is slight. This process is termed "mutation". Self propagation (analogous to asexual reproduction) of a single parent may also be permitted in some implementations. This survival of the fittest methodology can evolve populations of individuals which tend to approach global maxima even for complex problems exhibiting turbulent, high dimensional problem spaces. The GA is a methodology, rather than an algorithm, and there are five principal issues that must be resolved to successfully tailor the GA to a specific application:

1. How should a solution be represented by an individual's genome?
2. What are the mechanics of the cross-over and mutation processes?
3. How should the initial population be generated?
4. How should fitness be defined?
5. What should the population size be? Should it vary or remain static?

We address these issues *in situ*, during the explanation of our GA based relaxation process (Section 4.4.2).

The EP methodology is subtly different to that of the GA, in that no cross-over is performed. Offspring for successive generations are copied from a single parent, selected stochastically with a bias to fitness, and subjected to mutation; major mutations have a much lower probability than minor mutations. The genome tends not to be explicitly represented as a string, but as a point in the problem space. Mutation in many cases takes the form of a translation in the problem space of stochastically chosen direction and magnitude.

The ES methodology is very similar to that of EP, but again differs slightly. The selection process in EP is often "tournament based"; a pair of parents are picked at random to compete in a tournament — the outcome being decided by their fitness and the winner being allocated a "win point". Many individual tournaments take place when producing a successive generation, and the highest aggregate scoring parents are allowed to propagate. By contrast, with ES the weaker parents are deterministically culled from the population prior to propagation.

4.2.1 Genetic Algorithms in Computer Graphics

EA based techniques have been successfully applied to a number of areas within Computer Graphics. Early uses of EAs include Reynolds' distributed behavioural models [129]. These simulated many individual automata interacting via simple rules, from

which emerge more complex collective behaviours such as flocking and herding. A realistic model of fish locomotion was presented by Tu and Terzopoulos [160], which simulated perception, learning and, like Reynolds' automata, also exhibited group behaviour. A well known application of GAs to Computer Graphics is in the virtual creatures of Sims [141]. Each of Sims' creatures is defined genetically, and generations of creatures can evolve novel behavioural and graphical characteristics over time. Sims posited that creatures which evolve via simulated genetic processes should be capable of exhibiting more a complex and realistic biological evolutionary response than possible through explicit, procedural modelling. GAs have also been applied to content based image retrieval (CBIR) [2].

The majority of applications for GAs in the field of Computer Graphics address the problem of goal directed animation. GAs have been successfully applied to animate realistic flower growth [105], and also to develop stimulus response systems for human articulated motion [115]. Tang and Wan [157] described a GA based system which allows character motions to evolve in virtual environments, for example learning the optimal way to perform a jump to reach a goal. Motion planning and character animation techniques were also driven by GAs in [126] and [179] respectively.

Our paint by relaxation technique is GA based. To justify this, consider Haeberli's [62] abstraction of a painting as an ordered list of strokes (comprising control points, thickness, etc. with colour as a data dependent function of these); the space of possible paintings for a given image is clearly very high dimensional, and our optimality criterion makes this space extremely turbulent. Stochastic searches that model evolutionary processes, such as genetic algorithms, are often cited as among the best search strategies in situations of similar complexity [36]. This is due to the fact that GAs search from a population of points, not a single point, and that the mutation and cross-over processes integral to propagation cause jumps in the problem space which can mitigate against the attraction of local minima; these cause difficulty to other strategies such as gradient descent or simulated annealing. Furthermore, whilst it is difficult to explicitly model the complex relationships between stroke parameters during the creation of a painting, goal driven stochastic optimisers such as GAs are known to perform acceptably is the absence of such models.

4.3 Determining Image Salience

For the purposes of examination, please note that the primary investigator of the salience measure described in this section was Peter Hall of the University of Bath, with whom the author collaborated. Specifically, Hall developed the basic measure which operates at a single scale (Sections 4.3.1– 4.3.3). The multi-scale extensions to the measure were investigated and developed by the author (Section 4.3.4).

Our painterly process requires a method to automatically estimate the perceptual salience of images. That is, produce a mapping from a colour image to a scalar field in which the value of any point is directly proportional to the perceived salience of the corresponding image point. We now describe an approach to estimating this mapping, comprising three operators which respectively compute the rarity, visibility, and classification of local image artifacts. These three operators are computed independently yielding three probabilities (P_{rare}, $P_{visible}$, P_{class}). These are combined to estimate the final probability of an image artifact being salient as:

$$P_{salient} = P_{rare}P_{visible}P_{class} \tag{4.1}$$

Each of the three operators makes use of circular signals generated by sampling from concentric rings centred upon the pixel whose salience is to be determined. The first operator performs unsupervised global statistical analysis to evaluate the relative rarity (P_{rare}) of image artifacts. This process is similar to our original rarity based approach of Section 3.2, and the motivation for this operator is principally to adapt that rarity measure to be consistent with the circular sampling strategy. However the rarity based measure is augmented with two further operators. Not all rare artifacts should be considered salient; for example, normally invisible JPEG image compression artifacts can sometimes be regarded as salient using rarity alone. A prerequisite for salient artifacts is therefore that they should also be visible, motivating a second perceptually trained operator which estimates the visibility ($P_{visible}$) of image artifacts. The measure is refined by asserting that certain classes of artifact, for example edges or corners, may be more salient than others. This motivates use of a third operator, which users train the system by highlighting artifacts in photographs they regard as salient. Signals corresponding to these artifacts are clustered to produce a classifier which may be applied to artifacts in novel images in order to estimate their potential salience (P_{class}).

This definition allows for a more subjective measure of salience, and holds further advantages in that *classes* of salient features may be trained and classified independently. This allows stroke parameters to vary not only as a function of salience magnitude, but

also allows differentiation of rendering style according to the classification of salient regions (see Figure 4-8).

4.3.1 Determining Pixel Rarity

The first operator is an unsupervised technique for determining pixel rarity. The technique is very similar to that of Section 3.2 in that a model is constructed which encodes the statistical distribution of a set of measures locally associated with each pixel, and the outliers of this distribution are isolated.

For a given pixel $\underline{p} = (i, j)^T$ the operator examines a series of rings of radius ρ, each centred at $(i, j)^T$. The image is uniformly sampled around each ring's circumference at angular positions θ, hence obtaining a discrete signal $\underline{x}(\underline{p}) = (\rho, \theta) \in \Re^3$; colours are in RGB space. This signal is rewritten as a column vector. We have found a sampling rate of 16, and values of ρ ranging from 1 to 3 pixels in increments of 0.5, to yield good results in subsequent processing. As before, an eigenmodel is created from the collection of vectors $\underline{x}(.)$ resulting from each pixel within the image. The Mahalanobis distance $d(.)$ is then computed for all pixels $\mathcal{P}$ in the image.

$$d^2(\underline{x}(.)) = (\underline{x}(.) - \underline{\mu})^T \underline{\underline{U \Lambda U}}^T (\underline{x}(.) - \underline{\mu}) \tag{4.2}$$

The probability of an individual pixel $\underline{q} \in \mathcal{P}$ being rare is then written as a quotient measuring the fraction of the sample density which is less rare than the pixel $\underline{q}$:

$$\mathcal{Q} = \{\underline{r} : d(\underline{x}(\underline{r})) \leq d(\underline{x}(\underline{q})) \wedge \underline{r}, \underline{q} \in \mathcal{P}\} \tag{4.3}$$

$$P_{rare}(\underline{q}) = \frac{\sum_{\underline{p} \in \mathcal{Q}} d(\underline{x}(\underline{p}))}{\sum_{\forall \underline{p} \in \mathcal{P}} d(\underline{x}(\underline{p}))} \tag{4.4}$$

4.3.2 Determining Visibility

The second operator estimates the probability that a local image window contains a perceptually visible signal. The *just noticeable difference* (JND) between colours in RGB format is empirically measured. It is assumed that for each RGB colour $\underline{r}$ there is distance $\tau(\underline{r})$, also in RGB space. Together the colour and the distance specify a sphere of RGB colours $(\underline{r}, \tau(\underline{r}))$. No colour interior to the surface of the sphere can be perceptually discriminated from the centre colour, whilst all exterior colours can be so discriminated. The distance $\tau(\underline{r})$ is one JND at the colour $\underline{r}$. The sphere radius can vary depending on experimental conditions, and after several experimental trials τ emerges as the mean radius accompanied by an associated standard deviation σ. Although this is a simple colour model (an ellipsoid might better model JND surfaces) it has been found to perform satisfactorily, and the reader is referred to [66] for dis-

cussion and experimental details. Similar distance metrics are also described in [175] for luminance. The advantage of the this approach and [175], over other perceptually based colour spaces (such as CIELAB), is that unit distances in JND space correspond to colour distances that are only just discernible by the user.

To evaluate the visibility of artifacts local to a point $\underline{p} = (i, j)^T$, the image is sampled in a manner identical to Section 4.3.1 to obtain a signal (ρ, θ), the differential magnitude of which may be written as:

$$d(\rho, \theta; \underline{p}) = \left| \left| \frac{d\underline{c}(\rho, \theta; \underline{p})}{d\rho} \right|^2 + \left| \frac{d\underline{c}(\rho, \theta; \underline{p})}{d\theta} \right|^2 \right|^{1/2} \tag{4.5}$$

where $\underline{c}(\rho, \theta; \underline{p})$ returns the RGB value of the image at coordinates (ρ, θ) relative to $\underline{p}$. This is, however, not a perceptual distance and the probability $\phi(.)$ that this change is visible is computed as:

$$\phi(\rho, \theta) = erf((d(\rho, \theta) - \tau)/\sigma) \tag{4.6}$$

where τ and σ are the JND and its deviation for the colour sample at $\underline{c}(\rho, \theta)$ in the local window. The reasoning is that if a signal is visible in any ring, then it is visible for the whole ring but not for the whole disc, and so write:

$$P_{visible} = \sum_{\rho=1}^{\rho_{max}} \max(\phi(\rho, \theta)) \tag{4.7}$$

as the probability of the disc being visible. This definition ensures that if a signal grazes the edge of the disc it will register as visible, but not strongly because it will not pass through every ring. If, on the other hand, a signal pases through the centre of the disc then it passes through every ring, and a high visibility is obtained.

4.3.3 Classification of Image Artifacts

The final operator introduces a degree of subjectivity by allowing users to train the system to identify certain classes of low-level artifact as potentially salient.

For a given pixel $\underline{p}$, the image is sampled in an identical manner to that used for determining pixel rarity. However, each ring is treated separately, and so considers the classification of the colour signal $\underline{c}(\theta)$ at constant ρ (this transpires to be more stable than considering the disc as a whole). A feature vector is formed by first differentiating $\underline{c}(\theta)$, using Euclidean distance in RGB space, to obtain a periodic scalar signal $\underline{y}(\theta)$ (Figure 4-2).

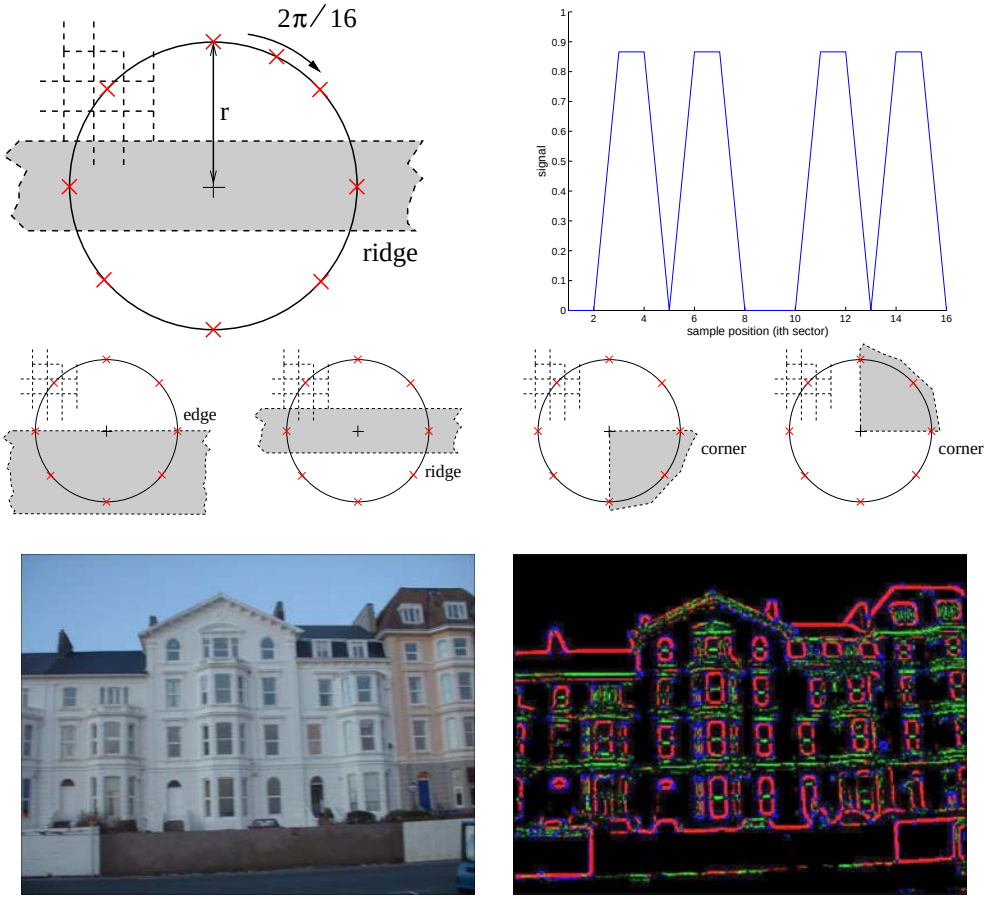


Figure 4-2 Circular descriptors (top left) are used to create signals (top right) from points specified as salient by the user, which are then processed and clustered in a high dimensional space. Features such as ridges, edges and corners (second row) create distinctive spectral signals (third row), which may be used to determine not only the salience of a point, but also its classification type. Bottom row: a photograph and its corresponding salience map with edges in red, ridges in green and corners in blue.

The absolute value of the Fourier components $|F[\underline{y}(\theta)]|$ are computed, normalised to unit power, and the d.c. (zeroth) component dropped. Thus for a given $\underline{y}(\theta)$ a feature is computed as follows:

$$\underline{f}(\omega) = \frac{|F[\underline{y}(\theta)]|}{\left(\sum_\theta |\underline{y}(\theta)|^2\right)^{\frac{1}{2}}} \tag{4.8}$$

$$\underline{f}(\omega) \leftarrow \underline{f}(\omega) \setminus \underline{f}(0) \tag{4.9}$$

by appeal to Parseval's theorem to compute power. Removing the d.c. component is equivalent to subtracting the mean, which makes this feature vector invariant to linear colour shifts. It is also invariant to orientation. Thus $\underline{c}(\theta)$, $\underline{c}(\theta)+\underline{\alpha}$, $\underline{c}(\theta+\beta)$ all map to the same point in feature space. The system has proven to be robust to more general

colour scalings, $\gamma\underline{c}(\theta)$, but cannot be invariant (suppose $\gamma = 0$).

It is these properties that principally motivated choice of circular sampling (after Smith and Brady who advocated the use of circular sampling [144] in their SUSAN system), since the classification of salient artifacts (for example, edges) should be invariant with respect to a cyclic shift of the signal. This contrasts with features based on standard derivative forms, in which edge signals, say, are thinly distributed across feature space (forming a closed one-dimensional manifold).

Training and Classification

Training is a supervised process that occurs over several images, and requires the user to interactively highlight artifacts they regard as salient during a pre-processing step. Moreover, the user may choose a number of classes of artifacts (such as edge, ridge, or corner), and identify a class label with each artifact they highlight. Training therefore results in multiple sets of artifacts, each set containing artifacts of identical class.

To build the classifier each artifact in a given set, k say, is converted into a feature vector as previously described. An estimate of the class conditional density $p(\underline{f}|k)$ for that set of features is then obtained using a Gaussian Mixture Model (GMM), fitted using Expectation Maximisation [41]. A prior, $p(k)$, is also estimated as the expected number of points — the ratio of the number elements in the given set to the number of points in all sets. This enables computation of the posterior likelihood $p(k|\underline{f})$ by appeal to Bayes theorem:

$$p(k|\underline{y}) = \frac{p(\underline{y}|k)p(\underline{y})}{\sum_{j=1}^{N} p(\underline{y}|j)p(j)} \tag{4.10}$$

During painting, classification of a pixel begins by sampling to obtain a new artifact. This is converted to a feature vector and the above probability vector is computed (one element per class). The L_1 norm of this vector is unity, and in fact we can simply add elements to estimate the probability that an artifact belongs to a subset of classes. For each classified pixel we therefore have a probability $p(k|\underline{y})$ of membership to each of the trained classes, and compute P_{class} as the maximum value over all $p(k|\underline{y})$. Later, this classification allows us to vary the stroke rendering style according to the class of salient artifact encountered — see Figure 4-8.

4.3.4 Selection of Scale for Classification

The above approach classifies artifacts at a constant ρ, and so at constant scale. However classification can vary over scale. For example, an artifact classified as an edge at

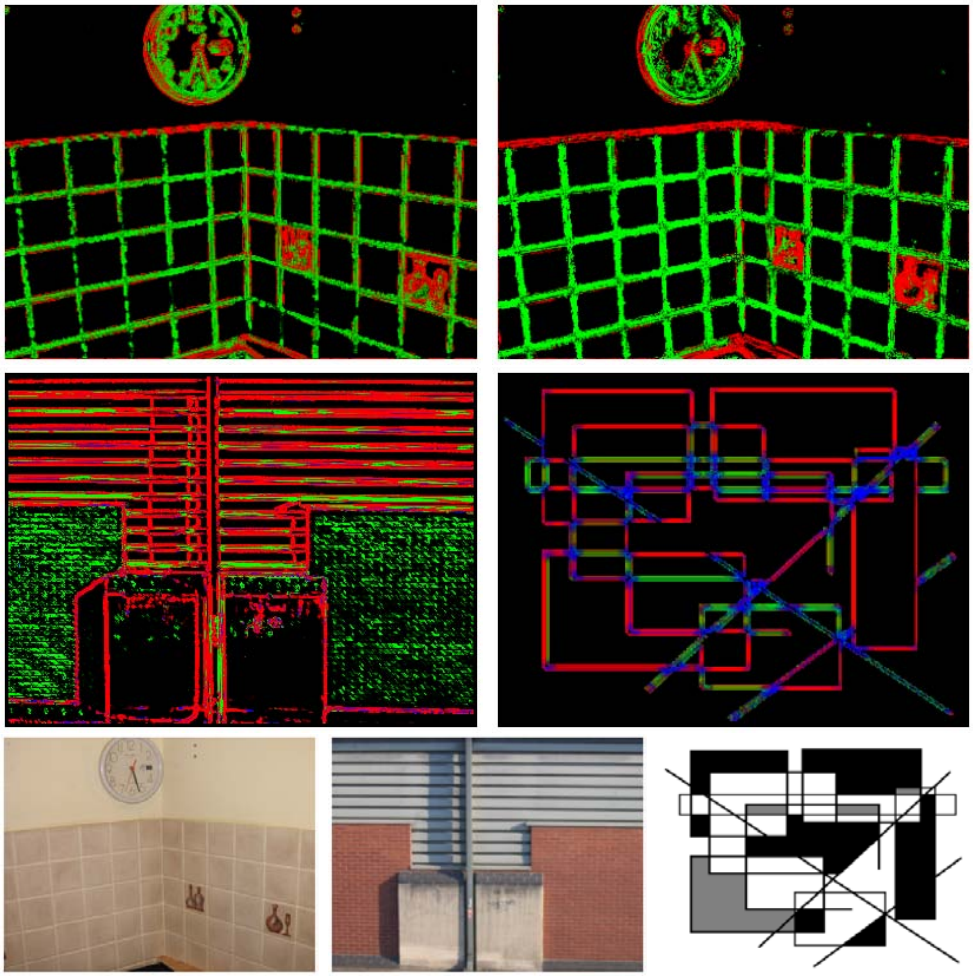


Figure 4-3 Top: Classifier output using a single scale $\rho = 3$ (left) and multiple scales (right). Note that ridges and edges around the tiles are more clearly discriminated using the multi-scale technique. Middle: Further multi-scale classified output on real and synthetic source images. Bottom: Source images.

small scales might be classified a ridge at larger scales; in such cases one would arguably prefer the final classification to be "ridge". By contrast corners remain relatively stable over scale variation, and it transpires that a range of heuristics exist for other such combinations. To opt for the most stable classification over scale is therefore insufficient, but to hard code heuristics specific to edges, ridges etc. is also a poor solution since these are but examples of more general features that users may identify as salient.

Our strategy is to perform the classification of a given point at several values of ρ; again using the range 1 to 3 pixels at increments of 0.5. At each scale we obtain a posterior probability vector $p(k|\underline{y})$, and concatenate these to form a column vector (in effect, a point in a higher-dimensional space that now encapsulates scale information). Since we know the user supervised classification of each point we may again perform clustering

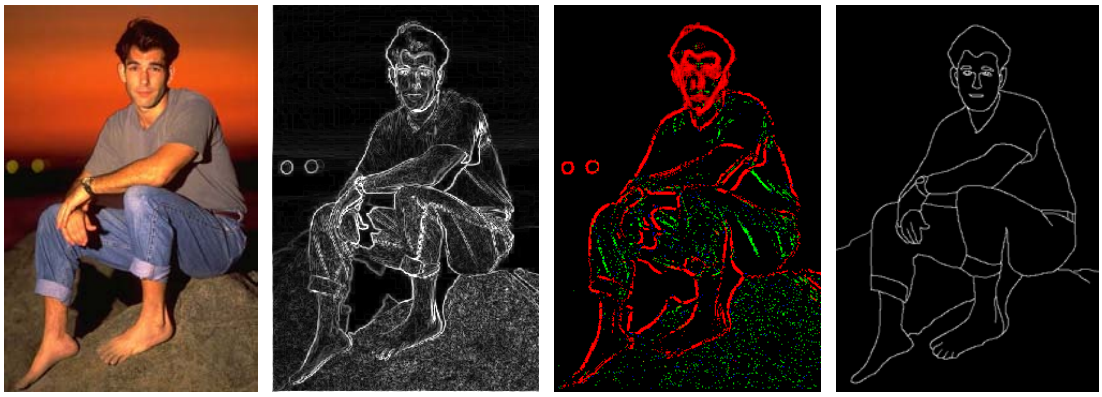


Figure 4-4 Applying the salience measure to an image from test set β, from left to right: original image, Sobel filtered image, multi-scale salience map (red lines indicate salient edges), and a manually specified ground truth of salient edges. The salience measure allows us to discriminate between salient edges, and the ridges and non-salient artifacts that comprise the remainder of the image. Such distinction is not possible using local measures. We later present a painterly rendering of this image in Figure 4-13, in which salient edges (for example the face) are emphasised, and other details (such as the rock texture) are abstracted away.

of salient feature classes, this time by fitting GMMs in this scale-dependent space. The advantage of our approach is that the aforementioned "heuristics" for classification are now implicitly learnt by example.

The extension of the feature classifier to operate at multiple scales impacts both a) the ability to determine salience magnitude, and b) the ability to classify the salient artifacts encountered — both are relevant to our painting process. We performed two experiments to measure each of these impacts respectively.

Experiment 1: Salience magnitude

We trained both the single-scale and multi-scale versions of the salience measure using the image set α (see Figure 4-5). The classes of feature trained on were edges, ridges and corners. In the case of the single-scale measure, we used a disc radius value of $\rho = 3$ for both the training and classification processes; this value has been found to work well over many images (see [66]). Once trained, we applied both measures to a further image set β (distinct from α), to determine salience magnitude within those images ($P_{salient}$, see equation 4.1). We also manually obtained a ground truth salience map for each image in β manually, from a human participant instructed to draw over the important features in the image. The ground truth, and the training, were supplied by the same participant in our experiment (a sample ground truth map is given in Figure 4-4). We compared the output of the salience measures with the ground truth salience map, to determine the performance of each salience measure.

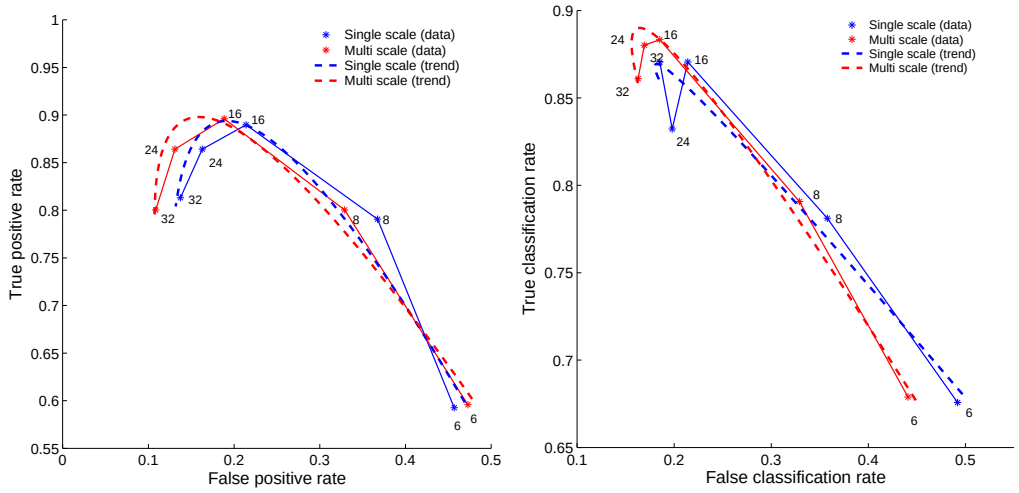
The upper left graph of Figure 4-5 summarises the results of this experiment. The graph shows the ROC curve for the detection of salient pixels, where $P_{salient} \geq 0.5$ (that is, where pixels are more likely to be salient than not). The curve plots sensitivity (true positive rate; the fraction of ground truth salient points classified as salient by the measure), versus specificity (one minus the false positive rate; the fraction of ground truth non-salient points classified as salient). These measures were averaged over the ten image test set (β). A range of radial sampling rates were tested $\{6, 8, 16, 24, 32\}$ for both the single and multi-scale versions of the salience measure. The performances of both measures over the range of sampling rates were plotted with accompanying trend lines.

Both measures perform well at sampling rates of around 16, providing a good compromise between true positives and false positives, and motivating the choice of 16 sampling intervals in our painting process. Both the true positive and false positive rates fall as sampling rate increase beyond the neighbourhood of 16; in general fewer points are identified as salient at these rates. Although the internal representation of the circular signal is superior at higher sampling rates, it is likely that the higher dimensionality of the space in which these signals are distributed inhibits clustering (since both true and false positive signals decline uniformly). Note that although the rate of true positives (pixels correctly identified as salient) does not increase greatly using the multi-scale approach, the false positive rate declines significantly, so improving performance.

Experiment 2: Accuracy of Classification

Our second experiment tested for any specific improvements in classifier accuracy due to the multi-scale extensions. The experiment applied the same single and multi-scale measures, trained over image set α, to the test image set β. However, the ground truth in this experiment was a manually specified map in which corners, edges and ridges had been manually identified (in a similar manner to the classifier training process itself). As with experiment 1, the same person trained the classifier as provided the ground truth data.

Figure 4-5 contains the results of this experiment. The upper right graph shows an ROC curve, which compares the true and false classification rates (with respect to the manually specified ground truth). This graph was generated by forming a "confusion matrix" for each test image. Examples of a confusion matrices for a real image (the kitchen image, Figure 4-3, bottom-left), and a synthetic image (the Mondrian image, Figure 4-3, bottom-right), and also given in Figure 4-5. The values in the confusion matrix represent the proportion of ground truth artifacts (specified by the horizontal,



Conf. matrices (kitchen)

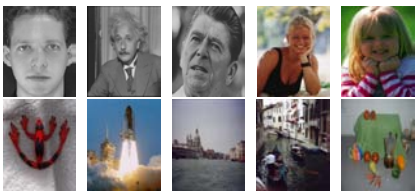
Single scale

	$\mathbf{c}_t$	$\mathbf{e}_t$	$\mathbf{r}_t$
$\mathbf{c}_d$	0.97	0.03	0.00
$\mathbf{e}_d$	0.01	0.81	0.05
$\mathbf{r}_d$	0.02	0.09	0.89

Multi-scale

	$\mathbf{c}_t$	$\mathbf{e}_t$	$\mathbf{r}_t$
$\mathbf{c}_d$	0.97	0.00	0.00
$\mathbf{e}_d$	0.00	0.91	0.04
$\mathbf{r}_d$	0.03	0.03	0.90

Training set (α)



Conf. matrices (synthetic)

Single scale

	$\mathbf{c}_t$	$\mathbf{e}_t$	$\mathbf{r}_t$
$\mathbf{c}_d$	0.89	0.15	0.17
$\mathbf{e}_d$	0.05	0.65	0.15
$\mathbf{r}_d$	0.03	0.14	0.64

Multi-scale

	$\mathbf{c}_t$	$\mathbf{e}_t$	$\mathbf{r}_t$
$\mathbf{c}_d$	0.98	0.18	0.16
$\mathbf{e}_d$	0.01	0.72	0.10
$\mathbf{r}_d$	0.00	0.09	0.73

Test set (β)

Figure 4-5 Comparison of salience measure performance under varying circular sampling rates, using single-scale vs. multi-scale classification (results discussed in Section 4.3.4). Top: Two ROC graphs of the single-scale (blue) and multi-scale (red) measures, showing performance over various sampling rates (labelled). Top-left: True and false positives wrt. detection of salient points. Top-right: True and false classifications rates of identified salient feature points. Bottom left: Confusion matrices when classifying a sample real and synthetic image, with and without multiple scale classification (c, e and r represent rates for corners, edges and ridges respectively. Subscript t indicates ground truth classifications, while subscript d indicates detected classification). Bottom right: The distinct training (α) and test (β) image sets.

subscript "t") deemed to be of a certain classification (specified by the vertical, subscript "d"). Note that columns in the matrix need not sum to unity, since it is possible that a true corner, for example, may not be picked out as salient by the measure (a false negative). Likewise, the rows need not sum to unity since salient artifacts identified by the measure need not have been specified as such in the ground truth (a false positive).

The diagonal of the matrix may be averaged to give the true classification rate for an image. The remainder of the matrix may be averaged to obtain the false classification rate. The values used for the ROC curve correspond to these true and false classification rates, averaged over the entire test image set (β). On average, the multi-scale approach to classification demonstrates superior performance than the single scale approach (exhibiting higher true positive, and lower false negative rates for classification).

Compare the confusion matrices for the single and multiple scale approaches. Performance for both the real and synthetic images is improved in both cases using the multi-scale approach. This may be visually verified by comparing the classification results of the real (kitchen) scene in Figure 4-3 with (upper-right) and without (upper-left) the use of multi-scale information; ridges and edges are discriminated more clearly in the former case. Indeed, the confusion matrices show the greatest performance increase is in the discrimination of these two feature classes — confirming our suggestion that scale is of great importance when deciding between a classification of "edge" or "ridge". The performance increase on the synthetic (Mondrian) image is less pronounced than that of the real scene. We suggest that this is possibly due to the training of the classifier on image set α, which consists entirely of real world images; the frequency characteristics of a synthetic scene may exhibit differences which impede the classification process, although acceptable classifications are produced by the system (see Figure 4-3, middle-right).

4.4 Generating the Painting

We now describe our algorithm for generating a painting from a 2D image. We begin by computing a salience map for the source image using the technique of Section 4.3. An intensity gradient image is also computed via convolution with directional Gaussian derivatives, from which a gradient direction field is obtained by taking arc tangents. In areas of low gradient magnitude the directional field can be unreliable, and so is interpolated smoothly from neighbouring pixels using a distance transform. The source image, direction field and salience map are used in subsequent stages of the painting algorithm. We first describe how individual strokes are placed to create a painting, and then describe the relaxation stage which results in the generation of an "optimal" painting.

4.4.1 Stroke placement algorithm

Our paintings are formed by compositing curved spline strokes on a virtual canvas. We choose piecewise Catmull-Rom splines for ease of control since, unlike β-splines (used in [58, 71]), control points are interpolated. We begin by placing seed points

on the canvas, from which strokes are subsequently grown bidirectionally. Seeds are placed stochastically, with a bias toward placement of seeds in more salient regions. As a heuristic we make provision for a stroke to be seeded at every other pixel; the salience map then governs the distribution of these strokes over the image. In practice we scatter 95 percent of the n strokes in this manner, the remaining 5 percent are scattered uniformly; this prevents holes appearing in areas of relatively low salience (Figure 4-6, right).

Bidirectional Stroke Growth

Strokes are grown to extend bidirectionally from seed points. Each end grows independently until it is halted by one or more preset criteria. Growth proceeds in a manner similar to Hertzmann's algorithm [71] in that we hop between pixels in the direction tangential to their intensity gradient. A history of visited pixels is recorded, and used to form the control points for the spline stroke.

We observe that noise forms a component of any real image, and any locally sampled direction estimate is better regarded as being sampled from a stochastic distribution (Figure 4-6, left). We assume that noise obeys the central limit theorem, and so model this distribution as a zero centred Gaussian, $G(0, \sigma)$; we determine σ empirically (see next subsection). Given a locally obtained gradient direction estimate θ we select a hop direction by adding Gaussian noise $G(0, \sigma)$. The magnitude of the hop is also Gaussian distributed, on this occasion $G(\mu', \sigma')$, both μ' and σ' being inversely proportional to the local value of the precomputed salience map. Provided that a preset minimum number of hops have been executed, the growth of a stroke end is halted when either the curvature between adjacent pixels, or the distance (in *JND* space) between the

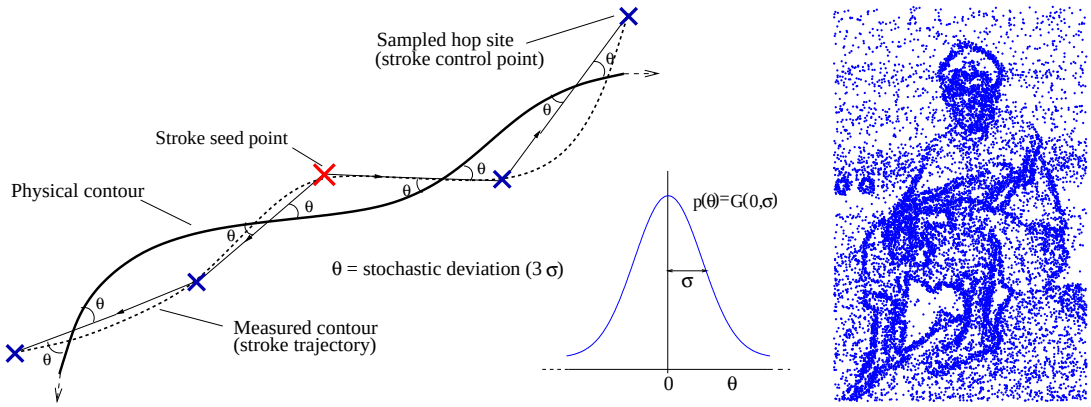


Figure 4-6 Left: Illustrating the stochastic growth of strokes from a seed pixel. We choose strokes with hop sites which minimise our objective function, under the constraint that hop angles are drawn from the distribution $p(\theta) = G(0, \sigma)$. Right: The salience-biased stochastic distribution of strokes, corresponding to the painting of Figure 4-13.

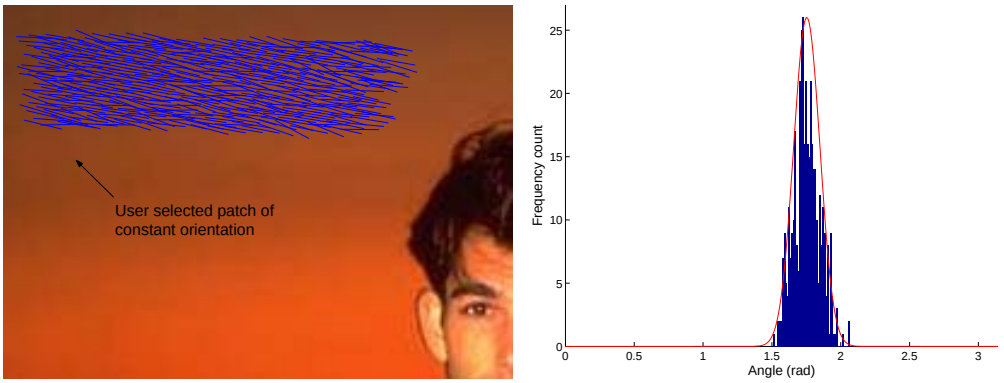


Figure 4-7 The system estimates the standard deviation of image noise by accepting a "ground truth" from the user. The user selects a region of the image within which they deem paint strokes should be orientated in a uniform direction (left). The gradient direction of pixels within this region is then measured (blue). We have observed that such image noise tends to obey the central limit theorem, and so is well modelled by a Gaussian (above, the mean direction is approximately 88°, standard deviation $\sigma \approx 5°$.

colour of the pixel to be appended and the mean colour of visited pixels, exceeds a threshold (preset at 3 JNDs).

This method initially yields a sub-optimal trajectory for the stroke with respect to our measure, described in Section 4.1. For a "loose and sketchy" painting this is often desirable (see Figure 4-8), but for painting styles exhibiting tighter stroke placement, trajectories must be closer to the optimal. The degrees of freedom resulting from each of the many stochastic hops combine to create a range of stroke loci, at least one of which will result in the maximal conservation of salient detail. The combination of these optimally positioned strokes comprises the optimal painting, and it is by means of breeding the fittest paintings to create successively superior renderings, that we search for such a painting via GA relaxation in Section 4.4.2. Our relaxation strategy is thus able to approach more globally optimal stroke trajectories, and these can out-perform trajectories based purely on local estimates of direction.

Calibration for image noise

The choice of σ significantly influences the stroke growth and relaxation process. A value of zero forces degeneration to a loose and sketchy painterly system, whilst a high value will lengthen the relaxation process unnecessarily and also may introduce unnecessary local minima. We propose a one time user calibration process to select this σ, typically performed during the training step of the perceptual salience measure.

The user is asked to draw around sample image regions where direction of image gradient is perceived to be equal; i.e. along which they would paint strokes of similar

orientation. This results in multiple samples of the gradient components, from which we compute gradient direction by taking arc-tangents. We have observed the natural distribution of these values to be Gaussian, confirming our assumption that such image noise obeys the central limit theorem (Figure 4-7). We therefore take the mean angle $\mu(.)$ as the common tangential angle. Similarly, we compute the unbiased standard deviation of the set of measured tangential angles which subsequently becomes the σ parameter for bidirectional stroke growth. We assume σ to be equal for all angles.

We typically obtain very similar σ values for similar imaging devices, which allows us to perform this calibration very infrequently. A typical σ ranges from around 2 to 5 degrees, with the larger deviations being attributed to digital camera devices (possibly as artifacts of lower CCD quality or JPEG compression). This variation allows between twelve and thirty degrees of variation per hop for noisy images which, given the number of hops per stroke, is a wide range of loci for a single stroke. Such observations add credence to our argument for the need of a relaxation process taking into account image noise; potentially large variations in stroke placement due to uncompensated image noise are likely to produce inaccurate stroke placements in single-pass (i.e. single iteration) painterly rendering systems [58, 71, 103, 140].

Rendering and Differential Styles

Stroke rendering attributes are set automatically as a function of *stroke salience*, taken as the mean value of the salience map under each control point. By default, stroke thickness is set inversely proportional to salience. Stroke colour is uniform and set according to the mean of all pixels encompassed in the footprint of the thick paint stroke. During rendering, strokes of least salience are laid down first, with more salient strokes being painted later. As with our previous algorithm (Section 3.3) this prevents strokes from non-salient regions encroaching upon salient areas of the painting.

The ability of our salience measure to differentiate between classes of salient feature also enables us to paint in context dependent styles. For example, we have described how we may discriminate between artifacts such as edges and ridges (Section 4.3.3). In Figure 4-8 we give an example of a painting generated by our system, in which the classification probability of a feature is used as a parameter to interpolate between three rendering styles (parameter presets) *flat*, *edge* and *ridge*. For the flat preset, rendering takes the default form described in the previous paragraph. For edges and ridges, the luminance of strokes is heavily weighted to create dark, outline strokes. In the case of edges, thickness of strokes is also boosted to create thick outlines — while with ridges the thickness is greatly reduced to produce thin wispy strokes. The σ value for ridges is also boosted to reduce accuracy and produce "sketchy" strokes. Since these

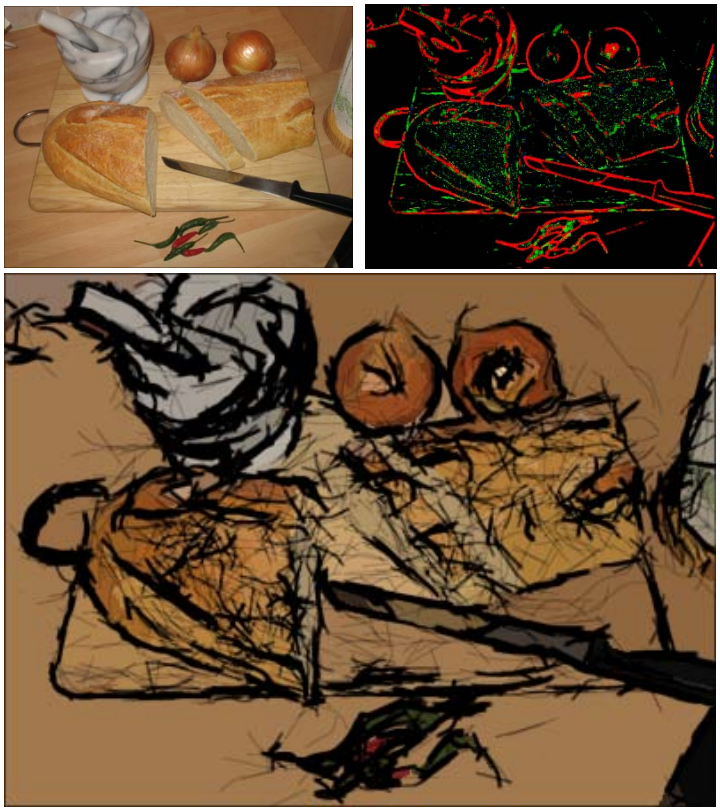


Figure 4-8 Top: a still-life composition and corresponding salience map. Bottom: the above rendered as a loose and sketchy painting, exhibiting differential stroke rendering styles determined by local feature classification. Edges are drawn with hard, precise thick strokes; ridges with a multitude of light, inaccurate strokes. Rendered prior to the relaxation step of Section 4.4.2.

preset rendering parameters (thickness, luminance decay, etc.) all vary by continuous multiplicative factors, interpolation between the presets according to the classification probability vector is straightforward. This process also adds semantic to the rendering, in that we render ridges as single strokes, rather than as two separate edge strokes. To the best of our knowledge the rendering of paintings in differential styles via an automated heuristic is a novel contribution to AR.

4.4.2 Relaxation by Genetic Algorithm

Genetic algorithms simulate the process of natural selection by breeding successive generations of individuals through the processes of cross-over, fitness-proportionate reproduction and mutation. In our algorithm such individuals are paintings; ordered lists of strokes and their associated attributes. Recall that we define the fitness of a given painting as proportional to the correlation between the salience map of the original image and level of (high frequency) detail within the corresponding painting.

Fitness and Selection

We begin by seeding an initial generation of paintings using the approach described in Section 4.4.1 and then enter the iterative phase of the genetic algorithm (Figure 4-11). First we perform evaluation; the entire population is rendered, and edge maps of each painting produced using by convolution with Gaussian derivatives, which serve as a quantitative measure of local, high frequency detail. The scale of the Gaussian is carefully chosen so as to smooth the fine inter-stroke edges, and prevent these influencing this detail measure in the painting. The generated edge maps are then compared to the precomputed salience map of the source image. The mean squared error (MSE) between maps is used as the basis for fitness measure $F(.)$ for a particular painting; the lower the MSE, the better the painting:

$$F(I, \psi) = 1 - \frac{1}{N} \sum |S(I) - E(\Psi(I, \psi))|^2 \tag{4.11}$$

The summation is over all N pixels in source image I. $\Psi(.)$ denotes our painterly process, which produces a rendering from I and an ordered list of strokes ψ (ψ corresponds to an individual in the population). Function $S(.)$ signifies the salience mapping process of Section 4.3, and $E(.)$ the process of convolution with Gaussian derivatives to produce an edge map. In this manner, individuals in the population are ranked according to fitness. The bottom ten percent are culled, and the top ten percent pass to the next generation. The latter heuristic promotes convergence; the fittest individual in successive generations must be at least as fit as those in the past. The top ninety percent are used to produce the remainder of the next generation through simulated natural selection. Two individuals are selected stochastically with a bias to fitness, and bred via cross-over to produce a novel offspring for the successive generation. This process repeats until the population count of the new generation equals that of the current.

Cross-over

We now describe the cross-over process in detail (Figure 4-9, below). Two difference images, A and B, are produced by subtracting the edge maps of both parents from the salience map of the original image, then taking the absolute value of the result. By computing the binary image $A > B$, and likewise $B > A$, we are able to determine which pixels in one parent contribute toward the fitness criterion to a greater degree than those in the other. Since the atoms of our painterly renderings are thick brush strokes rather than single pixels, we perform several binary dilations to both images to mark small regions local to these "fitter" pixels as desirable. A binary AND operation between the dilated images yields mutually preferred regions. We mask these conflicting regions with a coarse chequerboard texture (of random scale and phase offset) to decide

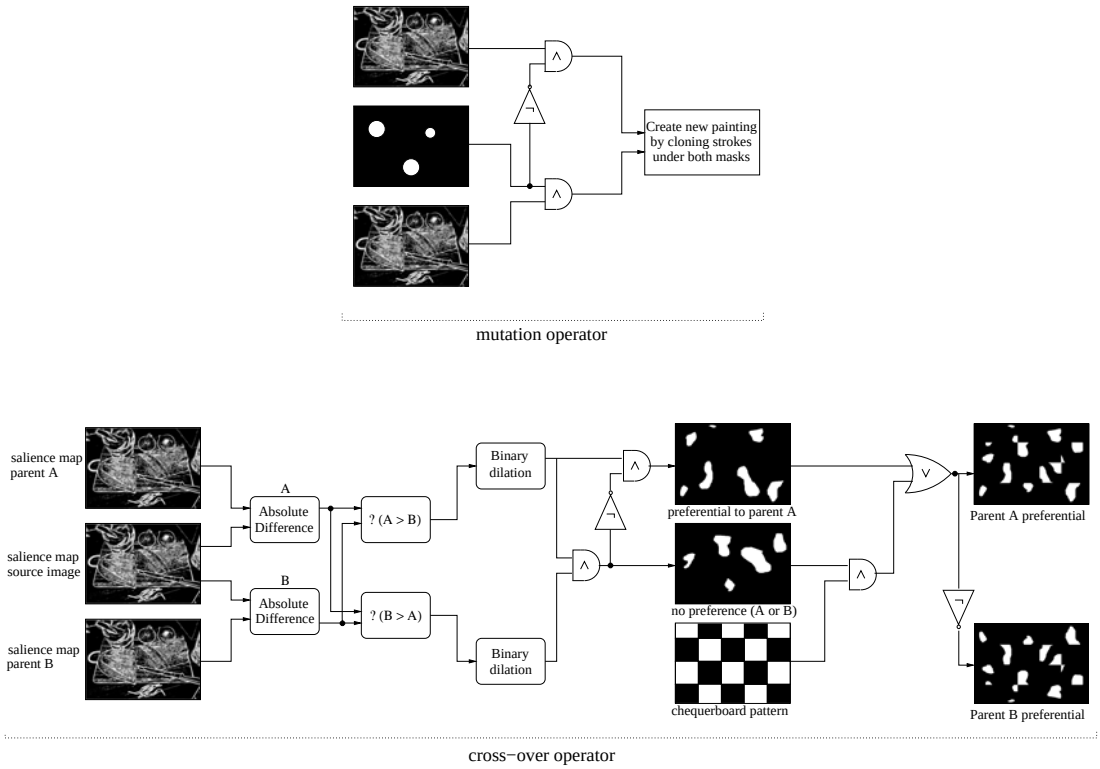


Figure 4-9 Genetic operators: the mutation and cross-over operators used during the relaxation process.

between parents in an arbitrary fashion. Finally, strokes seeded within the set regions in each parent's mask are cloned to create a new offspring.

Mutation

Finally, when a bred individual passes to a successive generation it is subjected to a random mutation. A new "spare" painting is synthesised (though never rendered), and a binary mask produced containing several small discs scattered within it. The number, location and radius of the discs are governed by random variates. Strokes seeded within set regions of the binary mask are substituted for those in the spare painting; the spare painting is then discarded. In our implementation large areas of mutation are relatively rare, averaging around four hundredths of the image area per painting.

Termination

The relaxation process runs until the improvements gained over the previous few generations are deemed marginal (the change in both average and maximum population fitness over sliding time window fall below a threshold Δ), at which point the search has settled into a minima (see Figure 4-20) of sufficient extent in the problem space that escape is unlikely (Figure 4-10, right). The fittest individual in the current pop-

ulation is then rendered and output to the user. Typically executions run for around one to two hundred iterations for values of σ between two and five degrees, which we found to be a typical range of standard deviations for image noise (see Section 4.4.1). Forcing larger values of σ can result in convergence but, we observe, at the cost of an exponential increase in execution time (Figure 4-10, left).

Parallel Implementation

In practice, evaluation is the most lengthly part of the process and the rendering step is farmed out to several machines concurrently. In our implementation we distribute and receive paintings via the Sun RPC interface, using XDR to communicate over a small heterogeneous (Pentium/UltraSPARC) compute cluster running on our local network. Each painting in the population represents one "job" of work. Execution is blocked until the entire population of paintings are rendered, that is, all jobs are complete. In order to maintain acceptable execution speeds it is therefore important to assign jobs to machines in an efficient manner.

The time between sending a job to a helper (slave) machine, and the return of results from that machine is recorded. A mean execution time is thus maintained and updated for each machine throughout the rendering of a population. In our original implementation, jobs were simply farmed out to the first available machine. However in a heterogeneous system, machines may be of varying speeds and capabilities, and a single relatively slow machines can severely impact the performance of the whole

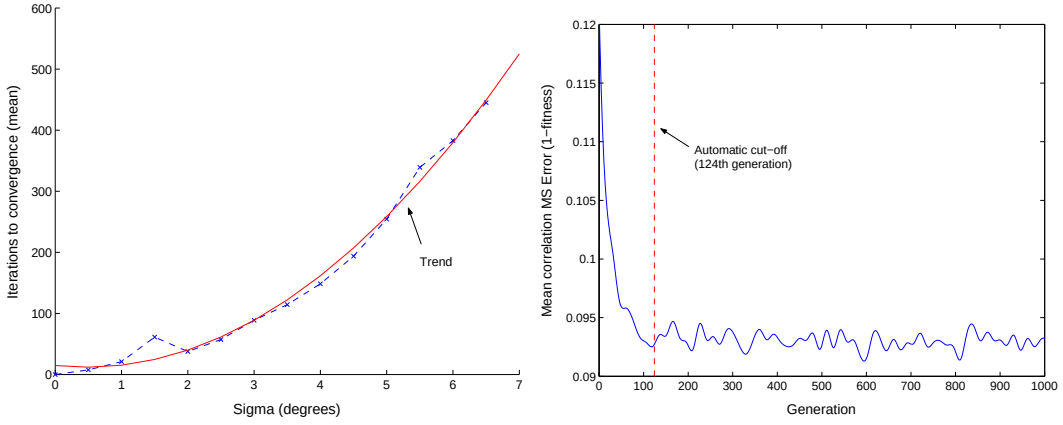


Figure 4-10 Illustrating GA termination. Left: Increasing the value of σ allows processing of noisier image at the cost of an exponentially increasing execution time. Right: the MSE (inverse fitness) averaged over the entire population. Automatic algorithm termination was suppressed, and the GA forced to run for 1000 iterations (data has been sub-sampled for presentation). This is representative of relaxation process' behaviour, and there is little advantage in exploring additional local problem space after the termination point.

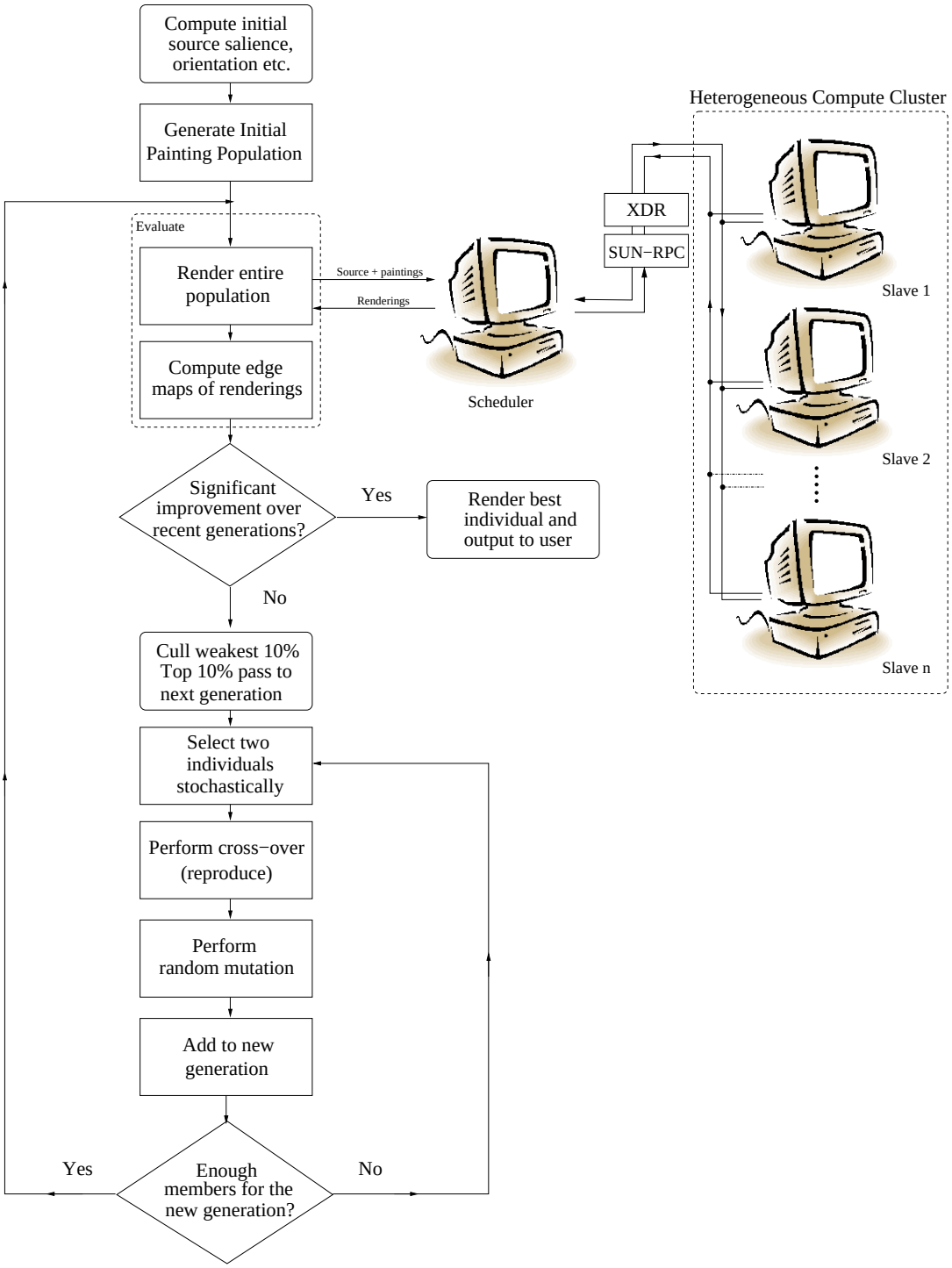


Figure 4-11 Illustrating flow of control in the genetic algorithm. The population evaluation stage is inherently parallel and rendering is farmed out to a distributed compute cluster.

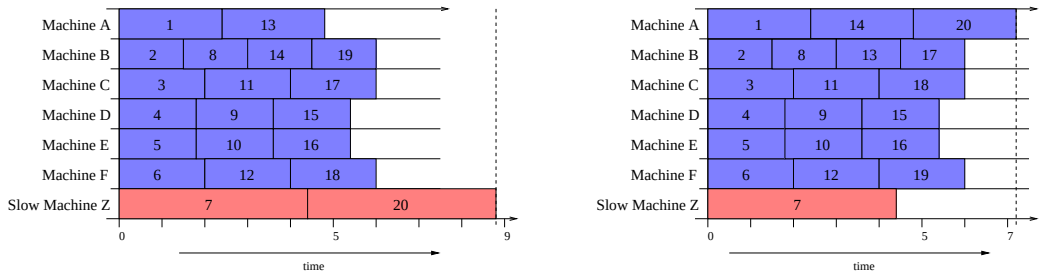


Figure 4-12 Illustrating the scheduling algorithm governing the distributed rendering of a 20 painting generation. Left: The simplistic "first free machine" scheduling strategy becomes time inefficient toward the last few jobs in the population. Right: Sending jobs to machines that we predict will complete them soonest maintains time efficiency over the entire population.

cluster using this strategy (Figure 4-12). Instead, our strategy is to predict which of the machines is likely to finish the pending job first, and then queue the job on that machine. This may involve queueing a job on a faster, busy machine, when a slower machine is idle. Note that for the first iteration of rendering we will not hold an estimate for the speed of machines, and so on this iteration all are assumed to render at equivalent speed.

The typical time to render a fifty painting generation at high (1024×768) resolution is approximately five minutes over six workstations. Relaxation of the painting can therefore take in the order of hours, but significant improvements in stroke placement can be achieved, as can been seen in Figure 4-16 and the accompanying video. The overhead of our task scheduler is low, and processing time falls approximately linearly as further machines of similar specification are added to the cluster.

4.5 Rendering and Results

We have generated a number of paintings to demonstrate application of our algorithm using the source photographs of Figure 4-17. The reader is also referred back to the dragon (Figure 4-1) and sketchy still-life (Figure 4-8) paintings, presented *in situ*. As a note, we have found that running the paintings through a standard sharpening filter [145] can assist presentation of our paintings on the printed page, and have applied such a filter to all paintings presented in this chapter.

The painting of the model in Figure 4-13a converged after 92 generations. Thin precise strokes have been painted along salient edges, while ridges and flats have been painted with coarser strokes. Observe that non-salient high-frequency texture on the rock has been abstracted away, yet tight precise strokes have been used to emphasise

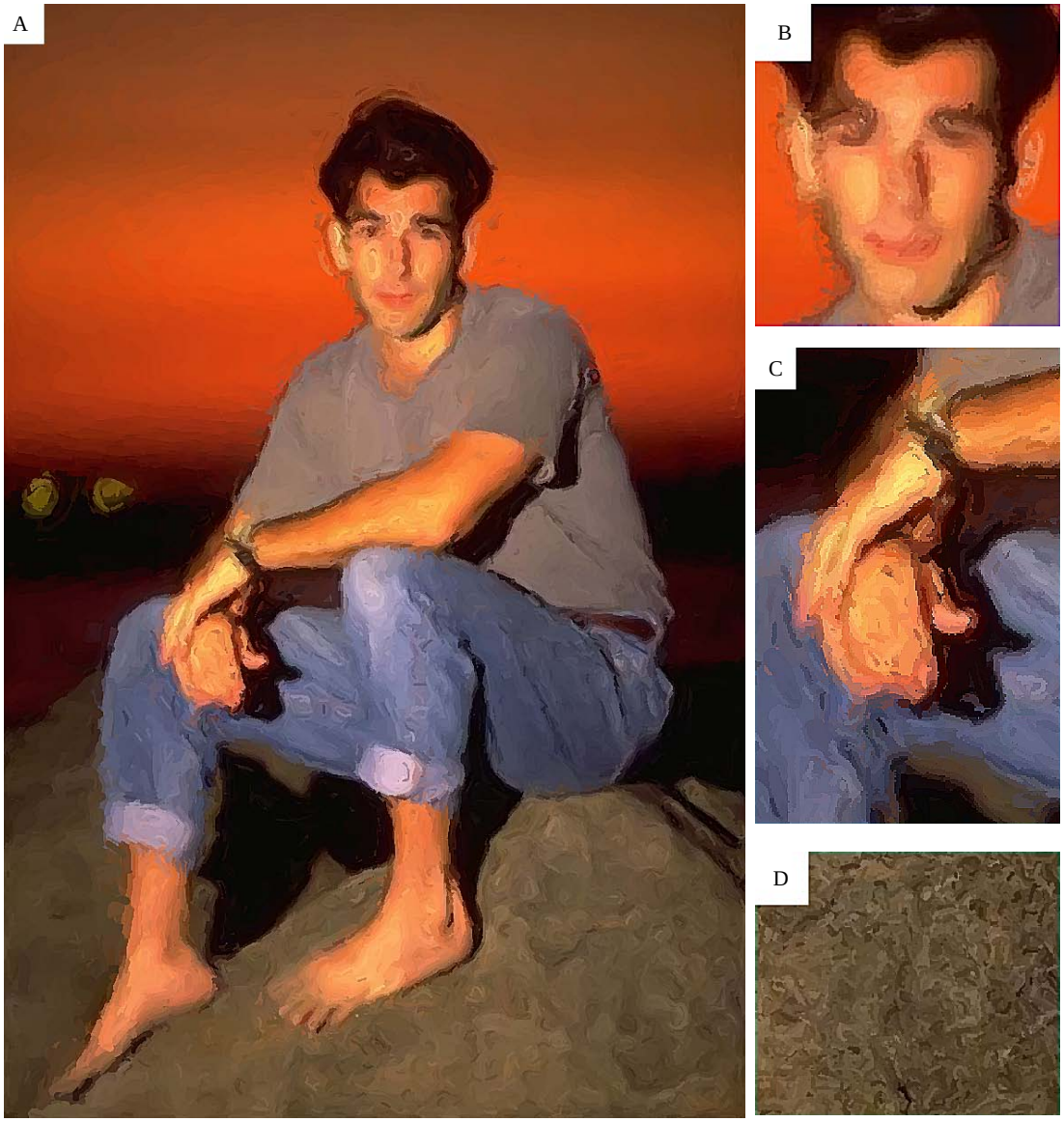


Figure 4-13 Man on rock: (a) final painting after convergence using our proposed method, close-up of hands in (c). (b) example of the face rendered with insufficient emphasis (d) example of rock texture rendered with too great an emphasis. Refer to the text of Section 4.5 for full explanation of (b) and (d), and how our salience adaptive painting avoids such difficulties.

salient contours of the face. In the original image the high frequency detail in both regions is of similar scale and edge magnitude; existing painterly techniques would, by contrast, assign both regions equal emphasis. With current techniques, one might globally increase the kernel scale of a low-pass filter [71] or raise thresholds on Sobel edge magnitude [103] to reduce emphasis on the rock (Figure 4-13c). However this would cause a similar drop in the level of detail on the face (Figure 4-13b). Conversely, by admitting detail on the face one would unduly emphasise the rock (Figure 4-13d). In our method, we automatically differentiate between such regions using a perceptual

Figure 4-14 Pickup truck after convergence. Observe salience adaptive emphasis of sign against background in (a). We have manually dampened the salience map in (b) to cause greater abstraction of detail; compare stroke placement here with the remainder of the car body. Original photo courtesy Adam Batenin.

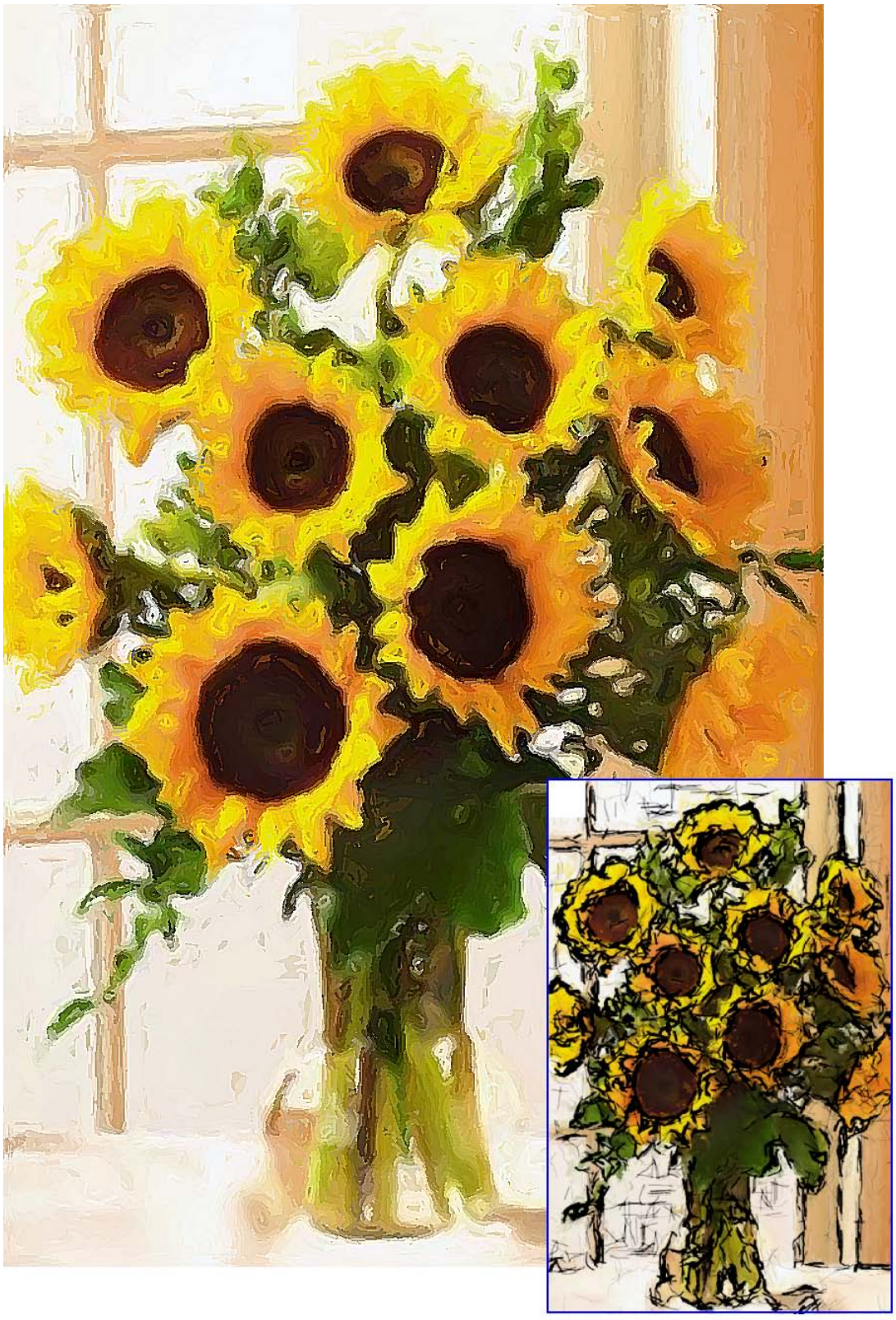


Figure 4-15 Sunflowers after convergence. Inset: a sketchy version of the sunflowers in the style of Figure 4-8, prior to relaxation.

Figure 4-16 Relaxation by genetic algorithm. Detail in the salient region of the "dragon" painting sampled from the fittest individual in the 1st, 30th and 70th generation of the relaxation process. Strokes converge to tightly match contours in salient regions of the image thus conserving salient detail (an animation of this convergence has been included with the electronic supplementary material in Appendix C).

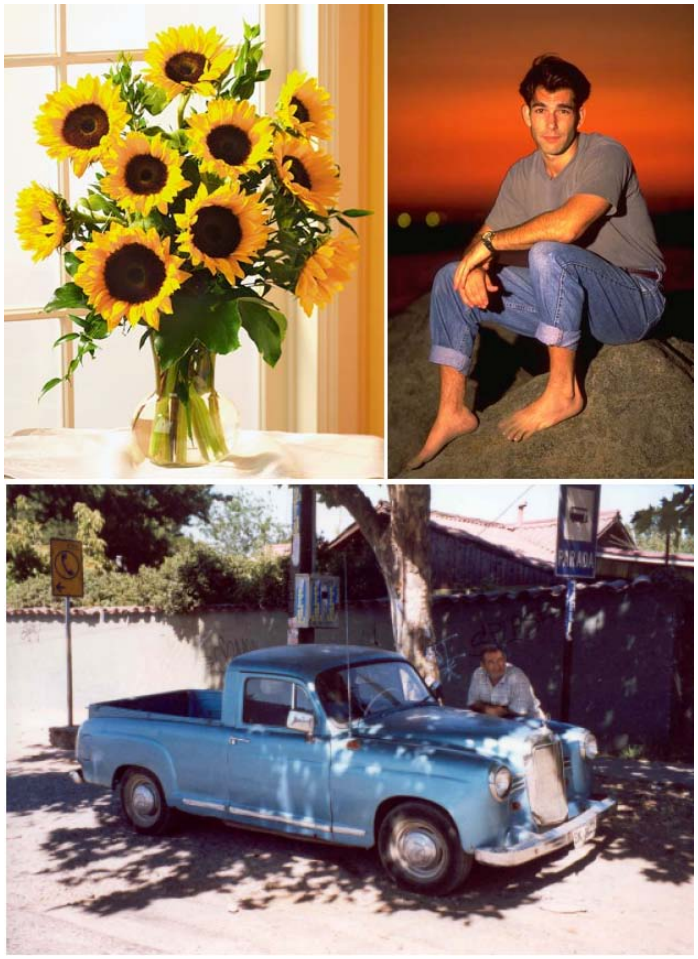


Figure 4-17 Source images used to generate the paintings of Figures 4-13, 4-14, and 4-15.

Figure 4-18 Left: The distinction between salient and non-salient detail can not be made by current image-space painterly techniques which use locally measures such as variance or the Sobel operator (rendered using [103]). All high frequency detail is afforded equal emphasis. Right: "Sheaf-binder" [Van Gogh, 1889]. The majority of Van Gogh's work is characterised by his use of elegant, curved sweeping strokes; our system is capable of producing strokes in ostensibly similar styles (Figure 4-15).

Figure 4-19 Detail from Figure 4-14, region A. Left: Section of the original photograph exhibiting non-salient background texture (shrubbery) and salient foreground (sign-post). Middle: All fine detail is emphasised using existing automatic approaches (here we use [103] as a representative example), which place strokes using only spatially local information. In this image, the high frequency detail of the background leaf texture has caused strokes to be clipped at edges, tending the process back toward photorealism. However attempts to mitigate this effect, by reducing the edge threshold for clipping, will further degrade salient detail on the sign. Right: Using our adaptive approach, salient detail is conserved, and non-salient detail is abstracted away.

salience map (Figure 4-4) — contrast this with the Sobel edge field in the same figure, in which no distinction between the aforementioned regions can be made.

We present a still-life in Figure 4-15 which achieved convergence after 110 generations. Inset within this figure we present a similar painting prior to relaxation, demonstrating differential rendering style as strokes with a high probability of being edges are darkened to give the effect of a holding line. Further examples of level of detail adaptation to salience are given in Figure 4-14. In region A, observe that the salient 'phone sign is emphasised whilst non-salient texture of the background shrubbery is not (also see Figure 4-19 for a enlarged, comparative example). For the purposes of demonstration we have manually altered a portion of salience map in region B, causing all detail to be regarded as non-salient. Contrast stroke placement within this region with that on the remainder of the car body. Variations in style may be achieved by altering the constants of proportionality, and also thresholds on curvature and colour during stroke placement. Paintings may be afforded a more loose and sketchy feel by increasing the halting threshold Δ and so decreasing the number of relaxation iterations; essentially trading stroke placement precision for execution time. A similar trade-off could be achieved by manually decreasing the σ parameter (Figure 4-10, left).

All of our experiments have used populations of fifty paintings per generation. We initially speculated that population level should be set in order of hundreds to create the diversity needed to relax the painting. However it transpires that although con-

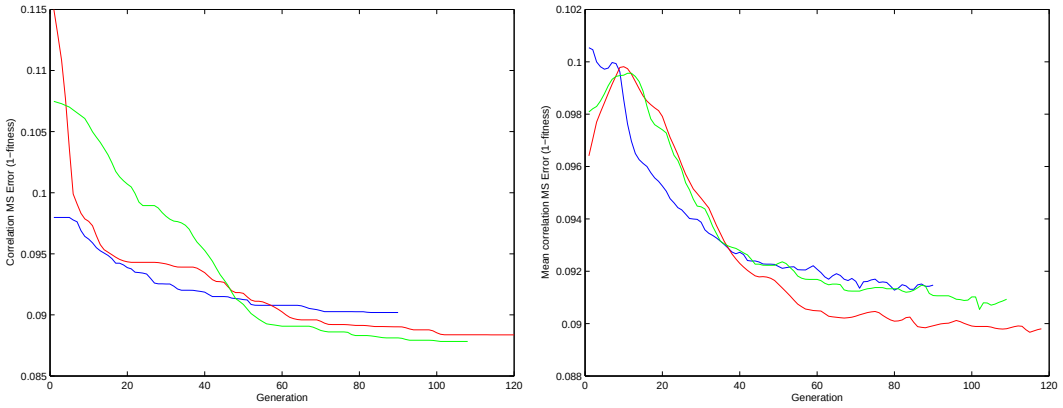


Figure 4-20 Three representative runs of the relaxation process; blue corresponds to the model (Figure 4-13), red the dragon (Figure 4-1) and green the abbey (Figure 4-21). The left-hand graph shows the MSE (inverse fitness) of the fittest individual over time, the right-hand graph shows the same measure averaged over each generation.

vergence still occurs with such population limits, it requires, on average, two to three times as many iterations to achieve. Such interactions are often observed in complex optimisation problems employing genetic algorithms [56]. We conclude that the diversity introduced by our mutation operator (Section 4.4.2) is sufficient to warrant the lower population limit.

During development we experimented with a number of alternative GA propagation strategies. Originally we did not carry the best individuals from the previous generation directly through to the next. Instead, the search was allowed to diverge, and a record of the "best painting so far" was maintained separately. This resulted in a more lengthly relaxation process, which sometimes produced marginally fitter paintings than the current method. However the marginal aesthetic benefit that resulted did not seem to warrant the large increase in run-time. Similar results were observed using another early strategy; if, after a number of generations, we observe no change in fitness, then we may have reached a plateau in the problem space. In such circumstances the probability of large scale mutation occurring was gradually increased until the search escaped the plateau. Again, this caused lengthly execution times for which the pay off in terms of quantitative change in the fitness function, and qualitative improvement in aesthetics, was marginal.

4.6 Summary and Discussion

In this chapter we have presented a novel automatic algorithm for creating impasto style painterly renderings from photographs. This work builds upon the pilot single-pass salience adaptive painterly technique introduced in the previous chapter, which

aimed to control emphasis in a painting through use of a global salience map. The principal disadvantage of single-pass algorithms is that strokes are placed only once and then remain fixed; image noise local to stroke location can produce inaccuracies in the rendering. By contrast, relaxation based painting techniques can iteratively tweak stroke parameters to converge the painting toward some procedurally defined "goal" state. Such an approach is of considerable benefit, since in the absence of a specific procedural model of the painting process, we can still approach a specific goal; in effect, we can specify where we want to be, and provide only vague instructions on how to get there.

We drew upon our observations of artistic practice in Chapter 3 to define the degree of optimality for a painting to be measured by the correlation between the salience map of the original image and level of detail within the corresponding painting. To search for this optimal painting, we developed a novel GA based relaxation technique. Our choice of GAs for relaxation was motivated by their superiority to other search techniques for high dimensional problem spaces with many local optima. Furthermore, although it is difficult to explicitly model the complex relationships between stroke parameters during creation of a painting, goal driven stochastic processes such as GAs are known to perform acceptably well without such models. The inherent parallelism of the population evaluation step also permits acceptable performance when combined with distributed computation techniques (Section 4.4.2).

The paintings generated by our algorithm serve to reinforce our argument that a higher level of spatial analysis is of benefit to AR, in terms of enhancing quality of output. The GA relaxation framework and salience measure serve as a more sophisticated means to the same end; that of generating paintings with a focus and emphasis driven by global image importance rather than simple local frequency content. For example, the salience adaptive discrimination between level of detail on the rock, and the model's face (see Figure 4-13), or the sign-pose and shrubbery (see Figure 4-19), would not have been possible using local, frequency driven approaches to AR.

We are not the first to describe relaxation approaches to painterly rendering. As we observe in the review of Section 2.4.1, the discussion of such approaches dates back to Haeberli [62], and the first algorithmic solutions was described in [155]. However the relaxation techniques that exist contain objective functions that specifically aim to maximise the preservation of high frequency detail in the final painting. We discussed the disadvantages of this spatially local approach to rendering in Chapter 3.

Our algorithm operates by compositing curved spline brush strokes, which are fitted

Figure 4-21 Bath Abbey after 110 generations of the relaxation process. The darker strokes outlining the arch and other salient edges are generated by interpolating between a default and "edge" preset according to the probability of salient artifacts being edges (see Section 4.4.1). Original photograph inset, top-left.

by sampling local gradient orientation. We argued that orientation measurements are better modelled as points sampled from a Gaussian distribution, due to the presence of noise (a component of any real image). Our algorithm is provided with a calibrated estimate of the level of noise in the image, and so can produce a stochastic distribution of potential values for measurements, such as orientation, taken from the image. Our relaxation based approach varies stroke parameters within this distribution to converge toward our optimality criterion. This explicit modelling of image noise is a novel contribution to AR. Furthermore, the estimate of noise (σ) need only be approximate and should be greater than or equal to the true level of noise in the image. The penalty for over-estimating noise is an unnecessarily long execution time (Figure 4-10, left), although this often still results in convergence unless a gross over-estimation of σ has been made. The penalty for under-estimating noise is that the painting may not converge to the optimal, since stroke attributes may not be allowed to vary sufficiently to reach the best configuration. Although the placement of strokes is governed primarily by the relaxation process, each stroke has a guaranteed number of hop sites, and hop lengths in inverse proportion to the salience of the image regions they cover. Selection of reasonable values for minimum hop length and count prevents the output of system tending toward photorealism, should such an attraction evolve.

Further contributions of our method were the use of a user trained measure of salience (which the author played a collaborative role in developing, see Section 4.3). The advantages of using this method were two-fold. First, the salient artifacts encountered were classified into one of several trained categories (such as edges, or ridges). This allowed us to automatically differentiate stroke rendering style over the image; a novel contribution. Second, the measure represented a more subjective approach to problem of estimating image salience.

There are a number of potential avenues for development of the GA based relaxation process. An interesting investigation might be to allow users, rather than our objective function, to choose survivors for each iteration of the GA, and to investigate whether such a system might assist the exploration of alternative rendering styles. However this would undoubtedly increase the demands on user time, which are currently very low for our system. We might also choose to introduce novel additional constraints into the relaxation process. For example, explicitly including an information theoretic constraint controlling the density of stroke coverage for a region; this is currently a stochastic decision performed once for each painting generated. More generally, we believe the most interesting direction for development would be to explore alternative objective functions for painting using constraints. A natural extension of the system would be to devise a library of constraints, by studying a range of artists' painting styles. The distributed GA painting system could then be applied to fit a particular painting model (selected by a user) to a photograph, thus painting that photograph in one of a range of established artistic styles. Depending on the commonalities identified between styles, this "paint by model fitting" approach could potentially yield a single versatile system capable of generating many artistic styles within a single parameter space.

Simplifying assumptions have been made in the salience measure. For example, the decision to use spherical JND surfaces in the visibility operator, and the use of a single Gaussian for clustering during rarity were made on the grounds of unattractive computational complexity during clustering. Work continues at Bath to investigate higher level models for the classifier of Section 4.3.3, which take into account neighbouring regions to produce a more context sensitive classification of artifacts.

As regards rendering, we might choose to texture strokes to produce more realistic brush patterns, although this should be a post-processing step so as not to introduce undue error in the comparison of salience maps. Many techniques apply texture mapping to strokes [71, 103, 140], and a bump mapping technique was also proposed in [73]. Highly realistic volume based hairy brush models have recently been proposed [177]

which could be swept along the Catmull-Rom spline trajectories generated by our algorithm. However, we have concentrated primarily upon stroke placement rather than media, and we leave such implementation issues open. We believe the most productive avenues for future research will not be in incremental refinements to the system, but rather will examine alternative uses for salience measures in the production of image-space artistic renderings.

The electronic Appendix C contains high resolution versions of all paintings presented in this chapter (see `/paintings`).

Part III

Video Paintbox: the benefits of Higher level temporal analysis

Chapter 5

Foreword to the Video Paintbox

In this chapter we give a brief introduction to the "Video Paintbox", a novel system for generating AR animations from video. This system is developed throughout the remainder of Part III (Chapters 6, 7, and 8); each chapter dealing with one of the three individual subsystems comprising the Video Paintbox. In this introduction we recall our discussion in Chapter 2 which highlighted the limited scope of current research into video driven AR, and use these observations to motivate development of the Video Paintbox. We give a brief overview of the paintbox's core subsystems, their principal contributions, and the arguments they make for higher level temporal analysis in AR.

5.1 Introducing the Video Paintbox

We observed in our literature review that although image-space AR techniques have become increasingly common in the literature, the problem of synthesising artistically rendered animations from video remains sparsely researched. A very small number of algorithms [75, 96, 103] exist for automatically producing such animations, and each of these propose a solution to the very specific problem of creating a painterly effect in video whilst maintaining temporal coherence. We argue that the current literature is limited in its approach to video driven AR animation in a number of respects.

- First, all algorithms which address the problem of automated painterly animation process video on a per frame sequential basis, rendering each frame of animation using data from only the current and previous frame. Processing begins by directly applying a standard, static AR algorithm to the first frame of video. Most methods [96, 103] then translate brush strokes from frame to frame using a vector field produced using an optical flow technique [7], which supplies a per pixel inter-frame motion estimate. Although the level of temporal coherence is, of course, determined by the optical flow technique implemented, optical flow does

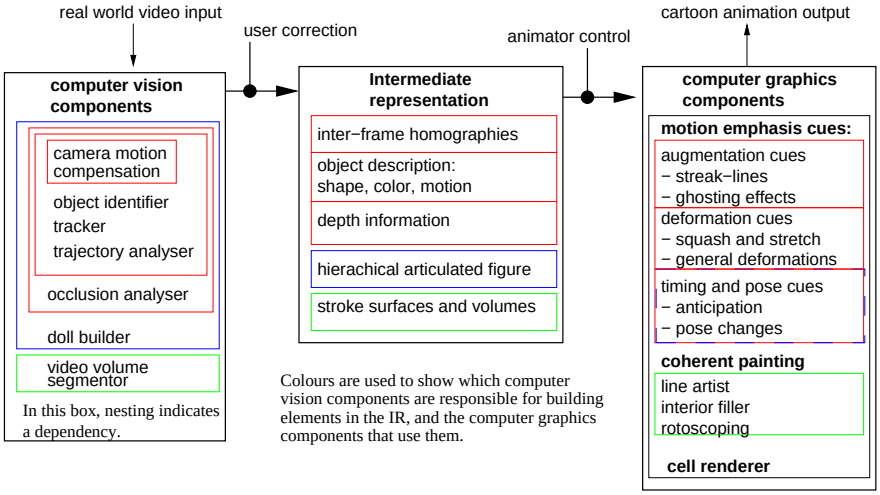


Figure 5-1 A schematic overview of the Video Paintbox. Video is parsed through application of Computer Vision techniques, to generate intermediate representations of the video content. These representations are then rendered, subject to user parameterisation, to generate the final animation. In the diagram, red items indicate use by the visual motion cue subsystem (Chapter 6), blue items the time and pose motion cue subsystem (Chapter 7), and green items the Stroke Surface (coherent artistic video shading) subsystem (Chapter 8).

not generally perform well in the presence of occlusion, varying lighting conditions, or within flat untextured regions [54]. Unfortunately, a general solution to the problem of creating an error-free motion field between two images is unknown to Computer Vision; the problem can be reposed as the under-constrained general "correspondence problem"[1] [145]. Estimation errors quickly accumulate and propagate forward through the video sequence, manifesting themselves as an uncontrolled "swimming" in the animation. Exhaustive manual correction is typically required to compensate for these errors and improve temporal coherence [61]. A faster, but less accurate, approach to producing painterly video was proposed by Hertzmann and Perlin [75], where an RGB differencing operation is used to "paint over" areas of the frame that are deemed to have moved, i.e. exhibit large inter-frame pixel differences. Under this algorithm, lighting changes or fast camera movements cause behaviour to degenerate to an independent painting of each video frame, reintroducing associated difficulties with swimming.

- Second, the scope of research into video driven AR has so far been very limited. Temporally coherent video rendering is a difficult problem, worthy of study. However, the concentration of research effort on this issue has been at the expense of other, in our opinion, equally worthy investigations. In particular, the characterisation and emphasis of movement which lends "personality" to an animation is at

[1] Simply stated, this is the problem of finding which pixels in one image correspond to which in a second image. For motion analysis, this assumes that motion is such that both frames overlap.

least as important as artistic style in which it is drawn [98]. However no existing literature addresses the problem of emphasising motion within a video sequence; in fact it could be argued that the efforts of previous video AR research have concentrated upon mitigating against, rather than emphasising, the presence of motion in the image sequence.

The temporally local nature of current video driven AR methods is a factor which limits both the temporal coherence of animations, and the potential artistic styles in which video may be rendered. Current techniques for artistic rendering of image sequences operate on a per frame sequential basis; none of these existing techniques achieve satisfactory levels of temporal coherence without manual correction. Furthermore, although simplistic motion effects are possible using low-level temporal processing (motion blur is one such operation — Figure 2-5), we show later in this chapter that most traditional cartoon style motion cues such as streak-lines, or squash and stretch deformation effects, demand a higher level of temporal analysis for application to video. This is because such cues not only depict and emphasise historic motion, but also anticipate future movements. A much larger temporal window than two consecutive frames is therefore required to correctly analyse motion and trajectories (in addition, consider that some cues emphasise acceleration, and so require at least three temporal samples for measurement via a finite difference scheme).

In Part III of this thesis we develop the Video Paintbox; a novel system for the automated synthesis of AR animations. Our paintbox accepts real world video as input and outputs an animation rendered into some artistic style influenced by the setting of user parameters (Figure 5-1). The results of the algorithms which comprise the Video Paintbox serve as evidence to our argument that higher level temporal analysis is of benefit to video driven AR; improving the diversity of animation rendering style (facilitating novel non-realistic temporal effects such as cartoon-like motion emphasis) and enhancing the temporal coherence of AR animations.

Throughout our Video Paintbox there is an underlying assumption that motion within the video sequence is smooth, that is, free from scene changes (for example cuts and cross-fades). A plethora of algorithms exist (mostly colour-histogram based) which can segment a video sequence into such "cut-free" chunks; we use [181]. This segmentation is a pre-processing step to our method.

Internally the system comprises three principal components: an *intermediate representation* provides an abstract description of the input video clip. This is parsed from the video by a *computer vision* component which is responsible for video analysis. This representation, together with any system parameters, forms the input to the *computer*

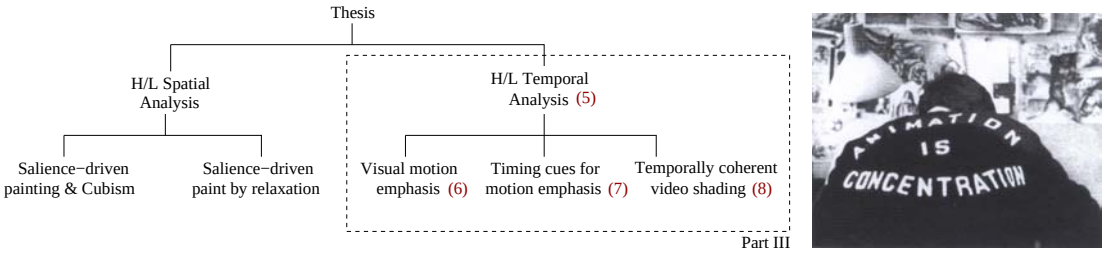


Figure 5-2 Left: Reminder of the thesis structure, in particular the layout of Part III. Chapter numbers are written in red. Right: Our motivation is to create an automated tool which alleviates much of the tedium of the animation process — yet retains the ability for the animator to direct the creative process at a high level (Photo credit: Frank Herrmann, reproduced from [169]).

graphics component, which is responsible for all decisions relating to synthesis, subject to animator control. The intermediate representation is in fact a suite of integrated representations (Figure 5-1). Each component of the representation describes the video content in a way that is useful to one or more tasks. Although we wish for a high degree of automation within our system, we by no means wish to completely remove the animator from the artistic process. Rather our aim is to develop tools for animators with which they may use to render artistic effects in video, by creatively directing at a high level. For example, in our system the animator might point to an object and specify it to be rendered in watercolour, or move with streak-lines — whereas in the traditional animation process (Figure 5-2, right) the animator would operate at a lower level, laying down the individual strokes to create either effect.

Our work is motivated by a desire to render 2D image sequences in cartoon-like styles, a problem that decomposes into two separable sub-goals, addressing each of the aforementioned benefits respectively:

1. Emphasising motion in the image sequence
2. Producing temporally coherent shading effects in the video

Chapter 6 is primarily concerned with the former sub-goal, and specifically with the problem of visually depicting motion through artistic rendering. Chapter 7 is also concerned with motion emphasis, though using a different class of motion cues which modify a subject's pose over time to emphasise its motion. We are able to incorporate both classes of motion cue into a single framework within our Video Paintbox. The artistic rendering of motion within a video sequence is a novel contribution to AR, and one that implies interesting new application areas for Computer Vision.

Chapter 8 is concerned with the latter sub-goal of producing temporally coherent artistic shading in video. We describe "Stroke Surfaces"; a novel spatiotemporal approach to

processing video, and demonstrate that by analysing the video sequence at a temporally higher level, as a block of frames (rather than on a per frame, per pixel basis as with current methods) we are able to generate artistically shaded video exhibiting a high degree of temporal coherence. This framework integrates with our motion emphasis techniques to form the completed Video Paintbox, capable of producing cartoon-style animations directly from video.

A show-reel (with commentary) which summarises the Video Paintbox's capabilities is included in Appendix C, see `/videos/vpshowreel`. A systems paper giving an overview of the Video Paintbox accompanies this show-reel, see `/papers/VideoPaintbox.pdf`.

Chapter 6

Cartoon-style Visual Motion Emphasis from Video

In this chapter we describe the first of three subsystems comprising the Video Paintbox; a framework capable of rendering motion within a video sequence in artistic styles, specifically emulating the visual motion cues commonly employed by traditional cartoonists. We are able to synthesise a wide range of augmentation cues (streak-lines, ghosting lines, motion blur) and deformation cues (squash and stretch, exaggerated drag and inertia) in this single encompassing framework. We also address problematic issues such as camera motion and occlusion handling. Effects are generated by analysing the trajectories of tracked features over multiple video frames simultaneously. Emphasis of motion within video, rather than mitigation against its presence for the purpose of maintaining temporal coherence, is a unique contribution to AR facilitated by the higher temporal level of analysis performed by the Video Paintbox.

6.1 Introduction

The character of an animation is influenced not only by the artistic style in which it is drawn, but also by the manner in which the animated objects appear to move. Although a number of video driven non-photorealistic animation techniques have been proposed to address the former issue, the problem of automatically emphasising and stylising motion within a 2D video sequence has not yet been addressed by the AR community (Section 2.5.3). In this chapter we describe a novel framework capable of rendering motion within a video sequence, and demonstrate the automated synthesis of a wide range of visual motion emphasis cues commonly used by animators[1].

[1]This work has been published in [27] and has been submitted to the journal "Graphical Models". An earlier version of this work was published as [25].

Figure 6-1 Examples of motion cues used in traditional animation (top) and the corresponding cues inserted into a video sequence by our system (bottom). From left to right: two examples of streak-line augmentation cues, the latter with ghosting lines. Two examples of deformation cues; squash and stretch and suggestion of inertia through deformation.

Animators have evolved various ways of visually emphasising the characteristics of a moving object (Figure 6-1). Streak-lines are commonly used to emphasise motion, and typically follow the movement of the tip of the object through space. The artist can use additional "ghosting" lines that indicate the trailing edge of the object as it moves along the streak-lines. Ghosting lines are usually perpendicular to streak-lines. Deformation is often used to emphasise motion, and a popular technique is squash and stretch in which a body is stretched tangential to its trajectory, whilst conserving area [98]. Other deformations can be used to emphasise an object's inertia; a golf club or pendulum may bend along the shaft to show the end is heavy and the accelerating force is having trouble moving it. The magnitude of deformation is a function of motion parameters such as tangential speed, and of the modelled rigidity of the object. In this chapter we develop a system which processes real video to introduce these motion cues; comparative examples of hand drawn and synthesised cues are given in Figure 6-1.

Recall that our aim is to develop tools for animators with which they may interact at a high level, to render artistic effects in video. Users interact with our visual motion emphasis framework by drawing around objects in a single frame and specifying the type of motion emphasis they would like to apply to those objects. This approach makes the problem of tracking features tractable from a Computer Vision point of view, since the general segmentation problem prohibits an automated solution to bootstrapping our feature tracker. It is questionable whether we would prefer to use an automated segmentation technique for this task, were one to exist. Animation by its very nature is driven by an individual's style and creative flair. Although putting traditional animation techniques into practice can be repetitive and tedious work (exhibiting potential

for automation), the imagination and direction governing the choice of techniques to apply is better considered to be a function of individual taste. The same idea motivates the well established use of teams of lesser skilled "inbetweeners" in commercial animation. These teams interpolate the key-frames of more experienced animators to cut production times — a practice pioneered back in the 1940s. Thus we find it desirable to allow artistic influence in our system through high level interaction; for example specifying which objects should undergo motion emphasis, and with which techniques. By contrast, automation should free the animator from the mechanics of how, for example, such objects deform — these remain largely constant between animations, as evidenced by the many animator's handbooks (for example, [169]) which offer guidance on motion emphasis techniques. However, these effects are not rigidly defined; their behaviour can be modulated to fit the animators needs. Likewise, our system allows performance parameters to be set on motion cues, for example constants controlling rigidity in a squash and stretch deformation. We describe these parameters with their respective effects *in situ*.

6.2 Overview of the Subsystem

We now describe the major components of the subsystem responsible for synthesising visual motion cues, leaving detailed explanation to subsequent sections of the chapter. The subsystem has two major components: the Computer Vision component which is responsible for tracking motion of features (e.g. arm, leg, bat or ball), camera motion compensation, and depth ordering of features; and the Computer Graphics component, responsible for the generation of motion cues, and their rendering at the correct depth. We wish for minimal user interaction with the Computer Vision component, which must be robust and general; currently users draw polygons in a single frame to identify features which are then tracked automatically. In contrast, the user is given control over the graphics component via a set of parameters which influence the style in which the motion cues are synthesised.

6.3 Computer Vision Component

The Computer Vision component is responsible for tracking features over the video sequence. A camera motion compensated version of the sequence is first generated; we do not desire camera motion to bias the observed trajectories of subjects. Features are then tracked over this compensated sequence. By analysing occlusion during tracking we determine a relative depth ordering of features, which is later used in the rendering stage to insert motion cues at the correct scene depth.

6.3.1 Camera Motion Compensation

The nature of the motions we wish to emphasise often demands a moving camera to capture them, so necessitating a camera (ego) motion compensation step as the first operation in our subsystem. Ego-motion determination allows the camera to move, to pan, and even compensates for camera wobble. We model inter-frame motion by a homography, and perform compensation by applying a robust motion estimation technique initially proposed by Torr [158].

A number of interest points are first identified within two given images (I and I') using a variant of the Harris corner detector, refined to operate with sub-pixel accuracy (details of this refinement are given in Appendix A.3). Correspondence between interest points is estimated using cross-correlation of local image signal — however this produces many erroneous correspondences. We thus search for an initial approximation to the homography $\underline{\underline{H}}$, between corresponding interest points, using RANSAC [47] to minimise the forward-backward transfer error $E(\underline{\underline{H}})$:

$$E(\underline{\underline{H}}) = \sum_{i=1}^{4} |\underline{\underline{H}}\underline{p}_i - \underline{q}_i| + |\underline{\underline{H}}^{-1}\underline{q}_i - \underline{p}_i| \tag{6.1}$$

where $\underline{\underline{H}}$ is a putative homography proposed by a RANSAC iteration, and $\underline{p}_{[1,4]}$, $\underline{q}_{[1,4]}$ are four pairs of potentially corresponding feature points chosen at random from the complete set of correspondences identified during cross-correlation. On each iteration $\underline{\underline{H}}$ is computed by solving a homogeneous linear system, comprising two linear equations for each inhomogeneous point correspondence $\underline{p}_i = (x_i, y_i)^T \mapsto \underline{q}_i = (x'_i, y'_i)^T$:

$$x'_i(h_7 x_i + h_8 y_i + h_9) - h_1 x_i - h_2 y_i - h_3 = 0 \tag{6.2}$$

$$y'_i(h_7 x_i + h_8 y_i + h_9) - h_4 x_i - h_5 y_i - h_6 = 0 \tag{6.3}$$

Where $h_{[1,9]}$ represent the nine elements of the matrix transformation for the planar homography (see equation 3.22). The homogeneous system is solved directly in matrix form, using standard linear methods (SVD) to obtain $h_{[1,9]}$:

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x'_1x_1 & x'_1y_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & y'_1x_1 & y'_1y_1 & y'_1 \\ & ... & & & & & & ... & \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x'_4x_4 & x'_4y_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & y'_4x_4 & y'_4y_4 & y'_4 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ ... \\ h_8 \\ h_9 \end{bmatrix} = 0 \tag{6.4}$$

Having established an estimate for $\underline{\underline{H}}$ using RANSAC, we refine that estimate using a

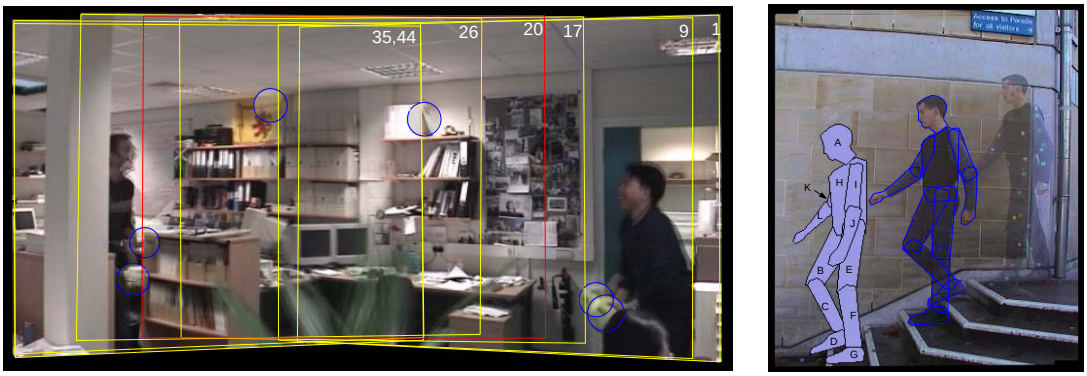


Figure 6-2 Left: The camera compensated *VOLLEY* sequence sampled at regular time intervals. The camera view-port at each instant is outlined in yellow, the tracked feature in blue (see `videos/panorama` in Appendix C). Right: *STAIRS* sequence. (Top) markers are required to track this more complex subject but are later removed automatically (middle). Recovery of relative depth ordering permits compositing of features in the correct order (bottom); labels **A**–**K** correspond to the graph of Figure 6-3.

Levenburg-Marquadt iterative search[2] (after [154]) to minimise the mean squared RGB error $E'(\underline{\underline{H}})$ between image pixels which overlap when transformed by $\underline{\underline{H}}$ (we write the subset of overlapping pixels within image I, as X):

$$E'(\underline{\underline{H}}) = \frac{1}{|X|} \sum_{\underline{x} \subseteq X} \left| I'(\underline{\underline{H}}\underline{x}) - I(\underline{x}) \right|^2 \tag{6.5}$$

Frames are projected to a single viewpoint via homography to produce a motion compensated sequence in which the tracking of features is subsequently performed. This process assumes a static background against which camera motion may be judged. This background should contribute over fifty percent of Harris points in the imaged scene; in practice this is rarely a difficult target to obtain, since the background is often reasonably well textured and occupies much of the imaged area.

6.3.2 Tracking Features through the Compensated Sequence

The general problem of tracking remains unsolved in Computer Vision and, like others, we now introduce constraints on the source video in order to make the problem tractable. Users identify features by drawing polygons, which are "shrink wrapped" to the feature's edge contour using snake relaxation [167], in an identical manner to Section 3.4.1. We assume contour motion may be modelled by a linear conformal affine transform (LCAT) in the image plane, a degenerate form of the affine transform consisting of 4 parameters (uniform scale, orientation, and spatial position — represented by the 4-tuple $\underline{h}$(t)=[s(t), θ(t), x(t), y(t)]). This allows planar motion plus scaling

[2]The Levenburg-Marquadt algorithm is a general purpose, non-linear smooth function optimiser — described in [123].

of features. Variation of these 4 parameters is assumed to well approximate a second order motion equation over short time intervals:

$$\underline{h}(t+\delta) \approx \underline{h}(t) + \frac{\mathrm{d}\underline{h}}{\mathrm{d}t}\delta + \frac{\mathrm{d}^2\underline{h}}{2\mathrm{d}t^2}\delta^2 \tag{6.6}$$

In homogeneous coordinates, a transform from time t to t' is instantiated by:

$$\underline{\underline{M}}_{tt'}(s,\theta,x,y) = \underline{\underline{T}}(x,y)\underline{\underline{R}}(\theta)\underline{\underline{S}}(s) \tag{6.7}$$

A feature, F, comprises a set of points whose position varies in time; we denote a point in the t^{th} frame by the column vector $\underline{x}_t = (x,y)_t^T$. In general these points are not pixel locations, so we use bilinear interpolation to associate a colour value, $I(\underline{x}_t)$ with the point. The colour value comprises the hue and saturation components of the HSV colour model; we wish to mitigate against variations in luminance to guard against errors introduced by lighting changes. Although the LCAT can be derived from two point correspondences we wish to be resilient to outliers, and therefore seek the LCAT $\hat{\underline{\underline{M}}}_{tt'}$ which minimises $E[.]$:

$$E[\underline{\underline{M}}_{tt'}] = \frac{1}{|F|}\sum_{i=1}^{|F|} |I(\underline{x}_{t'}^i) - I(\underline{\underline{M}}_{tt'}\underline{x}_t^i)|^2 \tag{6.8}$$

where the tt' subscript denotes the matrix transform from frame t to t'. In a similar manner to camera motion correction (Section 6.3.1), the transformation $\underline{\underline{M}}_{tt'}$ is initially estimated by RANSAC, and then refined by Levenburg-Marquadt search.

Note that this process assumes the animator is able to isolate an unoccluded example of the feature, in a single video frame; a reasonably light restriction given that there is no requirement for all features to be bootstrapped from within the same individual frame.

By default, several well distributed interest points are identified automatically using the Harris [68] operator (see sequences *CRICKET*, *METRONOME*). In some cases a distinctively coloured feature itself may be used as a marker (see *VOLLEY*, Figure 6-2). In more complex cases where point correspondences for tracking can not be found (perhaps due to signal flatness, small feature area, or similarity between closely neighbouring features), distinctively coloured markers may be physically attached to the subject and later removed digitally (see *STAIRS*, Figure 6-2). In these cases the Harris interest points are substituted for points generated by analysis of the colour distribution in a frame (see Appendix A.5) for full details).

Noise and Occlusion

So far we have overlooked the impact of noise and occlusion upon our tracker. The "best" $\underline{\underline{M}}_{tt'}$ measured will be meaningless in the case of total occlusion, and exact in the case of neither noise nor occlusion. The likelihood L_t of the feature being occluded at any time t may be written as a function of detected pixel error:

$$L_t = exp(-\lambda E_{[\underline{\underline{M}}_{tt'}]}) \tag{6.9}$$

where λ is the reciprocal of the average time an object is unoccluded (this is a data dependent constant which can be varied to tune the tracker), and $E[.]$ is the pixel-wise difference defined in equation 6.8. Thus at any time t we know the estimated LCAT $\underline{\underline{M}}_{tt'}$, and the confidence in that estimate $C_t = 1 - L_t$. We have incorporated a Kalman filter [90] into our tracker which processes video frames sequentially, accepting the LCAT estimate and confidence values over time, and outputting a best estimate value for each instant's LCAT. The Kalman filter "learns" a model of the tracked feature's motion over time; this motion model is factored in to the "best estimate" to output at time t. The credence given to the model, versus that given to the observation, is governed by the confidence in our observation at that time, C_t. We give further details of our Kalman tracking approach in Appendix A.4, but summarise that central to the filter is a 3×3 state transition matrix $\underline{\underline{S}}$ which represents a second order motion equation in the 4D parameter space of the LCAT. The coefficients $\underline{c}(.)$ of this equation are continuously updated, according to the confidence of measurements submitted to the filter, to learn the immediate history of the feature's motion. For example, in the case of a zero confidence ($C_t = 0$) input, the filter will produce an output based entirely on this learnt motion:

$$\begin{aligned} \underline{c}(t') &= \underline{\underline{S}}\underline{c}(t) \\ \underline{c}(t') &= \begin{bmatrix} 1 & t'-t & (t'-t)^2/2 \\ 0 & 1 & t'-t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{h}(t) \\ \frac{\mathrm{d}\underline{h}(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}^2\underline{h}(t)}{\mathrm{d}t^2} \end{bmatrix} \end{aligned} \tag{6.10}$$

Observe that if confidence C_t is low, then the object is likely to have been occluded, because we cannot match the template to the image. If $C_t > 0.5$ we say the object is more likely occluded than not. If the likelihood of occlusion is low, $C_t < 0.5$, and the modelled and observed LCAT disagree then the estimate, due to the second order motion equation, is likely to be in error. We can therefore use a low C_t to detect collisions (see Section 6.4.2).

The addition of a Kalman filter to the system permits the tracker to maintain a lock on features moving under occlusion providing that motion, whilst occluded, approximately

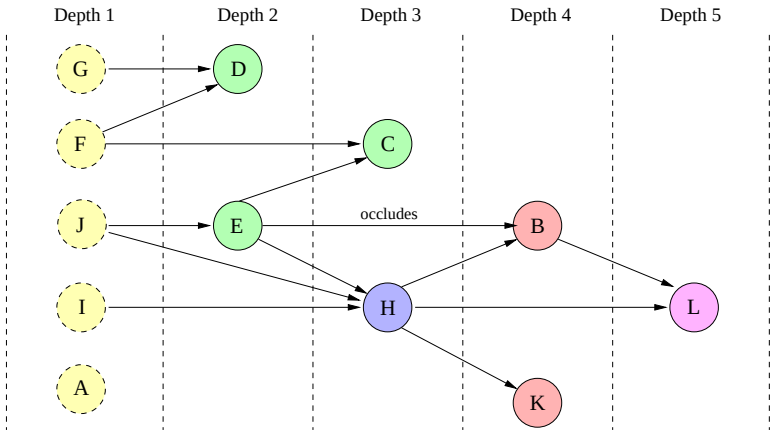


Figure 6-3 An occlusion graph constructed over 150 frames of the *STAIRS* sequence, node letters correspond with Figure 6-2. Dotted lines indicate unoccluded nodes.

follows the second order model estimated by the Kalman filter; this approach has been found sufficiently robust for our needs. However, although this lock is sufficient to re-establish accurate tracking immediately following an occlusion, the predicted LCAT during occlusion is often inaccurate. We refine the tracked motion by deciding at which time intervals a feature is occluded and interpolating between the parameters of known LCATs immediately before and after occlusion. Knowledge of the correct positions of features during occlusion is important when rendering as, although a feature will not be visible while occluded, any applied motion cues may be. Occlusion is also used to determine relative feature depth, and a knowledge of collision events is important later for squash-and-stretch effects.

6.3.3 Recovering Relative Depth Ordering of Features

We now determine a partial depth ordering for tracked features, based on their mutual occlusions over time. The objective is to assign an integer value to each feature corresponding to its relative depth from the camera. We introduce additional assumptions at this stage: 1) the physical ordering of tracked features cannot change over time (potential relaxation of this assumption is discussed later in Section 6.5) ; 2) a tracked feature can not be both in front and behind another tracked feature; 3) lengthly spells of occlusion occur due to tracked features inter-occluding.

For each instance of feature occlusion we determine which interest points were occluded by computing a difference image between tracked and original feature bitmaps. A containment test is performed to determine whether occluded points lie with the bounding contour of any other tracked features; if this is true for exactly one feature, then the occlusion was caused by that feature. We represent these inter-feature relationships as a directed graph, which we construct by gathering evidence for occlusion over the

whole sequence. Formally we have a graph with nodes $G_{1..n}$ corresponding uniquely with the set of tracked features, where $G_i \mapsto G_j$ implies that feature G_i occludes G_j (Figure 6-3). Each graph edge is assigned a weight; a count of how many times the respective occlusion is detected.

This graph has several important properties. First, the graph is directed and *should* also be acyclic (i.e. a DAG) since cycles represent planar configurations which cannot occur unless our previous assumptions are violated or, rarely, noise can cause false occlusion. Second, some groups of polygons may not interact via occlusion, thus the resulting graph may not be connected (a forest of connected DAGs is produced). Third, at least one node G_m must exist per connected graph such that $\forall G_{i=1..n} \neg\exists G_i \mapsto G_m$. We call such nodes "unoccluded nodes", since they correspond to features that are not occluded by any other feature over the duration of the video. There must be one or more unoccluding node per connected graph, and one or more graph in the entire forest.

First we verify that the graph is indeed acyclic. If not, cycles are broken by removing the cycle's edge of least weight. This removes sporadic occlusions which can appear due to noise. We now assign an integer *depth code* to each node in the graph; smaller values represent features closer to the camera. The value assigned to a particular node corresponds to the maximum of the hop count of the longest path from any unoccluded node to that node (Figure 6-3). The algorithm proceeds as follows:

1: **function** *determine_depth_coding*($G_{1..n}$)
2: **for** $i = 1$ to n **do**
3: Find the connected graph g s.t. $G_i \in g$.
4: Identify the unoccluded nodes $U \subseteq g$.
5: $code = 0$
6: **for** each unoccluded node $u \in U$ **do**
7: $code$=**max**($code$,**longestpath**(u, G_i)).
8: **end for**
9: **assign** depth order of node $G_i = code$
10: **end for**

We note that since connected graphs are encoded separately, depth coding is only consistent within individual connected graphs. By definition, features within disconnected graphs do not occlude each other; thus it is not possible to determine a consistent ordering over all connected graphs using occlusion alone. However since this data is required later only to composit features in the correct order, such consistency is not relevant.

6.4 Computer Graphics Component

In this section we show how to construct *correspondence trails*, *deformation bases*, and *occlusion buffers*, all of which are used when rendering motion cues within the video. We begin with an object, which has been tracked over the full course of a video, as already described. We analyse tracking data and show how to produce an *animated object* which is a re-presentation of the original, subject to the placement of augmentation cues (streak-lines and ghosting), deformation cues (squash and stretch and other distortions), and suitable occlusion handling. We work entirely with two-dimensional data. Each feature has two cels associated with it, one for *augmentation cues* such as streak-lines, the other for *deformation cues* such as squash and stretch. The cels are composited according to the depth ordering of features, from the furthest to the nearest; for a given feature, deformation cels are always in front of augmentation cels (Figure 6-11).

6.4.1 Motion Cues by Augmentation

Augmentation cues, such as streak-lines and ghosting, are common in traditional animation (Figure 6-1). Streak-lines can ostensibly be produced on a per frame basis by attaching lines to a feature's trailing edge, tangential to the direction of motion [80]. Unfortunately, such an approach is only suitable for visualising instantaneous motion, and produces only straight streak-lines. In addition, these "lines" often exhibit distract-

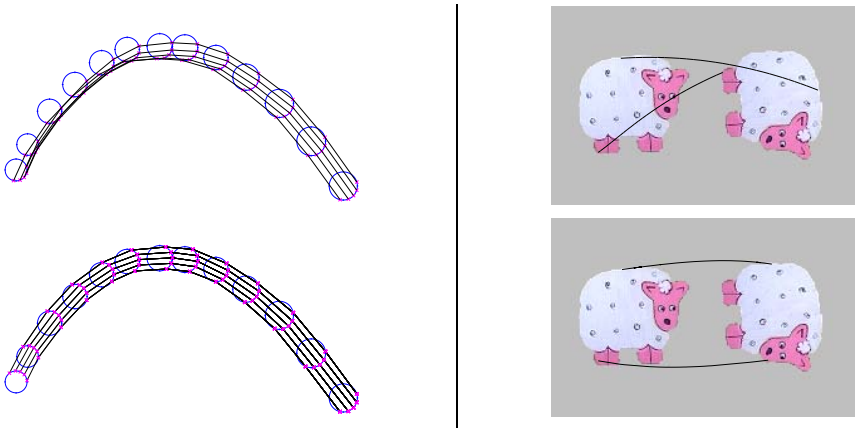


Figure 6-4 Demonstrating that point trajectories do not generally form artistic streak-lines. Corresponding the polygon vertices returned by the tracker for the ball in *VOLLEY* creates unsightly bunching of lines; in fact, in this example such trajectories are ambiguous due to symmetry. However the problem remains even in the case of unambiguous point trajectories, for example that of the sheep (right) where streak-lines converge and cross. We match trailing edges, form correspondence trails, and filter them using heuristics to place streak-lines in a manner similar to that of traditional animators. Our correspondence strategy produces a closer approximation the streak-lines drawn by an animator (bottom-left, bottom-right).

ing wobbles over time due to small variations in a feature's trajectory. By contrast, animators tend to sketch elegant, long curved streaks which emphasise the essence of the motion historically. In Figure 6-4 we show that the trajectories generated by imaged points do not necessarily correspond to an animator's placement of streak-lines. Readily available techniques, such as optical flow [54], are thus infeasible for producing streak-lines for two main reasons: (1) Optical flow algorithms tend to operate over periods of time that are much shorter than typical streak-lines represent. This makes them susceptible to noise, and they fail at occlusions. (2) In general, streak-lines do not trace the motion of a single point on an object (as is the intent with optical flow), but depict the overall "sense-of-motion" of an object.

Streak-line placement is therefore a non-trivial problem: they are not point trajectories, features tend to move in a piecewise-smooth fashion, and we must carefully place streak-lines on features. To produce streak-lines we generate *correspondence trails* over the trailing edge of a feature as it moves, we then segment trails into smooth sections, which we filter to maximise some objective criteria. We finally render the chosen sections in an artistic style.

We begin by sampling the feature boundary at regular spatial intervals, identifying a point as being on the trailing edge if the dot product of its motion vector with the external normal to the boundary is negative. Establishing correspondence between trailing edges is difficult because their geometry, and even connectivity, can vary from frame to frame (as a function of motion). The full LCAT determined during tracking cannot be used, as this establishes point trajectories (which we have observed are not streak-lines); we wish to establish correspondence between feature silhouettes.

We establish correspondence trails by computing the instantaneous tangential velocity of a feature's centroid $\underline{\mu}$. A translation and rotation is computed to map the normalised motion vector from $\underline{\mu}$ at time t to time t'. Any scaling of the feature is performed using the scale parameter of the LCAT determined during tracking. Points on the trailing edge at time t are now translated, rotated, and scaled. Correspondence is established between these transformed points at time t, and their nearest neighbours at time t', this forms a link in a correspondence trail. This method allows the geometry of the trailing edge to vary over time, as required.

Animators tend to draw streak-lines over smooth sections of motion. Therefore the correspondence trails are now segmented into smooth sections, which are delimited by G^1 discontinuities — knots in piecewise cubic curves. This results in a set of smooth sections, each with a pair of attributes: 1) a function $\underline{G}(s)$ where $s \in [0, 1]$ is an arc-

Figure 6-5 A selection from the gamut of streak and ghosting line styles available: Ghosting lines may be densely sampled to emulate motion blur effects (b,c,f) or more sparsely for ghosting (a,d). The feature itself may be ghosted, rather than the trailing edge, to produce Futurist-like echos (e,g). Varying the overlap constant w influences spacing of streak-lines (b,f). The decay parameters of streak and ghosting lines may be set independently (a).

length parameterisation of the curve trajectory; 2) a lookup table $g(.)$ mapping from an absolute time index t (from the start of the clip) to the arc-length parameter, i.e. $s = g(t)$, thus recording velocity along the spatial path $\underline{G}(s)$. The inverse lookup function $g^{-1}(.)$ is also available. Clearly the curve exists only for a specific time period $[g^{-1}(0), g^{-1}(1)]$; we call this interval the duration of the curve.

The association between each smooth section and its data points is maintained. These data points are used to filter the set of smooth sections to produce a subset σ of manageable size, which contains optimal paths along which streak-lines will be drawn.

Our filtering selects curves based on heuristics derived from the practice of animators [130] who favour placement of streak-lines on sites of high curvature and on a feature's convex hull. Long streak-lines and streak-lines associated with rapid motion are also preferred, but close proximity to other co-existing streak-lines is discouraged. We select streak-line curves using a greedy algorithm to maximise the recursive function $H(.)$, on each iteration i adding a new element to σ (initially empty).

$$\begin{aligned} H(0) &= 0 \\ H(i+1) &= H(i) + (\alpha v(x) + \beta L(x) - \\ &\quad \gamma D(x) - \delta\omega(x, \sigma; \eta) + \zeta\rho(x)) \end{aligned} \tag{6.11}$$

where x is the set of spatial points associated with a smooth section. $L(x)$ is the length of a smooth section, $v(x)$ is the "mean velocity" defined as $L(x)/t(x)$ in which $t(x)$ is the temporal duration of the smooth section. $\rho(x)$ is the mean curvature of feature boundary at points in x. $D(x)$ is the mean shortest distance of points in x from the

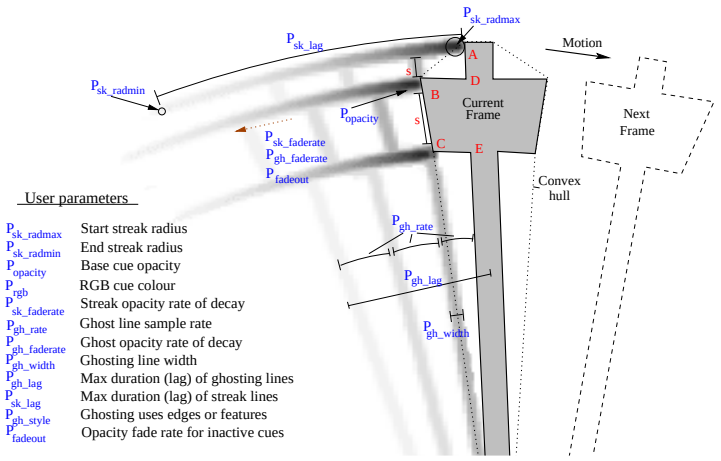


Figure 6-6 Left: User parameters used, in conjunction with weights of equation 6.11, to influence augmentation cue placement.

convex hull of the feature. $\omega(x, \sigma; \eta)$ measures the maximal spatiotemporal overlap between x and the set of streak-lines chosen on previous iterations. From each curve we choose points which co-exist in time, and plot the curves with width η returning the intersected area. Constant w is user defined, as are the constant weights $\alpha, \beta, \gamma, \delta$, and ζ; these give artistic control over the streak-line placement (see Figure 6-5). Iteration stops when the additive measure falls below a lower bound.

We are now in a position to synthesise two common forms of augmentation cue; streak-lines and ghosting lines — both of which are spatiotemporal in nature. A streak-line is made visible at some absolute time t and exists for a duration of time Δ. The streak-line is rendered by drawing a sequence of discs along the smooth section with which it is associated, starting at spatial location $\underline{G}(g(t))$ and ending at $\underline{G}(g(max(g^{-1}(0), t - \Delta)))$. The streak-line is rendered by sweeping a translucent disc along the smooth section (backwards in time) which grows smaller and more transparent over time. These decays are under user control.

Ghosting lines depict the position of a feature's trailing edge along the path of the streak-line, and are useful in visualising velocity changes over the course of the streak-line. Ghosting lines are rendered by sampling the trailing edge at regular time intervals as the streak-line is rendered, interpolating if required. The opacity of ghosting lines is not only a function of time (as with streak-lines) but also a function of speed relative to other points on the trailing edge; this ensures only fast moving regions of edge are ghosted. Users may control the sampling rate, line thickness, and decay parameters to stylise the appearance of the ghosting lines (Figure 6-6).

6.4.2 Motion Cues by Deformation

Deformation cues are further tools available to an animator. They are routinely used to depict inertia or drag; to emphasise the line of action through a frame; or to show the construction or flexibility of an object. Features are cut from the current video frame, and motion dependent warping functions applied to render the deformation cue cel for each feature.

Deformations are performed in a curvilinear space, the basis of which is defined by the local trajectory of the feature centroid, and local normals to this curve. This trajectory has exactly the same properties as any smooth section (see Section 6.4.1). A local time-window selects a section of the centroid trajectory, and this window moves with the object. The instantaneous spatial width of this window is proportional to instantaneous centroid velocity. At each instant, t we use the centroid trajectory to establish a curvilinear basis frame. First we compute an arc-length parameterisation of the trajectory: $\underline{\mu}(r)$ in which $r = \int_{i=t}^{t'} \left|\dot{\underline{\mu}}(i)\right| di$. Next we develop an ordinate at an arc-length distance r from the instant; the unit vector $\underline{n}(r)$ perpendicular to $\dot{\underline{\mu}}(t)$. Thus, at each instant t we can specify a point in the world-frame using two coordinates, r and s:

$$\underline{x}_t(r,s) = \underline{\mu}(r) + s\underline{n}(r) \tag{6.12}$$

We can write this more compactly as $\underline{x}_t(\underline{r}) = \underline{C}(\underline{r})$, where $\underline{r} = (r,s)^T$, and maintain the inverse function $\underline{r}_t(\underline{x}) = \underline{C}^{-1}(\underline{x})$ via a look-up table. This mapping, and its inverse, comprise one example of a *deformation basis*.

The above analysis holds for most points on the centroid trajectory. It breaks down at *collision points*, because there is a discontinuity in velocity. We define a collision point as one whose location cannot be satisfactorily predicted by a second order motion equation (constructed with a Kalman filter, see Section 6.3.2). This definition discriminates between G^1 discontinuities in trajectory which are formed by, say, simple harmonic motion, and true collisions which are C^1 discontinuous (see Figure 6-7).

At a collision point we establish an orthonormal basis set aligned with the observed collision plane. We assume the angle of incidence and reflection are equal, and hence the unit vector which bisects this angle is taken to be the ordinate. The abscissa lies in the collision plane. We define this new basis set as an additional instance of a deformation basis, and write the mapping to and from the world frame using notation consistent with the curvilinear basis. In addition, we compute the *impact parameter* of the collision, which we define as the distance (D) from the object's centroid to its

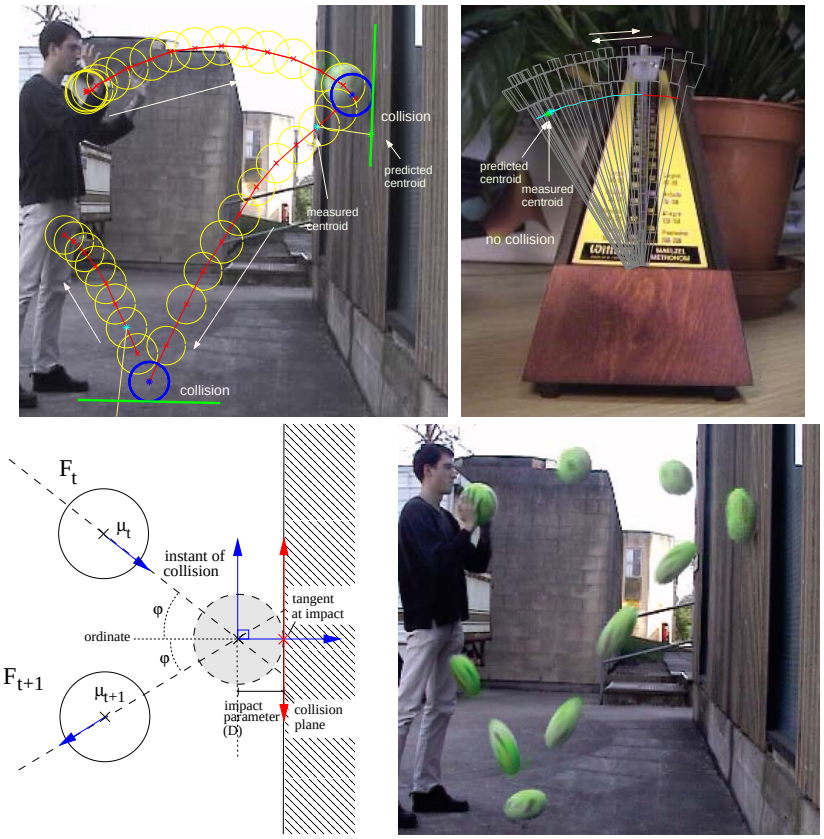


Figure 6-7 Top: bounces are detected as collisions (left), whilst the simple harmonic motion of the metronome is not (right). Predicted motions shown in green, observed position in cyan. Bottom: Collision geometry of the first impact in the *BOUNCE* sequence. The time lapse image (right) shows the resulting deformations (see `videos/bounce_deformationonly`).

boundary, in the direction of the negative ordinate. Note we compensate for temporal sampling around the true collision instant by intersecting extrapolated impact and rebound trajectories (Figure 6-7).

An animated object is, in general, a deformed version of the original. Squash and stretch tangential to instantaneous motion leads to visually unattractive results; it is better to not only squash and stretch, but also to bend the object along the arc of its centroid trajectory. Let $\underline{x}(t)$ be any point in the object. We transform this point with respect to a single deformation basis into a new point $\underline{y}(t)$, given by

$$\underline{y}(t) = \underline{C}(\underline{A}_t[\underline{C}^{-1}(\underline{x}(t))]) \tag{6.13}$$

where $\underline{A}_t[.]$ is some deformation function at time t. In the case of squash and stretch the deformation is an area-preserving differential scale that depends on instantaneous

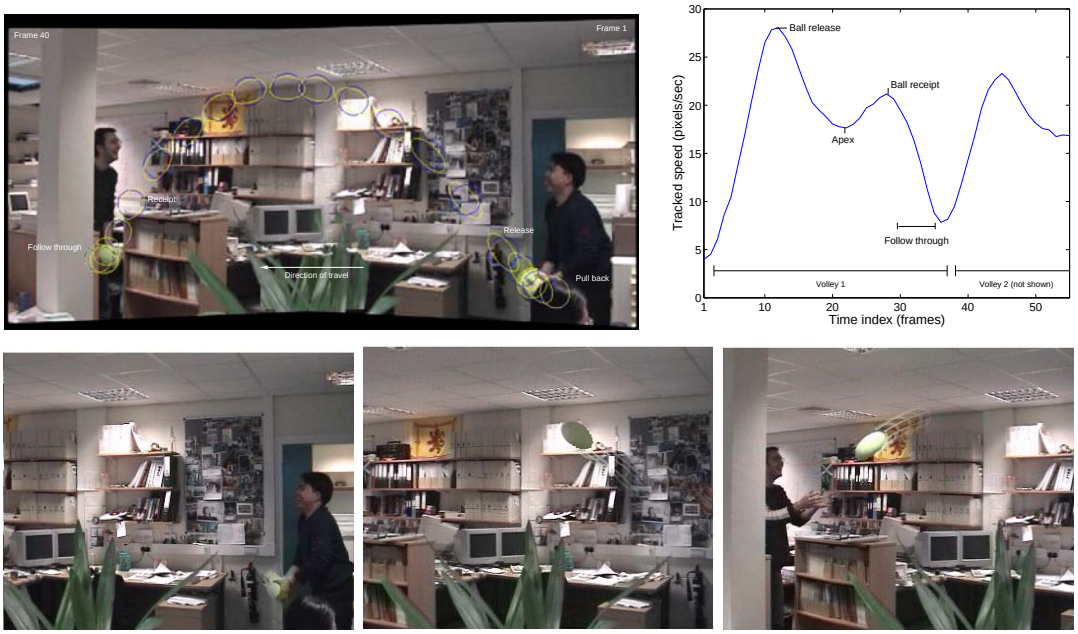


Figure 6-8 Top left: Illustrating the squash and stretch effect; eccentricity varies as a function of tangential speed (right). Bottom: Frames from the *VOLLEY* sequence exhibiting squash and stretch. Observe the system handles both large scale camera motion, and lighting variation local to the ball (`videos/volley_streaks`).

speed $|\dot{\underline{\mu}}(t)|$:

$$\underline{\underline{A}} = \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \tag{6.14}$$

$$k = 1 + \frac{K}{2}(1 - cos(\pi \frac{v^2 + 1}{2}))$$

$$v = \begin{cases} 0 \text{ if } |\dot{\underline{\mu}}| < V_{min} \\ 1 \text{ if } |\dot{\underline{\mu}}| >= V_{max} \\ (|\dot{\mu}| - V_{min})/(V_{max} - V_{min}) \text{ otherwise} \end{cases} \tag{6.15}$$

However during collision animators tend to vary the amount of "stretch" in proportion to acceleration magnitude, rather than speed. In addition, the deformation due to impact takes place in the deformation basis defined about the collision site, as discussed earlier in this Section.

We linearly interpolate between the deformation caused by a "standard" deformation basis with that caused by a "collision" deformation basis. Suppose $\underline{p}(t) \mapsto \underline{q}(t)$ and $\underline{p}(t) \mapsto \underline{q}'(t)$, respectively, then:

$$\underline{r}(t) = f(d)\underline{q}(t) + (1 - f(d))\underline{q}'(t) \tag{6.16}$$

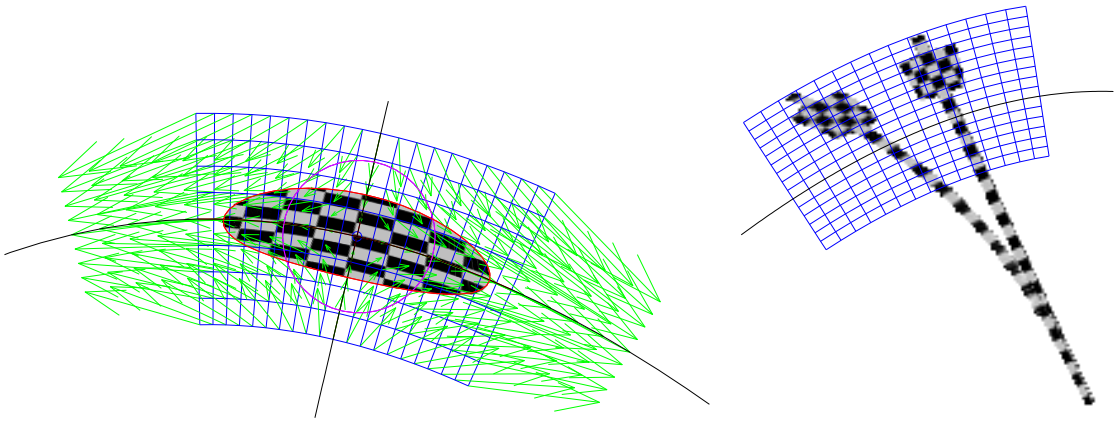


Figure 6-9 Examples of curvilinear warps used to create deformation cues in the *VOLLEY* (left) and *METRONOME* (right) sequences.

where $f(d) = sin(\frac{\pi}{2}\min(1, d/(sD))$ in which D is the impact parameter of the collision (defined previously), and s is a user parameter which controls the spatial extent over which collision influences the deformation. This transfer function gives a suitably smooth transition between the two deformation bases. As a note, the mapping to $\underline{q'}(t)$ not only has to scale the object but also translate it toward the impact point, so that the edge of the deformed object touches the collision plane. This translation vector is expressed within the collision basis as:

$$\begin{bmatrix} 0 \\ D(1-\frac{1}{k}) \end{bmatrix} \tag{6.17}$$

Non-linear deformations are also possible, and these can be used to create bending effects. We can form warping functions which depend on each point's velocity and acceleration as well as its position. We write $x' = \underline{C}(\underline{A}[\underline{C}^{-1}(\underline{x}); \dot{\underline{x}}, \ddot{\underline{x}}])$, where $\underline{A}(\underline{r}; .)$ is a functional used, for example, to suggest increased drag or inertia. A typical functional operates on each component of $\underline{r} = (r_1, r_2)^T$ independently; to create effects suggesting drag we use:

$$r_1 \leftarrow r_1 - F(\frac{2}{\pi} arctan(|\dot{\underline{x}}_i|))^P sign(\dot{\underline{x}}_i) \tag{6.18}$$

F and P are user defined constants which affect the appearance of the warp. For example, large F give the impression of heavy objects and P influences the apparent rigidity of objects. By substituting acceleration for velocity, and adding rather than subtracting from r_1 we can emphasise inertia of the feature (see Figure 6-10):

$$r_1 \leftarrow r_1 + F(\frac{2}{\pi} arctan(|\ddot{\underline{x}}_i|))^P sign(\ddot{\underline{x}}_i) \tag{6.19}$$

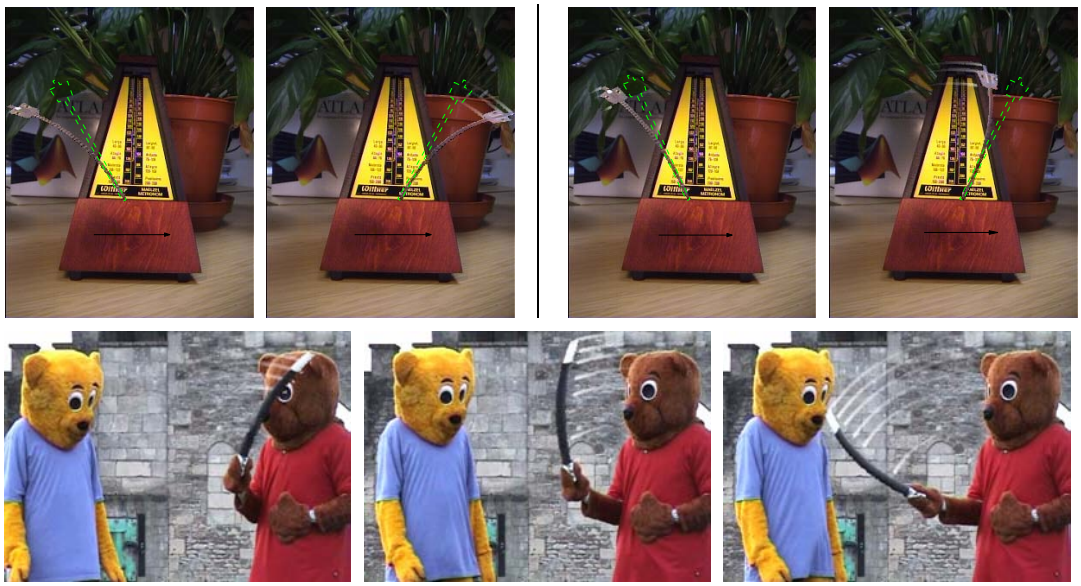


Figure 6-10 Top: Frames from *METRONOME* suggesting inertia (left) and drag (right) effects through deformation, original feature outline in green. Bottom: Deforming a rigid wand in the *WAND* sequence, using a velocity based "drag" effect — streak-lines are warped along with the object, adding to the effect (`videos/wand_cartoon`).

Finally, we ensure visual consistency by deforming not only features, but also their associated augmentation cue cels containing streak-lines or ghost lines.

6.4.3 Rendering in the Presence of Occlusion

In any real video a tracked feature may become occluded by other elements of the scene. Naïvely, all pixels inside the bounding contour of a feature will be included in that feature and so are subject to deformation. Consequently, it is easy to mistakenly warp parts of occluding objects; we now explain how to avoid producing these unwelcome artifacts.

We begin our explanation with an illustrative example. Consider the *BASKETBALL* video clip of Figure 6-12, in which a fast moving ball passes behind a hoop. We may identify pixels as belonging to an occluding object, in this case the ring, by forming a difference image (using the hue and saturation components in HSV space, again to affect simple invariance to illumination changes) between the feature region in the current frame and the feature itself. This yields a weighted mask of occluding pixels which are re-textured by sampling from feature regions in neighbouring unoccluded frames. An approximation to the unoccluded feature is thus reconstructed, and after deformation occluding pixels may be recomposited to give the illusion of the ball passing 'behind' occluding objects (Figure 6-11d).

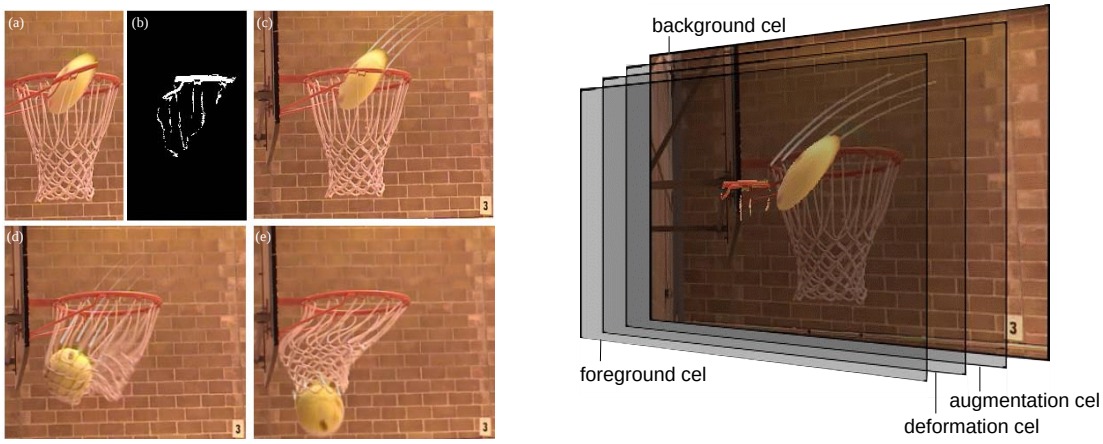


Figure 6-11 Left: (a) naïve deformation creates artifacts, but our method does not (c). An occlusion buffer is constructed over time (b), allowing augmentation cues to be handled in the wake of the feature. The system breaks down (d) after impact erratic movement of the netting causes the occlusion buffer to empty and cues to be incorrectly drawn in front of that netting. Right: Frames are rendered as layers, in back-to-front order. In this case a single object (the ball) is tracked to produce augmentation cues (streak-lines) and deformations (squash and stretch). Occluding pixels are overlaid in the foreground to produce the illusion of the ball deforming behind the hoop (`videos/basket_rendered`).

The situation becomes more complicated when one considers that a feature may be deformed (stretched, for example, as part of squash and stretch) outside its original polygonal boundary. Similarly, augmentation cues should pass behind occluding objects; these cues are also drawn outside of the original boundary. Fortunately we can make progress, since these cues traverse an identical path to that of the feature itself. We construct an *occlusion buffer* by examining a small temporal window of frames centred upon the current time instant, summing the difference images generated whilst handling occlusion in the manner described in the preceding paragraph. Using this buffer we may determine which pixels will occlude cues such as streak-lines, so long as those pixels do not change in the time interval between the real feature passing and the augmentation cues being drawn. The occlusion buffer contains a difference image for occlusion and the RGB value of each pixel. Pixels in the buffer are deleted when the difference between the stored colour for a pixel, and the measured colour of that pixel at the current time is significant. In this case the occluding pixel has moved, obsoleting our knowledge of it.

This algorithm works acceptably well over short time intervals, but in general we can hope only to mitigate against artifacts generated by occluding objects which we have not explicitly tracked. For example, Figure 6-11d demonstrates how streak-lines are able to correctly pass behind complex netting, whilst that netting is stationary. The system breaks down in Figure 6-11e when the netting moves erratically after ball im-

pact, causing the occlusion buffer to quickly empty and streak-lines to be incorrectly drawn in front of the netting.

6.4.4 Compositing and Rendering

The graphics component composits cels to create each frame of the animation. To render a particular frame, a *background cel* is first generated by subtracting features from the original video; determining which pixels in the original video constitute features by projecting tracked feature regions from the camera-compensated sequence to the original camera viewpoint. Pixels contributing to feature regions are deleted and absent background texture sampled from locally neighbouring frames in alternation until holes are filled with non-feature texture. This sampling strategy mitigates against artifacts caused by local lighting changes or movement.

Once augmentation and deformation cels have been rendered for each feature, cels are composited to produce an output video frame. Cels are projected by homography to coincide with the original camera viewpoint, and composited onto the background in reverse depth order. Finally, a *foreground cel* is composited, which contains all pixels that occlude identified objects. These pixels are taken from the occlusion buffer described in Section 6.4.4. Thus motion cues appear to be inserted into video at the correct scene depth (Figure 6-11).

6.5 Summary and Discussion

We have described and demonstrated a subsystem, within the Video Paintbox, for the artistic rendering of motion within video sequences. The subsystem can cope with a moving camera, lighting changes, and presence of occlusion. Aside from some (desirable) user interaction when boot-strapping the tracker, and the setting of user parameters, the process is entirely automatic. Animators may influence both the placement and appearance of motion cues by setting parameters at a high conceptual level; specifying the objects to which cues should be attached, and the type of motion emphasis desired. Parameters may also be set on individual motion cues to stylise their appearance, for example streak-line spacing or squash and stretch rigidity.

We have shown that through high level analysis of features (examining motion over blocks of video with large temporal extent, rather than on a per frame basis) we may produce motion cues closely approximating those used in traditional animation (Figure 6-1). For example, we have shown that streak-lines are generally poorly represented by point trajectories [80], and should instead be rendered as smooth curves capturing the essence of a trajectory over time. Although the former are obtainable via temporally

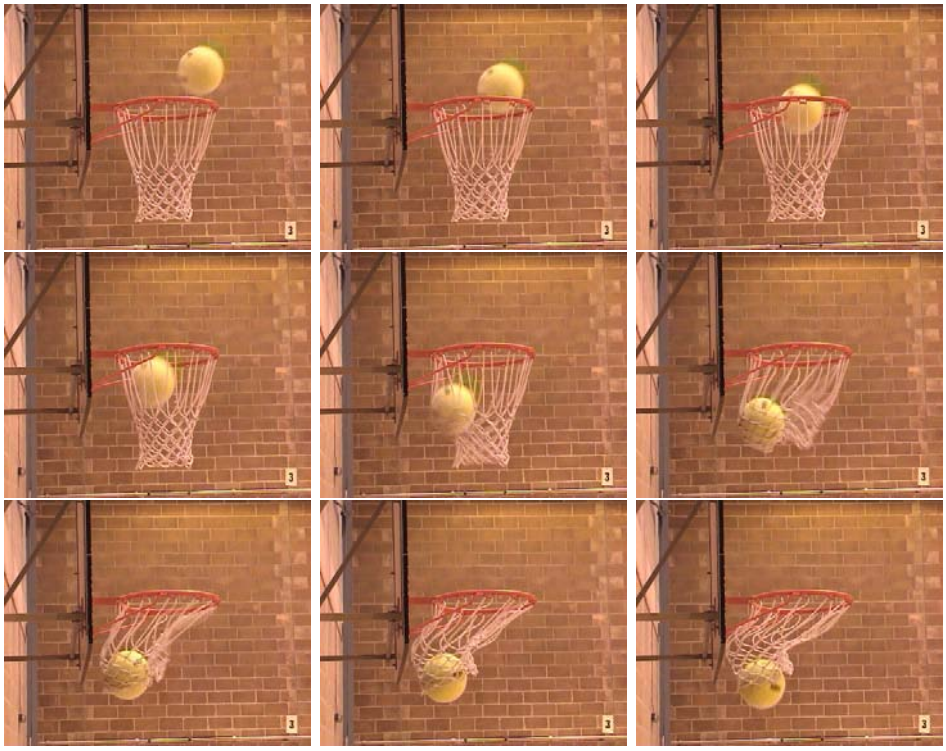
local processing, the latter require analysis which necessitates a higher level of temporal analysis for trajectory analysis. Likewise deformation effects such as squash and stretch demand detection and localisation of collisions, the analysis of which requires examination large temporal window around the collision instant. The spatiotemporal occlusion buffer is another example of higher level temporal analysis performed by the motion emphasis subsystem.

Further developments could address the relaxation of some of the assumptions made in the design of the subsystem. For example, violations of depth assumptions in Section 6.3.3 are detectable by the presence of heavily weighted cycles in the depth graph. It may be possible to segment video into a minimal number of chunks exhibiting non-cyclic depth graphs, and in doing so recover relative feature depth under more general motion. The tracker could also benefit from more robust, contemporary methods such as CONDENSATION [85], although this may involve imposing a more restrictive motion model than Kalman filtering requires, which is why we opted for the latter. We can imagine a system that uses CONDENSATION tracker for common objects, such as people walking, but defaults to a Kalman filter in more general cases. Further developments might evaluate the robustness of the algorithm using both ground truth comparisons for measures such as velocity, as well as processing sequences exhibiting distinctly non-planar motion.

Although these incremental changes would serve to enhance robustness, the most interesting extensions of this work lie in an expansion of the gamut of motion cue styles, and the incorporation of motion emphasis with the coherent artistic shading of video. These two developments are documented in Chapters 7 and 8 respectively.

A selection of source and rendered video clips have been included in Appendix C.

SOURCE *BASKETBALL* SEQUENCE



RENDERED *BASKETBALL* SEQUENCE

Figure 6-12 Stills from the original (`videos/basket_source`) and rendered (`videos/basket_rendered`) versions of the *BASKETBALL* sequence. This sequence not only demonstrates the use of squash and stretch deformations, streak-lines and ghosting, but also serves to illustrate our occlusion handling.

RENDERED *BOUNCE* SEQUENCE

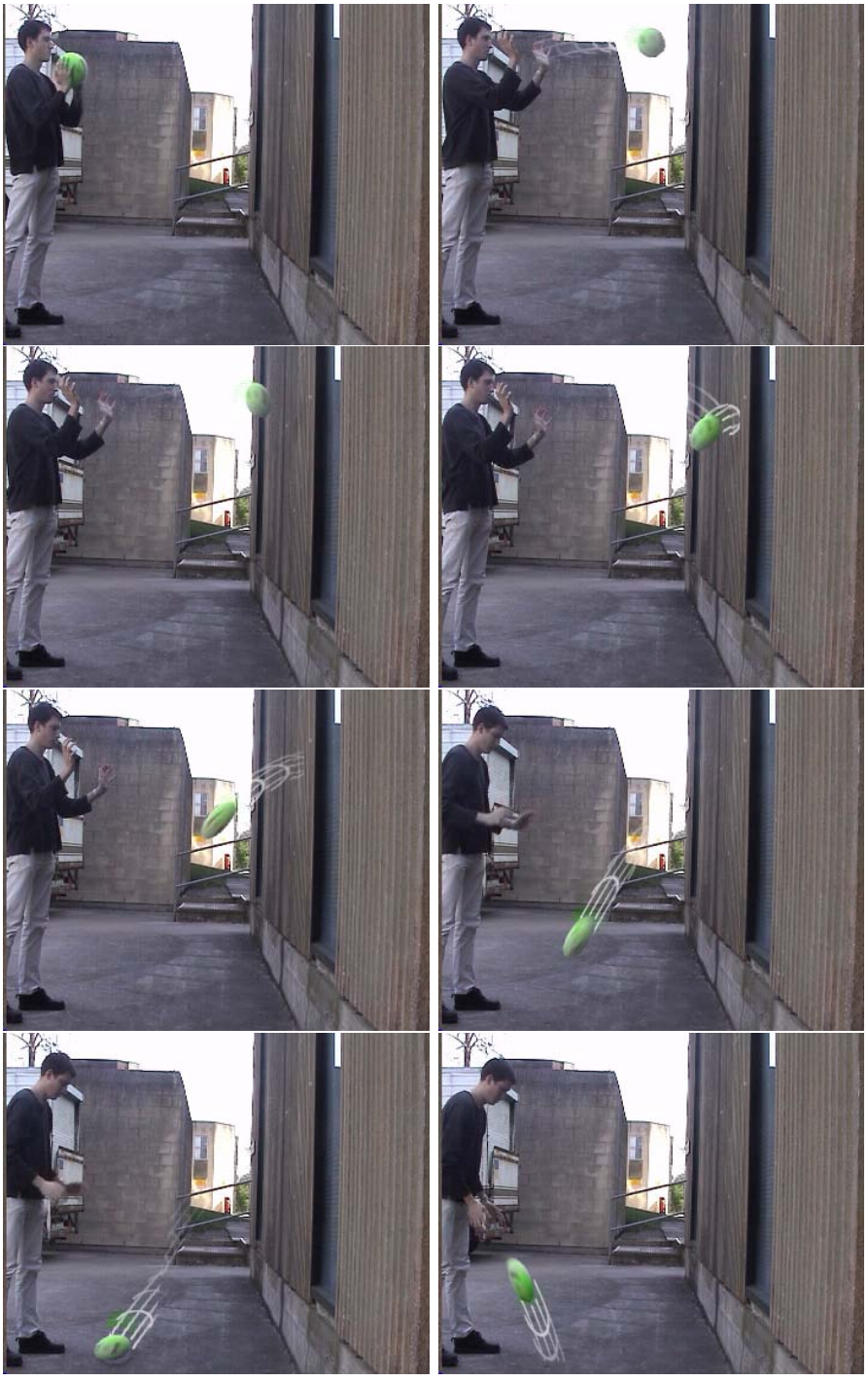


Figure 6-13 Stills from the rendered *BOUNCE* sequence, demonstrating both augmentation cues and our squash and stretch deformation effect (see `videos/bounce_motiononly`). This sequence demonstrates the correct detection and handling of collision events.

RENDERED *VOLLEY* SEQUENCE

Figure 6-14 Stills from the rendered *VOLLEY* sequence, demonstrating both augmentation cues and our squash and stretch deformation effect (see `videos/volley_motiononly`). This sequence demonstrates that the system is able to compensate for large scale camera motion, and handle local lighting changes (on the ball).

RENDERED *BALLET* SEQUENCE

Figure 6-15 Stills from the rendered *BALLET* sequence, demonstrating the insertion of augmentation cues in to the sequences at the correct scene depth (`videos/ballet_streaks`).

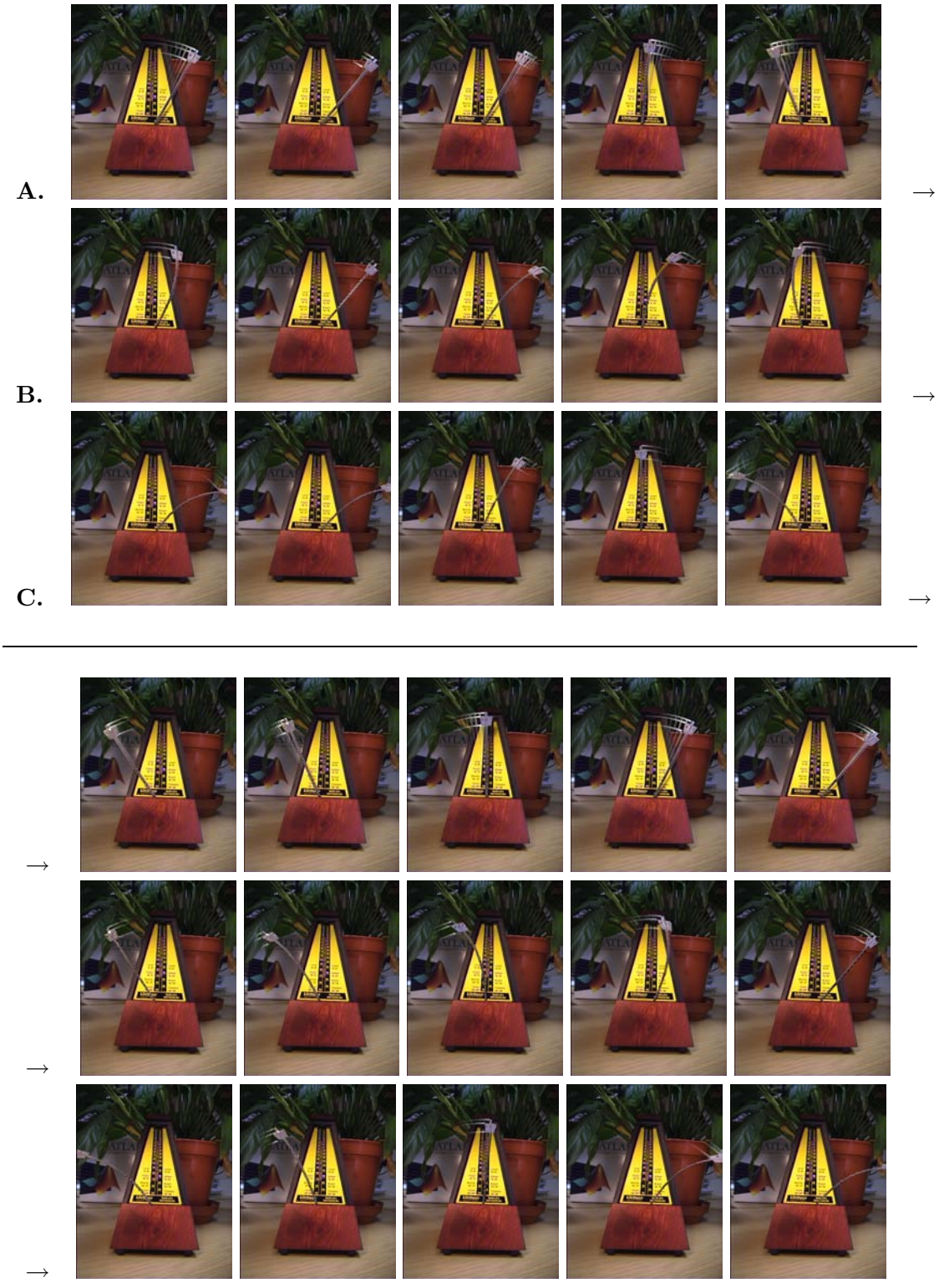


Figure 6-16 Ten stills taken simultaneously (time runs from left to right) over three different renderings of the *METRONOME* sequence: (A) Augmentation cues only (`videos/metro_streaks`); (B) Deformation emphasising drag (velocity dependent warp) with streak lines (`videos/metro_warp_veloc`); (C) Deformation emphasising inertia (acceleration dependent warp) with streak lines (`videos/metro_warp_accel`). Approximately one still per five frames.

Chapter 7

Time and Pose Cues for Motion Emphasis

In this chapter we extend the gamut of motion emphasis cues in the Video Paintbox to include animation timing effects; specifically, cartoon "anticipation" effects and motion exaggeration. Our process begins by automatically recovering articulated "dolls" of subjects moving in the plane. By fitting our previously tracked (Chapter 6) features to this articulated structure we are able to describe subject pose, at any instant, as a point in a high dimensional pose space. We then perform local and global distortions to this pose space to create novel animation timing effects in the video. We demonstrate that these timing cues may be combined with the visual cues of Chapter 6 in a single framework.

7.1 Introduction

Although animators may emphasise the motion of a subject using an array of purely visual tools (for example making marks, or performing deformations), they may also choose to manipulate the timing of an action to stylise an animation — we term this class of effects "time and pose" motion cues. In this Chapter we describe the subsystem for inserting these cues in to the video sequence, and explain how this work may be integrated with the motion cues of Chapter 6 to complete the motion emphasis framework within the Video Paintbox.

A common "time and pose" cue used by animators is "anticipation", which is used to indicate the onset of motion. Typically, an object exhibiting anticipation is observed to recoil briefly, as if energy is somehow being stored in preparation for a release into motion; the resulting effect is sometimes referred to as "snap" by animators (Figure 7-5). According to animator Richard Williams [169] (pp. 274–275) there are three rules

governing anticipation:

1. The anticipation is always in the opposite direction to where the main action is going to go.

2. Any action is strengthened by being preceded by its opposite.

3. Usually the anticipation is slower than the action itself.

A further motion emphasis technique is "exaggeration". Just as caricaturists often draw faces by exaggerating their characteristics [98], in a similar vein animators often exaggerate the individual characteristics of a subject's movement. Such manipulations of motion clearly demand large temporal windows for trajectory analysis. In the case of anticipation, at any given instant the system must be aware of future motion that may subsequently take place. Likewise the identification of patterns and characteristics within a subject's movement over time are a prerequisite to exaggerating those motions within the video sequence. A per frame, sequential approach to video driven AR therefore precludes the insertion of time and pose cues into the resulting animation.

The framework we propose operates by varying the pose of the subject over time, manipulating the features tracked by the Computer Vision component of Section 6.3 prior to their rendering by the Computer Graphics component of Section 6.4. Our initial experiments attempted to create time and pose cues by directly manipulating the tracked LCATs of features; this did not synthesise motion cues of satisfactory aesthetic quality for anything but the simplest of motions (for example a translating ball). We instead opted to recover the articulated structure of tracked subjects, to create a higher level description of subject motion — the subjects' pose. Manipulation of this pose over time allows us to augment the existing gamut of motion cues in the Video Paintbox with convincing temporal effects.

We begin by automatically recovering an articulated "doll" from the set of features tracked in Chapter 6. Once the "doll" structure has been recovered, the tracked polygons for each frame are fitted to the articulated structure, creating a series of "pose vectors". These pose vectors are points in a high dimensional "pose space", each of which specify the configuration of the articulated subject at a particular instant in time. As the articulated subject moves over time, the vectors can be thought of as tracing a smooth trajectory in this high dimensional pose space. By performing judicious transformations on the spatiotemporal trajectories within this space we may affect the pose of the subject over time, and so introduce "time and pose" cues into the video sequence.

Our desire to process post-production video of general content (for example a person, or a metronome) from a single view-point has restricted the range of literature we have

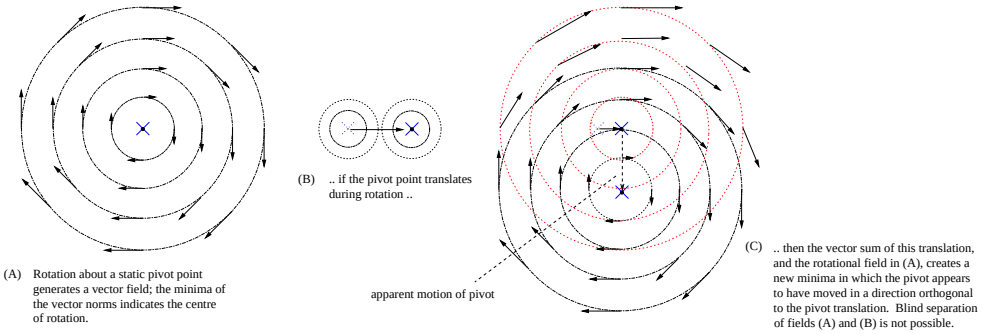


Figure 7-1 Illustrating the difficulty of recovering pivot point location from rotation, in the case of a moving pivot. Under instantaneous motion, the combination of rotation and pivot shift causes an apparent translation of pivot location orthogonal to the direction in which the pivot have moved (see Appendix A.6 for a proof).

been able to draw upon for the recovery of articulated structure and pose. Most passive motion capture systems assume provision of an *a priori* articulated model [17, 76], are tailored to the specific problem of tracking humans [10, 89], or require multiple camera viewpoints [11, 76, 88]. We have developed a technique for recovering articulated structure and pose that does not require such constraints, but does make a number of alternative assumptions which are more compatible with our application. First, we assume that the subject exhibits a rigid, hierarchical articulated structure. Second, we assume that the subject's motion is planar within the camera compensated sequence in which features are tracked (Section 6.3.1). Lastly, we assume that pivot points in the structure are static relative to their attached features. This avoids the problem of localising moving pivot points solely from polygonal data; this problem is intractable in principle (see Figure 7-1 and Appendix A.6).

7.2 Recovery of Articulated Pose in the plane

Our process begins by recovering the articulated structure of a subject moving in the plane; this is achieved by analysing the motion of the feature polygons tracked by the Computer Vision component in Chapter 6. The basic technique is to estimate a pivot point between each pair of tracked features, and then to assess the quality of these pivot points to determine which feature pairs are truly articulated in the physical world. These physically articulated feature pairs, and their associated pivot points, form a hierarchical representation of the rigid articulated subject being tracked. At each time instant we fit this articulated structure to the tracked feature polygons, creating a set of joint configurations represented numerically by a "pose vector". This pose representation is manipulated to introduce cartoon-like motion cues (Section 7.4).

7.2.1 Four Algorithms for Recovering inter-feature pivot points

We now describe four different algorithms for determining the pivot point between two, potentially articulated, features "A" and "B". Each feature is considered be static within its own reference frame. We write $\underline{\underline{F}}_{A(t)}$ and $\underline{\underline{F}}_{B(t)}$ as the affine transformations from each of these reference frames to world coordinates, respectively. In the first frame ($t = 1$) the reference frames are coincident with the world basis, and these transforms are the identity.

We have to assume a static pivot point in our work; by this we mean the pivot of A about B will be static relative to both frames $\underline{\underline{F}}_{A(t)}$ and $\underline{\underline{F}}_{B(t)}$ — in world coordinates the pivot can, of course, move freely.

In our descriptions of each algorithm we consider the motion of A about B, within $\underline{\underline{F}}_{B(t)}$; i.e. we assume feature B's motion has been "subtracted" from feature A's. We denote the vertices of feature A as $\underline{\underline{A}}_t$ (a series of column vectors representing the homogeneous world coordinates of each vertex at time t). Likewise we write $\underline{\underline{B}}_t$ for feature B. In homogeneous form, we may express the motion of feature A in the frame of B as $\underline{\underline{A}}'_t$, where:

$$\begin{aligned} \underline{\underline{A}}'_t &= \underline{\underline{F}}^{-1}_{B(t)} \underline{\underline{A}}_t \\ &= (\underline{\underline{B}}_1 (\underline{\underline{B}}_t)^{+}) \underline{\underline{A}}_t \end{aligned} \tag{7.1}$$

where superscript + specifies the Moore-Penrose "pseudo inverse" [123]. We use the notation $\underline{\underline{A}}'_t$ throughout our description of the four algorithms.

7.2.2 Closed Form Eigen-solutions

Algorithm 1: Decomposition of affine transform via SVD

Our first algorithm accepts two samples of the feature polygon at different time instants $[t, t + \Delta]$ (where Δ is a user defined temporal interval), and outputs a least squares algebraic approximation to the pivot point location. This is achieved by decomposing the compound affine transform for rotation about an arbitrary point in 2D. The motion of A over time interval $[t, t + \Delta]$ may be described as $\underline{\underline{A}}'_{t+\Delta} = \underline{\underline{M}}\,\underline{\underline{A}}'_t$, where $\underline{\underline{M}}$ is an affine transform defined as follows:

$$\underline{\underline{M}} = \begin{bmatrix} \underline{\underline{R}}_{(\theta)} & -\underline{\underline{R}}_{(\theta)}\underline{p} + \underline{p} \\ 0 & 1 \end{bmatrix} \tag{7.2}$$

Where $\underline{p}$ denotes the pivot point, $\underline{\underline{R}}_{(\theta)}$ the 2D rotation matrix and θ the degree of

rotation. Introducing the notation $\underline{m} = (m_1, m_2)^T$:

$$\underline{m} = -\underline{\underline{R}}_{(\theta)}\underline{p} + \underline{p} \tag{7.3}$$

$$\underline{\underline{M}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & m_1 \\ \sin(\theta) & \cos(\theta) & m_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

Each of the unknowns in $\underline{\underline{M}}$ is recovered by forming a homogeneous linear system, using all n vertices of the feature A at time t ($[x_{1..n}, y_{1..n}]^T$) and at time $t+\Delta$ ($[x'_{1..n}, y'_{1..n}]^T$). The system is solved using SVD:

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 & -x'_1 \\ y_1 & x_1 & 0 & 1 & -y'_1 \\ & ... & & ... & \\ x_n & -y_n & 1 & 0 & -x'_n \\ y_n & x_n & 0 & 1 & -y'_n \end{bmatrix} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ m_1 \\ m_2 \\ 1 \end{bmatrix} = 0 \tag{7.5}$$

Rearrangement of $\underline{m}$ yields an expression for $\underline{p}$:

$$\underline{m} = (\underline{\underline{I}} - \underline{\underline{R}})\underline{p} \tag{7.6}$$

$$\underline{p} = (\underline{\underline{I}} - \underline{\underline{R}})^{-1}\underline{m} \tag{7.7}$$

In cases of small rotary motion the value for $\underline{p}$ becomes increasingly unreliable — very small errors in the measurement of tracked feature vertices have a very large influence on the estimated pivot location. If we write out the components of $\underline{p} = (p_x, p_y)^T$ then this behaviour is explained by the large denominators that result as θ tends to zero:

$$p_x = (m_2 \sin\theta - m_1(1 - \cos\theta))/2(\cos\theta - 1) \tag{7.8}$$

$$p_y = (m_1 - p_x(1 - \cos\theta))/\sin\theta \tag{7.9}$$

With no rotational component at all, $\underline{p}$ is undefined.

Extension to Algorithm 1: Estimation over multiple frames

With only two frames (t and $t+\Delta$) to estimate a pivot point, we can do no better than this single estimate. However, since we assume the pivot to be static in $\underline{\underline{F}}_{B(.)}$, we can perform several such estimates over the video sequence and compute an average for $\underline{p}$ over all time. We have observed that large scale rotations produce a better estimate for $\underline{p}$ than small scale rotations. We therefore take a weighted mean of the estimates

for $\underline{p}$:

$$\frac{1}{N}\sum_{t=1}^{N}\omega(\theta(t))\underline{p}(t;\Delta) \tag{7.10}$$

Where N is the number of frame estimates, and $\omega(.)$ is a confidence weight for the pivot estimate between time t and $t+\Delta$. We model this as a smoothly varying distribution over θ — zero in the case of no rotation (or equivalently, a full rotation of $\theta = 2\pi$), and unity under maximum rotational shift ($\theta = \pi$).

$$\omega(\theta) = |\sin(2\theta)| \tag{7.11}$$

θ is obtained from the solution to equation 7.5 using arc-tangents.

Algorithm 2: Minimum of inter-frame motion field

Our second algorithm accepts two samples of the feature polygon at differing time instants $[t, t+\Delta]$ (where Δ is a user defined temporal interval). We obtain an affine transformation $\underline{\underline{M}}$ which maps polygon vertices at the former time instant to the latter:

$$\underline{\underline{M}} = \underline{\underline{A'}}_{t+\Delta}(\underline{\underline{A'}}_{t})^{+} \tag{7.12}$$

Now consider three non co-linear vertices of $\underline{\underline{A'}}_t$ at locations $\underline{\underline{X}} = [\underline{x}_1 \;\; \underline{x}_2 \;\; \underline{x}_3]$ transformed to locations $\underline{\underline{Y}} = \left[\underline{y}_1 \;\; \underline{y}_2 \;\; \underline{y}_3\right]$:

$$\underline{\underline{MX}} = \underline{\underline{Y}} \tag{7.13}$$

We define a static point $\underline{p}$ with reference to the feature, in barycentric coordinates as $\underline{\alpha}$:

$$\underline{p} \;\; = \;\; \underline{\underline{X}\alpha} \tag{7.14}$$

Applying transformation $\underline{\underline{M}}$ to the point $\underline{p}$:

$$\underline{\underline{Mp}} \;\; = \;\; \underline{\underline{MX}\alpha} \tag{7.15}$$

$$\underline{\underline{Mp}} \;\; = \;\; \underline{\underline{Y}\alpha} \tag{7.16}$$

we observe that $\underline{\alpha}$ remains constant relative to the feature reference frame, which has changed. The distance $\underline{d}(\underline{p})$ which point $\underline{p}$ moves, relative to reference frame $\underline{\underline{F}}_{B(.)}$ is:

$$\underline{d}(\underline{p}) = |\alpha_1(\underline{x}_1 - \underline{y}_1) + \alpha_2(\underline{x}_2 - \underline{y}_2) + \alpha_3(\underline{x}_3 - \underline{y}_3)| \tag{7.17}$$

Consider a field $d(.) \in \Re$ defined over all points. In cases of pure rotation, the minimum should be zero valued and coincident with the location of the static pivot i.e. $\underline{\underline{M}}\underline{p} = \underline{p}$:

$$\underline{\underline{M}}\underline{p} - \underline{p} = \underline{0} \tag{7.18}$$

$$\underline{\underline{M}}\underline{\underline{X}}\underline{\alpha} - \underline{\underline{X}}\underline{\alpha} = \underline{0} \tag{7.19}$$

$$(\underline{\underline{M}} - \underline{\underline{I}})\underline{\underline{X}}\underline{\alpha} = \underline{0} \tag{7.20}$$

Introducing the notation $\underline{\underline{V}} = (\underline{\underline{M}} - \underline{\underline{I}})\underline{\underline{X}}$, we can solve the following homogeneous linear system to locate the pivot point $\underline{\alpha}$ in barycentric coordinates:

$$\underline{\underline{V}}\underline{\alpha} = 0 \tag{7.21}$$

As with Algorithm 1, in noisy conditions angular velocity governs the accuracy of the estimate. In the case of no rotation, for example pure scale or translation, there will not be a single minimum. We therefore extend this algorithm to operate over multiple frames as with Algorithm 1; averaging pivot estimates over each frame and weighting these estimates according to the amount of rotary motion detected. Since a quantitative estimate for the amount of rotary motion is unavailable using this algorithm, we obtain an estimate of θ (equation 7.11) for this weighting using the algebraic technique of Algorithm 1.

7.2.3 Evidence Gathering, Geometric Solutions

Algorithm 3: Circle fitting method

Our third algorithm accepts three samples of the feature polygon at different time instants $[t, t + \frac{\Delta}{2}, t + \Delta]$ (where Δ is a user defined temporal interval), to produce an estimate of pivot point location.

If the motion of vertices $\underline{\underline{A}}'_t$ in frame $\underline{\underline{F}}_{B(t)}$ is approximately described by rotation about a fixed pivot $\underline{p}$, we may consider the trajectory of single vertex $\underline{a}'_t$ of that feature to follow a circular path. We sample the position of $\underline{a}'_t$ at three distinct time intervals, and fit a circle to interpolate those points; described by both a centre $\underline{p} = (i, j)^T$ and a radius r. The circle fitting method first computes the two chords of the circle, $\underline{H}_1 = \underline{a}'_{t+\Delta} - \underline{a}'_{t+\frac{\Delta}{2}}$ and $\underline{H}_2 = \underline{a}'_{t+\frac{\Delta}{2}} - \underline{a}'_t$. We then intersect the perpendicular bisectors of $\underline{H}_1$ and $\underline{H}_2$, to obtain $\underline{p}$. Radius r is obtained as the L_2 norm of the vector from $\underline{p}$ to either of the midpoints on $\underline{H}_1$ or $\underline{H}_2$. The centre $\underline{p}$ of the fitted circle is the pivot point. This process is repeated for each vertex $\underline{a}'_t$ in $\underline{\underline{A}}'_t$ and an average pivot point computed over all vertices.

As with the previous algorithms, this method extends to take into account multiple temporal samples to compute an improved estimate for the pivot location. However we found that averaging estimates for $\underline{p}$ over time did not give reliable results since outliers, which occur frequently due to noise, greatly skew the average. These outliers often exhibit very different radii from inlier estimates. Rather than throw away this useful indicator, we use the information to help combine estimates for $\underline{p}$ using a Hough-like accumulator approach.

By fitting multiple circles over different time intervals we accumulate "votes" for circle parameters. We define a 3D accumulator space, and for every vote a 3D Gaussian of constant standard deviation and mean is added to the space, centred at $[\underline{p}^T \quad r]^T$. An associated "confidence" weight $\omega(\theta)$ is assigned to each of these votes using an identical scheme to algorithms 1 and 2 (see equation 7.11). In this algorithm, we obtain θ from the inner product of the normalised perpendicular bisectors. The values of each vote's distribution, in accumulator space, are weighted by $\omega(\theta)$. Similar circle parameterisations accumulate votes to form heavily weighted distributions in local regions of space. Outliers are separated from this distribution by their poor estimates for r. After several temporal samples, the maximum vote in 3D space is deemed to represent the best fit circle parameters. The centre of this circle corresponds to the pivot point estimate. The use of Gaussians, rather than points, to cast votes in the accumulator space enables rapid vote accumulation without demanding a large number of temporal intervals to be sampled.

Algorithm 4: Linear accumulation method

Our fourth algorithm is an adaptation of algorithm 3, again requiring three temporal samples of the feature polygon (at time instants $[t, t + \frac{\Delta}{2}, t + \Delta]$). We reconstruct the two chords of the circle as before, and compute the two perpendicular bisectors. However we do not intersect the two bisectors to obtain a single pivot point centre and radius as with algorithm 3. Rather, we densely sample the distribution of points which lie both within the bounds of the video frame, and upon the infinitely long lines congruent with the two perpendicular bisectors. Each of these points is regarded as a potential pivot point, and cast into a 2D accumulator array. As with algorithm 3, the votes are weighted according to the magnitude of rotary motion using the functional of equation 7.11. Over multiple temporal samples, votes accumulate in this space creating maxima around the best estimates for pivot point location.

7.2.4 Summary of Algorithms

We have described four algorithms for pivot point recovery: the former two driven by the closed form solution of eigen-problems, and the latter two driven by iterative,

Hough-like evidence gathering procedures. All algorithms can be applied to estimate motion between two instants t and $t + \Delta$ (i.e. over a single interval). Under zero noise (i.e. synthetic) conditions all algorithms recover the pivot point location exactly. However the presence of such noise typically renders single interval estimates unusable, and as we will show, performance of each of the four algorithms degrades differently as noise increases.

To improve the accuracy of the estimate under noise, we have described how multiple temporal intervals may be examined simultaneously using each of the four algorithms. In the case of the eigen-problem solutions, we have described a method of combination for multiple frame pairs using a weighted average (where greater credence is attributed to pivot measurements resulting from larger rotary motion). This works well for algorithms 1 and 2, however the approach is impractical for algorithms 3 and 4. In the case of these evidence gathering approaches, the accumulator space is instead populated with multiple "votes" to create an improved estimate over multiple temporal intervals.

We now present the results of comparing the performance of each algorithm, using both real and synthetic data.

7.2.5 Comparison of Pivot Recovery Algorithms

For the purposes of algorithm comparison and evaluation we constructed a test rig (shown in Figure 7-12), and used the Computer Vision component of Chapter 6 to track the planar motion of the several coloured rectangles pivoting upon one another. This assembly was mounted upon a sliding base, which allowed the rig to translate. We filmed 800 frames of the rig being manipulated in a random fashion by a human operator; these form the *CONTRAPTION* sequence which we used as the sample of "real" data when evaluating the four algorithms. The physical pivot points in the sequence were also manually located via a point-and-click operation to provide a ground truth for *CONTRAPTION*, which we used to compare the performance of the algorithms.

Behavioural Predictions

All the algorithms described use samples of either two or three frames in the sequence (spanning a temporal interval $[t, t+\Delta]$), over which a single estimate of the pivot point location is derived. Recall that in cases of slight rotary motion, the estimated pivot is likely to be in error since very small movements of the polygons (perhaps, due to noise) will cause large shifts in the predicated pivot positions. By contrast, we predict large rotations should cause the estimated pivot location to be much more robust to noise. Therefore, our first prediction is that the estimation of the static pivot should improve in accuracy when we use larger temporal intervals (Δ), which are more likely

to span movements with large rotary components. Second, when the estimates from multiple temporal intervals are combined together, we predict the system will perform with greater accuracy. Third, we predict that increasing levels noise in the positioning of the tracked polygons will cause the estimated pivot location exhibit increasing levels of error.

Our three test variables are therefore temporal interval size (Δ), the number of temporal intervals to be combined in producing the estimate, and the level of measurement noise in the positions of polygon vertices.

Variation of temporal interval size (Δ)

We first applied each of the algorithms to a synthetic data set, in which a polygon from the *STAIRS* sequence was subjected to rotation within 70° over 100 frames. All four algorithms recovered the pivot point exactly when examining single temporal intervals (subject to some very small numerical error), regardless of the temporal interval size (Δ) used. Figure 7-3 gives a representative result.

We then assessed performance on real data, by applying each algorithm to features corresponding to the white and pink slabs within the *CONTRAPTION* sequence. We tested only a single temporal interval within this sequence, and examined the effect of varying the temporal interval size (Δ). For all algorithms, as we increased Δ, the estimated location of the pivot point tended toward the manually specified ground truth. Figure 7-4 (right) gives a representative plot showing algorithm two's performance on this sequence. An optimal value for Δ is video dependent, since such a choice is determined by the nature of motion within the temporal window spanned by $[t, t + \Delta]$. However the trend suggests that higher values of Δ produce improved results, and our experimentation on other slab combinations in the *CONTRAPTION* sequence led to similar conclusions. Our empirical evaluation suggests that a temporal interval size of 25 frames is suitable for accurate pivot point recovery on this sequence. Keeping the interval constant at 25, we again applied each of the algorithms to the *CONTRAPTION* sequence. The resulting error (Euclidean distance between the estimated and ground truth pivot locations) is shown in Figure 7-4, left. Algorithm 2 exhibited superior performance (i.e. minimum Euclidean distance between estimated and ground truth pivot points) relative to the other algorithms.

Combining multiple temporal intervals

The scattering of individual pivot estimates for real data clearly demonstrates the need for combining estimates from multiple temporal intervals (Figure 7-4, bottom). Individual estimations form an approximately Gaussian distribution of pivot points, the

mean of which closely approximates the ground truth location for the physical pivot. As the number of temporal intervals used to create a pivot estimate increases, a smaller error (Euclidean distance between the estimated and ground truth pivot points) was observed for all four algorithms. However the improvement was most notable for the evidence gathering algorithms (3 and 4). Algorithms 1 and 2 required fewer temporal intervals than algorithms 3 and 4 to produce reliable mean estimates of the pivot location. A likely explanation is that the vote accumulation approaches adopted by the latter two algorithms require a larger number of temporal samples to be recorded in the accumulator space before maxima begin to emerge above the level of noise.

Impact of tracker noise on performance

Although algorithm 2 exhibited superior accuracy for the *CONTRAPTION* real video sequence, this is by no means conclusive since different video sequences may exhibit differing degrees of noise (so affecting the accuracy of pivot point recovery). We therefore created a synthetic simulation of two articulated features — the positions of polygon vertices in this simulation were subjected to zero centred, additive Gaussian noise $G(0, \sigma)$. The performance of each algorithm was compared under increasing standard deviations σ of this additive noise. The temporal interval size (Δ) was held constant at 25 frames (1 second) during these experiments, and the number of temporal intervals tested was held constant at 74. Thus pivot point estimation was based on 100 frames (4 seconds) of data.

Figure 7-2 (left) shows how increasing noise causes an increase in error (measured as the Euclidean distance between measured and ground truth pivot locations) for all four algorithms. By inspection we make the qualitative observation that algorithm two exhibits superior robustness to such noise. We can quantify this robustness using Student's t-test (Figure 7-2, right). Student's t-test [152] is a standard method used to determine, to a specified level of significance, whether two distributions (in our case the ground truth and estimated pivot distributions) are significantly different. The process involves deriving a scalar "t-value" from the means and variances of both distributions. This "t-value" is compared with a corresponding value from a statistical table representing a particular confidence level. If the measured "t-value" is greater than the tabulated value, then the distributions are different.

In our experiments we increased the level of noise (σ) until we were "significantly sure" (95% certain) that the distributions differed. We entered the t-table at this level of certainty, and obtained a threshold t-value of 2.02. Thus the value of σ that caused the measured t-value to rise above 2.02 (marked as a black line in Figure 7-2), corresponds to the level of noise at which the algorithm became unreliable. The results confirm our

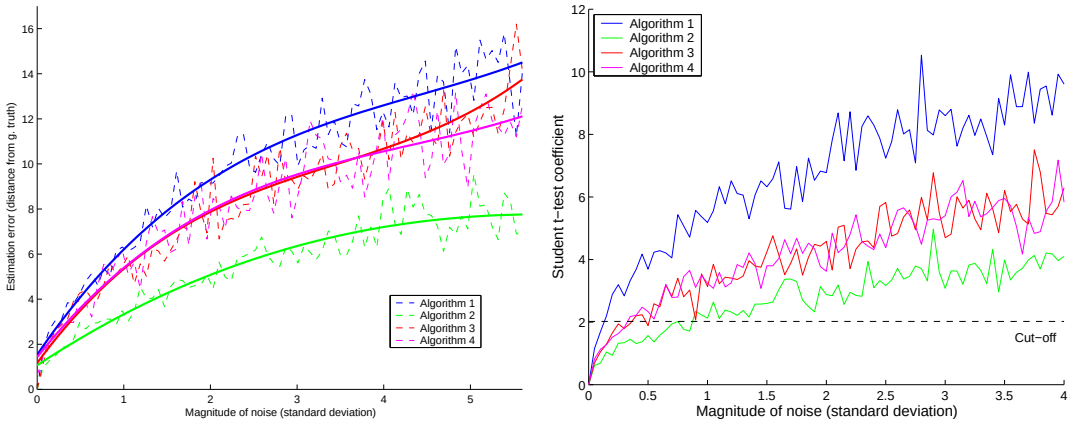


Figure 7-2 Comparing the performance of the four pivot point location algorithms using simulated data. The polygon vertices were subjected to zero mean Gaussian noise of increasing standard deviation, resulting in increasing levels of estimation error when recovering the pivot point — error measured as the Euclidean distance between estimated and ground truth pivot (left). The trend is plotted as a solid line, measurements as a dotted line. The test ran over one hundred simulated frames, sampling frame pairs at time intervals of 25 frames (1 second). Student's t-test (right) was employed to decide the level of noise at which the distribution of estimated pivots differed from the ground truth significantly (i.e. with 95% confidence). This threshold value (derived from statistical tables) is plotted in black. Both graphs show algorithm 2 to out-perform the others, the latter graph demonstrates tolerance of up to one standard deviation of noise.

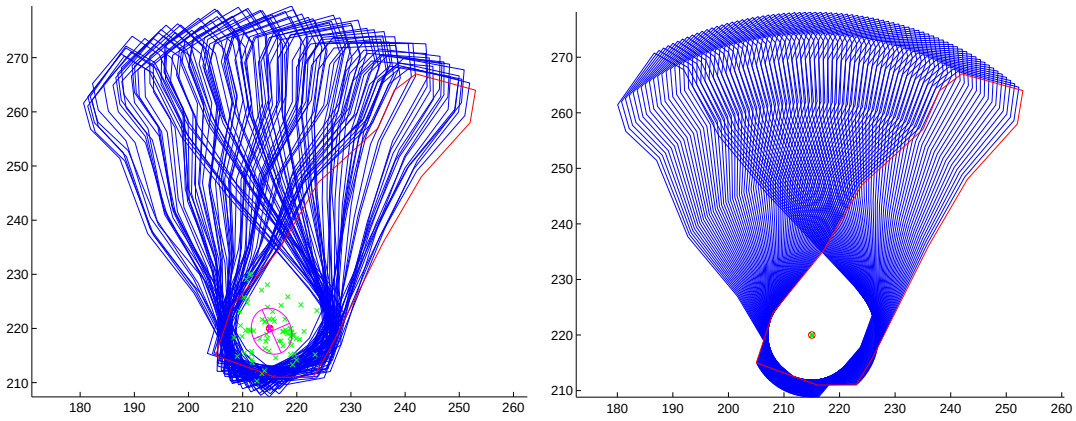


Figure 7-3 Examples of pivot point recovery using algorithm two, forming part of the synthetic test case used to generate Figure 7-2. A limb (feature E, original position in red — see Figure 6-2) from the *STAIRS* sequence was subjected to synthetic rotation within 70° for 4 seconds (100 frames). Left: polygon vertices perturbed by two standard deviations of zero centred Gaussian noise, produces a small cluster of potential pivots the mean of which closely corresponds to the ground truth (two standard deviations plotted in magenta). Right: in the zero noise case the pivot is recovered exactly — this is true for all four algorithms.

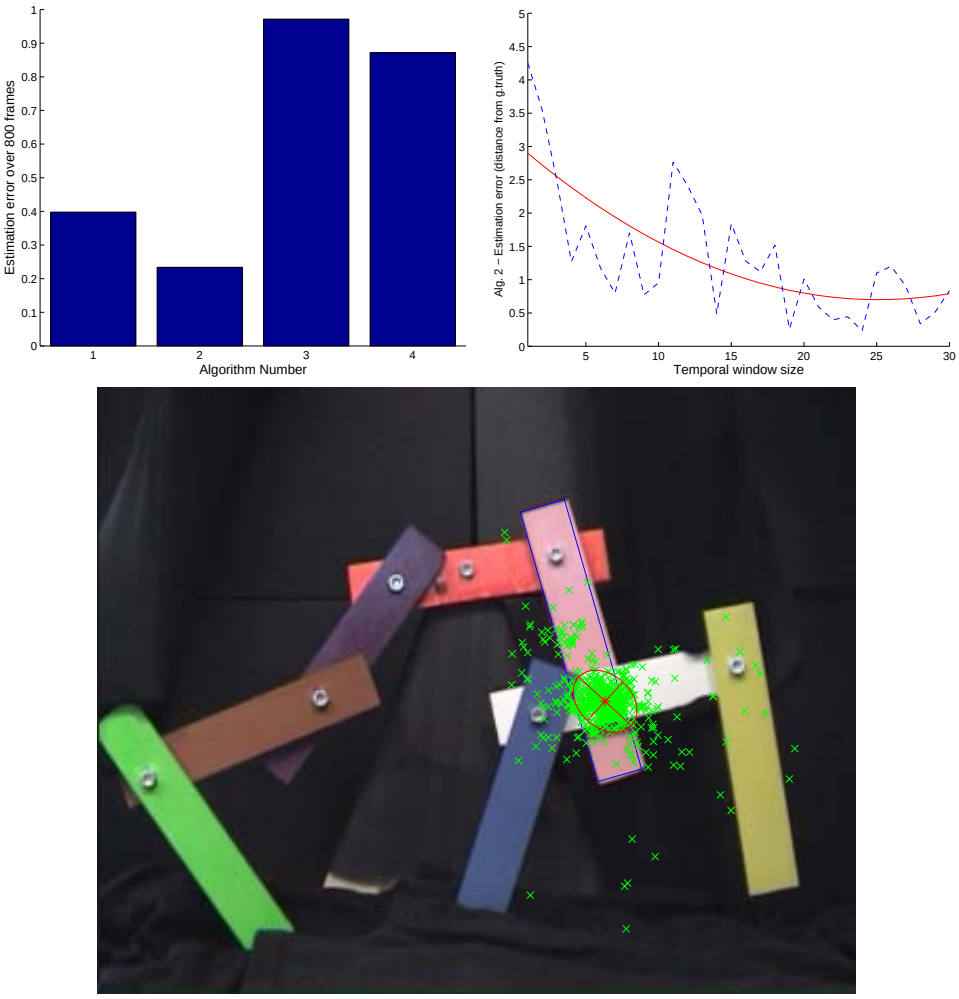


Figure 7-4 Applying the algorithms to 800 frames of the *CONTRAPTION* sequence, specifically to determine the pivot point between the pink and white slabs. Bottom: The estimated pivot points due to algorithm 2, between multiple frame pairs sampled at an interval of $\Delta = 25$; the error distribution of estimated pivots about the ground truth approximates a zero mean Gaussian. The estimated (weighted) mean pivot location is very close to the ground truth (red star). Top left: per algorithm comparison of errors in pivot position after 800 frames. Top right: Showing that pivot estimation error declines as the sampling distance between frames in the frame pair increases (algorithm 2).

earlier observations; algorithm two exhibits superior robustness to noise — differing from the ground truth significantly at around $\sigma = 1$. The values for σ at which the other algorithms became unreliable were, at best, a third of this value.

7.3 Recovering Hierarchical Articulated Structure and Pose

Algorithm 2 appears to estimate pivot points with superior accuracy and robustness than the other proposed algorithms. We therefore applied this algorithm to recover one pivot point for each ordered pair of tracked features in the video (denoting a sin-

gle ordered pair as $\{A, B\}$). Observe that due to noise in the estimation process, the computed pivot of A about B, and B about A, may not be identical; thus the pivots resulting from pairing $\{A, B\} \neq \{B, A\}$. Computing a pivot for each ordered pair of features requires $2C_2^n$ ordered pairings for n tracked features; this is not a large number for typical sequences (for example, 132 for the *STAIRS* sequence), and this computational load can be further reduced by only testing feature pairs whose polygons intersect for a significant proportion of their lifetime.

Many of the computed pivots do not correspond to physically existing articulations between feature pairs, and should be discarded. We decide which to discard by evaluating a "quality" metric for each estimated pivot, and discarding pivots that score poorly. This filtering process leaves us only with true pivot points, and so with a final articulated structure for the tracked subject. We can may then fit the original tracked polygons to the derived articulated structure, so generating a pose vector for each frame.

Quality Metric

If two features A and B are articulated, then in cases where we can observe those features with no measurement noise, the following properties hold:

1. The world coordinate position of the pivot between features A and B should, at all times, lie within the area of intersection of both features A and B.

2. The motion of feature A relative to feature B should be well described by a rotation of A in the reference frames of B, about a static pivot.

3. The motion of feature B relative to feature A should, similarly, be well described by a rotation of B in the reference frame of A, about a static pivot.

4. The world coordinate position of the pivot point of A on B and the pivot point of B on A should at all times be coincident.

Our assumption is that if these properties are violated, then features are not articulated. Of course estimation errors may cause violation of these properties, and so they are better viewed as heuristics — compliance with these heuristics indicates a "good quality" estimated pivot point, and so a likely articulation between A and B. We construct the quality function for a potentially articulated feature pair $\{A, B\}$ as follows.

For a well estimated pivot point $\underline{p}$ in $\underline{\underline{F}}_{B(t)}$, the motion of $\underline{\underline{A}}'_t$ over time should be well described by a rotation $\underline{\underline{R}}_{\theta(t)}$ about $\underline{p}$; thus we desire a small residual r_1 averaged over

all n frames:

$$r_1 = \frac{1}{n}\sum_{t=2}^{n}\left|\underline{\underline{A}}'_{t-1} - \underline{\underline{R}}_{\theta(t)}(\underline{\underline{A}}'_t - \underline{p}\underline{1}^T) + \underline{p}\underline{1}^T\right| \tag{7.22}$$

This r_1 forms the first term in the quality function.

Now consider that the world coordinates of the pivot of A with respect to B at time t (which we write as $\underline{p}_{A(t)}$) and the pivot of B with respect to A (which we write as $\underline{p}_{B(t)}$) should be coincident under zero noise conditions. Thus we desire a small residual r_2 averaged over all n frames:

$$r_2 = \frac{1}{n}\sum_{t=1}^{n}\left|\underline{p}_{A(t)} - \underline{p}_{B(t)}\right| \tag{7.23}$$

The position of the pivot $\underline{p}_t$ between features A and B at time t is computed as $\underline{p}_t = \frac{1}{2}(\underline{p}_{A(t)} + \underline{p}_{B(t)})$.

Finally, we add a penalty term for the distance that the pivot lies outside the area of intersection of A and B; specifically the Euclidean distance between $\underline{p}_t$ and the closest pixel within the intersection area. This distance is averaged over time to obtain penalty term Φ. We write the complete quality function $Q[A, B]$ as a sum of weighted terms:

$$Q[A, B] = \exp(-k(r_1 + r_2 + \Phi)) \tag{7.24}$$

where k controls the decay of a pivot point's viability as this sum of error terms increases, we find a value of around $k = 0.1$ suitable for our source data.

Note that our system does not yet cater for cases where features are rigidly joined, with little or no rotary motion present. Currently such features are classified as non-articulated, and we assume for the time being that such groups of features have been segmented as a single rigid component.

Recovering Pose Vectors

The algorithms described thus far are capable of recovering general articulated motion of features in the plane. For ease of representation and later manipulation, we assumed that such features form a hierarchical structure. Although this is a specialisation of the system, most common subject matter, for example people, are admitted under this model. It is a simple matter to detect the presence of cycles in the recovered articulated structure, and currently we return an error in such situations, specifying that time and pose cues can not be applied to that particular object.

Like most hierarchical pose representations, we specify the position of a node, say an arm, in relation to a component higher in the hierarchy, which in turn may be specified in relation to a component higher still in the hierarchy, and so on recursively to the root node. This can produce positional inaccuracies in the leaf nodes due to a build up of multiplicative errors as we descend the hierarchy. We therefore carefully choose a feature to serve as a root node, such that this choice minimises the longest hop count from the root node to a leaf node. In the case of the *STAIRS* sequence, the choice of root feature is the torso. In the *CONTRAPTION* sequence, the root is the red slab.

We wish to construct a numerical representation of the subject's pose at each time instant t, by searching for a best fit of the recovered articulated structure to the set of tracked features. Recall that each feature was tracked independently in Section 6.3.2 — the resulting tracked polygons each represent the "best" estimate obtainable for a feature's position in the raw footage, given that no global model or constraints were available at the time. By combining the raw feature position estimates with the recovered articulated model, we not only produce a numerical "pose vector" for each time instant, but also impose physical constraints to further refine the accuracy of tracked polygons. We form an initial estimate for this "pose vector", then search for an "optimal" pose local to this using a Nelder-Mead search [114]. For our purposes, the optimal pose is the configuration which minimises the Euclidean distance between the tracked feature polygon vertices, and the vertices generated by re-synthesising the feature polygon positions from the putative "optimal" pose.

The structure and initial estimate of the pose vector is formed as follows. The first four elements of the vector are a representation of the four LCAT parameters which transform the root feature from its initial position (in the first frame), to its current position (in frame t). This is extracted directly from the feature tracker. The translational component of the LCAT is converted from Cartesian to polar form, thus the first four elements of the pose vector are $[\phi,\ r,\ \theta,\ s]$; where ϕ is the direction of translation, r is the translation magnitude, θ is the orientation of the root object and s is a uniform scaling.

$$\underline{V}(t) = \left[\begin{array}{cccccccc} \phi & r & \theta & s & \theta_1 & \theta_2 & \ldots & \theta_n \end{array} \right]^T \tag{7.25}$$

Features in the structure are visited via a pre-order traversal of the hierarchy. Each subsequent entry in the pose vector specifies the angle at which the newly visited feature is orientated, relative to its parent feature (i.e. about the parent-child pivot point). This angle is relative to the child's position in the first frame; at time $t = 1$ all such angles will be zero — the angle in the pose vector encodes the change in orientation between frame 1 and frame t. In our initial estimate, this angle is extracted from the

LCAT between the child's position relative to the parent in frame 1 and frame t. In equation 7.25 each of these angles is denoted θ_i where $i = [1, n]$, n being the number of features in the hierarchy, and i being an index into the ordering in which the feature hierarchy is traversed.

7.4 Temporal Re-sampling

The process of Section 7.2 results in a series of vectors $\underline{V}(t)$ for each frame t. These vectors form individual points in a high dimensional space, representing the subject's pose at a given instant. The trajectory of these points encode the pose of the subject moving over time. Manipulating this pose space gives rise to novel time and pose cues which may be used to emphasise motion in our system. We may choose to manipulate only a local region of one dimension of this space; affecting the position of one joint over a small temporal window. We term these "local" pose transformations, and show in the next subsection that cartoon "anticipation" effects can be created by this class of transformation. We may also choose to scale or deform one or more dimension of this pose space globally, i.e. over all time. This has the effect of modifying the basis of the pose space. We refer to these as "global" pose transformations, and discuss these in subsection 7.4.2.

7.4.1 Temporally Local transformation (anticipation)

Anticipation is an animation technique applied to objects as they begin to move; the technique is to create a brief motion in the opposite direction, which serves to emphasise the subsequent large scale movement of an object (Figure 7-5). The anticipation cue communicates to the audience what is *about to* happen. Anticipation acts upon a subject locally — only within a temporal window surrounding the beginning of the movement to be emphasised, and only upon the feature performing that movement.

We have implemented anticipation as a 1D signal filtering process. Each individual, time varying component of the pose vector $\underline{V}(.)$ (for example, the angle a metronome beater makes with its base) is fed through an "anticipation filter", which outputs an "anticipated" version of that pose signal (Figure 7-5). The filter also accepts six user parameters which control the behaviour of the anticipation motion cue. The filtering process operates in two stages. First, the 1D signal is scanned to identify the temporal windows over which anticipation should be applied. Second, the effect is applied to each of these windows independently.

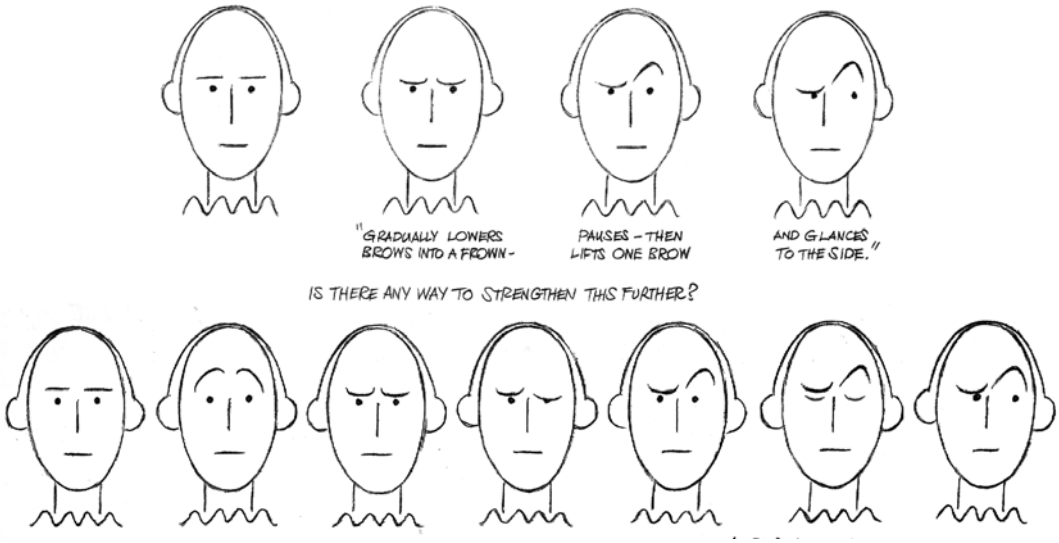


Figure 7-5 Illustrating how animators can apply anticipation to emphasise motion, in this case a Disney-style facial animation (reproduced from [169]).

Identifying Temporal Windows for Anticipation

Given a 1D input signal, the filter first identifies temporal windows for application of anticipation. These are characterised by the presence of high acceleration magnitudes (above a certain threshold), which exist for a significant number of consecutive frames (a further threshold) in the signal. These two thresholds form part of the set of user parameters that control the effect. This process allows us to identify a set of temporal windows corresponding to large motion changes, which an animator would typically emphasise using anticipation. A high acceleration magnitude may or may not generate a change of direction in the signal and, after numerous conversations with animators [130], we have determined that the manifestation of the anticipation cue differs slightly between these two cases:

Case 1. First, consider the case where acceleration causes a change of direction in the 1D signal; for example, a pendulum at the turning point of its swing. Regardless of the acceleration magnitude of the pendulum beater (which may rise, remain constant, or even fall during such a transition), the anticipation effect is localised to the instant at which the beater changes direction i.e. the turning point of the signal; the *minimum of the magnitude of the first derivative with respect to time*. In the case of the *METRONOME* sequence (Figure 7-15), a brief swinging motion would be made, say to the left, just prior to the recoil of the metronome beater to the right. The object then gradually "catches up" with the spatiotemporal position of the original, un-anticipated object at a later instant.

Case 2. Now consider the second case where acceleration does not cause change of direction in the 1D signal; for example, a projectile already in motion, which acquires

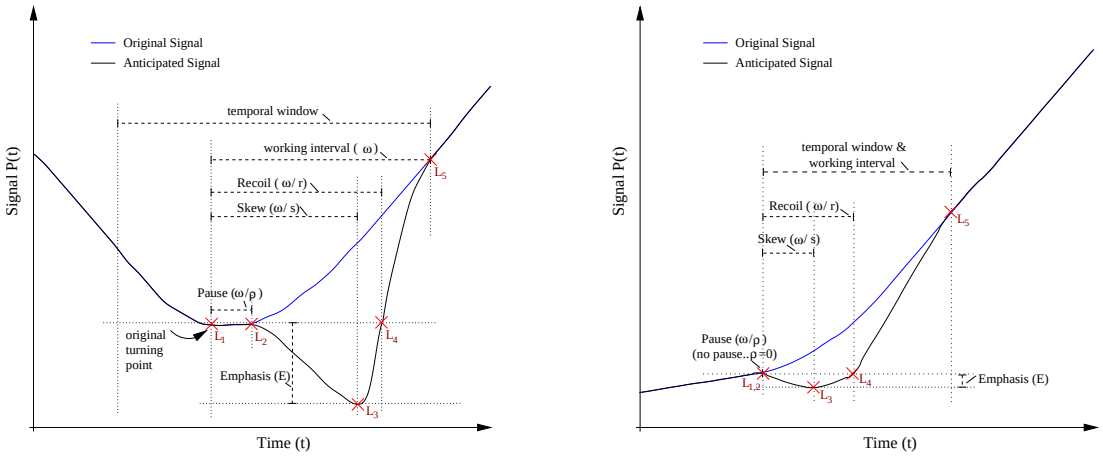


Figure 7-6 Schematic examples of the anticipation filter under case one (signal direction of motion changes) and case two (signal direction of motion unaffected). Case two has been illustrated with pause parameter $\rho = 0$. Section 7.4.1 contains an explanation of the user parameters ρ, s, r, and ϵ which influence the behaviour of the effect.

a sudden burst (or decrease) in thrust, i.e. a change in acceleration magnitude. The anticipation effect is manifested as a short lag just prior to this sudden acceleration change; i.e. at the *maximum in the magnitude of the third derivative with respect to time*. As with case 2, the projectile swiftly accelerates after anticipation to catch up with the spatiotemporal position of the original, unaffected projectile. Interestingly a projectile moving from rest is equally well modelled by either the first or second case, since the locations of zero speed (minimum first derivative) and maximum acceleration change (maximum third derivative) are coincident.

Synthesising Anticipation within a Temporal Window

Each temporal window identified for application of the anticipation cue is processed independently, and we now consider manipulation of one such a window. The first task of the "anticipation filter" is to scan the pose signal to determine whether a change of direction occurs within the duration of the temporal window. This test determines which criterion from the respective case (1 or 2) is used to determine the instant at which anticipated motion should be "inserted" into the sequence; we denote this time instant by τ. We define a temporal "working interval" as the time window within which we pose is varied from the original signal, in order to introduce anticipation. This working interval extends from time τ to the end of the temporal window, which we write as $\tau + \omega$. In all cases the direction of the anticipatory motion will be in opposition to the direction in which acceleration acts. We refer the reader to Figure 7-6 to assist in the explanation of the subsequent signal manipulation.

We create the anticipation effect by modifying the 1D pose signal to follow a new curve, interpolating five landmark points in space $[t, P(t)] \in \Re^2$, where $P(t)$ indicates the value of the pose signal at time t. Aside from the two parameters used to control activation of the effect, there are four user parameters ρ, s, r, and ϵ (where $\rho \leq s \leq r$). These influence the location of the five landmark points $[\underline{L}_{1\ldots5}]$, which in turn influences the behaviour of the anticipation. We now explain the positioning of each of the five landmarks and the effect the user parameters have on this process. Throughout, we use notation $p(t)$ to indicate the original (unanticipated) pose signal at time t, and $p'(t)$ to denote the new, anticipated signal.

$\underline{L}_1$. The first landmark marks the point at which the original and anticipated pose signals become dissimilar, and so $\underline{L}_1 = (\tau, p(\tau))^T$. Recall τ is the determined by the algorithm of either case 1 or 2, as described in the previous subsection.

$\underline{L}_2$. At the instant τ, a short pause may be introduced which "freezes" the pose. The duration of this pause is a fraction of the "working interval" — specifically ω/ρ frames, where ρ is a user parameter. The second landmark directly follows this pause, and so $\underline{L}_2 = (\tau + \omega/\rho, p(\tau))^T$.

$\underline{L}_3$. Following the pause, the pose is sharply adjusted in the direction opposite to acceleration, to "anticipate" the impending motion. The magnitude (E), and so the emphasis of, this anticipatory action is proportional to the magnitude of acceleration: $E = \epsilon|\ddot{p(\tau)}|$. Here ϵ is a user parameter (a constant of proportionality) which influences the magnitude of the effect. A further user parameter, s, specifies the instant at which the anticipation is "released" to allow the movement to spring back in its original direction. We term s the "skew" parameter, since can be used to skew the timing of the anticipation to produce a long draw back and quick release, or a sharp draw back and slow release. Referring back to Williams' guidelines for anticipation, one would typically desire the former effect ($s > 0.5$), however our framework allows the animator to explore alternatives. The third landmark is thus located at the release point of this anticipated signal, and so $\underline{L}_3 = (s, E)^T$.

$\underline{L}_4$. The rate at which the feature springs back to "catch up" with the unanticipated motion is governed by the gradient between the third and fifth landmarks. This can be controlled by forcing the curve through a fourth landmark $\underline{L}_4 = (\tau + \omega/r, p(\tau))^T$.

$\underline{L}_5$. Finally the point at which the anticipated and original pose signals coincide is specified by the final landmark, $\underline{L}_5 = (\tau + \omega, p(\tau + \omega))^T$.

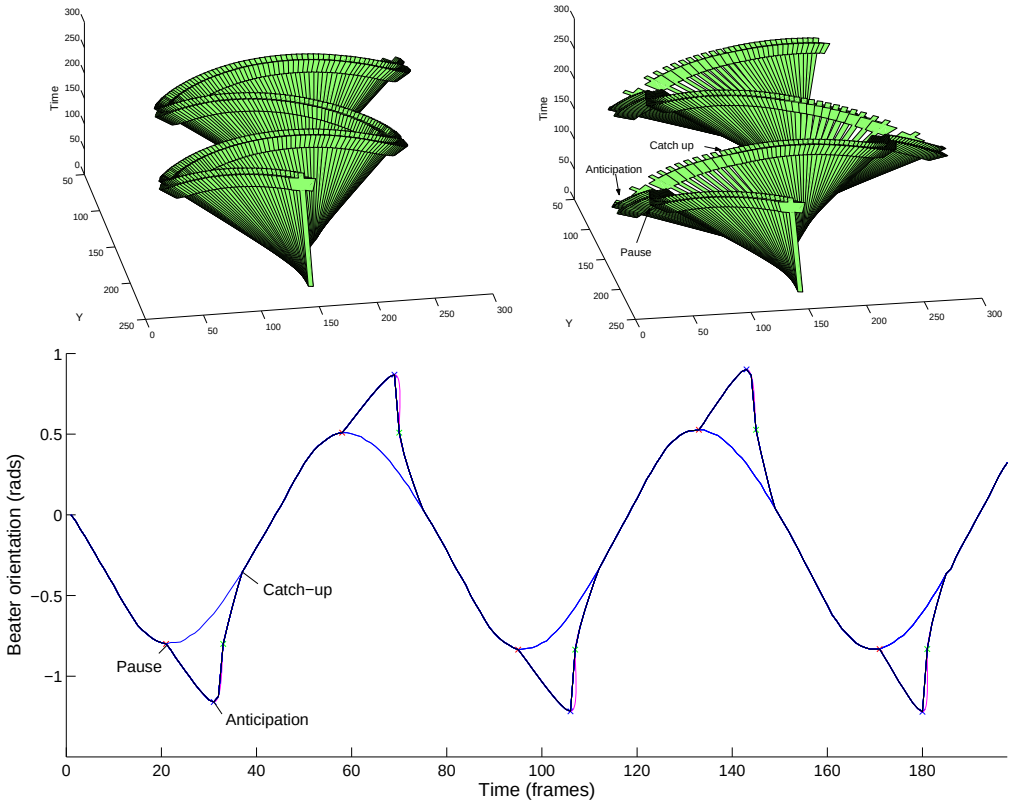


Figure 7-7 Top: Time lapse representation of the beater in the *METRONOME* sequence, before (left) and after (right) application of the anticipation filter to the pose vectors. Bottom: Visualisation of the 5^{th} element of the *METRONOME* pose vector (encoding the angle between metronome beater and body), before (blue) and after (black) passing through the anticipation filter with $\rho = 0$, $\epsilon = 150$, $s = 0.8$, $r = 0.9$. Green and red vectors indicate the gradients at $\underline{L}_4$ and $\underline{L}_5$ used to interpolate those two control points.

The anticipated signal $p'(t)$ at any time t within the temporal window is created by interpolating these five landmarks, the following manner. In the first stage of the anticipation, landmarks $\underline{L}_1$ and $\underline{L}_2$ are linearly interpolated to create a simple pause in the signal. The pause component of the anticipation is created using the parametric line $\underline{\pi}_1(c)$ where $c = [0, 1]$:

$$\underline{\pi}_1(c) = \underline{L}_1 + c(\underline{L}_2 - \underline{L}_1) \tag{7.26}$$

The second stage of the anticipation is the movement in the opposite direction to the impending motion. We interpolate landmarks $\underline{L}_2$, $\underline{L}_3$ and $\underline{L}_4$ using a cubic Catmull-Rom spline [51], creating a smooth transition between the pause stage and the anticipatory

action. Using notation $\underline{\pi_2}(c)$ we have:

$$\underline{\pi_2}(c) = \begin{bmatrix} \underline{L}_2 & \underline{L}_2 & \underline{L}_3 & \underline{L}_4 \end{bmatrix} \begin{bmatrix} -0.5 & 1 & -0.5 & 0 \\ 1.5 & -2.5 & 0 & 1 \\ -1.5 & 2 & 0.5 & 0 \\ 1 & -0.5 & 0 & 0 \end{bmatrix} \begin{bmatrix} c^3 \\ c^2 \\ c \\ 1 \end{bmatrix} \tag{7.27}$$

where the 4×4 matrix term is the Catmull-Rom cubic blending matrix. Finally, we interpolate between landmarks $\underline{L}_4$ and $\underline{L}_5$ using a cubic Hermite spline [51]. This family of splines ensures C_1 continuity at both landmarks, so blending the anticipated signal smoothly with the original signal. The Hermite curve requires specification of position and velocity at both landmarks. Again, using c as a dummy parameter, this segment of the anticipation is described by $\underline{\pi_3}(c)$:

$$\underline{\pi_3}(c) = \begin{bmatrix} \underline{L}_4 & \underline{L}_5 & \dot{\underline{L}}_4 & \dot{\underline{L}}_5 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} c^3 \\ c^2 \\ c \\ 1 \end{bmatrix} \tag{7.28}$$

where the 4×4 matrix term is the Hermite cubic blending matrix. We can obtain the velocity at $\underline{L}_5$ using a simple finite difference approach on the original, discrete signal at $p(\tau + \omega)$. The velocity at $\underline{L}_4$ is obtained from the partial derivative of equation 7.26 with respect to c.

Figure 7-6 gives two schematic examples of signals undergoing anticipation in cases 1 and 2. Figure 7-7 (bottom) shows the original and anticipated signals used for rendering the *METRONOME* sequence, a time lapse representation of which is given in Figure 7-7 (top). The reader is referred to Appendix C for this and other rendered video clips.

7.4.2 Temporally Global transformation (motion exaggeration)

A portrait caricaturist will emphasise unusual or characteristic features of a subject's face. Likewise, cartoonists will emphasise unusual motion characteristics, for example a limp, exhibited by their subjects [98]. Such characteristics may be considered to be outliers in the cartoonist's mental model, for example, of people; the more an individual's characteristics diverge from the "norm", the more those characteristics tend to be emphasised.

Although our system is not equipped with a model of the population, we can learn the pattern of simple repetitive motions made by a tracked subject, and exaggerate variations in this pattern. Periodic motion, such as human gait, causes pose vectors to trace

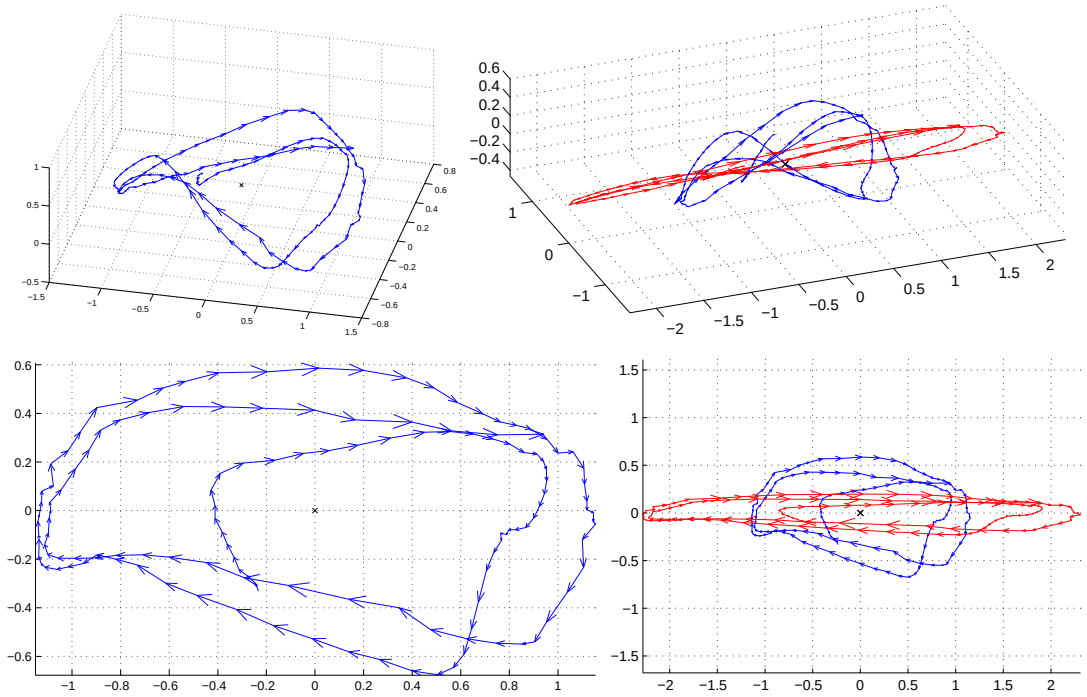


Figure 7-8 Visualisation of the pose space for *STAIRS* before (blue) and after (red) motion exaggeration with $\mathcal{F} = 2$. Graphs produced by projecting the high dimensional space down into first 3 (above) and 2 (below) principal axes, centring the mean upon the origin. Arrowheads indicate the direction of time.

a cyclic trajectory within a subspace of the pose space (consisting of all dimensions minus the first two, which represent translation of the root feature). If several cycles of motion exist within the video sequence (for example, *STAIRS*), then it is possible to reliably compute a mean point in the pose space. Performing a global scaling transformation on the space, with centre of projection at the mean, serves as a basis for exaggerating variations in the motion over time.

This simple approach not only emphasises variations, for example in gait, but also any noise introduced by the tracking process. In our motion cartooning application we desire exaggeration of only the important, principal, motions leaving noise components unchanged. Our strategy is to perform a principal component analysis (PCA) of the pose samples taken over duration of the video sequence, to isolate the sub-space within which the motion principally varies. We produce an eigenmodel of all pose samples $\underline{V}(t)$ over time (yielding a mean pose $\underline{V}_\mu$, a collection of column eigenvectors $\underline{\underline{U}}$ and a diagonal matrix of eigenvalues $\underline{\underline{\Lambda}}$). By scaling the basis specified in each eigenvector by a multiplicative factor $\mathcal{F}$ of its corresponding eigenvalue, we produce a novel set of pose vectors $\underline{V'}(t)$ in which motion appears to be exaggerated.

$$\underline{V'}(t) = \underline{\underline{U}}^{+} \mathcal{F} \underline{\underline{\Lambda U}} (\underline{V}(t) - \underline{V}_\mu) + \underline{V}_\mu \tag{7.29}$$

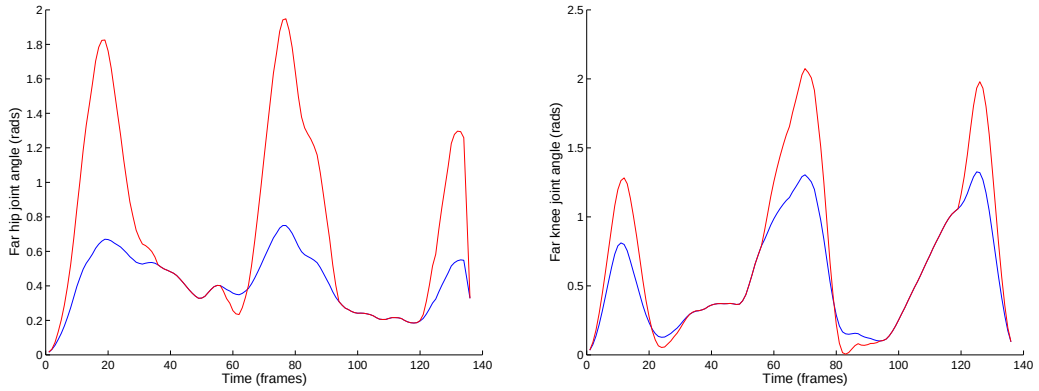


Figure 7-9 Visualisation of signal within two dimensions of the pose space for *STAIRS*, corresponding to orientation of the far hip joint (left) and of the far knee joint (right). Observe the periods of invariance between the original signal (blue) and exaggerated signal (red), generated by the constraints imposed by the animator.

Figure 7-15, sequence A superimposes the modified positions of feature polygons when $\mathcal{F} = 2$ over the original video.

Introducing Physical Constraints

While sequence A in Figure 7-15 demonstrates exaggerated motion, the manipulation of the pose does not yet take into account physical constraints upon the articulated structure. For example, the feet appear to glide over the steps, and do not appear to interact with the ground surface in a naturally plausible manner. Constraints to counter this behaviour must be interactively specified by the animator; we allow users to select temporal windows over which certain features (for example the feet), are to remain coincident with the original sequence. Although a complete solution to the problem of motion exaggeration under constraints lies within the realms of inverse kinematics, we have found that our combination of localised input from the animator with the global transformation of the motion exaggeration, to produce aesthetically acceptable output.

To simplify matters we state that if, at a given instant, a feature is marked to be held in its original position, then all features between that feature and the root feature must also to match their original position. Thus we can derive a series of temporal windows for each feature during which their motion should not be emphasised.

We encode these temporal windows in a matrix of weights, which we will write as $\underline{\underline{\omega}}$. This matrix is formed as a concatenation of column vectors $\underline{\omega}(t)$, each corresponding to the pose vector $\underline{V}(t)$ at time $t \in [1, n]$.

$$\underline{\underline{\omega}} = \left[\begin{array}{cccc} \underline{\omega}(1) & \underline{\omega}(2) & \ldots & \underline{\omega}(n) \end{array} \right] \tag{7.30}$$

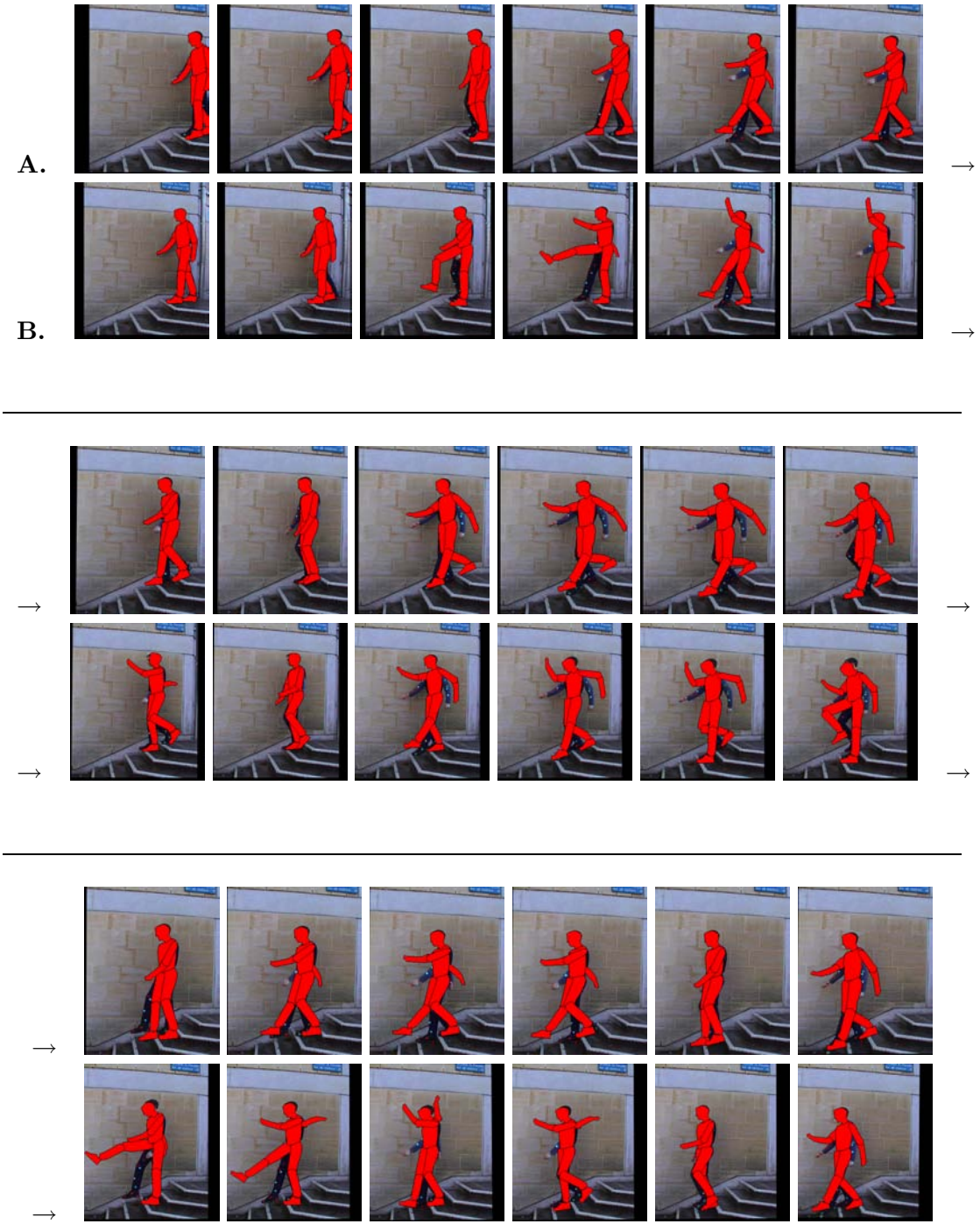


Figure 7-10 Stills from the source *STAIRS* sequence, with the positions of the exaggerated feature polygons overlaid. Motion has been exaggerated using physical constraints in sequence B, and without constraints in sequence A. Observe the improved placement of the feet in B. Approximately one still per five frames (see animations `videos/stairs_exaggerate` and `videos/stairs_exaggerate_polys`).

Figure 7-11 Four stills from the `videos/stairs_exaggerate` animation in Appendix C. We have introduced motion exaggeration into the *STAIRS* sequence, and applied moustache and pith helmet (using the rotoscoping features of the Video Paintbox, Section 8.4.3) to produce an animation reminiscent of Monty Python's Flying Circus [BBC 1969–74].

If, at particular instant t, we desire the i$^{\text{th}}$ component of the pose to follow its original rather than exaggerated trajectory, we set the respective matrix element $\omega_{t,i}$ to be zero. Consecutive zero valued elements within a particular row of $\underline{\underline{\omega}}$ therefore correspond to temporal windows during which the motion of a particular pose component should not to be exaggerated. We iterate through each row of $\underline{\underline{\omega}}$ in turn. Elements in the row are assigned values in proportion to their distance from the nearest zero valued element on that row. These values are normalised to rise to unity halfway between two given temporal windows. Each matrix element $\omega_{t,i}$ now contains a scalar weight, representing the degree of exaggeration to apply to each pose component i at each time t. To produce our exaggerated, yet physically constrained pose $\underline{V}''(t)$ at time t we linearly interpolate between the original pose ($\underline{V}$(t)) and the unconstrained, exaggerated pose ($\underline{V}$'(t), see equation 7.29) using:

$$\underline{V}''(t) = \underline{V}(t) + \underline{\omega}(t)(\underline{V}'(t) - \underline{V}(t)) \tag{7.31}$$

Figure 7-10 (sequence B) gives an example of the resulting animated sequence using the *STAIRS* data. As with sequence A, feature polygons have been superimposed on the original video to compare the two sequences. The animator has interactively specified various temporal windows within which the feet should coincide with the original data (whilst they are on the ground), and also specified that all features should start and end coinciding with the original data.

7.5 Video Re-synthesis

Once the pose vectors have been manipulated to introduce motion cues, it remains to re-synthesise the video sequence using the novel pose data. This involves painting each feature polygon with a suitable texture sampled from the video (and then composit-

Figure 7-12 Pose recovery and manipulation. Left: A frame from the *CONTRAPTION* sequence, with pivot points and articulated structure recovered automatically. The recovered pose vector is shown inset. Right: Overwriting the last dimension (angle between white and yellow slabs) of the first 50 frames with values from 0 to 2π in equal increments.

ing the polygons in the correct depth order, in accordance with the depth information recovered in Section 6.3.3). This re-texturing may introduce difficulties, since the modified pose may expose regions of features that were occluded in the source video.

Our solution to sampling texture is similar to that used in the occlusion buffer of Section 6.4.3. We sample as much unoccluded texture as possible at a time instant t, and then fill in remaining "holes" in the texture by sampling neighbouring frames in alternation, i.e. $t-1$, $t+1$, $t-2$, $t+2$, and so on. This gives a reasonable reconstruction of the feature texture local to time t, so allowing for some variation in the reconstructed texture due to temporally local illumination changes in the video. If any holes remain in the texture they are filled using the mean RGB colour of the recovered texture.

All that remains is to decide the instant t from which to begin sampling texture. In the occlusion buffer of Section 6.4.3, the feature to be re-textured invariably occupied the same spatiotemporal position as in the original video. Thus if we wished to re-texture an object in frame i, we began sampling texture at time $t = i$. In the case of time and pose cues, this is not necessarily true; the spatiotemporal position of a feature may differ considerably from its position in the original video, due to our pose manipulations. We therefore begin sampling texture from an instant t in the original video where the feature is in approximately the same position as the feature in the manipulated pose, to take into account local lighting changes. Furthermore, there may be many such instants in the original video and we should sample from the instant closest in time to the current frame being rendered; this permits local lighting to vary over time.

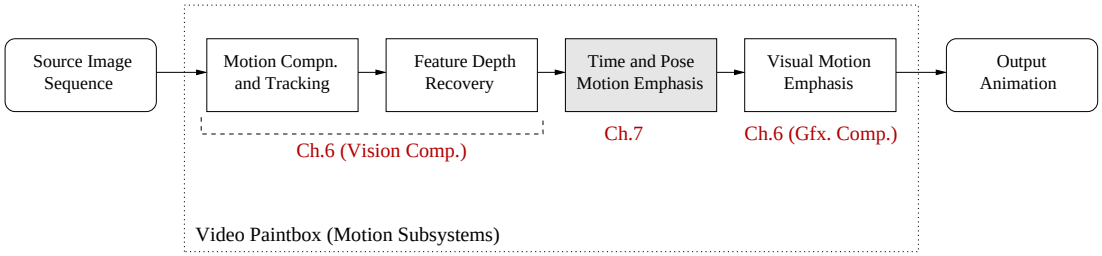


Figure 7-13 Schematic illustrating the flow of control and data in the Video Paintbox subsystems dealing with motion emphasis.

Fortunately t is straightforward to determine, since we have access to pose vectors which describe the feature's position in both the original and motion emphasised sequences. Suppose we wish to render a particular frame i. We first determine the feature's position at time i in the emphasised pose. We then examine each vector j of the original pose computing a spatial distance measure d_j — the mean Euclidean distance between the vertices of the original and emphasised feature at time j. We choose the start instant t as:

$$t = \text{argmin}_t(\alpha d_t + \beta|t - i|) \tag{7.32}$$

Choice of α and β depend on the rate of lighting variation in the video; in rapidly varying lighting conditions β should greatly outweigh α. For our data we have used $\alpha = 1$, $\beta = 0.1$. Figure 7-12 demonstrates the results of re-texturing the *CONTRAPTION* sequence following pose manipulation. We have simply overwritten the last dimension (angle between white and yellow slabs) of the first 50 frames with values from 0 to 2π in equal increments; this causes the slab to appear to spin about its pivot, whilst its motion remains coherent with the remainder of the contraption. Occlusions between the yellow and white slabs are correctly handled.

7.6 Integrating Time and Pose Cues within the Video Paintbox

Time and pose motion cues are applied directly to the output of the Computer Vision (Chapter 6), which tracks objects within a camera motion compensated version of the source video sequence. The result is a modified version of the video and associated tracked features, which are then passed to the Computer Graphics component of Chapter 6. Visual augmentation and deformation cues are then inserted into the animation — so completing the motion emphasis framework of the Video Paintbox. The benefit of this arrangement is that visual motion cues, such as object deformations, are seen to react to the changes in pose caused by effects such as anticipation. Figure 7-15 gives such an example where the *METRONOME* sequence has been subjected to the

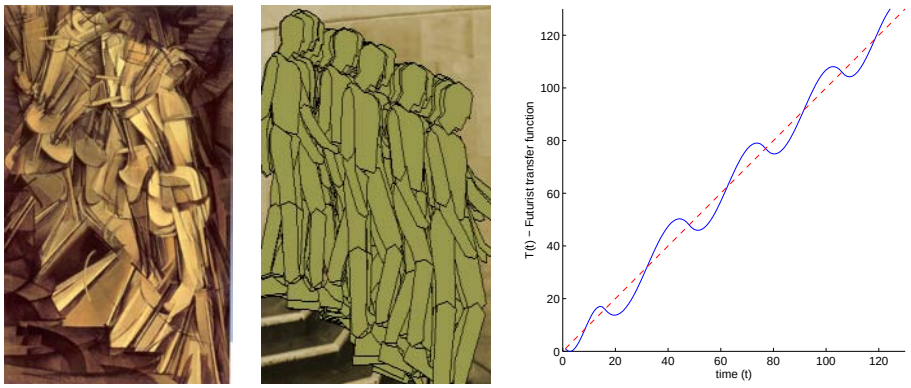


Figure 7-14 Toward Futurist-like rendering via irregular temporal sampling of the Video Paintbox output. Left: Duchamp's "Nude Descending a Staircase II" [1912], and samples of corresponding source photography by Muybridge [1872]. Middle: Our approximation to Futurist art using a temporal sampling strategy modelled upon Duchamp's work. Right: The transfer function $\mathcal{T}(.)$ which created our artwork in blue, with the standard (identity) transfer function in red.

anticipation process, after which a non-linear (velocity based) deformation has been used to emphasise drag.

7.7 Varying the Temporal Sampling Rate

Our explanation so far has assumed that the animator desires only to render pose vectors at uniform temporal intervals, and at precisely the same frame rate as the source video. However there is no technical reason why the animator should be restricted to this sampling strategy. As a final step we introduce a transfer function $\mathcal{T}(t)$ which accepts a time instant (i.e. a frame) to be rendered (t), and outputs the time instant of the animation to synthesise via our previously described framework. Standard rendering would simply require $\mathcal{T}(t) = t$. However, interesting effects can be created by choosing an alternative $\mathcal{T}(.)$.

The Futurist art movement created static artworks, many of which depict motion through the composition of time lapse images sampled at irregular intervals [81]. One classic example of such a work is Marcel Duchamp's "Nude Descending a Staircase II" [1912], which took its inspiration from the photographic work of Edwaerd Muybridge [1872] (see Figure 7-14, left). As an investigation to conclude our temporal motion emphasis work, we considered whether it would be possible to combine our tracked features (for example, arms and legs), with our temporal analysis to produce artwork similar in spirit to Duchamp's. To do so required the compositing of frames into a single image (a minor modification) and a more general functional $\mathcal{T}(.)$.

We observe that, rather than painting regular temporal instants, Duchamp painted

salient key-frames of the subject descending the staircase (Figure 7-14, left). These glimpses of motion are enough to suggest the full course of the movement, much as a few salient lines are enough to suggest form in a well drawn piece of artwork. We say the instants chosen by Duchamp are "temporally salient".

With this observation in mind, consider the smooth sections of Chapter 6 which form streak-line motion cues. The temporal regions around the start and end points of these streak-lines have high temporal salience relative to the remainder of the cue. A few points around each end of the streak-line are often sufficient for one to correctly predict the smooth trajectory of the object. Temporal salience is high at the streak-line origin, but decays as the streak-line progresses — falling to a minimum halfway along the trajectory. Temporal salience then increases again as we approach the terminus of the streak-line.

We have devised a non-linear transfer function $\mathcal{T}$, which varies sampling rate according to the magnitude of temporal salience at the current instant (i.e. proportional to the minimum time difference between the start and end of a streak-line). A plot of the resulting functional $\mathcal{T}(.)$ for the *STAIRS* sequence is shown in Figure 7-14, right. By compositing frames to create a single still image (we have chosen to also assign painting order in this composition to be proportional temporal salience), we obtain a result such as that of Figure 7-14, middle. Although the aesthetics of this output do leave something to be desired (static AR techniques could be applied to improve the final rendering), the composition of the artwork is, ostensibly, of a similar genre to that of Duchamp. We suggest that the notion of temporal salience warrants further investigation, and that the ability to define a user functional $\mathcal{T}(.)$ as a final stage in the motion emphasis pipeline serves as a simple, but interesting, means to access novel temporal rendering effects.

7.8 Summary and Discussion

We have described a subsystem within the Video Paintbox for automatically introducing "time and pose" cues into a video driven AR animation. This class of cue manipulates the timing of the animation, and includes traditional animation effects such as motion anticipation and exaggeration (motion cartooning). The "time and pose" subsystem integrates well with the visual motion emphasis work of the previous Chapter, and serves to broaden the gamut of motion cues available through our Video Paintbox.

We produce our time and pose cues by manipulating the positions of tracked features

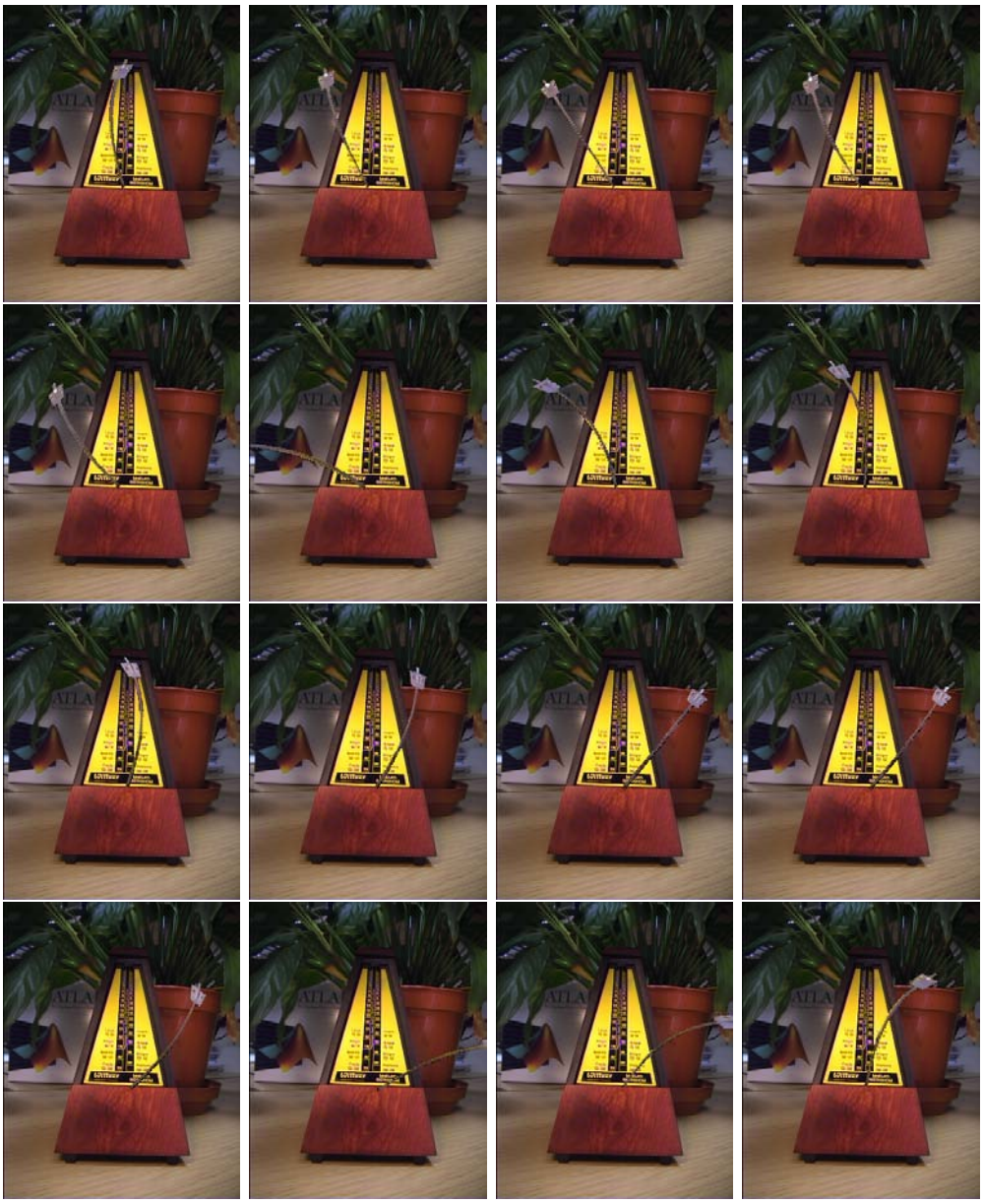


Figure 7-15 Stills taken from a section of the rendered *METRONOME* sequence, exhibiting the anticipation cue combined with a deformation motion cue emphasising drag (described in Section 6.4.2). Approximately one still per five frames. Observe that the visual deformation cues enable us to characterise movement in the scene, using only these still frames (see `videos/metro_warp_anticipate` and `videos/metro_anticipate` for animations).

over time. Recall that our initial experiments manipulated features independently by varying their LCAT transforms. The disappointing results which emerged motivated us to manipulate features using a hierarchical articulated structure, resulting in aesthetically superior animations. It is likely that the conceptually high level model of the articulated structure created more believable movement, because it more closely matches our mental model of the manner in which objects move — a subject's motion

is constrained by its inter-connecting joints, rather than allowing free motion of each component. If we refer back to the hand-drawn example of anticipation in Figure 7-5 it is clear that features of the face, for example eyebrows, are not anticipated using functions of their rotation, translation, etc. but according to a mental model of how facial parts move. This is again an example of how "time and pose" cues require a high level underlying model, in this case a facial muscle model rather than a rigid hierarchical structure. Future work might allow substitution of the current hierarchical articulated model for other models, so improving the generality of the system. We have also shown that the very nature of "time and pose" motion cues demands large temporal windows for analysis of the video sequence. Both the use of high level spatial models, and large temporal windows for motion analysis, are pre-requisites to synthesising time and pose cues.

There are a number of ways in which we might improve the work in this Chapter. We might seek to relax the assumptions on the nature of the motion, perhaps extending the system to emphasise non-planar motion. Alternatively we might revisit the problem of localising moving pivot points by allowing the animator to introduce a model of pivot motion. It may also be possible to improve accuracy of the pivot recovery algorithm (and so of subsequent pose recovery) using a Kalman filter to take advantage of the Gaussian distribution of error in pivot estimates, and so refine the pivot estimate over time.

A selection of source and rendered video clips have been included in Appendix C.

Chapter 8

Stroke Surfaces: Temporally Coherent Artistic Animations from Video

In this chapter we describe the third and final subsystem comprising the Video Paintbox: a framework capable of creating artistically shaded animations directly from video. We demonstrate that through automated analysis of the video sequence at a higher spatiotemporal level — as a block of frames rather than on a per frame, per pixel basis as with current methods — we are able to generate animations in a wide variety of artistic styles, exhibiting a uniquely high degree of temporal coherence. In addition to rotoscoping, matting and novel temporal effects unique to our method, we demonstrate the extension of "traditional" static AR styles to video including painterly, sketchy and cartoon shading effects. We demonstrate how our coherent shading subsystem may be combined with the earlier motion emphasis subsystems (Chapters 6 and 7) to produce complete cartoon-styled animations from video clips using our Video Paintbox.

8.1 Introduction

In this chapter we propose a solution to the long-standing problem of automatically creating temporally coherent artistically shaded animations from video[1]. As observed in Chapter 5, this problem has been sparsely researched. Existing video driven AR methods [75, 96, 103] address only the problem of producing painterly effects in video, and typically produce animations exhibiting poor levels of temporal coherence. AR techniques are predominantly stroke based, and temporal incoherence occurs principally when either:

[1]This work appeared in [26] and an overview presented at the BMVA Symposium on Spatiotemporal Processing (March 2004). This work has also been submitted to BMVC 2004.

1. the motion of strokes, and so motion within the resulting animation, does not agree with the motion of content within the source video sequence.

2. the visual attributes of strokes fluctuate rapidly, creating flicker in the animation.

The manifestation of temporal incoherence is as an uncontrolled motion and rapid flickering in the animation, termed *"swimming"*. Swimming severely damages the aesthetics of an animation, and tends to produce perceptual cues which distract from the content of the image sequence itself. This observation is supported by psychophysical research. For example the Gestalt "common fate" cue [95], describes how objects moving in a similar manner become grouped. Conflicts between the motion of phantom objects perceived due to grouping, and physical objects, contribute to the distracting nature of swimming. Wong *et al* [172] observe that rapidly flickering dots are perceived to "pop-out" from their neighbours; explaining the confused sense of depth apparent in an animation with poor stroke coherence. Unfortunately, we observe that this pop-out effect manifests most strongly at around 6Hz, close to the aesthetically optimal frame rate for coherent painterly animations determined by Hertzmann and Perlin [75].

Swimming in AR animations is therefore a significant practical problem, and one that can be solved only by smoothly moving strokes in a manner consistent with motion in the scene. Numerous object-space AR techniques based upon this principal have been published in recent years, and are capable of creating coherent AR animations from 3D models [32, 65, 108, 111]. Broadly speaking, object-space methods operate by fixing strokes to surfaces in the 3D geometry which move coherently when the object moves relative to the camera (see Chapter 2 for details of specific approaches). As we observed in Chapter 1, the problem statements of object-space and video driven AR are thus somewhat different. With the former there is no requirement to recover structure and motion prior to stroke placement, since scene geometry is supplied. With the latter, we must analyse pixel data to recover missing structure and motion prior to rendering.

All existing automatic 2D video AR algorithms [75, 96, 103] largely disregard spatial structure by moving brush strokes independently, and attempt motion recovery using inter-frame comparisons; motion is estimated from one frame to the next, and brush strokes translated accordingly. Both per frame optical flow [96, 103] and frame differencing [75] approaches to motion estimation have been applied to AR (Section 2.5.2 contains details), however both approaches fall far short of producing temporally coherent animations. We have argued (Chapter 5) that there are in-principal difficulties with analysing video on a temporally local, per frame progressive basis, when attempting to produce a globally coherent AR animation. These difficulties include the rapid accumulation and propagation of error over subsequent frames due to poor motion estimates, and the limitations of the motion estimate techniques employed (especially

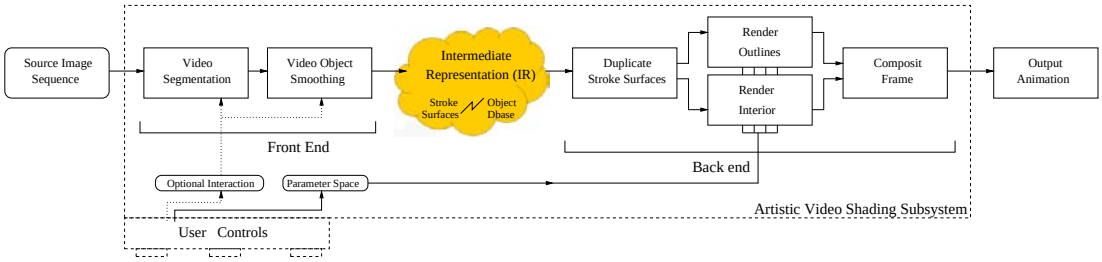


Figure 8-1 Illustrating the rendering pipeline of the artistic rendering subsystem. Video is parsed into an intermediate representation (IR) by the front end, using Computer Vision techniques. This representation is then rendered by the back end, under the influence of user parameters which may be varied to stylise the output according to the animator's wishes.

when operating upon flat textured objects, or on objects undergoing occlusion). If one were processing video for interaction or real-time animation then a frame by frame approach would be justified (an example is Hertzmann's "Living Painting" [75]). However the motivation of the Video Paintbox is primarily to create a tool for animators, with which they are able to process video for post-production effects. A global analysis over all frames available during offline rendering seems a more promising avenue for producing temporally coherent animations.

We therefore argue for a higher level of spatiotemporal analysis than that employed by existing automatic techniques. Spatially, we operate at a higher level by segmenting images into homogeneous regions, which correspond well with individual objects in the scene. Using the novel approach we describe in this Chapter brush stroke motion is guaranteed to be consistent over entire regions — contradictory visual cues do not arise, for example where stroke motion differs within a given object. Temporally we work at a higher level, automatically corresponding regions over time to carve smooth trajectories through the video, and smoothing region attributes, such as colour, over blocks of adjacent frames to mitigate swimming; this is in contrast to all existing AR video methods. We believe the paradigm of automatically processing video at this higher spatiotemporal level to be a novel and valuable approach to the problem of synthesising AR animations from video.

The remainder of this chapter describes our novel framework for the production of temporally coherent AR animations from video, which comprises the third and final subsystem with the Video Paintbox. Our approach is unique (among automated AR video methods) in that we treat the image sequence as a spatiotemporal voxel volume; a stack of sequential frames in which time forms the third dimension. The interpretation of video as a volume, rather than as a disjoint set of frames, is a powerful abstractive technique proposed as far back as the early eighties [86] simplifying analysis and

exploitation of frequency patterns in both the spatial and temporal dimensions. Applications of spatiotemporal processing have been identified both within the field of Computer Vision, for example in motion estimation [139] and content based image retrieval [118], and in Computer Graphics for interactive video editing [6, 93] and more commonly for visualisation [34, 128, 173]. We demonstrate that by manipulating video in this representation we are able to synthesise a wide gamut of artistic effects, which we allow the user to stylise and influence through a parameterised framework. The diversity of artistic style, and level of temporal coherence, exhibited by our animations further evidence our central argument for a higher level of spatiotemporal analysis in image-space AR.

8.1.1 Overview and Capabilities of the Subsystem

In a similar manner to the motion emphasis subsystems (Chapters 6 and 7), the artistic shading subsystem consists of a single rendering framework which may be broken into a front and back end. The front end (Section 8.2) is responsible for parsing the source video to create an "intermediate representation" (or "IR", Section 8.3), and is automated through application of Computer Vision techniques. This abstracted video representation is then passed to the back end (Section 8.4), where it is rendered in one of a range of artistic styles. The user is given control over the back end of the system via a set of high level parameters which influence the style of the resulting animation (Figure 8-1).

The artistic shading subsystem operates in the following manner. We begin by segmenting video frames into homogeneous regions, and use heuristics to create semantic associations between regions in adjacent frames. Regions are thus connected over time to produce a collection of conceptually high level spatiotemporal "video objects". These objects carve sub-volumes through the video volume delimited by continuous isosurface patches, which we term "Stroke Surfaces". The video is encoded by a set of such boundaries and a counter-part database containing various properties of the enclosed video objects. The surfaces and database respectively form the two halves of the IR, which is passed to the back end for rendering. To render a frame at time t the back end intersects the Stroke Surfaces with the plane $z = t$, to generate a series of splines corresponding to region boundaries in that frame. By manipulating the IR (for example, temporally smoothing the Stroke Surfaces), the back end is able to create temporally coherent animations in a range of artistic styles, under the high level direction of the animator.

Although our spatiotemporal framework was originally motivated by our goal of creating coherent, flat-shaded cartoons from video, it now encompasses many artistic styles.

Figure 8-2 Top: Stills from the hand-illustrated music video to A-Ha's "Take On Me" [Barron, 1985]. A woman enters a world inside a comic strip and appears non-photorealistic (sketchy), whilst interacting with a number of photorealistic and non-photorealistic beings and objects. Bottom: Our automatic video AR framework is capable of similar "mixed media" effects (left, we show photorealistic people against AR background; middle, vice versa), as well as many other artistic styles such as oil paint, watercolour, flat shaded cartoon (right), and can create a range of novel temporal effects too. Motion emphasis cues from the previous two chapters may be readily combined with this subsystem to create complete cartoon animations from video (right).

In addition to novel temporal effects unique to our framework, we demonstrate the extension of numerous static AR styles to video including oil and watercolour paint, sketchy, cartoon shading effects, as well the ability to create "mixed media" effects (Figure 8-2). An application to rotoscoping and video matting is also identified, in fact rotoscoping and stroke based AR techniques (such as painterly rendering) are unified under our framework. A potential application to abstracted, low bandwidth transmission of video content is also identified, resulting from the compact, continuous vector representation of the IR. Furthermore, we are able to combine our coherent shading framework with our earlier motion cue work (Chapters 6 and 7) to produce polished cartoon-styled animations from video clips using the completed Video Paintbox.

8.2 Front end: Segmenting the Video Volume

In this section we describe the video segmentation process of the front end. The front end is responsible for the smooth segmentation of the video volume into sub-volumes in a voxel representation, which describe the trajectories of features. These volumes are

then encoded in our "IR", which is passed to the back end for rendering. We describe the nature of this IR, and the encoding process, in Section 8.3.

There are three stages to the video segmentation algorithm, each of which we overview briefly here, and describe in detail in subsections 8.2.1, 8.2.2, and 8.2.3 respectively. We begin by independently segmenting video frames into connected homogeneous regions using standard 2D Computer Vision techniques (we describe these momentarily). The second stage of processing creates associations between segmented regions in each frame, to regions in adjacent frames. A filtering process removes spurious associations. The result of this second step is a set of temporally convex sub-volumes carved from the spatiotemporal video volume; we introduce the term *"video objects"* to describe these sub-volumes. These video objects are associated over time in a graph structure, which we refer to as the *"object association graph"*. The third and final stage performs a coarse temporal smoothing of the boundaries of video objects. This smoothing, combined with the filtering process the second stage, mitigate temporal incoherence in the video segmentation. The trajectory of a single feature through the video volume is represented by a collection of one or more associated video objects; we describe the union of video objects, which comprise such a trajectory, as a *"feature sub-volume"*.

8.2.1 Frame Segmentation

We now explain the first step of our process, which segments each video frame into homogeneous regions. The criterion for homogeneity we have chosen for our system is colour (after [38]). Many segmentation techniques also subscribe to this approach [5, 39, 42, 164], under the assumption that neighbouring regions are of differing colour. Each frame is independently segmented to create a class map of distinct regions. Associations between regions in adjacent frames are later created. Choice of segmentation algorithm influences the success of this association step, as segmentations of adjacent frames must yield similar class maps to facilitate association. Robustness is therefore an important property of the segmentation algorithm chosen to drive our subsystem: given an image I, a robust segmentation process $S(.)$, and a resulting class map of regions $S(I)$, small changes in I should produce very similar class maps $S(I)$. Although most published 2D segmentation algorithms are accompanied by an evaluation of their performance versus a ground truth segmentation, to the best of our knowledge a comparative study of algorithm robustness, as we have defined it, is not present in the literature. Consequently, we investigated the robustness of several contemporary 2D segmentation algorithms, with an aim of selecting the most robust algorithm to drive our video segmentation process.

We evaluated the robustness of five contemporary algorithms on twenty short clips

(around 100 frames each) of artificial and natural scenes. For our purposes, a natural scene is one of complex, cluttered content (such as Figure 8-5, *SOFA*), whilst an artificial scene is typically an uncluttered, low complexity scene with few homogeneous colour regions (such as Figure 8-5, *POOHBEAR*). In all clips the footage was of a static scene, with a gradual change of camera viewpoint over time. We tested the following algorithms in both RGB and HSV spaces:

1. Recursive histogram split (RHS) [145] A colour histogram of the image is built, and the colour which corresponds to the largest peak in the histogram is identified. The largest connected region of that colour is isolated, and "cut" from the image to form one segmented region. This process is iterative, and "cut" regions do not contribute to the histogram on subsequent iterations. The process terminates when the height of the histogram peak falls below a threshold.

2. Split and Merge (SM) [78] A two stage algorithm, comprising a "split" and a "merge" step. A tree T is constructed, initially with the whole image as the only node (root). We iterate through each node, splitting the node's region into quarters (creating four children) if that region is not "sufficiently homogeneous". The "split" step terminates when all nodes have been tested, and will split no further. Each leaf in T is a homogeneous region, though the image may be over-segmented. The "merge" step mitigates the over-segmentation by examining each leaf node in T, combining regions which are both homogeneous and spatially connected.

3. Colour Structure Code (CSC) [127] A form of split and merge technique which operates upon the image using small neighbourhoods with a hexagonal topology. Hierarchies of hexagonal "islands" are grown and merged iteratively to form homogeneous regions.

4. EDISON [19] A synergistic approach to segmentation which fuses boundary information from: 1) homogeneous regions produced by a colour based mean shift segmentation [29]; 2) edges detected in the luminance channel of the image.

5. JSEG [42] A technique which first performs colour quantisation, and then proceeds to grow pixel clusters to form regions of homogeneous colour texture. These clusters must exhibit low "J values"; these values are a novel texture variance descriptor introduced in the paper.

Algorithms (1) and (2) are popular, classical segmentation techniques. The remainder are comparatively recent segmentation techniques, which are reported by their authors [19, 42, 127] to perform well upon general images without the imposition of additional models or constraints.

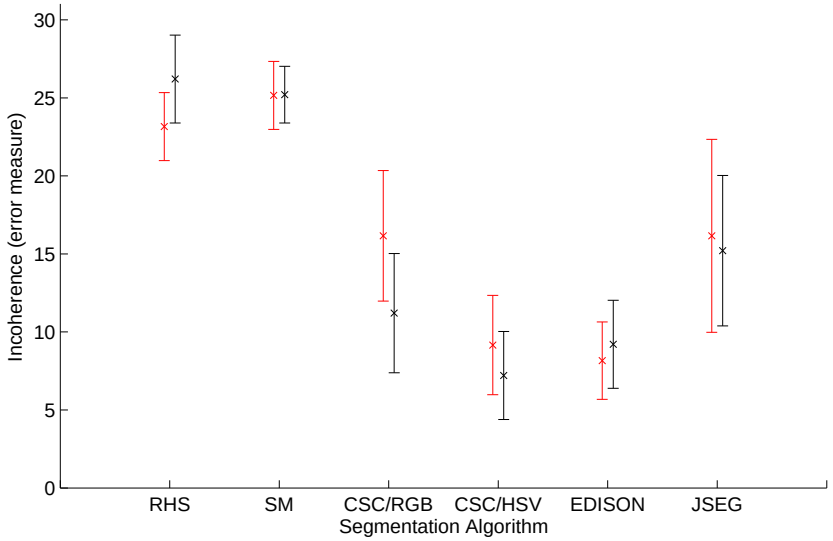


Figure 8-3 Results of comparing the robustness of segmentation algorithms. The lower the error measure on the ordinate, the more robust the algorithm. All algorithms operated in the RGB colour space unless otherwise stated. This graph shows the six most robust scenarios. Red and black plots indicate the results of processing ten "natural" and "artificial" scenes respectively. The crosses indicate the mean error measure for each scenario over all ten clips, the error bars indicate one standard deviation.

We measured robustness between adjacent frames in the following manner. Distance transforms [145] were computed for both frames' class maps. This resulted in two rasters whose individual pixel values correspond to the shortest distance from that pixel to a region boundary in the respective class map. We then computed the absolute difference between these distance transforms. The mean pixel value in this difference map was used the metric for robustness of segmentation between the two frames. The mean value of this inter-frame metric, computed over all consecutive frame pairs, was used as the metric for measured robustness of segmentation over the entire sequences. In practice the pixel values of the distance transform were thresholded at 30 pixels distance to prevent large errors from skewing the mean value (such errors emerged in cases of gross disparity between temporally adjacent class maps).

All segmentation algorithms operated in RGB colour space, but we also experimented with processing in HSV colour space — although this produced worse results for the most-part, results improved in the case of the CSC algorithm. Figure 8-3 summarises the means and variances of the robustness error measure for the six most robust scenarios, computed over ten "natural" and ten "artificial" source clips. Results indicate EDISON to be preferable in the case of natural scenes, and that artificial scenes were best segmented using the CSC algorithm operating in HSV space. However the associated error bars imply that there is little significant difference in one algorithm's performance over the other.

In our Video Paintbox we have chosen to use EDISON by default, but enable the user to switch to CSC/HSV if results of the segmentation are poor. The chosen algorithm is then independently applied to each frame in the source video clip to generate a set of segmented regions in each frame. Associations are then created between regions in adjacent frames to produce spatiotemporal video objects. We explain this process in the subsection 8.2.2, but first briefly justify our decision to opt for this associative approach to volume segmentation.

Choice of Video Segmentation Strategy: 2D plus time or 3D?

We have opted for a 2D segmentation followed by a temporal association step; this approach is commonly referred to as a "2D plus time" ($2D+t$) technique in Computer Vision literature. However, our initial development work centred upon an alternative methodology — performing a colour segmentation via a three dimensional flood-fill approach, and so processing the video as a single $3D$ volume (similar to [40]). Although a volumetric methodology is, perhaps, more in keeping with our spatiotemporal approach, there are a number of reasons that we opted for a $2D + t$ heuristic driven, associative approach over a $3D$ segmentation:

- Attributes such as the shape, colour or shading of a region are permitted to evolve gradually over time by the heuristics of our $2D + t$ approach. Such variation is difficult to accommodate within the framework of a single 3D segmentation (without introducing complicated 3D models and constraints, encapsulating the expected evolution of these attributes over time).

- Small, fast moving objects may form disconnected volumes in $3D$, resulting in temporal over-segmentation. However these discontinuities do not arise with our proposed $2D+t$ association scheme between frames, providing an upper threshold on search distance (parameter Δ, described later in Section 8.2.2) is set correctly.

- For pragmatic reasons. The problem of 2D image segmentation has received extensive study from the Computer Vision community, in contrast to 3D segmentation (exceptions lie within medical imaging, but do not deal with the problem domain of video imagery). However the modular nature of our framework (Figure 8-1) is such that the rendering process is loosely coupled with the segmentation technology used; thus we allow for substitution of segmentation algorithms as novel, improved technologies become available in the literature.

Alternative $2D + t$ approaches which associate contours over time have been described in the literature [53, 55, 92]; however we differ in that we create temporal association using region based properties rather than edges alone (the unconstrained problem of associating edge contours over time produces poor tracking solutions, and often requires

restrictive motion models to be imposed [84, 85]). We now explain the temporal region association process in detail.

8.2.2 Region association algorithm

The problem of associating sets of regions, each within multiple frames, is combinatorial in nature and an optimal solution can not be found through exhaustive search for any practical video. We propose a two step heuristic solution to the association problem, which we have found to perform well (that is, results in a locally optimal solution where associated objects exhibit an acceptable level of temporal coherence) and has quadratic complexity in the number of regions per frame. First, for each frame we generate associations between regions in that frame and those in frames adjacent to it. These associations are made according to heuristics based on mutual colour, area, spatial overlap, and shape. Second, the resulting chains of associated regions are filtered using a graph based search which removes sporadic associations. Association is complicated by the fact that objects may merge or divide in the scene. For example, a ball passing behind a post might appear to split into two regions, and then recombine. In our system it is satisfactory to represent a single occluded object as multiple imaged regions since, as we will describe, these regions become linked in a graph structure as a product of the region association process.

We observe that in a robust video segmentation: 1) properties of regions such as shape, colour, and area are typically subject only to minor change over short periods of time. The exceptions are the instants at which regions merge or divide; 2) although regions may appear or disappear, merge or divide over time, such events should be for the relative long-term (given a video frame rate of 25 frames per second) and not be subsequently reversed in the short-term. The first observations influences the choice of heuristics for the first stage of processing (region association), whilst the second observation governs the operation of the second stage (filtering).

Step 1: Iterative Association Algorithm

Consider a single region $r \in R_t$, where R_t denotes the set of segmented regions in frame t. We wish to find the set of regions in adjacent frames with which r is associated. We compute this by searching sets R_{t-1} and R_{t+1} independently — examining potential mappings from r to R_{t-1}, and then from r to R_{t+1}. Thus r could potentially become associated with zero or more regions in adjacent frames (Figure 8-4a). The suitability for two regions in adjacent frames $r \in R_t$ and $\rho \in R_{t\pm1}$, to be associated may be evaluated using an objective function $E(r, \rho)$. We describe this function momentarily, but first complete our description of the association algorithm.

For the purposes of illustration, let us consider the creation of associations between r and regions in the set R_{t+1}. The area (in pixels) of r is first computed, to give an "area-count". A potential set of associating regions in R_{t+1} is identified, whose centroids fall within a distance Δ of the centroid of r. These regions are sorted into a list in descending order of their score $E(.)$. Next a cumulative sum of their area counts is computed, working from the start of the list and storing the cumulative sum with each region. The area-count of r is subtracted from each cumulative area term in the list. The resulting set associated regions extends down this list until either the score $E(.)$ falls below a lower bound, or the area measure becomes less than or equal to zero. It is therefore possible for no associations to be created to past or future regions; in such circumstances a feature appears or disappears in the video, perhaps due to occlusion either by other objects or by the edge of the frame. The process is repeated for each frame t independently. The final set of associations for the sequence is taken to be the union of associations created for all frames.

Associated regions are thus semantically linked over time to create connected feature sub-volumes such as that in Figure 8-5c,d. These sub-volumes are broken into, possibly many, temporally convex video objects. Note we consider only the exterior boundary of these objects, disregarding "holes" in a volume produced by other nested objects; these are represented by their own external boundaries. A property of the temporally convex representation is that two separate objects will merge to produce one novel object, and an object division will produce multiple novel objects (see Figure 8-4b, Figure 8-5, bottom left). This representation simplifies later processing. The associations between video objects are also maintained; this mesh graph structure is also useful in later processing stages, as we refer to it hereafter as the "object association graph".

Heuristics for Association

We made use of a heuristic score $E(r, \rho)$ in our association algorithm, which determines the suitability of two regions in adjacent frames $[r \in R_t, \rho \in R_{t\pm1}]$ to be associated. This score may be written as a weighted sum of terms:

$$E(r,\rho) \quad = \quad \begin{cases} 0 \text{ if } \delta(r,\rho;\Delta) > 1 \\ w_1\sigma(r,\rho) + w_2\alpha(r,\rho) - ... \\ w_3\delta(r,\rho;\Delta) - w_4\gamma(r,\rho) \text{ otherwise} \end{cases} \tag{8.1}$$

The function $\delta(.)$ is the spatial distance between the region centroids as a fraction of some threshold distance Δ. The purpose of this threshold is to prevent regions that are far apart from being considered as potentially matching; $E(.)$ is not computed unless the regions are sufficiently close. We have found $\Delta = 30$ pixels to be a useful threshold. Constants $w_{1..4}$ are user defined weights which tune the influence of each

of four bounded ($[0,1]$) heuristic functions. It is through variation of these constants that the user is able to "tune" the front end to create an optimal segmentation of the video sequence; in practice less than an order of magnitude of variation is necessary. We have found $w_1 = 0.8, w_2 = 0.6, w_3 = 0.6, w_4 = 0.4$ to be typical values for the videos we present in this thesis. $\gamma(.)$ is the the Euclidean distance between the mean colours of the two regions in *CIELAB* space (normalised by division by $\sqrt{3}$). $\alpha(.)$ is a ratio of the two regions' areas in pixels. $\sigma(.)$ is a linear conformal affine invariant shape similarity measure, computed between the two regions. We now describe this final, shape similarity measure in greater detail.

We wish to compare the shape of two regions A and B. This comparison should be invariant to rotation, translation and uniform scale, since such transformations are common in imaged regions caused by an object undergoing motion relative to the camera. Regions are first normalised to be of equal area (affecting uniform scale invariance). We compute $\sigma(A, B)$ by analysis of the external regions' boundaries in the following manner.

We begin by computing the Fourier descriptors [31] of the "angular description function" of each boundary. The angular description function discretises a region's boundary into n vectors of equal arc-length $\underline{v_1}, \underline{v_2}, ..., \underline{v_n}$, and encodes that boundary as a series of variations in the angles between adjacent vectors. We write the angular descriptions for each region, A and B, as the scalar functions $\Theta_A(s)$ and $\Theta_B(s)$, where $sin[0, n]$ is a continuous dummy parameter which iterates around the boundary. Observe that $\Theta(.)$ is periodic, and invariant to translation of the region. We compute the Fourier transform of each $\Theta(.)$, to obtain spectral representations $F[\Theta_A(.)]$ and $F[\Theta)B(.)]$. Shape similarity is inversely to proportional to Euclidean distance between the magnitude vectors of these Fourier descriptors (disregarding phase for rotational invariance):

$$\sigma = 1 - \frac{1}{N} \sum_{\nu=1}^{N} \left| \frac{|F[\Theta_A(\nu)]| - |F[\Theta_B(\nu)]|}{(\sum_{s=1}^{n} (\Theta_A(s) - \Theta_B(s))^2)^{\frac{1}{2}}} \right| \tag{8.2}$$

where the summation is over the N lowest frequency components of the signal $|F[\Theta(.)|$. We have found that only the first eight components are desirable for inclusion in the summation ($N = 8$); the remaining high frequencies principally describe sampling errors due to the discrete, raster origin of the boundary descriptions.

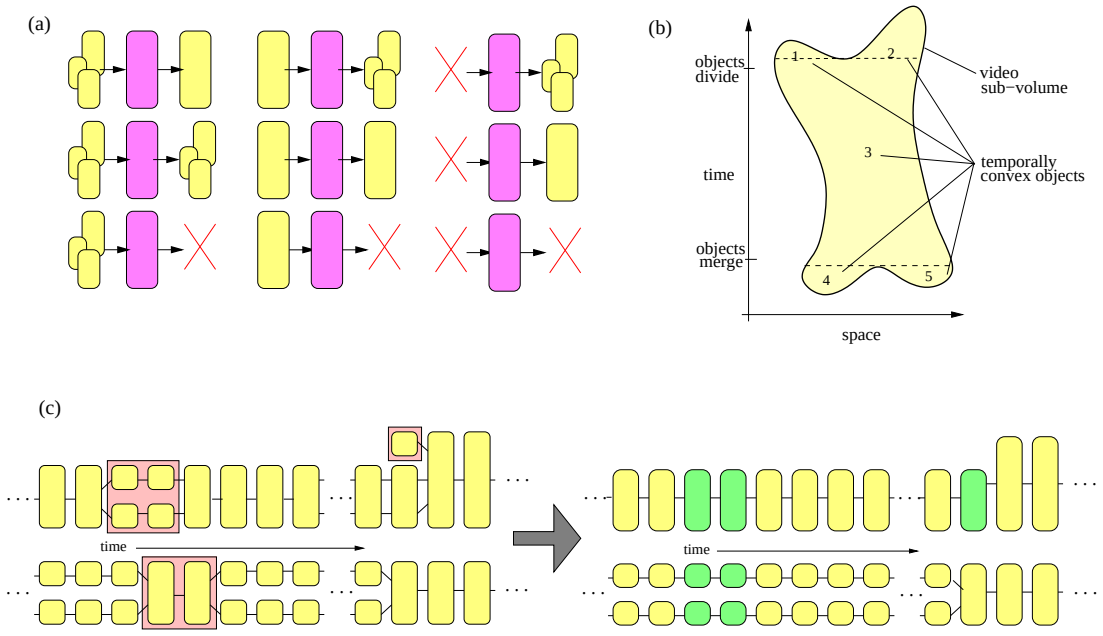


Figure 8-4 Illustrating the region association and filtering process (a) Nine cases of region association. Associations are created between a region in the current frame and potentially many regions in adjacent frames, through iterative evaluation of equation 8.1. (b) Example of a single video-sub-volume split into five temporally convex objects (c.f. Figure 8-5d). (c) An object association graph before and after graph filtering. Sporadic associations (red) are removed, and object boundaries interpolated (green) from neighbours.

Step 2: Filtering Sporadic Associations

Sporadic associations are sometimes incorrectly created between regions due to noise. We have observed that associations maintained over short time intervals may be categorised as noise, and filter out these artifacts by examining the object association graph's structure.

Since new objects are created for every merge or divide encountered in a feature sub-volume, one can identify sporadic merges or divisions by searching the graph for short-lived objects. We specify a short-lived object as an object which exists for less than a quarter of a second ($\leq$ 6 frames). This constant (as well as the search parameter Δ in equation 8.1) may be adjusted according to the assumed maximum speed of objects in the video. Short-lived objects deemed to correspond to false associations are removed by "cutting" that object from the graph and filling the gap by extrapolating an intermediate object from either neighbour (by duplicating the nearest neighbour, see Figure 8-4c). A serendipitous effect of this process is that poorly segmented areas of the video exhibiting high incoherence, tend to merge to form one large coherent object. This is subsequently rendered as a single region, abstracting away detail that would otherwise scintillate in the final animation.

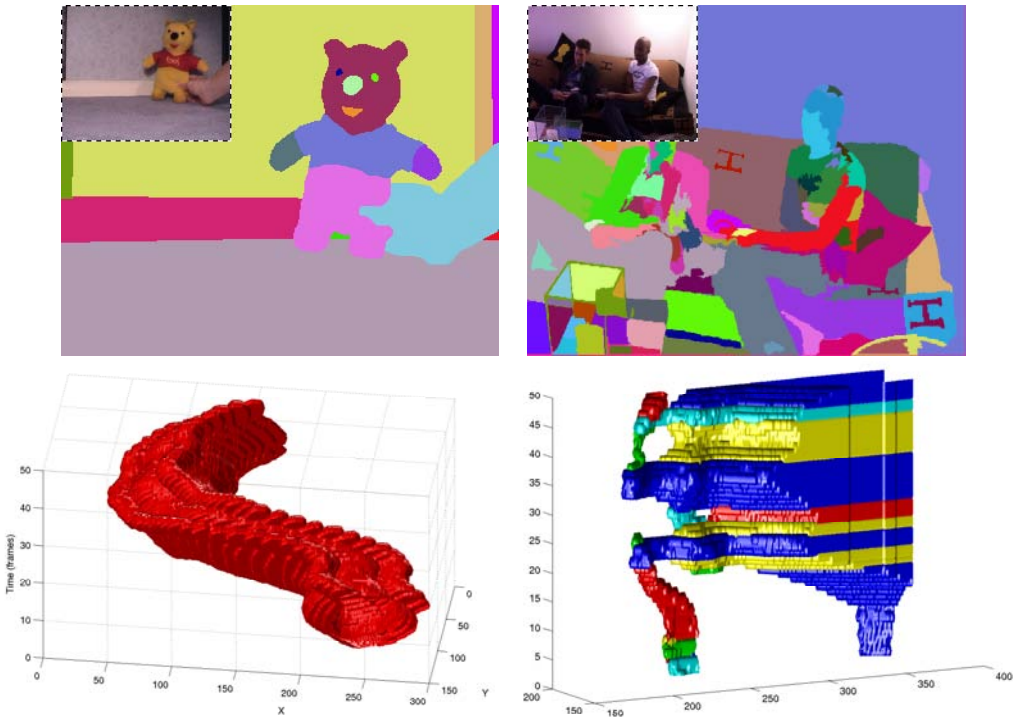


Figure 8-5 Above: Sample segmented frames from the *POOHBEAR* and *SOFA* sequences with original footage inset. Below: Two visualisations from the *POOHBEAR* video volume, corresponding to the head and the right hand section of the skirting board. Observe that whilst the head produces a single video object, the skirting board is repeatedly occluded by the bear's hand during movement causing division of regions. This resulting volume consists of a connected structure of several temporally convex video objects (coloured individually for clarity of illustration).

8.2.3 Coarse Temporal Smoothing

The segmentation and association processes result in an object association graph, and a number of video objects in a voxel representation. It is desirable that the boundaries of these video objects flow smoothly though the video volume, thereby creating temporally coherent segmentations of each video frame. Therefore, as a final step in our video segmentation algorithm we perform a temporal smoothing of video objects.

We fit constrained, smooth surfaces around each video object, and then re-compute the sets of voxels bounded by these surfaces to generate a new set of smoothed video objects. We describe this process in the remainder of this subsection. Note that this surface fitting process is a means to smooth the voxel video objects, and quite distinct from the surface fitting operations we introduce in Section 8.3 to encode the results of the front end in the IR.

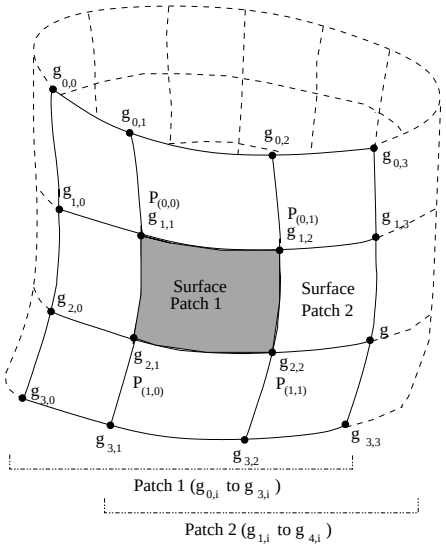


Figure 8-6 Coarse smoothing: A piecewise bi-cubic surface is fitted around each temporally convex video object. Multiple bi-cubic Catmull-Rom patches are stitched together with overlapping control points, to form a C_2 continuous bounding surface [14, 51].

Surface fitting by relaxation

Each video object is temporally convex, and as such its external boundary may be described by a continuous 2D surface — disregarding its"end caps", i.e. planes of constant time. These "end caps" are represented implicitly by the minimum and maximum temporal coordinates of the bounding surface's control points.

We fit a constrained 2D parametric surface to each segmented video volume, which we describe piecewise with a series of bi-cubic Catmull-Rom patches [14]. Each individual patch may be represented in a 2D parametric form $\underline{P}(s,t)$, where s and t are the spatial and temporal parameterisations, as:[2]

$$\underline{P}(s,t) = \underline{t}^T \underline{\underline{M}}^T \begin{pmatrix} \underline{g}_{0,0} & \underline{g}_{0,1} & \underline{g}_{0,2} & \underline{g}_{0,3} \\ \underline{g}_{1,0} & \underline{g}_{1,1} & \underline{g}_{1,2} & \underline{g}_{1,3} \\ \underline{g}_{2,0} & \underline{g}_{2,1} & \underline{g}_{2,2} & \underline{g}_{2,3} \\ \underline{g}_{3,0} & \underline{g}_{3,1} & \underline{g}_{3,2} & \underline{g}_{3,3} \end{pmatrix} \underline{\underline{M}}\underline{s} \tag{8.3}$$

where $\underline{t} = \left(t^3, t^2, t, 1\right)^T$, $\underline{s} = \left(s^3, s^2, s, 1\right)^T$, and $\underline{\underline{M}}$ denotes the Catmull-Rom cubic blending matrix [14] (given in equation 7.27). This class of spline function was chosen since it both interpolates all control points $\underline{g}_{i,j}$ (allowing greater local control over the

[2]Equation 8.3 represents a mild abuse of notation (used in the text of Foley *et al.* [51]) to avoid introduction of tensor notation. The 4×4 matrix of control points should be read as containing sixteen vector valued points $\underline{g}_{i,j}$ — rather than as an expansion of those points to create a 12×4 matrix of scalars.

surface during the fit), and ensures C_2 continuity between adjacent patches, i.e. whose control points overlap. We introduce the notation $\underline{Q}(s,t)$ to describe the entire 2D bounding surface, consisting of multiple piecewise bi-cubic patches (see the example of Figure 8-6).

The C_2 continuity of the Catmull-Rom spline class is important to our application since surface discontinuities in the first and second derivatives with respect to time ($\frac{\partial \underline{Q}}{\partial t}$, $\frac{\partial^2 \underline{Q}}{\partial t^2}$) have been found to produce jerky and incoherent motion in resulting animations. However, we note that spatial discontinuities in derivatives ($\frac{\partial \underline{Q}}{\partial s}$, $\frac{\partial^2 \underline{Q}}{\partial s^2}$) may be desirable when, for example, representing sharp corners. Fitting is performed via a generalisation of 1D active contours [91] to 2D surfaces (after [21]):

$$E_{surf} = \int_0^1 \int_0^1 E_{int}(\underline{Q}(s,t)) + E_{ext}(\underline{Q}(s,t))dsdt \tag{8.4}$$

the internal energy is:

$$E_{int} = \alpha \left| \frac{\partial \underline{Q}}{\partial s} \right|^2 + \beta \left| \frac{\partial^2 \underline{Q}}{\partial s^2} \right|^2 + \gamma \left| \frac{\partial \underline{Q}}{\partial t} \right|^2 + \zeta \left| \frac{\partial^2 \underline{Q}}{\partial t^2} \right|^2 \tag{8.5}$$

and the external energy is:

$$E_{ext} = \eta f(\underline{Q}(s,t)) \tag{8.6}$$

Function $f(.)$ is the Euclidean distance of the point $\underline{Q}(s,t)$ to the closest voxel of the video object. Hence constant η controls the influence of the data in the fit (we present this to unity). The pairs of constants $[\alpha, \beta]$, and $[\gamma, \zeta]$ bias the internal energy toward certain spatial and temporal characteristics respectively. Reducing constants α and γ cause more elastic spacings of control points, whilst reducing constants β and ζ allow greater curvature between control points. We preset α and β at 0.5 and 0.25 respectively, to permit high curvatures (for example, sharp corners), and so close fits in the spatial dimension. γ is set to 0.5 to maintain temporal spacing of control points. Constant ζ is the most interesting, since this parameter dictates the energy penalties associated with high curvatures in the temporal dimension. Recall that we observe high curvatures in this dimension correlate strongly with temporal incoherence. Thus by varying this parameter the we may vary the level of temporal smoothing in the sequence; we have chosen to make the value of this temporal constraint available as a user variable. In this manner the animator may tailor the degree of temporal smoothing to the speed of motion in the sequence, or even chose to retain some of the noise present as a consequence of the segmentation process; we draw analogy with presence

of film grain in a movie.

Surfaces are fitted within the video volume using an adaptation of Williams' algorithm to locate surface points [167] through minimisation of E_{surf} (equation 8.4). Note that this algorithm relaxes penalties imposed due to high magnitudes in the second derivative during the final iterations of the fitting process. We inhibit this behaviour in the temporal dimension to improve temporal coherence. Initial estimates for surface control points are obtained by sampling the bounding voxels of the object at regular intervals. The result of the process is a fitted 2D surface about each video object. The video volume is then re-segmented using these fitted surfaces to create a set of smoothed video objects in a voxel representation. At this stage, the fitted 2D surfaces are discarded, since they have served their purpose in smoothing the video objects.

In practice, smoothing the surface of each video object independently is problematic. If the position of the surface bounding a video object is adjusted, voxel "holes" may appear in the video volume, that is, some voxels are no longer assigned to any video object. Alternatively, a single voxel might now fall within the fitted, bounding surfaces of more than one video object; we term these voxels "duplicates". We take the following simple steps to remove "holes" and "duplicates". First, any "duplicate" voxels are relabelled as "holes". Second, "holes" are filled by repeatedly "growing" video objects via binary morphological dilation; only voxels marked as "holes" are overwritten during this process. Video objects are dilated in random order to promote an unbiased, and even filling of holes. The process terminates when all holes are filled.

8.3 Front end: Building the Representation

The results of the video segmentation process (Section 8.2) are a set of segmented, smooth, video objects in a voxel representation, and an object association graph describing how video objects are connected over time. From this information we generate an intermediate representation (IR) of the video, to be passed to the back end. This IR consists of a series of "Stroke Surface" patches and a counter-part database. In this Section we describe each of these components in turn, and explain how they are created.

In our IR, the spatiotemporal locations of objects are represented in terms of their interfacing surfaces. This is preferable to storing each object in terms of its own bounding surface as boundary information is not duplicated. This forms a more compact, and more manipulable representation (which is useful later when we deform object boundaries, either for further fine smoothing, or to introduce temporal effects) see Figure 8-10. By manipulating only these interfacing surfaces, we do not create "holes" as before.

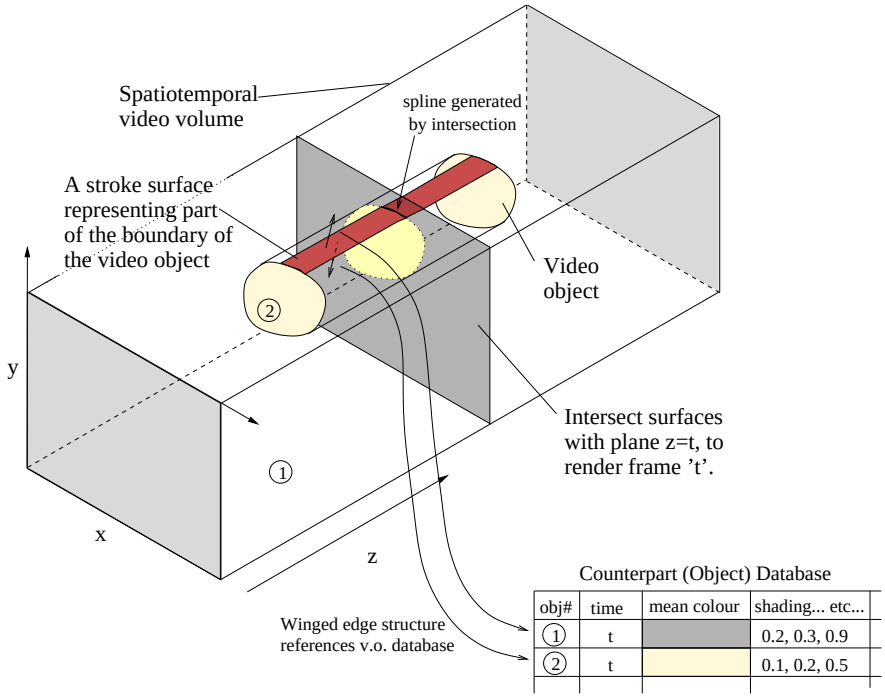


Figure 8-7 Illustrating the IR of the artistic shading subsystem, and the means by which it is rendered. Video is parsed by the front end, and represented as a collection of Stroke Surface patches — each accompanied by a winged edge structure which references the counter-part database. The attributes of video objects are stored in this database during parsing, here we demonstrate two objects, a yellow blob and a grey background. The back end renders the IR by intersecting a time plane with Stroke Surfaces; intersected surfaces form splines (holding lines) which bound regions. A region is rendered according to the graphical database attributes referenced by its bounding Stroke Surface(s).

However, in this representation it is much harder to vary the topology of the scene, for example the adjacency of the video objects. This motivated the coarse smoothing step of subsection 8.2.3.

8.3.1 Stroke Surface Representation

When two video objects abut in the video volume, their interface (in planes non-parallel to z) is represented in our IR in piecewise form by a collection of one or more disconnected surface patches. We introduce the term "Stroke Surface" to describe these individual patches.

As in Section 8.2.3 we use a piecewise Catmull-Rom bi-cubic form to represent a Stroke Surface, and a 2D parameterisation $\underline{Q}(s,t)$ to describe points upon it. Each Stroke Surface is contiguous. However, under certain interface geometries it is possible that internal holes may exist within the surface's external boundary — as such, regions of surface's parameter domain ($s,t \in [0,1]$) may be invalid. We address this repre-

sentational problem in a manner similar to binary CSG [51]. Each Stroke Surface is described by a continuous, externally bounded "primary" surface, from which a set of zero or more continuous secondary (or "hole") surfaces are subtracted[3] (Figure 8-8). We specify these holes as 2D polygonal regions in the parameter space $[s,t] \in \Re^2$.

To further illustrate the concept of Stroke Surfaces, consider the following two examples:

1. Consider two plasticine bricks (A and B). If the bricks were pressed together around a long metal pole, A and B would touch on either side of the pole, but the pole would "split" the interfacing surface between A and B into two discontinuous surfaces. The pole, A, and B are analogous to video objects. The interface between A and B would be represented by two Stroke Surfaces.

2. Now consider two new plasticine bricks (C and D), which are pressed together to completely surround a third object E. In this scenario, only one Stroke Surface is required to represent the interface between C and D. However, that surface will contain an internal "hole" where C touches E, rather than brick D. In fact the C-E and D-E interfaces will be represented by two further Stroke Surfaces.

8.3.2 Fitting Stroke Surfaces

Stroke Surfaces are fitted to the voxel video objects in the following manner. Each video object is systematically tested against each of its peers to determine whether the pair are spatially adjacent, that is, if there is an interface non-parallel to the z plane (temporal adjacency is not encoded via Stroke Surfaces, as this information is already stored in the object association graph). If an interface exists between the two objects, then that interface must be encoded in the IR by one or more Stroke Surfaces. We now describe the process by which we fit a single Stroke Surface between two example voxel video objects; we write the coordinate sets of voxels comprising these video objects as $\mathcal{O}$ and $\mathcal{P}$.

Fitting the primary surface

We begin by fitting the "primary" surface. This is accomplished in a manner similar to Section 8.2.3, using Williams' relaxation algorithm to fit a 2D parametric surface to the voxels which represent the interface between the two video objects. Internal parameters α', β, γ, ζ and η are set as described in Section 8.2.3, as we desire similar geometric properties in our fitted surface. However a new function $f(.)$ is substituted for the external energy term (see equation 8.6), which determines error between the

[3]Read "subtraction" in the context of binary constructive solid geometry (CSG). For example if two objects are represented by sets of voxels A and B, then A subtract B may be written as $A \setminus B$.

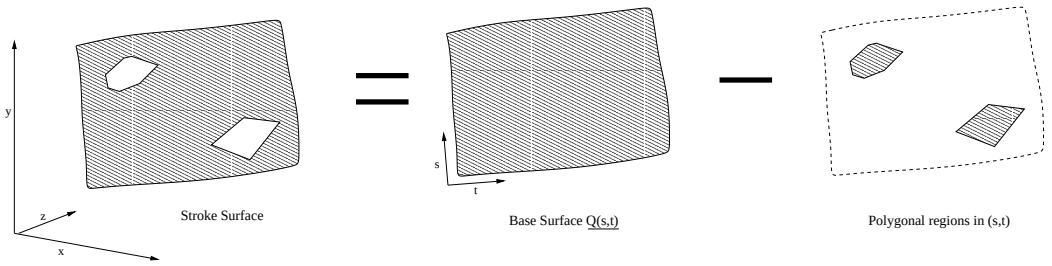


Figure 8-8 Stroke Surfaces are represented as a primary parametric surface $\underline{Q}(s,t) \in \Re^3$, minus a collection of one or more polygonal regions in space $[s,t] \in \Re^2$ which describe "holes" in that surface.

fitted surface and the data:

Given two voxel video objects $\mathcal{O}$ and $\mathcal{P}$, we compute a distance field $|\vec{\mathcal{OP}}| \in \Re^3$ from each voxel in $\mathcal{O}$ to the nearest voxel in $\mathcal{P}$. Likewise we compute field $|\vec{\mathcal{PO}}| \in |\Re^3$ from each voxel in $\mathcal{P}$ to the nearest voxel in $\mathcal{O}$. We generate a field $F = \min(|\vec{\mathcal{OP}}|, |\vec{\mathcal{PO}}|)$. The manifold $F = 0$ now represents the interface between $\mathcal{O}$ and $\mathcal{P}$, and points in field F encode distance from this interface. We wish to fit our surface to the manifold $F = 0$. Thus F forms the basis of the external function $f(.)$, which measures distance between points on the surface, and the interface to which that surface is to be fitted. The distance field F is also useful later to identify "holes" within this fitted, primary surface (see next subsection).

All that remains is to define an initial estimate for the surface control points. We threshold field F at a distance of one voxel, and apply standard morphological thinning [182], to obtain a field F'. The binary voxel map that results is a good approximation to the manifold $F = 0$. We sub-sample the voxels upon this manifold at equal temporal instants, and at equal spatial arc-length increments, to establish initial estimates for the control points. An optimal configuration of surface control points is then sought using Williams' algorithm, and so the primary surface $\underline{Q}(s,t)$ is fitted. We re-parameterise s and t to be normalised arc-length parameters, using standard methods (linear approximation, see [51]).

Fitting holes within the primary surface

We wish to find the set of polygons in space $[s,t] \in \Re^2$, which describe holes within the interior of the primary surface $\underline{Q}(s,t)$. Recall that these are the regions which do not lie on the interface between video objects $\mathcal{O}$ and $\mathcal{P}$. We sample the field $F(\underline{Q}(s,t))$ to create a 2D scalar field which denotes distance to the video object interface at point (s,t); i.e. the error in the fit of the primary surface $\underline{Q}(s,t)$. In cases where (s,t) does

not lie close to the interface of video objects $\mathcal{O}$ and $\mathcal{P}$, then $F(\underline{Q}(s,t))$ will be large. We threshold this field to obtain a collection of connected binary components which correspond to regions of $\underline{Q}(s,t)$ which do not lie on the interface. We obtain the chain codes [52] of the external boundaries of each component, and use a standard chain code vectorisation technique [119] to obtain a polygonal description of each region.

Each Stroke Surface is thus represented as a single function $\underline{Q}(s,t)$ (encoded in piecewise bi-cubic form), and a series of 2D polygonal regions which describe "invalid regions" of the parameter domain $[s,t] \in \Re^2$. We store the complete set of Stroke Surfaces for the video as one half of the IR. Each stroke surface holds an additional *winged edge structure* which contains two pointers corresponding to the two objects which it separates (one for each normal, Figure 8-7). These pointers reference records in the counter-part database, as we now describe.

8.3.3 Counter-part Database

A database is maintained as the second half of the IR, containing one record per object in the video volume. This counterpart database is referenced by the pointers held in the Stroke Surfaces' winged edge structure, and encapsulates the object association graph created by the region association process (Section 8.2.2), as well as various graphical attributes about each object at each frame of its existence. We now briefly summarise the information stored in the counterpart database, describing how the database is populated with this information in the next subsection (Section 8.3.4).

For each temporally convex video object (referenced by the Stroke Surfaces), the database maintains a record containing the fields:

1. **Born:** The frame number (B) in which the video object comes into existence.
2. **Died:** The frame number (D) in which the video object determines.
3. **Parent object list:** A list of video objects which have either split, or merged with other video objects, at time B to form the current video object. If this list is empty, the video object "appeared" as a novel feature in the sequence — all video objects in frame 1 will have be born in this manner, as will any regions which were segmented in a frame but were too different to be associated to any regions in previous frames (these often appear just after an occlusion).
4. **Child object list:** A list of video objects which the current video object will become one of (possibly many) parents for at time D. If this list is empty, the video object simply disappears (possibly due to occlusion, or because the end of the video sequence has been reached).

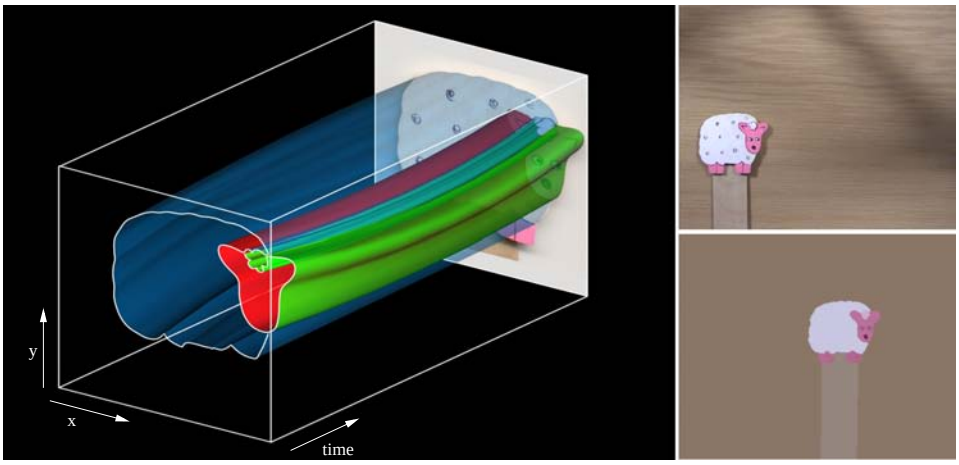


Figure 8-9 A visualisation (left) of the Stroke Surfaces generated from the *SHEEP* sequence (upper right), which when intersected by the a time plane generate a coherent segmentation (lower right). Observe how a single video object, for example the sheep's body, may be described by multiple Stroke Surfaces; in the example of the sheep's body these are the blue surface meeting the background object, and the red surface meeting the head object (see `videos/sheep_source` and `videos/sheep_flatsegment`).

5. **Colour list:** A list of $B - D + 1$ RGB colour vectors which encode the mean colour of the region in the original video, during each frame of the video object's existence.

6. **Homography list:** A list of $B - D + 1$ 3×3 matrices which describe the projective transform mapping the texture within the region, at each frame of the video object's existence, to the texture within the region in the previous frame. The first element in this list is the identity matrix, and is redundant.

7. **Shading list:** A list of $B - D + 1$ triples, containing the linear gradient shading parameters described in Section 8.3.4.

Attributes (1-4) encode the object association graph generated by the region association process of Section 8.2.2). Attributes (5-7) encode graphical information for each time instant t ($B \leq t \leq D$) useful to the back end during rendering. This signal in these graphical attributes is low-pass filtered (smoothed) over period $[B, D]$ by the front end to improve temporal coherence within the animation. For clarity of reading, we have deferred justification of this smoothing process to Section 8.4.2 of the back end, where the benefits of smoothing are more easily demonstrated in the context of cartoon flat-shading.

8.3.4 Capturing Interior Region Details in the Database

We begin by computing the mean colour of each region for each frame of the video. This information is stored in the respective video object record of the counterpart

database. At this stage in our explanation, each feature within the video is therefore stored in our representation only in terms of a series of linked, coherent spatiotemporal object boundaries and their mean interior colours. This is sufficient to recreate a basic, flat shaded animation, but can prove insufficient to render some artistic styles — the internal texture that has been abstracted away often forms perceptually important visual cues (for example, the cross-hatching marks sketched by artists to represent illumination, and so depict depth in a scene). We now describe two approaches we have developed to introduce internal detail into the animation in a temporally coherent manner.

First, we fit a linear shading gradient to each object on a per frame basis. The gradient at time t over an object may be described as a triple $\underline{G_t} = [g_0, g_1, \theta]$, where g_0 and g_1 are the start and end shading intensities respectively, and θ specifies the direction of shading over the region (as an angle). An optimal $\underline{G_t}$ is computed by a search [114] aiming to minimise the error $E(.)$:

$$E(\underline{G_t}, F_t) = \frac{1}{|P|} \sum_{p \in P} |I(\underline{G_t}) - F_t|^2 \tag{8.7}$$

Where $I(\underline{G_t})$ is a image created from the gradient triple $\underline{G_t}$, using the hue and saturation components of the object mean colour from the database and varying the luminance as defined by $\underline{G_t}$. F_t is the video frame at time t, and $\mathcal{P}$ is the set of pixels inside the object region at time t. The application of this gradient alone, when rendering, can dramatically improve the sense of depth in an image (Figure 8-14).

Second, images of the original object in a small temporal window around t are differenced with the optimal $I(\underline{G_t})$; the result is a difference image containing the detail thus far abstracted away by our representation. We pass these images through our salience measure (described in Section 3.2), which correlates salience and rarity; these maps indicate the salient marks in the interior of the region. We form a local motion estimate for the object over the temporal window by assuming the object to be approximately planar, and so its motion relative to the camera to be well approximated by a homography. Object regions over the window are projected to the reference frame of the object at time t. A initial degenerate estimate of the homography is obtained by taking the 2nd order moments of the two regions. This estimate is then refined using a Levenburg-Marquadt iterative search to minimise mean square pixel error between the interiors of the regions, using an identical optimisation approach to that described in Section 6.3.1. The computed salience maps are projected by homography to the reference frame at t, and averaged to form a final map. This results in the suppression of sporadic noise and reinforcement of persistent salient artifacts (under

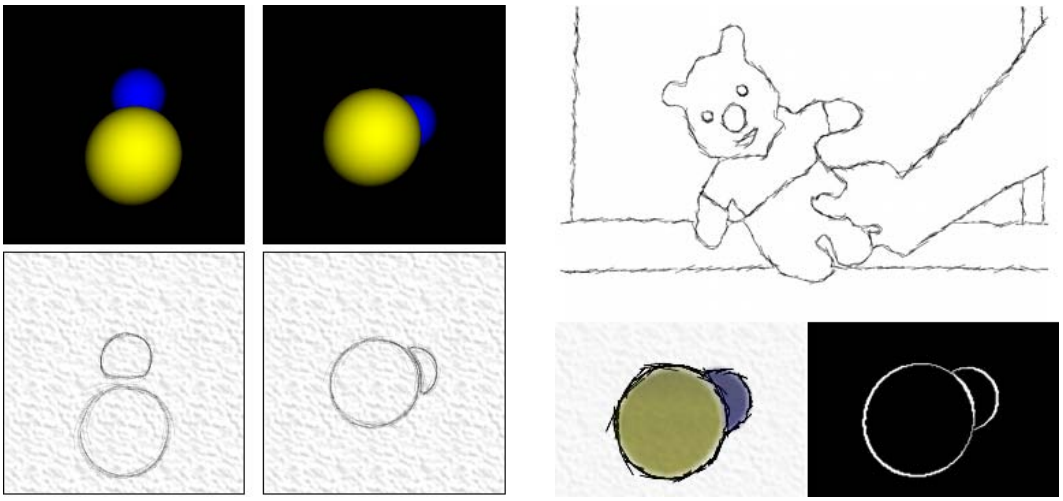


Figure 8-10 Left: Early experiments applied to a synthetic (bouncing spheres) test scene represented each video object individually in the IR by means of the individual bounding surfaces. This led to duplication of surface information, causing boundaries to no longer appear coincident following surface manipulation (for example after creating the sketch rendering effect described in Section 8.4.4). This led to unappealing artifacts such as the double edges between spheres (2^{nd} image set on left). Right: Our IR now represents video objects by means of their interfacing surfaces (Stroke Surfaces), which do not create artifacts such as double edges or holes when they are manipulated in the video volume.

the assumption that noise obeys the central limit theorem). We threshold the map, and apply morphological thinning. Stroke Surface patches are then fitted around each disconnected artifact as before. Finally, the patches are transformed to their original reference frames via the inverse homographies used to generate the map.

One shading gradient triple $\underline{G_t}$ and a homography are stored in the supplementary database, per frame of the video object. The homographies are of additional use later for rotoscoping and stroke based rendering. The new interior Stroke Surfaces are added to those of the existing representation, but with both sides of their winged edge structure set to point to the video object in which they originated. Thus interior edge markings are also encoded in our representation.

8.4 Back end: Rendering the Representation

We now describe how the IR is rendered by the back end, to produce animations in a variety of artistic styles.

Stroke Surfaces are first duplicated to produce two sets; one for the interior shading stage, and one for the line rendering stage, of the back end (Figure 8-7). These sets of surfaces may then be manipulated in some manner; for example, fine scale temporal smoothing may be applied to boost temporal coherence, or novel frequency components

introduced to the surfaces to create "coherent wobbles" (Section 8.4.1).

To render a particular frame at time t, the sets of Stroke Surfaces embedded in the video volume ($[x, y, z] \in \Re^3$) are intersected with the plane $z = t$. The frame is then rendered in a two stage process comprising: 1) shading of the object's interior region (processing the first set of surfaces); 2) rendering of the object's outline (referred to by animators as the *holding line*) and any interior cue lines also present (processing the second set of surfaces) . This separation allows us to create many novel video effects, such allowing interior shading to spill outside of the holding lines.

The remainder of this section is organised as follows. We first describe how Stroke Surfaces may be smoothed, or otherwise manipulated in frequency space, to create a number of temporal effects (subsection 8.4.1). We then describe how interior regions bounded by these surfaces are rendered, by drawing on data in the graphical attributes in the database component of the IR (subsection 8.4.2). Rotoscoping, matting and stroke based AR styles are also obtainable within this single framework (subsection 8.4.3). Finally, we describe the mechanism for rendering holding and interior lines in the animation (subsection 8.4.4).

8.4.1 Surface Manipulations and Temporal Effects

The representation of video as a set of spatiotemporal Stroke Surfaces simplifies manipulation of the image sequence in both spatial and temporal domains, and enables us to synthesise novel temporal effects which would be otherwise difficult to produce on a per frame basis. In this subsection we describe a method for manipulating the geometry of Stroke Surfaces in frequency space. We show that by applying a low-pass filter in the temporal dimension we may further improve temporal coherence, and by introducing novel frequency components we can produce coherent "wobbles" in the video reminiscent of popular commercial cartoons, for example "Roobarb and Custard" [Grange Calveley, 1972].

Planar offset parameterisation

We now describe our mechanism for manipulating a Stroke Surface $\underline{Q}(s,t)$ to produce small scale deformations, and so create spatiotemporal effects in the animation. We subsample the fields $s \in [0,1], t \in [0,1]$ at equal, user defined intervals, to obtain a grid of well distributed points on the manifold $\underline{Q}(s,t)$. By creating a simple 3D triangular mesh using these regular samples as vertices (see the geometry of Figure 8-11), we are able to create a piecewise planar approximation to $\underline{Q}(s,t)$, which we write as $\underline{P}(s,t)$.

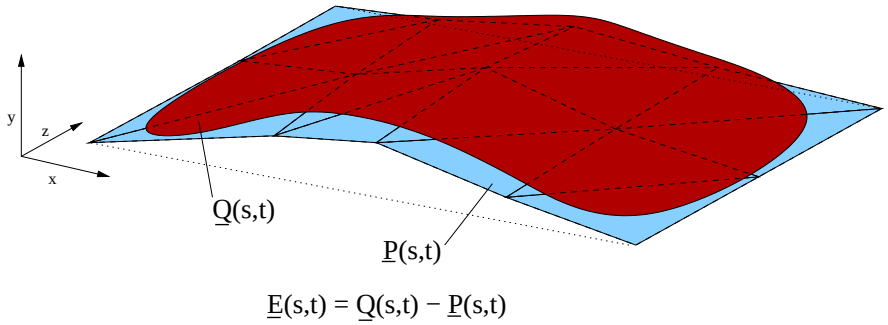


Figure 8-11 The planar offset parameterisation $\underline{E}(s,t)$ is derived by differencing corresponding points on the Stroke Surface $\underline{Q}(s,t)$ with those on a piecewise planar approximation to the surface $\underline{P}(s,t)$. $\underline{P}(.)$ is created by sub-sampling and linking points on the manifold of $\underline{Q}(.)$.

The error (offset) of this approximation is:

$$\underline{E}(s,t) = \underline{Q}(s,t) - \underline{P}(s,t) \tag{8.8}$$

Suppose we now hold $\underline{P}(.)$ constant, and allow manipulation of the offset field (we write the modified field as $\underline{E}'(.)$). We obtain a modified Stroke Surface:

$$\underline{Q}'(s,t) = \underline{P}(s,t) + \underline{E}'(s,t) \tag{8.9}$$

In our framework we allow manipulation of $\underline{E}'(.)$ through variation of a 2D scalar field $S(.)$:

$$\underline{E}'(s,t) = S(s,t)\frac{\underline{E}(s,t)}{|\underline{E}(s,t)|} \tag{8.10}$$

The identity in this framework is $S(.) = |\underline{E}(.)|$. However alternative $S(.)$ are possible, and variation of this field forms the basis for temporal effects in our framework.

To maintain continuity between neighbouring Stroke Surfaces we must reduce the magnitude of any deformations local to the surface's boundaries (both the external surface edges where either s or t are close to 0 or 1, and around the edges of any internal hole within the Stroke Surface). We define a 2D field of weights in (s,t) space by producing a binary map of these internal and external boundaries and performing a 2D distance transform. We then raise values in this field to a user defined power φ and normalise to obtain the weight field, which we write as $W(s,t)$. Parameter φ may increased by the animator to decrease the apparent rigidity of the Stroke Surface during these deformations. We revise equation 8.9 to weight any change in $\underline{Q}(s,t)$ by $W(s,t)$:

$$\underline{Q}'(s,t) = \underline{P}(s,t) + (1 - W(s,t))\underline{E}(s,t) + W(s,t)\underline{E}'(s,t) \tag{8.11}$$

Frequency manipulation of the offset field

Manipulation of the 2D field $S(s,t)$ is performed in the frequency domain by defining a complex transfer function $\mathcal{T}(\mathcal{C}^2)$ such that:

$$S'(.) = F^{-1}[\mathcal{T}(F[S(.)])] \tag{8.12}$$

where $S'(.)$ is the modified scalar field, $F[.]$ and $F^{-1}[.]$ specify the standard and inverse discrete Fourier transforms respectively, and $\mathcal{T}(.)$ is a general 2D transfer function which maps a 2D complex field to some other 2D complex field. The user is free to develop custom transfer functions, however we now give two useful examples:

1. **Temporal Smoothing:**

   We can apply a low-pass filter $S(.)$ to remove any high frequency undulations in the temporal dimension of the surface. A standard method to achieve this is through convolution with a Gaussian of user defined size σ, i.e. the following multiplicative transfer function in the frequency domain:

   $$\mathcal{T}_{smooth}(F;u,v) = \frac{F(u,v)}{\sqrt{2\pi}\sigma} e^{-\frac{v-d}{2\sigma^2}} \tag{8.13}$$

   where $F[u,v]$ specifies a component in the 2D Fourier domain, and d is a scalar such that the line $v = d$ intersects the coordinates of the d.c. component in $F[u,v]$. Depending on the implementation of the discrete Fourier transform, d may be located at zero or at the centre of the v axis.

2. **Coherent Wobbles:**

   We are able to introduce controlled wobbles and distortion effects by perturbing Stroke Surfaces, producing a distinctive pattern of incoherence in the animation. Such effects are easily generated by introducing several impulses in the frequency domain. The i^{th} impulse is defined by a randomly generated triple $[f_i, a_i, \omega_i]$; f_i is the frequency of the wobble effect; a_i is the amplitude of the effect; ω_i is the angular speed of the effect (the rate at which crests and troughs in wobble appear to "rotate" around a region's perimeter over time). We write the modified signal as:

   $$\mathcal{T}(F[\underline{x}]) = F[\underline{x}] + J(\underline{x}) \tag{8.14}$$

   where $F[\underline{x}]$ represents a the frequency component $\underline{x} = (u,v)^T$ in the 2D Fourier domain, and $J(\underline{x})$ the field containing impulses:

   $$J(\underline{x}) \quad = \quad \sum_{i=1}^{n} \begin{cases} a_i \text{ if } |\underline{x} - (\underline{D} + \underline{\nu}_i)| = 0 \\ 0 \text{ otherwise} \end{cases} \tag{8.15}$$

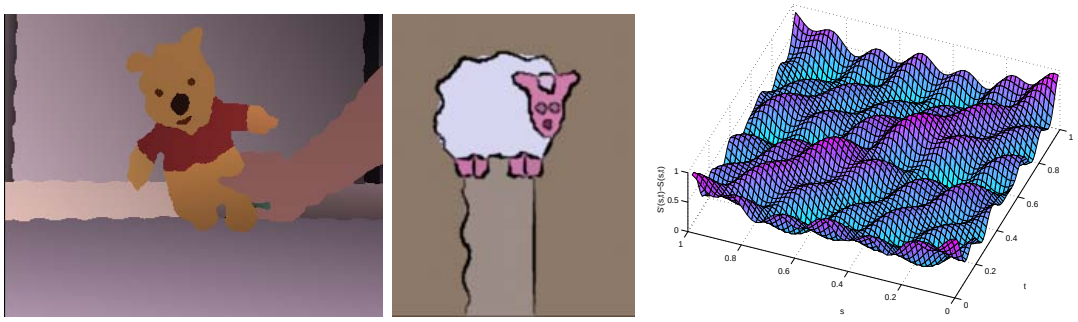


Figure 8-12 Not only can we enforce temporal coherence, but also introduce novel forms of incoherence. Left: example of a coherent wobbling effect in the gradient shaded *POOHBEAR* animation (`videos/pooh_wobblygradshade`). Middle: A example still from *SHEEP* demonstrating how the animator may transform one set of Stroke Surfaces (used to render holding and interior lines), yet leave the surfaces for interior regions unaffected (`videos/sheep_wobblycartoon`). Right: A visualisation of the random displacement field $(S'(.) - S(.))$ for a single Stroke Surface from *POOHBEAR*.

introducing the notation $F[\underline{D}]$ to specify the d.c. component in the Fourier domain, and n as the total number of impulses. Inside the summation, the term $\underline{\nu_i}$ is defined as:

$$\underline{\nu_i} = \begin{bmatrix} \cos\omega_i & -\sin\omega_i \\ \sin\omega_i & \cos\omega_i \end{bmatrix} \begin{bmatrix} f_i \\ 0 \end{bmatrix} \tag{8.16}$$

This mechanism creates smooth, periodic displacements in the 2D signal $S(.)$. This in turn produces smoothly varying undulations upon the Stroke Surface. As a result the wobbles of region boundaries appear to move coherently over time in the animation.

Since adjacent objects are represented in terms of their interfacing surfaces, residual temporal incoherences in those boundaries may be dampened by smoothing the surfaces. There is no danger of introducing "holes" into the video volume as with the coarse smoothing step (Section 8.2.3) — if volume is lost from one video object, it is gained by surrounding video objects. Similar consistency applies to all temporal effects created by this mechanism.

Figure 8-12 contains stills from animations produced using the second of these transfer functions, demonstrating that a simple periodic displacement function over Stroke Surface patches can produce dramatic and novel temporal effects in the animation. The random displacement field $S'(.) - S(.)$ for a single Stroke Surface in the *POOHBEAR* sequence has has also been visualised.

8.4.2 Rendering the Interior Regions

The process of intersecting the plane $z = t$ with the spatiotemporal Stroke Surfaces in the IR produces a series of splines which are scan converted into a buffer. Bounded regions in this buffer correspond to the interiors of video objects. In this section we describe how these interior regions are rendered to form part of the animation.

For a given region, we begin by determining exactly which video object that region is a "slice" of. This information is obtained by examining the the winged edge structures attached to the bounding Stroke Surfaces. Thus each region inherits a pointer referencing a record in the database component of the IR, which contains all the rendering information about that region at time instant t. This information may be used to render regions in a number of artistic styles.

Cartoon-style Flat Shaded Animations

Arguably the most straightforward strategy for rendering a region interior is to flat shade with a single, mean colour computed over that region's footprint in the original video. Recall that the front end recorded this colour information (at each time instant) in the database component of the IR. The cartoon-like "flat shading" effect that results goes some way to satisfying the second (shading) sub-goal of our original motivation — the automated generation of cartoons from video. However as video objects divide and merge over the course of the clip, the mean colour of their imaged regions can change significantly from frame to frame (perhaps due to shadows). This can cause unnatural, rapid colour changes and flickering in the video (see the left hand skirting board in Figure 8-13).

Thus, it is not enough to obtain temporally smooth segmentation (Stroke Surface) boundaries in an animation. One must also apply shading attributes to the bounded regions in a temporally smooth manner. The flickering of region colour we demonstrate is symptomatic of the more general problem of assigning the graphical attributes stored in the IR database, to regions in a coherent way. We draw upon our spatiotemporal representation to mitigate against this incoherence — specifically, by smoothing database attributes over time, as previously alluded in Section 8.3.

Smoothing Database Attributes

Recall that objects are associated via a graph structure; pointers to each video object's child and parent objects are stored in the database component of the IR. We analyse this graph to obtain a binary voxel map describing the union of all video objects within

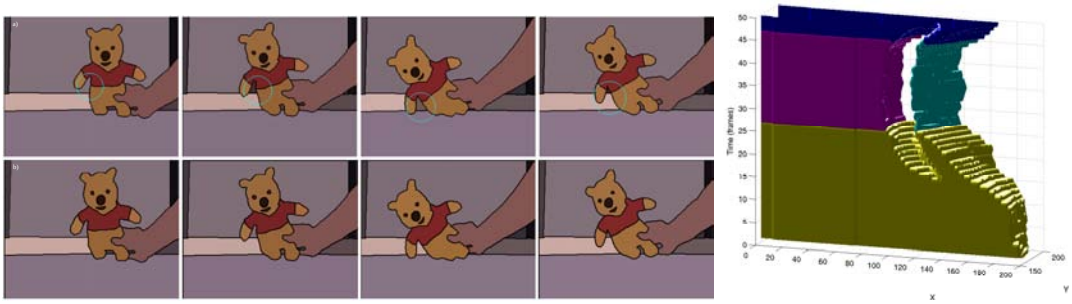


Figure 8-13 Demonstrating the temporal coherence of our cartoon-style flat shading. Top: Temporal incoherence (highlighted in blue), in the form of flickering colour, is caused by shading on a per frame basis (`videos/pooh_incoherentshade`). Bottom: Our spatiotemporal representation allows us to mitigate against these incoherences by smoothing attributes, such as colour, over the video volume (`videos/pooh_coherentshade`). Right: A volume visualisation of the left hand skirting board, comprised of a four associated video objects. Smoothing attributes with respect to this, and other volumes in the sequence, improves temporal coherence.

the subgraph containing the video object corresponding to the region being rendered. By averaging graphical database attributes (5-7, see Section 8.4.2), such as colour, over the volume we can create a smooth transition of those attributes over time (even if objects appear disjoint in the current frame but connect at some other instant (in the past or future). Such coherence could not be obtained using the per frame sequential analysis performed by current video driven AR methods [75, 96, 103].

The size of the temporal smoothing window is defined by the user; the larger the window size, the smoother the transitions of attributes over time; but consequently, rapid motions or lighting changes may not be reproduced faithfully in the animation. The animator thus varies the temporal window size until they are satisfied with the result — typical window sizes range between five and ten frames.

Gradient Shaded Animations and Morphological effects

We observed in Section 8.3.4 that the highly abstracted nature of a flat shaded video can be unappealing for certain applications; artists often make use of shading and cue marks to add a sense of lighting and depth to a scene. We can augment the flat shaded regions by rendering the gradient shading attributes fitted earlier, smoothing the parameters in a similar manner to colour to ensure coherence (Figure 8-14). Interior line cues may also be added by rendering the interior Stroke Surfaces of the object (Figure 8-17) although the rendering of such cues occurs later in the line rendering stage (Section 8.4.4).

We are also able to apply morphological operators to interior regions, prior to rendering. Figure 8-18 demonstrates a water-colour effect (combined with a sketchy outline — see

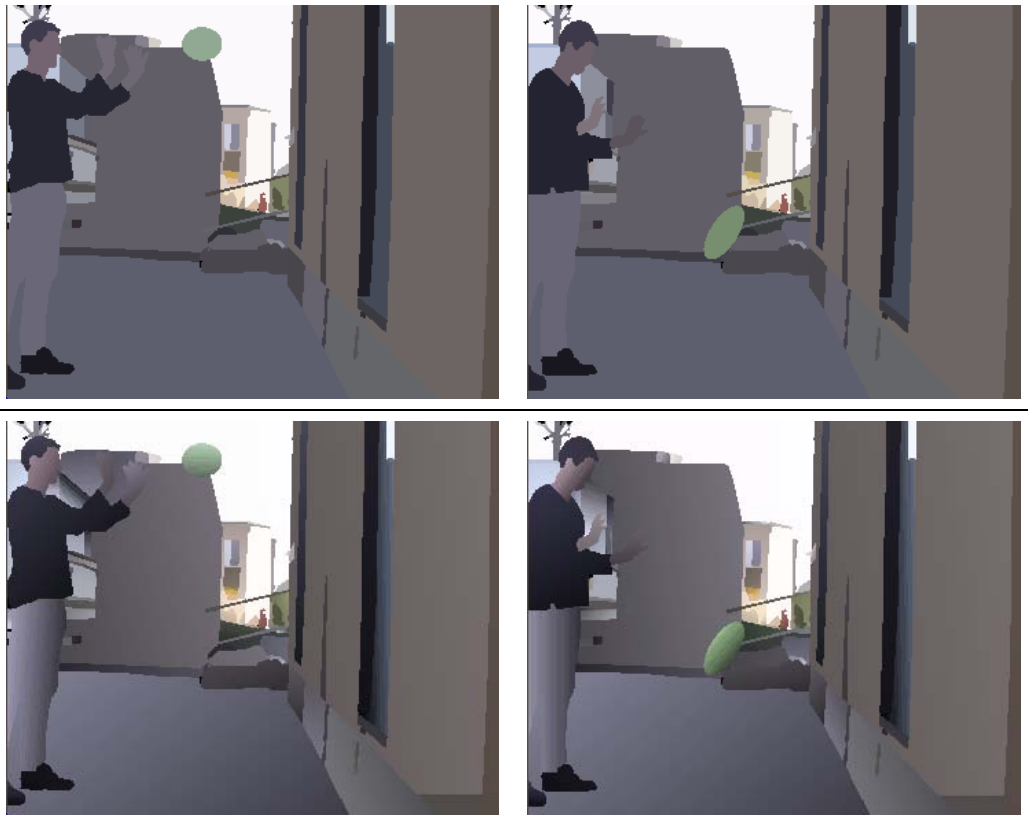


Figure 8-14 Examples of the temporally coherent flat shaded (top) and linear gradient shaded (bottom) interior rendering styles available in our framework (see `videos/bounce_flatshade`, `videos/bounce_gradshade`).

Section 8.4.4 for details), in which we have applied an erosion operator to the region prior to rendering; this gives the aesthetically pleasing impression of brush strokes stopping "just short of the edges". The effect is coherent over time (see Appendix C, `videos/pooh_watercolourwash`). The water-colour wash texture was produced via multiplication of pixel values with pre-supplied texture map.

8.4.3 Coherent Reference Frames and Stroke Based Rendering

The labour saving technique of "rotoscoping" was pioneered by cartoonists in the late 1930s, and was instrumental in the production of Disney's first feature length animated movie "Snow White" [Disney, 1937] — we refer the reader to Section 2.5.2 for further examples of rotoscoping. Traditionally, rotoscoping is the manual tracing over photographs, or film stills, to produce the stylised cels of an animation [169].

As with rotoscoping, AR algorithms draw over photorealistic footage to produce stylised output. In this sense, we argue that image-space static AR methods (for example stroke based renderers [27, 71, 103] or adaptive textures [65]) may be considered a form of modern day rotoscoping. This parallel between AR and rotoscoping is of further rele-

vance to video driven AR when one considers that both share a common objective; to produce stylised animations from video in which motion is coherent with the motion of the underlying source image sequence.

In this subsection we demonstrate that rotoscoping, video matting, and static AR can be unified in a single framework to produce temporally coherent animations from video. Our system creates temporally coherent motion estimates of image regions, requiring only a single key-frame to produce rotoscoping effects. The same framework may be used to place coherently moving brush strokes on image regions, enabling us to produce AR animations automatically from video.

Reference Frames and Rotoscoping

Recall the local motion estimate for video objects computed in Section 8.3.4, and stored in the database component of the IR. This estimate models inter-frame motion as a homography, and was previously used to create correspondence between region texture in adjacent video frames to enable recovery of internal edge cues. However this same motion estimate may be used to to implement automated rotoscoping in our framework. Animators draw a design, and attach it to a key-frame in the original footage. The inter-frame homography stored in the IR database is then used to automatically transform the design from frame to frame — the design remains static within the reference frame to which it is attached; it appears to be rigidly attached to the video object. Figure 8-15 demonstrates the attachment of this rigid frame using a synthetic chequerboard pattern. There are many useful applications for this automated rotoscoping; for example, to add personality to an animation by rotoscoping an expression on to a face (Figure 8-15, bottom). Such rotoscoping is particularly useful in our Video Paintbox, since small scale features such as eyebrows or mouths sometimes fail to be captured by the region segmentation and association processes of the front end.

Video Matting

As a special case of rotoscoping, we may set the inter-frame homography estimates for a region to the identity matrix (implying no region motion). By supplying these regions with video as a texture, rather than hand-drawn animations, we are able to replace shaded video objects with alternatively sourced video footage. This facilitates video matting within our framework, as we demonstrate in Figure 8-15 (top right) by substituting a novel background into the *SHEEP* sequence. We can also use the source video footage itself as a texture, and so reintroduce photorealistic objects from the original video back into the non-photorealistic animation. This technique produces the "mixed media" effects demonstrated in Figure 8-2 (`videos/bounce_mixedmedia`).

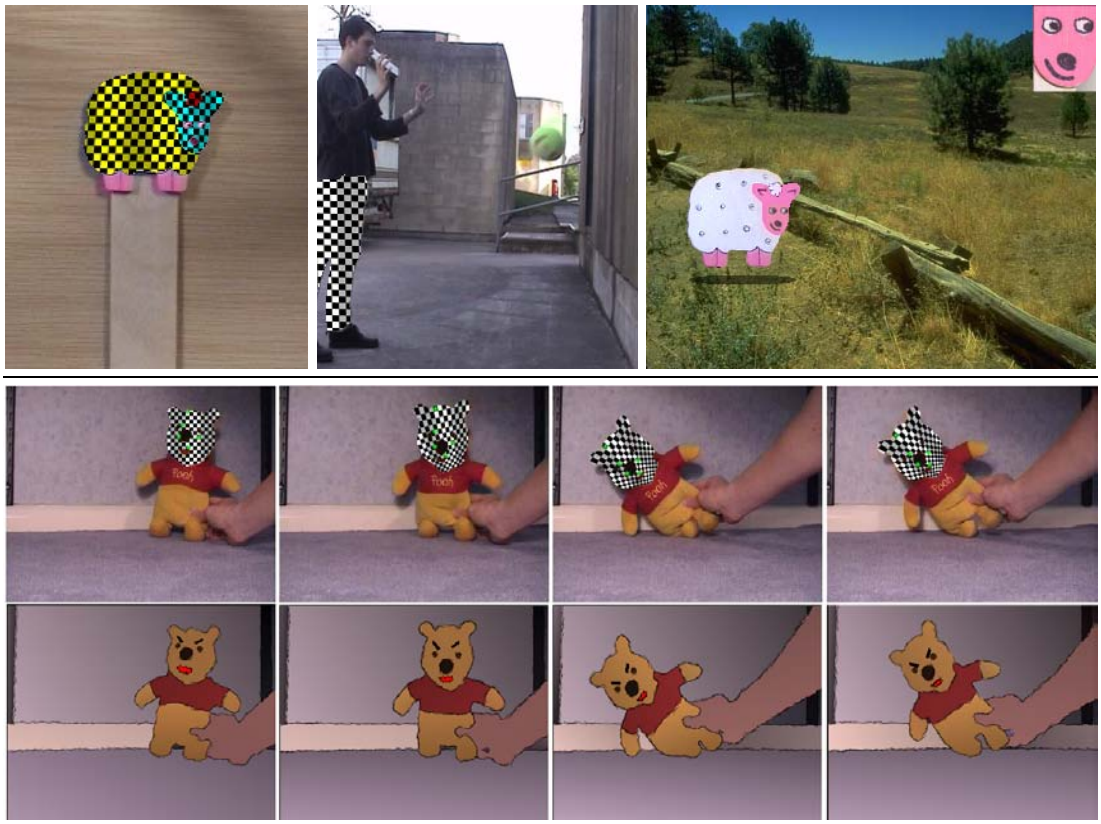


Figure 8-15 Rotoscoping and matting effects are possible within our AR video framework. Top left, middle: Rigid reference frames are attached to individual video objects via inter-frame homography (demonstrated here by a synthetic chequerboard pattern which moves coherently over time — `videos/sheep_reframe`, `videos/bounce_reframe`). Top right: The attached reference frame allows us to place markings such as illustrations, upon an object, and have those markings appear to move with the object: a form of automated rotoscoping. We may also substitute video objects (for example the background) for alternative textures to create matting effects `videos/sheep_rotomatte`. Bottom: A further example of rotoscoping, applying a new facial expression in the *POOHBEAR* sequence (`videos/pooh_angrybear`). Observe the green crosses on the chequerboard pattern; these symbolise the paint strokes which we may also attach to reference frames to produce coherent painterly effects (`videos/pooh_reframe`). Rotoscoping and stroke based AR are closely related in our rendering framework.

Note that all the effects described in this subsection require external information in addition to that stored in the IR — specifically, access to the relevant video source, image, or animator supplied illustration depending on the effect desired.

Coherent Stroke Based Rendering

Using a similar principal to rotoscoping, we are able to extend static, image-space AR methods to operate over video with a high level of temporal coherence. We observed (Section 8.1) that temporal coherence requires that:

1. Strokes should move in a manner consistent with the video content.

2. Strokes should not exhibit rapid fluctuations of their visual attributes, for example colour.

The first of these criteria may be met by rigidly attaching strokes to the local reference frames of video objects. From the viewpoint of the user, this causes stroke locations to exhibit motion matching that of the object to which they are attached. In this manner, strokes move coherently over time (see the green stroke centres in Figure 8-15). The reference frame attached to a video object defines a 2D plane of infinite extent, on which we place brush strokes. We subsample the plane at regular spatial intervals, to create potential locations for brush stroke centres. Strokes are instantiated at locations upon the sub-sampled plane when a potential stroke centre moves within the "footprint" of a video object — the region generated by intersecting the video object with the time plane at a given instant. We have found that, in contrast to rotoscoping, the full eight degrees of freedom of the homography are superfluous to needs when produced stroke based AR animations. A minimum least-squares error affine approximation to the homography produces aesthetically superior results.

The second criterion may be met by smoothing the visual attributes of strokes over time, in a similar manner to how visual database attributes in the IR are smoothed. Visual attributes, such as colour, are assigned to instantiated strokes; as with static AR, these attributes are functions of pixel data in the neighbourhood of the stroke's centre. These attributes are updated from frame to frame by re-sampling image data, causing the visual attributes of strokes to adapt to the video content. However, the state of attributes are also functions of their state in past and future time instants. The nature of this function is to constrain the values of attributes to evolve smoothly over time, so mitigating against stroke flicker. Once instantiated, a stroke is never destroyed — a record of the stroke's attributes persists for the remainder of the video sequence, even if the stroke moves out of the bounds of the footprint and becomes invisible. This retention of stroke attributes helps reduce flicker caused when strokes move repeatedly in and out of a footprint. However, note that visibility itself is one of many attributes assigned to strokes, and is therefore smoothed such that strokes do not rapidly alternate their visibility, but appear and disappear over lengthy time periods.

We now give an illustrative example of our coherent, stroke based rendering process by extending our pointillist painterly rendering technique of Chapter 3 to video.

Recall that each brush stroke in the painterly method of Chapter 3 formed a z-buffered conic of superquadric cross-section. Each stroke has seven continuous visual parameters:

1. α – Superquadric form factor

2. a – Aspect ratio of stroke
3. b – $1/a$
4. θ – Orientation of stroke
5. h – Height of stroke in z-buffer (implicit ordering of stroke)
6. $\underline{c}$ – Colour of stroke (an RGB triple)
7. r – Scale factor of superquadric base

In addition to these parameters each stroke also has two further attributes:

8. $(x, y)^T$ – the 2D location of the stroke's centre, relative to the video object's reference frame.
9. v – a continuous "visibility" value $v = [0, 1]$, where a value of $v \geq 0.5$ implies that the stroke is visible (i.e. should be painted)

The collection of attributes (1-9) represents the complete state of a stroke (written as a vector, $\underline{S}_t$) at some time offset t from that stroke's creation. Upon instantiation (at $t = 0$) we construct $\underline{S}_0$ as follows. Attributes 1-7 are determined from the footprint texture, in exactly the same manner as described in Chapter 3. Attribute 8 (location) is determined by the stroke's location on the reference frame, and is constant relative to this reference frame for the duration of the video. Attribute 9 (visibility) is set to 0.5.

At later instants ($t > 0$) we produce a vector of updated stroke attributes $\underline{S}_t$ in a similar manner. Attributes 1-6 are sampled from the footprint texture at time t. A value "ρ" for attribute 7 (scale of base) is determined as before, but modified to take into account any scaling applied to the reference frame. This is necessary, since if the reference frame is scaled up by a large factor, then the fixed, regular spacing of stroke could cause unpainted gaps to appear in the canvas. The scale component "s" of the affine transformation applied to the reference frame can be obtained using SVD (s is a product of the eigenvalues, computed for the transformation's linear component). The value r for attribute 7 is simply $r = \rho s$. Attribute 8 is a constant, and so simply copied from time S_{t-1}. Attribute 9 is set to 0 if the stroke's centre is outside the footprint, or 1 if it is inside.

The state of each stroke $\underline{S}_t$ is computed for all t, prior to the synthesis of any animation frames. To mitigate against stroke flicker, for example small, rapid changes in colour or visibility due to video noise, we smooth each stroke's attributes over time by independently low pass filtering each component of the signal $\underline{S}_t$. By default we use a Gaussian with a standard deviation of 3 frames, but allow the user to override this if

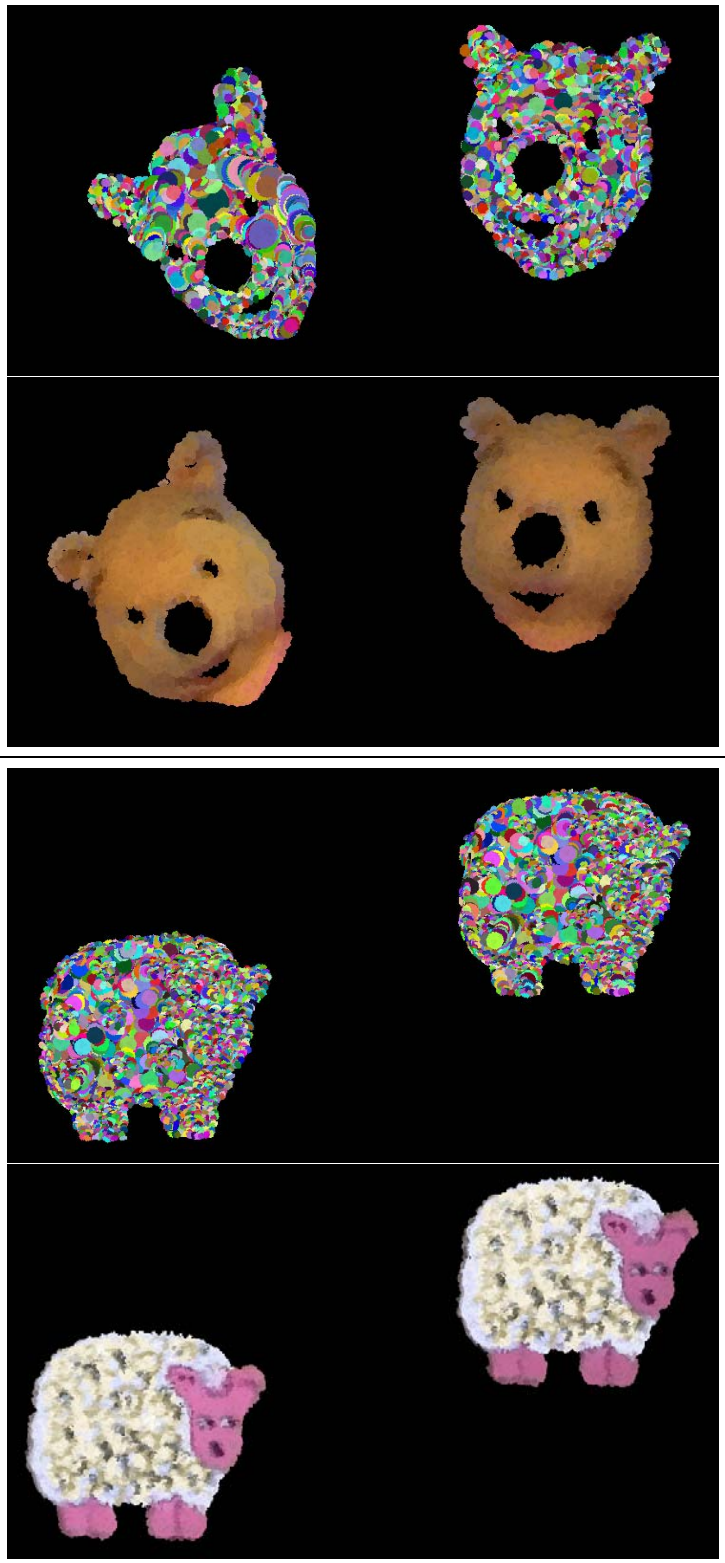


Figure 8-16 Illustrating the coherent painterly effect produced by extending our pointillist-style painting algorithm (Chapter 3) to video. We have used false colour to illustrate the coherence of individual strokes within the animation, however the reader is referred to Appendix C where complete animations demonstrating this effect have been included.

desired. For example, smaller scale filters could be used where stroke visual attributes must change at a faster rate — in these cases, rapid attribute changes are due to video content rather than video noise.

Implicit in our decision to perform temporal smoothing on stroke attributes is the assumption that video noise obeys the Central Limit Theorem. As the temporal window over which we integrate increases, signals will be reinforced and sporadic noise will be cancelled out. Similar assumptions regarding video noise have been by others; for example Capel [13] registers and averages video frames to achieve noise reduction in video surveillance images.

Rendering strokes with their smoothed attributes yields painterly animations exhibiting a uniquely high level of temporal coherence. Figure 8-16 gives examples of painted output generated by our system, including visualisations of strokes in false colour, where attribute 6 (colour $\underline{c}$) was fixed at a random, constant value for the duration of the sequence. The reader is referred to the source videos and corresponding painted animations available in Appendix C which demonstrate the temporal coherence of our algorithm. We compare our approach with the state of the art in video painting [103], in Section 8.6.

8.4.4 Rendering the Holding and Interior Lines

Having concluded our explanation of interior region rendering, we now describe the edge rendering process of the back-end. Recall the surface intersection operation by which we determine the Stroke Surfaces to be rendered a particular frame. The splines which result from this intersection form trajectories along which we paint long, flowing strokes, which are stylised according to a user selected procedural AR brush model. This produces attractive strokes which move with temporal coherence through the video sequence; a consequence of the smooth spatiotemporal nature of the Stroke Surfaces. We have used an implementation of Strassman's hairy brush model [150] (Section 2.2.1) to render our splines, producing thick painterly strokes to depict the exterior (holding) lines of objects (Figure 8-17, left).

Brush models (for example, Strassman's model) often invoke a stochastic process to simulate effects such as brush bristle texture. Without due care this non-determinism can cause swimming in the animation. Such models require only the *illusion* of randomness for aesthetics, and we have found that seeding the pseudo-random number generator with a hash of the unique ID number of a stroke surface is a convenient way of creating "reproducible" randomness for the duration of a Stroke Surface patch; a simple manifestation of a "noise box" [51].

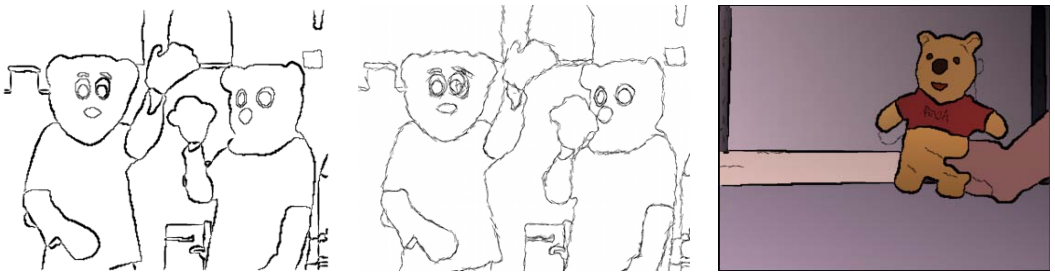


Figure 8-17 Examples of various line rendering styles available in our system. Left: Thick painted brush strokes generated by sweeping Strassman's [150] brush model over spline trajectories produce by intersecting a time plane with Stroke Surfaces in the *WAVE* sequence. Middle: A sketchy effect applied to the same video, generated by duplicating, shattering and jittering Stroke Surfaces prior to intersection. Each resulting splines has been rendered using a fine brush tip. Right: Combining the thick brush tip style on exterior cue (holding) lines (Stroke Surfaces with heterogeneous winged edge pointers) and a thin brush tip style on interior cue lines (Stroke Surfaces with homogeneous winged edge pointers).

The Stroke Surfaces whose intersection results in the spline strokes may be manipulated, prior to rendering, to produce a number of effects. In Section 8.4.1 we described a coherent wobbling effect that may be applied to both outline and interior Stroke Surface sets. However, effects specific to line rendering may also be applied, as we now explain.

Sketchy stroke placement

Artists often produce sketchy effects by compositing several light, inaccurate strokes on canvas which merge to approximate the boundary of the form they wish to represent. We may apply a similar, coherent, sketchy effect to video using a similar, two stage, technique applied to our Stroke Surfaces.

Stroke Surfaces are first shattered into many smaller individual Catmull-Rom [14] bi-cubic patches (Figure 8-18). The spatial and temporal intervals for this fragmentation are user parameters through which the appearance of the final animation may be influenced; small spatial intervals create many small sketchy strokes, and very large temporal intervals can create time lag effects. Each of these bi-cubic patches becomes a stroke in its own right when later intersected and rendered.

Second, each bi-cubic patch is subjected to a small "jitter" — a random affine transformation $\underline{\underline{M}}$ — to introduce small inaccuracies in the positioning of patches. Specifically:

$$\underline{\underline{M}} = \underline{\underline{T}}(\underline{\tau})\underline{\underline{T}}(-\underline{c})\underline{\underline{S}}(\sigma)\underline{\underline{R}}(\rho)\underline{\underline{T}}(\underline{c}) \tag{8.17}$$

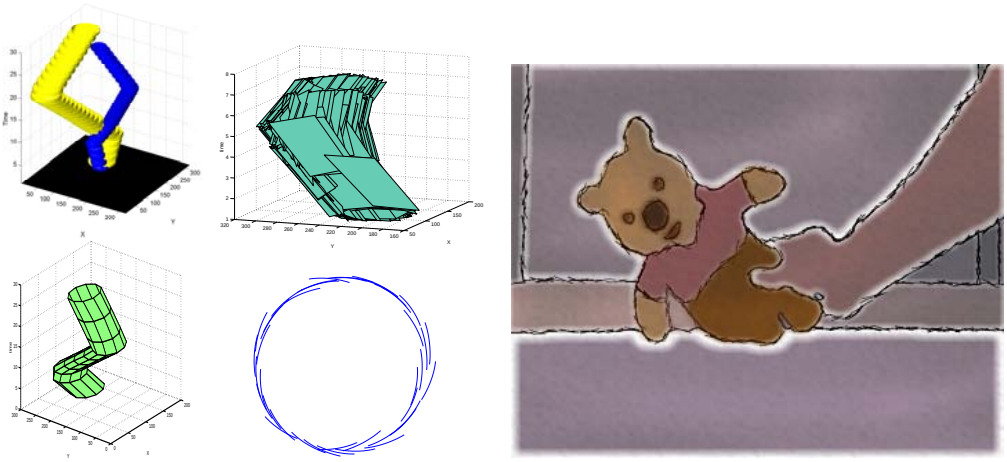


Figure 8-18 Left: A test sequence of bouncing spheres is segmented (top left). We visualise the Stroke Surface about one sphere, prior to (bottom left), and following (top right) surface shattering; we approximate surfaces here with piecewise linear patches for ease of visualisation. When shattered patches are intersected, coherent sketchy effect is formed (bottom right). Right: A still from a coherent animation produced from the *POOHBEAR* footage. Here the sketch effect on the holding line has been combined with a watercolour wash on the interior (`videos/pooh_watercolourwash`).

where:

$$\underline{\underline{T}}(\underline{x}) = \begin{bmatrix} \underline{\underline{I}} & \underline{x} \\ 0 & 1 \end{bmatrix}, \quad \underline{\underline{R}}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \underline{\underline{S}}(s) = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8.18)$$

$\underline{c}$ is the centroid of the patch, and $\underline{\tau}, \sigma, \rho$ are normal variates which control the jitter, and are typically of low magnitude (the user is given control over their limits). Figure 8-18 gives a visualisation of the resulting, perturbed patches.

The bi-cubic patches are intersected with a time plane ($z = t$) as before, and the resulting splines rendered with a fine tipped brush to yield a sketchy effect. Each stroke is guaranteed to be coherent over its temporal extent, since it results from intersection with a smooth bi-cubic patch embedded in the spatiotemporal volume. Finally, the density of sketching may be increased by further duplicating the Stroke Surfaces prior to shattering and jittering in the manner described.

It is a matter of artistic taste whether sketch lines should be re-sketched at a constant rate for the duration of the video, or whether they should only be re-sketched as the object moves. Currently our framework subscribes to the former point of view, but could be easily modified to produce the latter. By substituting the normal variates τ, σ, ρ for values picked from a noise-box parameterised by spatiotemporal location,

strokes would appear to be "re-sketched" only when the location of a Stroke Surface changes i.e. during object motion.

Other line styles

We do not concern ourselves unduly with the stylisation of individual strokes. We defer the problem of artistic media emulation to the literature, being concerned primarily with stroke placement rather than stroke rendering. However we have found that interesting results can be obtained by varying line weight according to some transfer function. Some successful transfer functions we have experimented with vary the line weight in proportion to:

- the maximum of the speed of the two objects it bounds.
- the maximum area of the two objects.
- the intensity gradient between the two objects.

The latter suggestion helps to mitigate against artifacts produced when a feature has been over-segmented, leading to, say, a face broken into two features divided by a thick black line. If there is little evidence for an edge in the image at that boundary, then the stroke may be omitted.

Note that interior cue lines generated in Section 8.3.4 are naturally accommodated into this framework, since they are simply further examples of Stroke Surfaces. Exterior and interior lines can be easily discriminated if desired, by examination of the Stroke Surfaces' winged edge structure, and may be rendered in differential styles (see Figure 8-17).

8.5 Interactive Correction

Feature sub-volumes may become over-segmented in the video volume, producing two distinct graphs of video objects where one would suffice. This situation arises when the initial segmentation algorithm consistently over-segments a feature over several video frames, often because of local illumination variance, to the extent that the region association process does not recombine the over-segmented regions. Since Computer Vision is unable to provide a general solution to the segmentation problem, such errors are unavoidable in our system.

We therefore provide an interactive facility for the user to correct the system by merging video objects as required. This operation takes place directly after the region

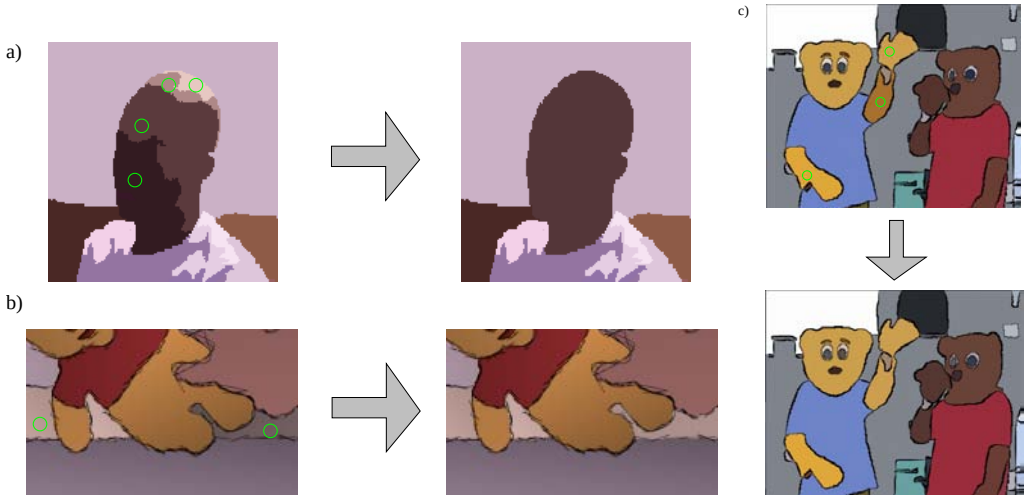


Figure 8-19 Users may create manual associations between objects to tune the segmentation or composition. (a) User creates a physical link between over-segmented objects, a single object replaces four. (b,c) User creates a semantic link between two objects. The objects remain distinct, but associations are created in the object association graph; during rendering attributes are blended between regions to maintain temporal coherence.

association and filtering process. Objects are linked by point-and-click mouse operations in a single frame, and those changes propagated naturally through to all other frames in which the object exists (since objects are spatiotemporal in their nature). We tend to bias the parameters to the EDISON [19] segmentation algorithm (see front end, Section 8.2.2) slightly toward over-segmentation, since over-segmentation is much more easily resolved via merging objects, than under-segmentation (which introduces the complicated problem of specifying a spatiotemporal surface along which to split video objects). The user may form two types of corrective link:

1. **Soft Link:**

   The animator semantically links two objects by creating edges in the object association graph. Objects remain as two distinct volumes in our representation, but the graph is modified so that any smoothing of graphical attributes (such as region colour) occurs over all linked objects (see Figure 8-19b,c). This type of link is often used to encode additional, semantic knowledge of the scene (for example, the continuity of the skirting board in Figure 8-19b).

2. **Hard Link:**

   The animator physically links two adjacent video objects by merging their voxel volumes. The objects to be merged are deleted from the representation and replaced by a single object which is the union of the linked objects (see Figure 8-19a). This type of link is often used to correct over-segmentation due to artifacts such as shadow, or noise, and is preferable in this respect to "soft" linking, since

the merged volume will undergo subsequent coarse smoothing as a single video object.

Interactive correction can also be used to control focus in the composition, for example by coarsening the scale of chosen region in the video; this is similar to the process driven automatically by our salience measure in the painterly rendering of Chapter 3. In the future we would also like to drive this video process automatically using a perceptual model.

Extensive correction of a sequence is rarely required, and correction itself takes little time since one click can associate spatiotemporal objects over multiple frames. Similarly, we allow the user to selectively chose rendering parameters for specific objects — for example, in Figure 8-15 (top right) specifying a region should be photorealistically matted rather than animated using the global specified settings. Again, such interaction requires only a couple of mouse clicks to modify parameters for video objects, and is a level of interaction which we wish to retain in our Video Paintbox to allow creative direction of the artistic process by the animator.

8.6 Comparison with the State of the Art

The majority of artistic styles that may be synthesised by the Video Paintbox have no parallel in the existing video driven AR literature; these techniques address only the single, specific problem of producing temporally coherent painterly animations. This improved diversity of style is one of the principal contributions of the Video Paintbox. However, we also wish to demonstrate the contribution made due to the improvements in temporal coherence available through our framework.

We begin by recapitulating and expanding on our description of existing video-driven painterly algorithms, which make use of either inter-frame differencing (Section 8.6.1) or inter-frame optical flow (Section 8.6.2) to improve temporal coherence. We then perform a comparison (Section 8.6.3) between these and the technique we developed in Section 8.4.3, which applied the coherent stroke based rendering component of our proposed framework to produce painterly animations from video.

8.6.1 RGB Differencing

Hertzmann and Perlin [75] proposed a video-driven painterly technique which computes the RGB difference between adjacent frames of video to locate regions of high motion magnitude. Their contribution was to paint the first frame of video using Hertzmann's static technique [71], and then update, or "paint over", regions with high motion magnitude in the next frame. Recall that animations swim if each frame is

repainted individually. Hertzmann and Perlin thus attempt to localise swimming to only the "moving" regions of the animation.

There is one clear advantage that this method holds over optical flow driven techniques (Section 8.6.2). Real-time frame rates (reportedly up to 6 frames per second) are possible on standard PC hardware, facilitating interactive applications. Without specialised hardware, reliable optical flow estimation algorithms are still too slow to facilitate real-time rendering. The same disadvantage applies to our artistic rendering subsystem, primarily because of its high level temporal nature (examining both "past" and "future" events in the video volume, rather than rendering on a per frame basis).

However there are three disadvantages to the frame differencing methodology:

1. First, although expedient to process, RGB differencing is a very poor method of determining the motion magnitude of imaged objects in a scene. The approach fails under varying lighting conditions, and assumes neighbouring objects are highly contrasting. Slow moving objects are hard to detect, since the threshold used to exclude small inter-frame differences caused by video noise, also exclude low speed motions.

2. Second, there is no internal consistency of motion within an object, since motion estimates are made on a per pixel rather than a per region basis — no higher level spatial grouping is used to assist motion estimation (in contrast to the higher level spatial analysis of our spatiotemporal method). As a result, moving flat-textured objects tend to be detected as collections of moving edges with static interiors. In Section 8.6.2 we will see that similar objections hold for optical flow.

3. Third, there is the problem discussed previously regarding the per frame sequential nature of processing. Errors in motion estimation accumulate and propagate through subsequent frames. For example, a region of the video containing an arm may move, but be misclassified as stationary by the differencing operation. This leaves a "phantom image" of the arm at one location in the painting which persists throughout the animation until something moves over the phantom arm's location causing "paint over" to occur again. The reduction in flicker is typically at the cost of an inaccurate and messy painting.

Temporal coherence of painterly animations is improved using Hertzmann and Perlin's method, but only within static regions. Temporal coherence is not attainable within moving regions, since these regions are repainted with novel strokes, laid down in a random order. Unlike optical flow based methods [96, 103] it is not possible to translate strokes with the video content, because frame differencing produces estimates only for motion magnitude, rather than both magnitude and direction. Recognising

the improved accuracy (but non-interactive frames rates) of optical flow, Hertzmann and Perlin describe how to adapt their system to use optical flow at the end of their paper. This enables their system to be adapted for offline video processing.

8.6.2 Optical Flow

Litwinowicz [103] proposed the first optical flow driven painting algorithm for video[4]. The algorithm paints short, linear brush strokes to produce a painterly effect on the first frame of video. Brush strokes are then translated from frame to frame to produce subsequent frames of animation. The vector field for the translation of strokes is determined by an inter-frame motion field, computed using an optical flow technique [7]. As discussed in Chapter 5, the per frame sequential processing of video causes the accumulation of large positional errors over time. This error is manifested either by a rapid flickering, or by the motion and painted content of the animation becoming non-representative of the underlying video.

Translation can cause dense bunching of strokes, or the density of strokes to thin causing holes to appear within the canvas (Figure 8-20, middle). Thus after the stroke translation stage there is a further *stroke density regulation* stage. Strokes in areas deemed too dense are culled at random, until the density falls below a preset "acceptable" threshold. New strokes are inserted into sparsely populated regions until the stroke density becomes acceptable. The order in which strokes are painted is preserved between frames to mitigate against flickering. Any new strokes inserted into the sequence are "mixed in" with old strokes by inserting them at random into this ordering.

The principal advantage of the optical flow methodology is that the strokes move with the content of the underlying video sequence — rather than moving regions being repainted afresh, as with frame differencing. This yields a substantial improvement in temporal coherence. However, there are also a number of disadvantages of this methodology:

1. First, the non-determinism of the stroke insertion strategy (due stroke density regulation) causes a significant level of flickering (Figure 8-20).

2. Second, the temporal coherence of the animation is only as good as the underlying optical flow algorithm and, in general, optical flow algorithms perform poorly on scenes exhibiting flat textures, occlusion, or lighting changes. No accurate, general, optical flow algorithm exists since the task demands solution of an underconstrained "correspondence problem". The resulting errors accumulate over

[4]Kovacs and Sziranyi [96] published a similar optical flow driven algorithm some years later in the Computer Vision literature (see Section 2.5.2 for a comparison of this and Litwinowicz's algorithm).

Figure 8-20 Identical frames four seconds into the *BOUNCE* sequence rendered with our technique (right) and SoA (left, middle). The errors that have built up in the cumulative optical flow estimate at this stage cause large scale distortion and flickering in the image. We have omitted the stroke density regulation stage (SoA-) in the left-hand image; this improves temporal coherence. However large holes are created in the canvas (red circle), and elsewhere tight bunching of strokes causes the scene to tend back toward photorealism. The reader is referred to the videos in Appendix C where the improved aesthetics and temporal coherence of our method are clearly demonstrated (see `videos/bounce_painterly_SoA`, `videos/bounce_painterly_SoA-` and `videos/bounce_painterly_ourmethod` for left, middle and right respectively).

time, as a consequence of the per frame sequential processing model employed. After only a short while (around 1 or 2 seconds depending on video content) this causes a gross disparity between source video and animation in terms of both content and motion.

3. Third, as with RGB differencing the motion of each brush stroke is estimated individually. No higher level spatial grouping of strokes into semantic regions is performed which could be exploited to improve temporal coherence (for example, to ensure consistency of stroke motion within a single object).

8.6.3 Comparison Methodology

The intended application of our Video Paintbox is as a post-production video tool; real-time frame rates are not required. Thus for our application, the more accurate optical flow methodology (rather than RGB differencing) is regarded as the current state of the art. We have implemented Litwinowicz's optical flow driven algorithm [103] (hereafter referred to as "state of the art", or "SoA") and now draw comparisons between the temporal coherence of this, and our proposed, painterly technique[5].

Recall our two criteria for a temporally coherent animation:

1. The locations and visual attributes of rendering elements should not vary rapidly

[5]For reference, the full paper detailing Litwinowicz's method is included in Appendix C.

(flicker) over time. Assessment of this criterion requires measuring the rate of change of these properties in the *rendered* image sequence.

2. The motion of strokes should agree with motion of content with the source video. Assessment of this criterion requires measurement, and comparison, of the motion fields of both *source* and *rendered* image sequences.

The second criterion is problematic since it demands an error-free means to determine the magnitude and direction of motion in the *source* video — if such a technique was known, we would have already built it into our Video Paintbox! Thus we require an externally supplied ground-truth motion field to evaluate this criterion.

We now describe two experiments which test each of these criteria respectively. We document the results of these experiments separately in Section 8.6.4, and draw conclusions from the results.

Experiment 1: Measurement of Flicker in the Animation

Our first experiment quantifies the level of stroke flicker in the rendered animation. Stroke flicker is a contributor to temporal incoherence, and is manifested as:

1. Rapid fluctuations in stroke location

2. Rapid fluctuations in stroke visual attributes

In both SoA and our algorithm, a stroke's location is defined by a 2D vector; we write $\underline{L}_t = (x, y)^T$ to specify the normalised coordinates (x, y) of a brush stroke's centre within the frame. Likewise, a stroke's visual attributes may be defined as a point in a high-dimensional space. Each stroke in our painterly method has seven visual attributes (see Chapter 3), in SoA there are two: colour and orientation. For the purposes of comparison we specify visual attributes as points in a 4D space $\underline{A}_t = (r, g, b, \theta)^T$ where the triple $(r, g, b \in [0, 1])$ specifies stroke colour and $\theta \in [0, \pi]$ specifies a rotation from the vertical (strokes are symmetrical about their principal axes) in both algorithms. We concatenate both location and visual parameter vectors to form a single vector representing the stroke state: $\underline{V}_t = [\underline{L}_t^T \;\; \underline{A}_t^T]^T$. We normalise the space of $\underline{V}_t$ so that distance $|(0, 0, 0, 0, 0, 0)^T - (1, 1, 1, 1, 1, \pi)^T|$ is unity; this normalises our flicker measure to range [0,1].

For a single stroke, the absolute rate of fluctuation of $\underline{V}_t$ at time t may be computed using a finite difference scheme:

$$R(t) = \big||\underline{V}_t - \underline{V}_{t-1}| - |\underline{V}_{t+1} - \underline{V}_t|\big| \tag{8.19}$$

This measure quantifies the absolute rate of change of visual state for a single, existing stroke. For strokes that are created or destroyed due to the stroke density regulation step of SoA, we set $R(t) = 1$. Note that this density equalisation step does not form part of our painting algorithm.

We write $\hat{R}(t)$ as the absolute rate of change of visual state averaged over all strokes present in frame t — hence quantifying the level of flicker present in frame t. Our objective measure of the average level of stroke flicker over all n frames of the animation is the mean of $\hat{R}(t)$ over all t.

Additionally, we may obtain a qualitative estimate of the level of flicker by visual inspection of the animation.

Experiment 2: Measuring the similarity between motion in the source and rendered sequences

The smaller the error between the motion vector fields of the source video and target animation, the greater the temporal coherence of the animation. We therefore produce a source image sequence, for which we know the ground truth motion, and render that image sequence using the algorithms under evaluation. The motion vector field for the target animation is obtained from the motion vectors of the brush stroke centres. We then perform a comparison of the ground truth and brush stroke motion vector fields.

A number of synthetic source test sequences are used for this experiment; examples of both flat and textured image regions, which are independently subjected to translation, rotation and scaling transformations. The ground truth motion vector field at time t, which we write as $\underline{S}_t(i,j) = (r(t), \theta(t))$, is compared with the motion field of the target animation at the identical instant $\underline{T}_t(i,j) = (r(t), \theta(t))$. These motion fields are in polar form ($r(.)$ is the magnitude of displacement, $\theta(.)$ is the direction of displacement — both $r(.)$ and $\theta(.)$ are normalised to range between zero and unity). The mean squared error (MSE) between the two fields is computed to obtain a measure of the similarity between the two motion fields at time t; we write this measure as $C(t)$:

$$C(t) = \frac{1}{xy} \sum_{i=1}^{x} \sum_{j=1}^{y} |\underline{S}_t(i,j) - \underline{T}_t(i,j)|^2 \tag{8.20}$$

where x and y are the frame width and height respectively. Our objective measure $\mathcal{C}$ of the average level of motion dissimilarity for the animation is the mean of $C(t)$ over all time. As with experiment one, this measure can be verified by qualitative visual inspection of the source and animated image sequences. In this manner qualitative

	SoA	SoA-	Our method
SPHERES	0.62	0.45	0.05
BOUNCE	0.79	0.65	0.11
SHEEP	0.78	0.61	0.13

Table 8.1 Table summarising the "flicker level" (see experiment 1) present in an animation produced by each of three algorithms (horizontal), over each of three source video sequences (vertical). Flicker level ranges from zero (no flicker) to unity (severe flicker).

estimates may be made for real source video sequences, for which there is no available ground truth.

8.6.4 Results and Discussion

Both experiments were conducted to compare the temporal coherence of animations produced by both our algorithm and SoA. We now discuss the results.

Experiment 1: Flicker in synthesised animation

We applied both algorithms to one synthetic sequence (*SPHERES*) and two real sequences. Of these two real sequences, one (*SHEEP*) contained regions of predominantly flat texture and the other (*BOUNCE*) was a natural scene containing moderately complex texture. We also tested SoA with, and without, the stroke density regulation step (we refer to SoA without this regulation step as SoA-). Table 8.1 summarises the levels of flicker measured within the resulting animations. Note that this measure of flicker quantifies the rate of change of stroke properties in the animation. By this definition, all animations should exhibit some degree of "flicker". However for a single video, this measure will be comparatively large in the case of frequent, rapid changes in stroke attribute and position (flicker), and lower in cases of smooth, less sporadic motion caused by movement of content in the video.

Our results indicate that SoA exhibits about half an order of magnitude greater stroke flicker than our method, on both real and synthetic video sources. All algorithms appear to perform consistently regardless of the level of texture in the video. This can be explained, since although optical flow tends to produce worse motion estimates in cases of flat texture, this experiment measures only flicker in the animation, not accuracy of motion (this is addressed by experiment 2). Omission of the stroke density regulation stage (SoA-) does reduce the level of flicker, but damages the aesthetics of the animation as "holes" appear upon the canvas.

We conclude that our painterly approach produces animations exhibiting significantly less flickering than the state of the art. Visual inspection of the resulting painterly

animations verifies these results (see Appendix C). The reduction in flicker can be explained by:

1. Our use of a robust motion estimation technique, which takes advantage of spatial grouping (segmentation) to move all strokes attached to objects in a similar way — rather than moving each stroke in isolation of all others. Error in motion estimation is spread over the entire region, making the approach more tolerant to noise that per stroke optical flow techniques.

2. The absence of a stochastically driven "stroke insertion" step in our algorithm. Unlike SoA, our approach does not require a "stroke insertion stage" since there are potentially an infinite number of strokes rigidly attached to the planar reference frame that moves with video objects. The state of strokes on the plane persists even whilst a stroke is not visible in a frame of the animation. This is not true with SoA — when strokes disappear, for example due to occlusion, they are deleted (as strokes bunch together), and then reinitialised after the occlusion (triggered by strokes spreading too thinly) without taking into account their previous state. The strategy of inserting new strokes at a random order was chosen in SoA to prevent regular artifacts from appearing in the video sequence during stroke insertion. We observe that similar use of non-determinism has been used elsewhere in Computer Graphics (for example the noise introduced by Cook's [30] distributed ray tracing), and also in many static AR algorithms to mask the regular, machine origins of the artwork produced. However non-determinism must be used sparingly in the context of AR animations, since it introduces flicker. Since our method does not require a stroke insertion step, stochastically driven or otherwise, the temporal coherence of the resulting animations is greatly improved.

3. In our system the visual attributes of strokes, for example orientation and colour, are sampled from each frame in the video, but smoothed over time under the assumption that noise obeys the Central Limit Theorem. In SoA these attributes are sampled from the video frame, and directly applied to strokes without taking into account the state of those attributes in previous or future frames. Point-sampling is highly susceptible to video noise (especially when sampling from derivative fields — such intensity gradient to obtain orientation). With SoA the sporadic fluctuations in sampled values result in the unsmoothed sporadic fluctuation of visual stroke attributes. This is not so with our approach.

4. The thresholded Sobel edges used to clip strokes in SoA are prone to scintillation over time. This in turn causes the lengths of strokes to fluctuate causing increased flickering. This clipping mechanism is not a component of our approach, and so introduces no such difficulty. Instead, we smoothly vary the continuous "visibility" attribute of strokes over time.

Experiment 2: Similarity between motion fields of source and animation

Experiment two measures the degree of similarity between the motion fields of both source and rendered image sequences. We have used only synthetic source sequences for this experiment, since we require an accurate ground truth motion estimate for the input video. Both our algorithm and SoA were modified to output a vector field corresponding to stroke motion in the animation, rather than an animation itself. We tested both flat (*SMILE*) and textured (*SPICES*) image regions under translation, rotation and scaling transformation, comparing the motion fields in both source and rendered footage.

Figures 8-21 to 8-26 present the results of this experiment. In all figures the top row shows the original and transformed images (i.e. the first and second frames of the video sequence). The second row shows the ground truth motion field, and a colour visualisation of motion magnitude. The third row shows the dense motion field used to move strokes in our system. The fourth row shows the optical flow generated motion field used by SoA. In all cases, we observe that the motion field generated by our method closely matches the ground truth. The MSE (equation 8.20) between our field and the ground truth is approximately 0.05 for all transformation classes. The MSE between the optical flow derived stroke motion field and the ground truth varies between 0.5 and 0.7; an order of magnitude less accurate.

We draw attention to the false negative readings returned by optical flow for flatly textured regions (regions of near constant intensity) within the images. By contrast optical flow estimates around most of the edges appear to reasonably accurate. The results of this error can be seen most clearly in the resulting painterly animations, where brush strokes in the centres of flat regions remain static, but strokes around the edges move. This conveys contradictory motion cues to the viewer; strokes around the edges of an object appear to move in the opposite direction relative to those in the middle of the object. By contrast the homography based motion field generated by our method is computed over the entire image region. This produces an accurate motion estimate even within flatly textured regions, and ensures that stroke motion is consistent within individual regions in the video.

In summary, the higher spatial level of processing performed by our technique performs motion estimation on a per object, rather than per pixel basis. Errors in motion estimation are thus distributed over the entire object, rather than individual strokes. Similarly the higher level of temporal processing performed by our technique smooths stroke attributes over time. Measurements of stroke attributes in adjacent frames combine to reinforce each other and cancel out noise. Robustness to such errors, produced

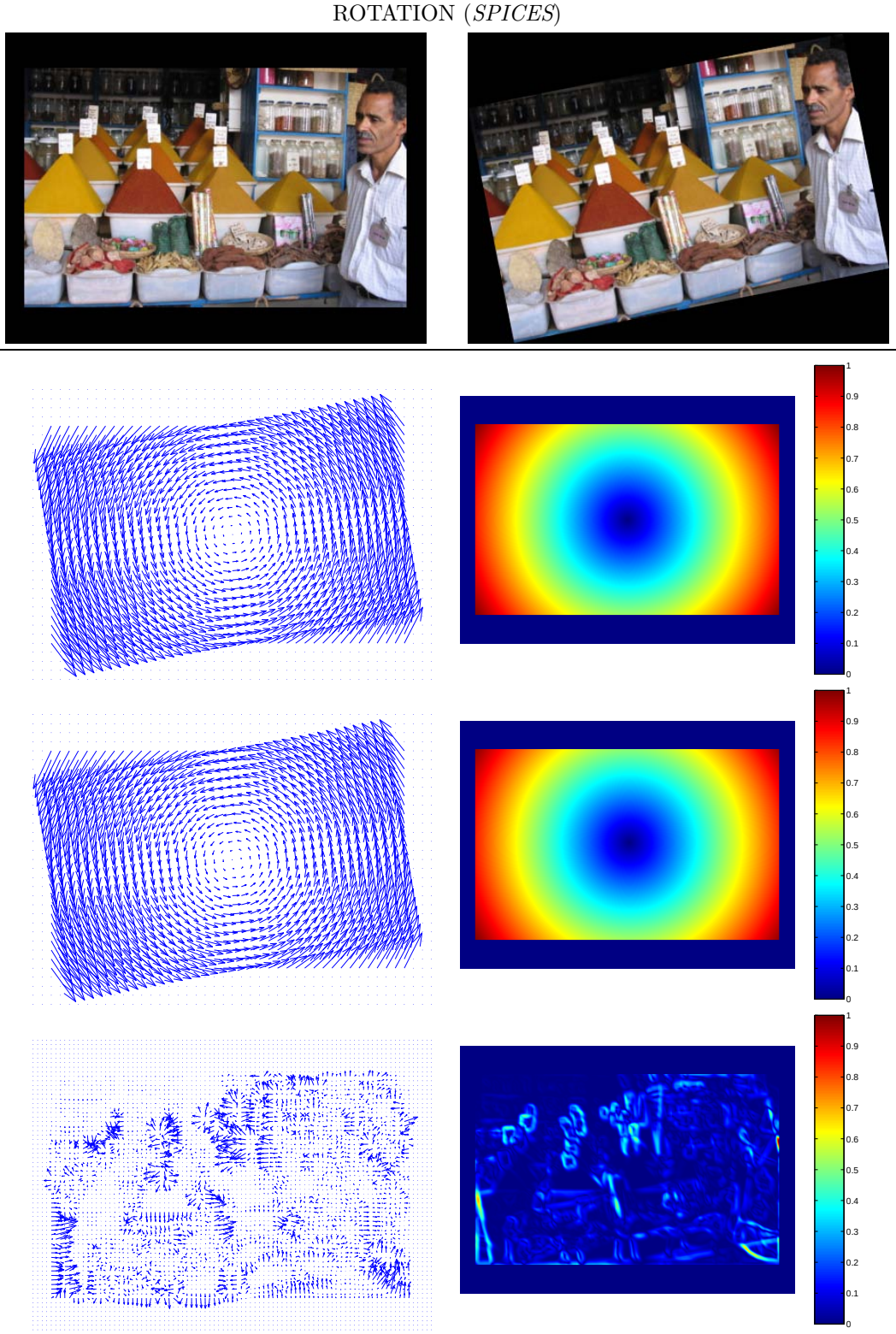


Figure 8-21 Rotation of a highly textured example *SPICES* (12° anticlockwise). First row: source and transformed image. Second row: ground truth motion vector field (left) and colour temperature visualisation of ground truth vector magnitude (right). Third row: Estimated vector field using our method, and visualisation of vector magnitude. Fourth: Estimated vector field using optical flow, and visualisation of vector magnitude.

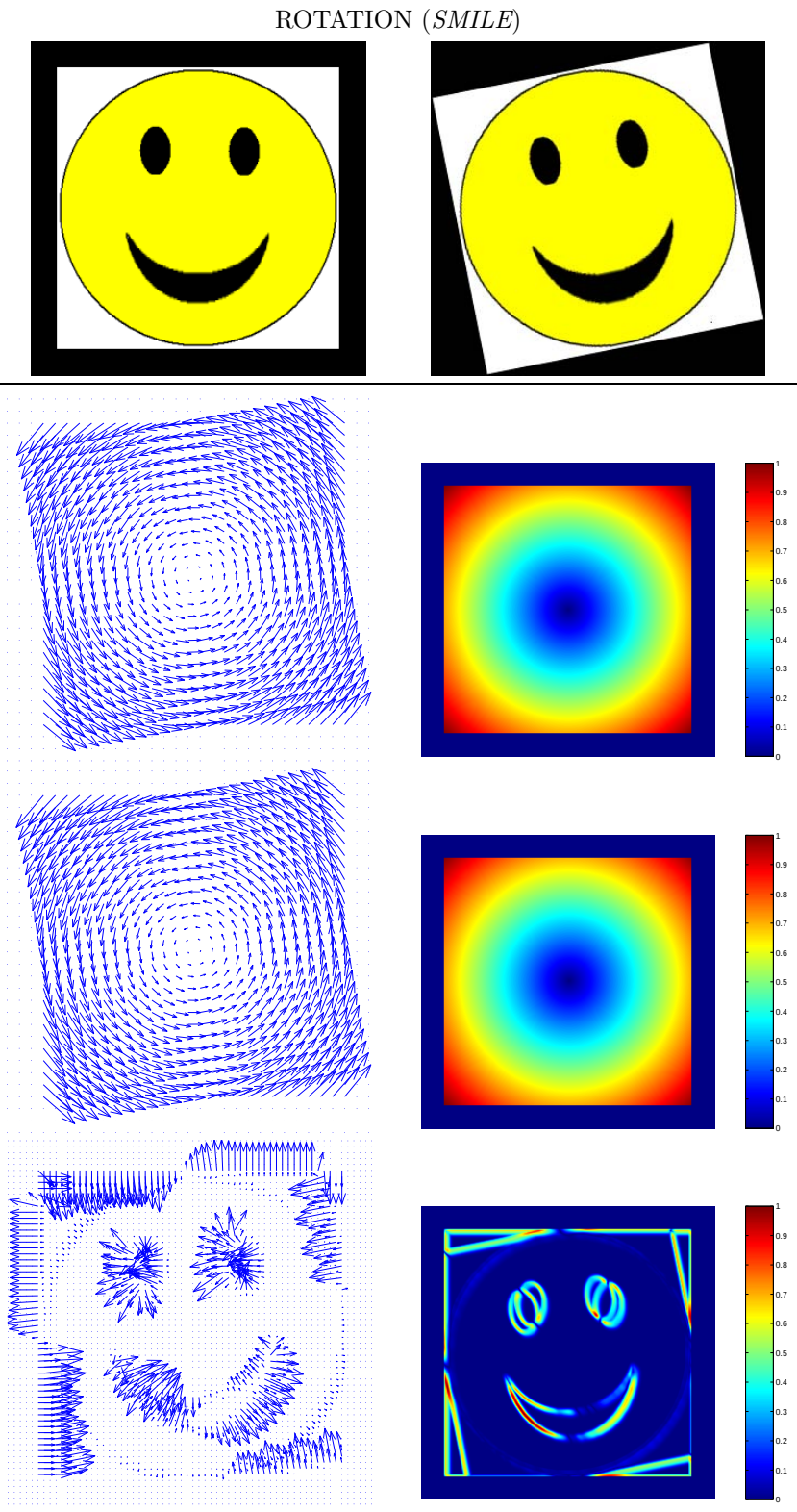


Figure 8-22 Rotation of an example with predominantly flat texture *SMILE* (12° anti-clockwise). First row: source and transformed image. Second row: ground truth motion vector field (left) and colour temperature visualisation of ground truth vector magnitude (right). Third row: Estimated vector field using our method, and visualisation of vector magnitude. Fourth: Estimated vector field using optical flow, and visualisation of vector magnitude.

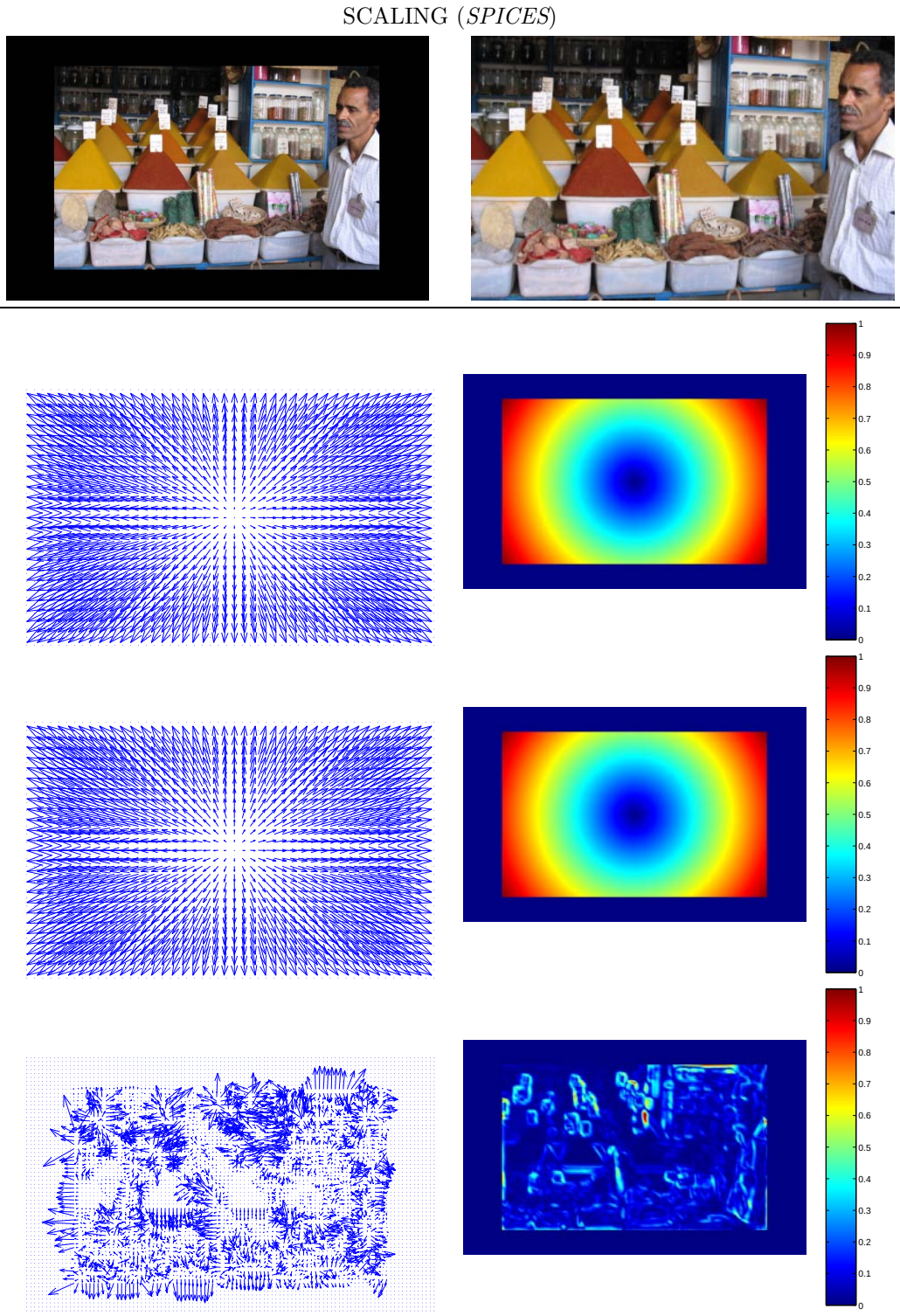


Figure 8-23 Uniform scaling of a highly textured example *SPICES* (scale factor 1.3). First row: source and transformed image. Second row: ground truth motion vector field (left) and colour temperature visualisation of ground truth vector magnitude (right). Third row: Estimated vector field using our method, and visualisation of vector magnitude. Fourth: Estimated vector field using optical flow, and visualisation of vector magnitude.

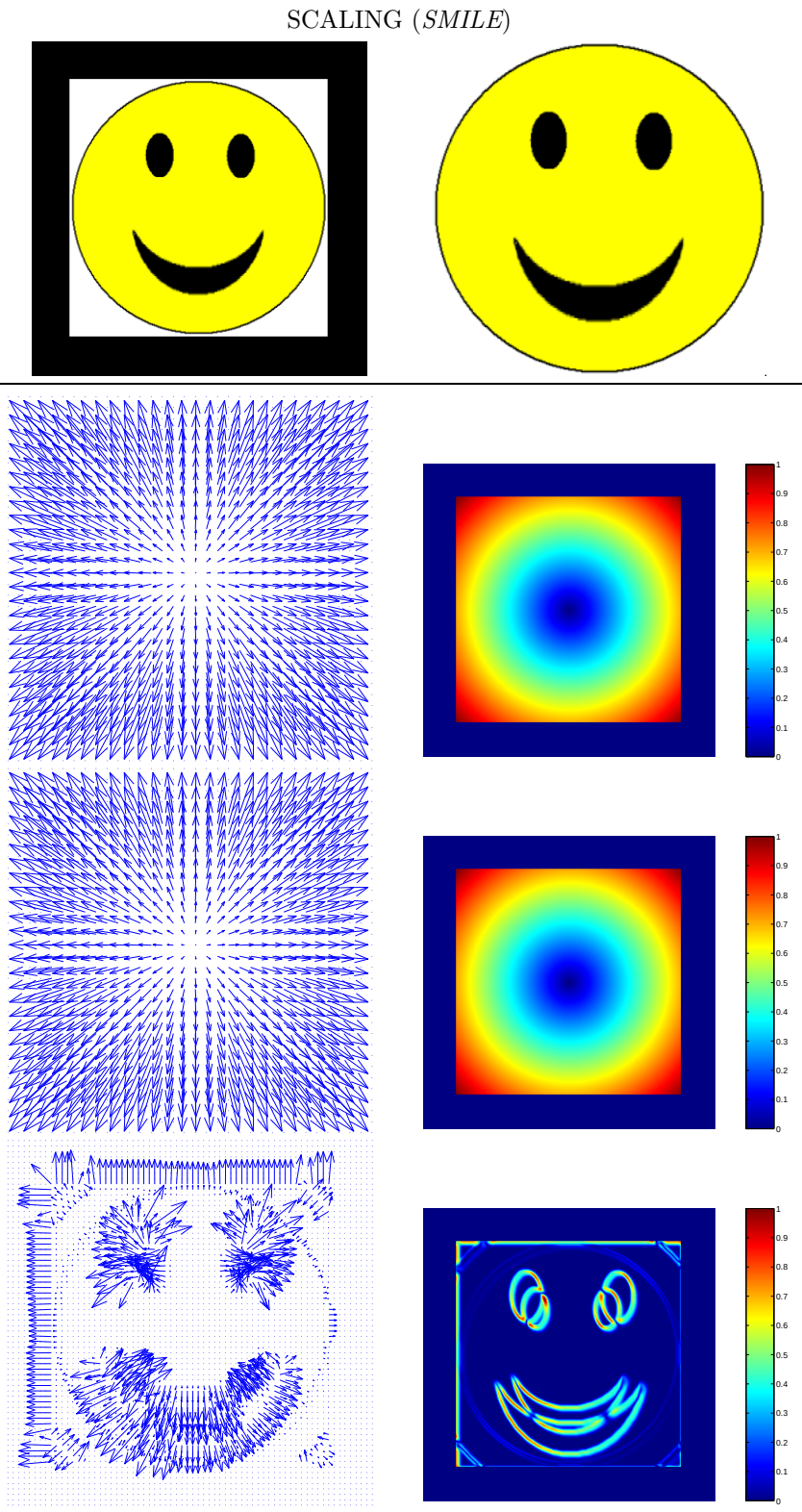


Figure 8-24 Uniform scaling of an example with predominantly flat texture *SMILE* (scale factor 1.3). First row: source and transformed image. Second row: ground truth motion vector field (left) and colour temperature visualisation of ground truth vector magnitude (right). Third row: Estimated vector field using our method, and visualisation of vector magnitude. Fourth: Estimated vector field using optical flow, and visualisation of vector magnitude.

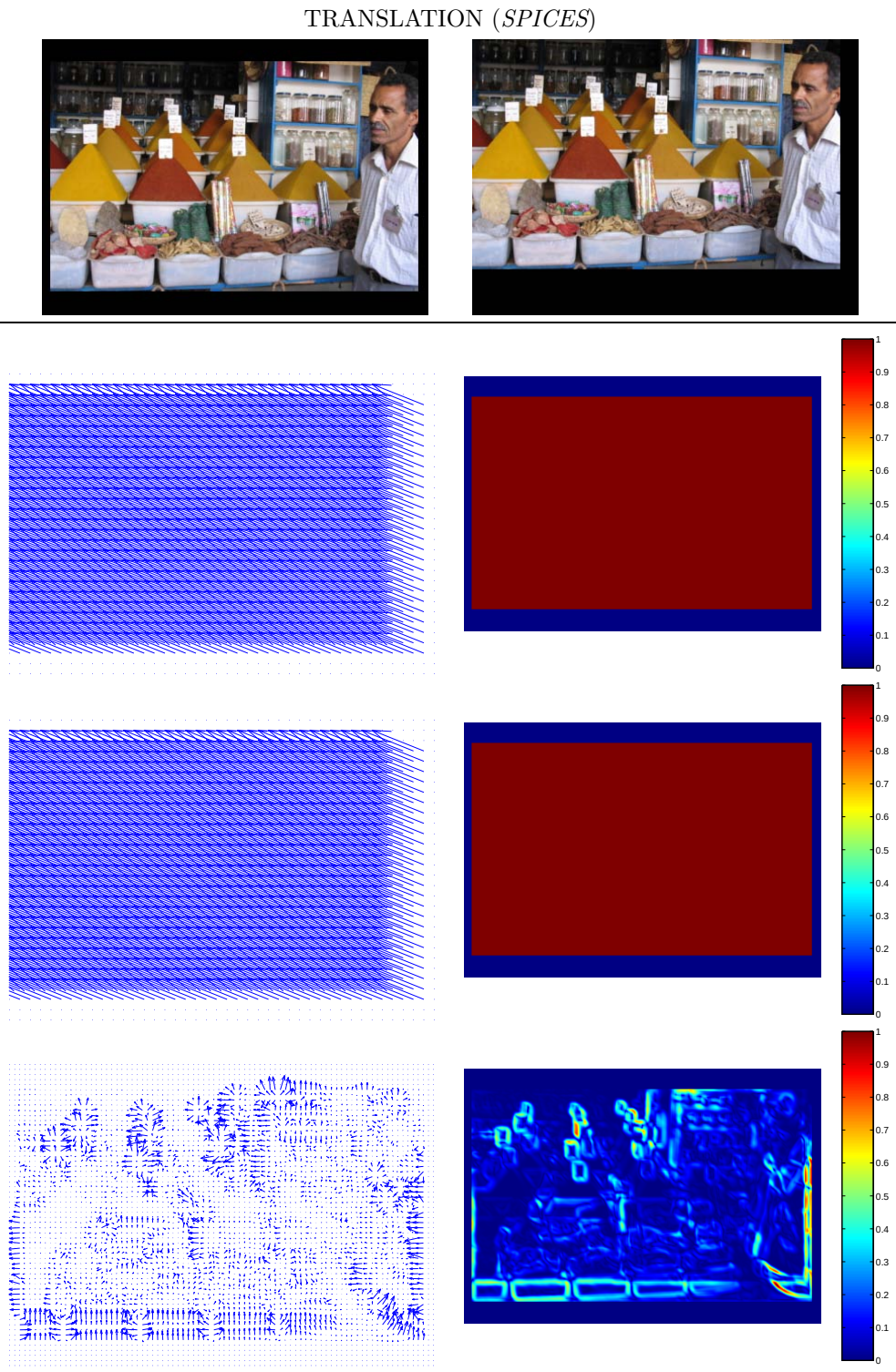


Figure 8-25 Translation of a highly textured example *SPICES* (shift by $(-20, -50)^T$). First row: source and transformed image. Second row: ground truth motion vector field (left) and colour temperature visualisation of ground truth vector magnitude (right). Third row: Estimated vector field using our method, and visualisation of vector magnitude. Fourth: Estimated vector field using optical flow, and visualisation of vector magnitude.

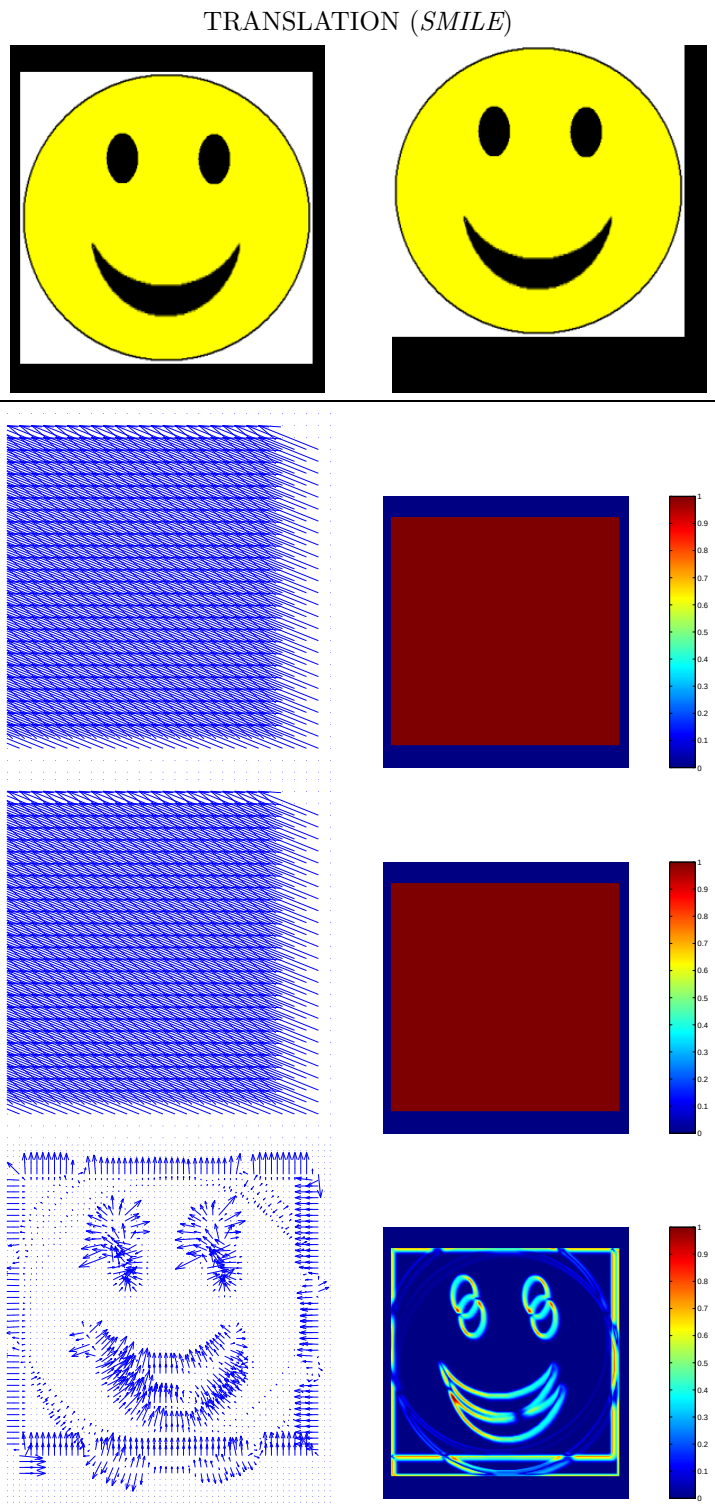


Figure 8-26 Translation of an example with predominantly flat texture *SMILE* (shift by $(-20, -50)^T$). First row: source and transformed image. Second row: ground truth motion vector field (left) and colour temperature visualisation of ground truth vector magnitude (right). Third row: Estimated vector field using our method, and visualisation of vector magnitude. Fourth: Estimated vector field using optical flow, and visualisation of vector magnitude.

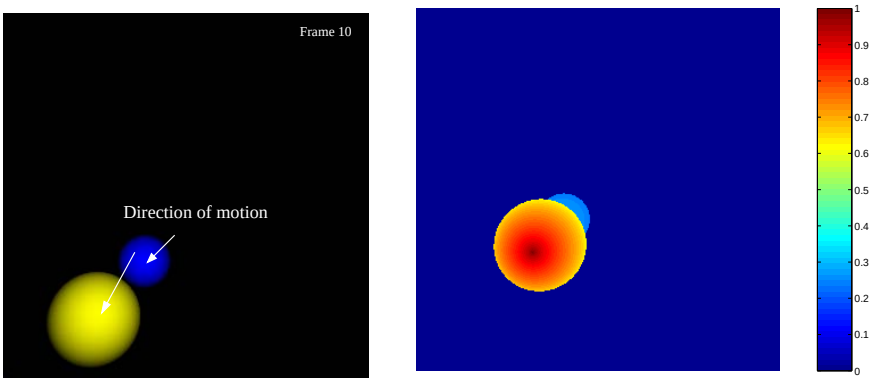


Figure 8-27 A synthetic test sequence (*SPHERES*, `videos/spheres_source`) consisting of two front lit, Lambertian shaded spheres (left). The spheres are of identical size but at differing scene depths (relative proximity to camera is illustrated by the depth map on the right). The spheres translate in planes parallel to the image plane, inter-occluding and casting shadows; a ground truth motion vector field is also projected to the camera plane for reference. We use this sequence to illustrate the limitations of the homography based motion model (see text).

for example by camera noise, is especially important with modern day equipment such as DV cameras whose compression algorithms often create artifacts in stored images.

Limitations of our technique

As with optical flow, our painterly rendering technique is governed by a number of assumptions. These are principally:

1. that the video to be painted is segmentable (some video, for example crowd scenes or water, are difficult to segment)

2. that the change of viewpoint of an object over time is well modelled by a homography (plane to plane transformation) between imaged regions of that object

Violation of assumption (1) will prevent a video being rendered by the artistic rendering subsystem. Violation of assumption (2) is non-fatal, since the video may be still processed into a painterly form, but one that exhibits a lesser level of temporal coherence. However, as we now show in a final experiment, this reduced level of temporal coherence can still represent a significant improvement over the coherence afforded by optical flow based painting techniques.

The synthetic *SPHERES* sequence represents a situation where assumption (2) is violated. The sequence contains two diffuse shaded, spheres (Figure 8-27) which undergo translation in planes parallel to the image plane. We rendered this sequence using both our method and SoA to obtain two painterly animations (see `spheres_painterly_SoA` and `spheres_painterly_ourmethod` in Appendix C). We obtained motion fields for

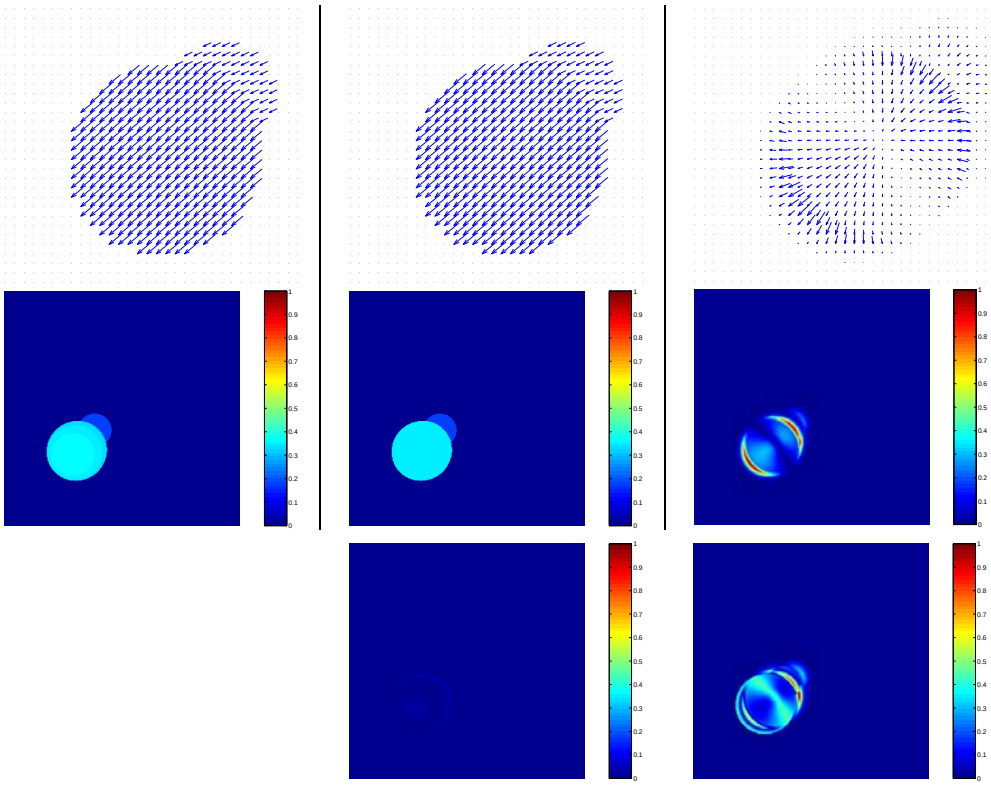


Figure 8-28 Top: Motion vector fields between frames one and two of *SPHERES*. From left to right; ground truth field (true motion of 3D objects projected to the camera plane); field for stroke motion determined via our method; field determined via optical flow [7]. Middle: Visualisation of motion magnitude, ordered as (top) and using identical colour temperature scales. Bottom: Difference between ground truth motion magnitude and that of our (left) and optical flow (right) using identical colour temperature scales for the purposes of comparison. Observe optical flow performs poorly in interior regions, whereas our method does not. Our method exhibits around half an order of magnitude less error than optical flow. We have normalised (left) in Figure 8-29 to illustrate the distribution of estimation error over the region.

brush strokes in the two animations, and modified our ray tracer to output a ground truth motion field of the spheres, projected to the 2D image plane.

The resulting source and rendered motion fields, corresponding to the imaged spheres, are shown in Figure 8-28 (first column). Observe that in the ground truth motion field, distant points exhibit lesser motion magnitudes due to parallax.

Optical flow performs poorly when recovering this motion field (Figure 8-28, second column). The computed fields suggest that the flat textured interiors of the spheres do not move at all, whilst the edges perpendicular to the direction of motion are deemed to have moved. Edges parallel to the direction of movement are not deemed to have moved, since the spatially local nature of the optical flow algorithm can not determine such motion due to the "aperture problem" [145].

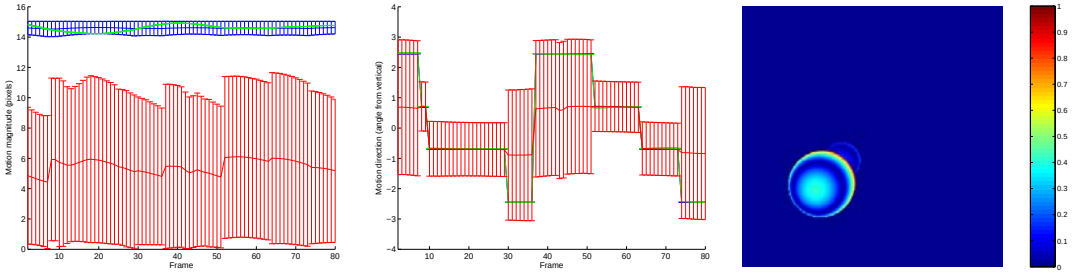


Figure 8-29 Plots illustrating errors in both our, and optical flow's, motion estimate for the yellow sphere. The left graph shows the mean value of motion vector magnitude within the yellow sphere region, at each time instant. The error bars show one standard deviation. Blue is the ground truth, green is our motion estimate, red is the optical flow estimate. Observe that our method produces a consistent, uniform vector field estimate that closely matches the mean of the ground truth. The optical flow motion estimate is inconsistent over the region, as evidenced by high standard deviation. Similar observations apply to the middle plot, which shows mean direction of motion over time. The right-hand figure is a normalised absolute difference map between ground truth motion magnitude, and motion magnitude estimated by our system. Our method produces a single motion estimate for each region, distribution estimation error over that region. In the case of the yellow sphere, the result is a motion estimate corresponding to around middle distance on the sphere; approximately the mean of all motion vectors.

Our method interprets the imaged spheres as translating planar discs, with a radial shading pattern upon their surfaces. The resulting motion field is thus a uniform set of vectors specifying a translation approximately equal to the mean of the ground truth vectors (Figure 8-28, third column). Although this uniform vector field is erroneous, the residual error due to our method is much lower than that of optical flow (see Figure 8-28). The internal consistency of motion vectors generated by our method is also much closer to that of the ground truth, whereas there is very little internal consistency within the sphere according to optical flow (observe the error bars depicting standard deviation in Figure 8-29, and Figure 8-28 top right).

8.7 Integration with the Motion Emphasis Subsystems

Our video shading subsystem meets the aims of the Video Paintbox's second sub-goal, generating temporally coherent artistic animations from video. We may combine this subsystem with the earlier motion emphasis work of Chapters 6 and 7, to meet our original aim of producing full, cartoon-like animations from video. Figure 8-30 contains a schematic of the entire Video Paintbox, illustrating how we combine the shading and motion emphasis work. We now describe the complete Video Paintbox rendering process:

A camera motion compensated version of the video clip is first generated (Section 6.3.1).

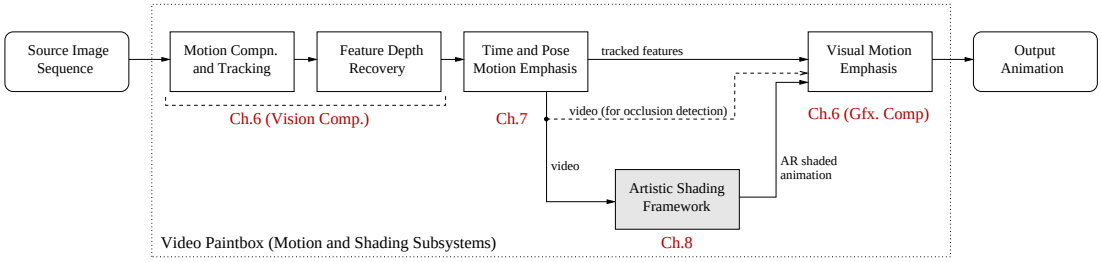


Figure 8-30 Schematic of the complete Video Paintbox, showing where the artistic rendering subsystem, described in this Chapter, fits into the rendering framework.

The user identifies features for the application of motion cues, and these features are tracked through the camera compensated sequence. Their locations and relative depths are recorded (Section 6.3.3). This processing is performing by the Computer Vision component of Chapter 6.

Time and pose cues (Chapter 7) are then applied, synthesising a novel video (for example, exhibiting anticipation) and altering the recorded locations of any tracked features. At this stage we discard the source video clip, and use this "time and pose" emphasised video as input to subsequent stages.

The "time and pose" emphasised video is passed to the artistic shading subsystem, which synthesises an AR version of the video in the requested artistic style. We now have three pieces of information: the tracked feature data, a photorealistic version of the "time and pose" emphasised video, and an artistically rendered version of the "time and pose" emphasised video.

As a final step we pass both the tracked feature data, and the AR version of the video, to the Computer Graphics component of the visual motion cue subsystem (Section 6.4). In our original description of that subsystem we stated that the Computer Graphics component accepts as input:

1. tracker data output by the Computer Vision component

2. the original, photorealistic video

We have altered the rendering pipeline, so that the Computer Graphics component accepts an artistically rendered version of the video sequence in place of (2) — see Figure 8-30. The final output is an artistically shaded animation which also exhibits motion cues. Stylisation of both the shading and motion cues is controlled by the animator at a high level (by requesting particular rendering styles and motion effects, and by setting parameters upon those special effects).

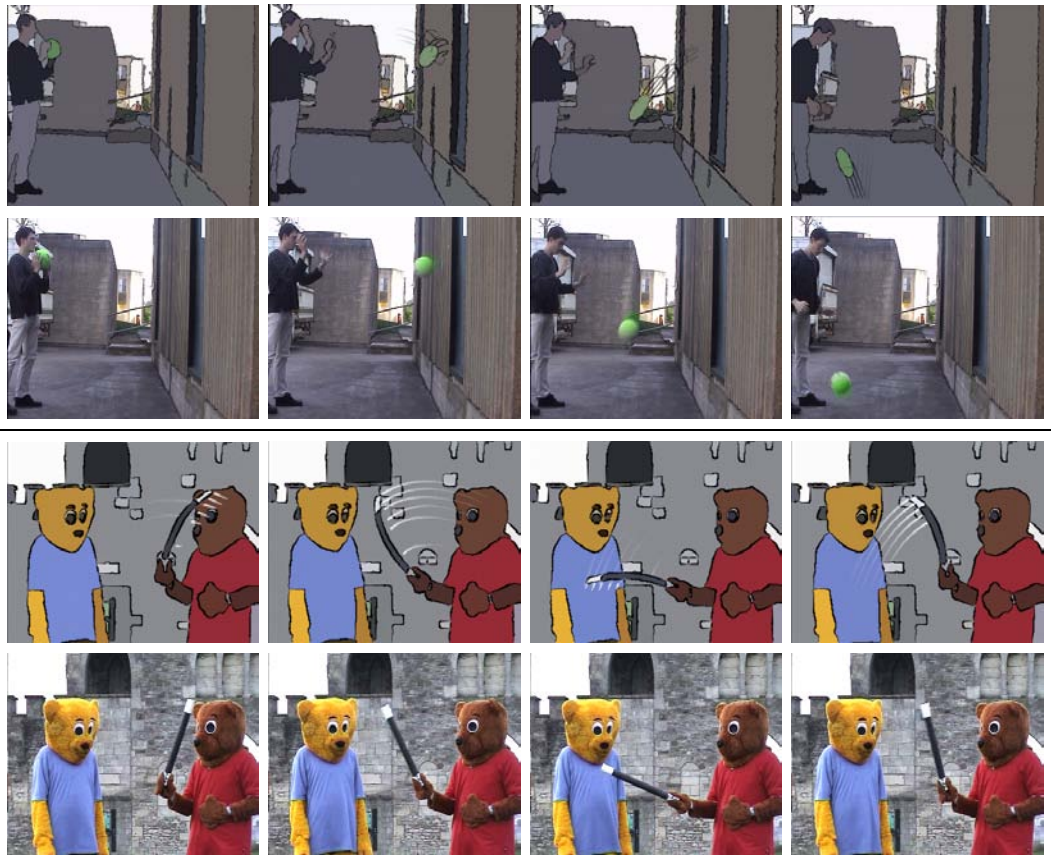


Figure 8-31 The artistic rendering subsystem is combined with the motion emphasis work of the previous two chapters, to produce complete cartoon animations from video (see `bounce_fullcartoon` and `wand_cartoon`).

As a practical note, we have observed the occlusion buffer system (Section 6.4.3) to operate with markedly less precision when using the non-photorealistic video in the manner described. We therefore make a small modification to the rendering pipeline, as described, which allows the occlusion buffer system access to the photorealistic version of the video sequence to perform its inter-frame differencing operations. This access is represented by the dashed arrow of Figure 8-30.

8.8 Benefits of an Abstract Representation of Video Content

Recall the basic architecture of the artistic rendering subsystem — the Computer Vision driven front end parses source video into an intermediate representation (IR), which the back end subsequently renders into one of many artistic styles. The IR therefore encodes an abstracted video representation, encapsulating semantic video content but not instantiating that content in any particular artistic style.

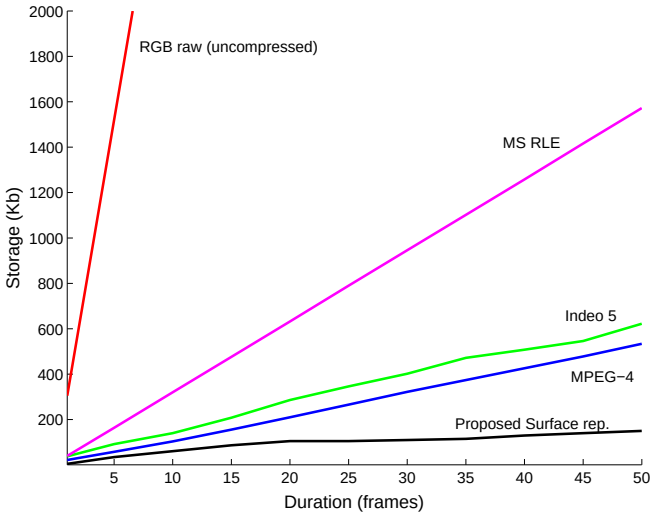


Figure 8-32 Demonstrating the comparatively low storage requirements of the surface representation when transmitting cartoons. Our test comparison uses up to 50 frames of the gradient shaded *POOHBEAR* sequence.

We draw an analogy with our IR and the popular XML data format. XML also divorces content from presentation style, deferring the problem of visualising information to the client who must process the XML content against an XSLT style-sheet to produce a legible document. This separation of information (XML) and stylisation (XSLT) creates a compact, and more manipulable format for information storage and exchange. Likewise, the IR is a highly compact representation of video content. Video objects are represented by a set of continuous spatiotemporal vector surfaces, and a small supplementary database. These properties present a number of interesting directions for future development of our IR and of the Video Paintbox system.

Hand-held mobile devices are no longer fundamentally limited by speed of CPU, or by storage constraints. The primary challenge for the next generation of wireless (3G) devices, at the time of writing, is to achieve a suitably high bandwidth wireless connection over which video content may be delivered. We suggest that, with further development, our IR could provide interesting new perspectives on both the issue of bandwidth, and upon the nature of the content delivered over the network. Figure 8-32 summarises details of a brief comparative investigation, contrasting the storage requirements of the IR with those of common video compression technologies, to store a cartoon. Approximately 150KB were required to store 50 frames of the *POOHBEAR* sequence. The compact nature of the IR can be seen to compare favourably with the other video compression algorithms tested; although we note that the spatiotemporal nature of our representation prohibits real-time encoding of video. The caveat is that the video must be abstracted and stylised in a cel animated fashion. However, the IR may be rendered into a wide gamut of artistic styles once downloaded, creating a novel level of

abstraction for video in which the server (implementing the front end) determines the video content, whilst the client-side (implementing the back end) determines the style in which that video is rendered. Splitting the responsibilities of video content provision and content visualisation between the client and server is a promising direction for development of our Video Paintbox architecture.

Aside from the benefits of compact representation and abstraction, also of interest is the continuous spatiotemporal nature of the Stroke Surfaces in the IR. This provides a highly manipulable vector representation of video, akin to 2D vector graphics, which enables us to synthesise animations at any scale without pixelisation. Indeed many of the figures in this Chapter were rendered at a scale factor greater than unity to produce higher resolution images than could be captured from a standard PAL video frame. Future developments might investigate the use of temporal scaling to affect the frame rate of animations.

8.9 Summary and Discussion

In this Chapter we have described a novel framework for synthesising temporally coherent non-photorealistic animations from video sequences. This framework comprises the third and final subsystem of the "Video Paintbox", and may be combined with the previously described motion emphasis work to produce complete cartoon-style animations from video.

Our rendering framework is unique among automated AR video methods in that we process video as a spatiotemporal voxel volume. Existing automated AR methods transform brush strokes independently between frames using a highly localised (per pixel, per frame) motion estimate. By contrast, in our system the decisions governing the rendering of a frame of animation are driven using information within a temporal window spanning instants before and after that frame. This higher level of temporal analysis allows us to smoothly vary attributes such as region or stroke colour over time, and allows us to create improved motion estimates of objects in the video. Spatially, we also operate at a higher level by manipulating video as distinct regions tracked over time, rather than individual pixels. This allows us to produce robust motion estimates for objects, and facilitates the synthesis of both region based (e.g. flat-shaded cartoon) and stroke based (e.g. traditional painterly) AR styles. For the latter, brush stroke motion is guaranteed to be consistent over entire regions — contradictory visual cues do not arise, for example where stroke motion differs within a given object. We have shown that our high level spatiotemporal approach results in improved aesthetics and temporal coherence in resulting animations, compared to the current state of the art.

Much of the discussion of the relative merits of our approach over optical flow can be found in Section 8.6.

We have demonstrated that automated rotoscoping, matting, and the extension of many "traditional" static AR styles to video, may be unified in a framework. Although we have experimented only with the extension of our own pointillist-style painterly method (Chapter 3) to video, we believe this framework to be sufficiently general to form the basis of a useful tool for the extension of further static stroke based AR techniques to video. The application of our framework to other static AR styles is perhaps the most easily exploitable direction for future work, though does not address the limitations of our technique, which we now discuss.

Perhaps the most limiting assumption in our system is that video must be segmented into homogeneous regions in order to be parsed into the IR (and so subsequently rendered). As discussed in Section 8.6, certain classes of video (for example crowd scenes, or running water) do not readily lend themselves to segmentation, and so cause our method difficulty. Typically such scenes are under-segmented as large feature subvolumes, causing an unappealing loss of detail in the animation. This is not surprising; the segmentation of such scenes would be a difficult task even for a human observer. Thus although we are able to produce large improvements in the temporal coherence of many animations, our method is less generally applicable than optical based flow methods, which are able to operate on all classes of video — albeit with a lower degree of temporal coherence. The problem of compromising between a high level model for accuracy, and a lower level model for generality, is an issue that has repeatedly surfaced in this thesis, and we defer discussion of this matter to our conclusions in Part IV. However we summarise that as a consequence we view our method as an alternative, rather than a replacement, for optical flow based AR.

The second significant limitation of our system stems from the use of homographies to estimate inter-frame motion from an object's internal texture. We assume regions to be rigid bodies undergoing motion that is well modelled by a plane to plane transformation; in effect we assume objects in the video sequence may be approximated as planar surfaces. There are some situations where lack of internal texture can cause ambiguities to creep in to this model; for example if an object moves in-front of an untextured background, is that background static and being occluded, or is that background deforming around the foreground object? Currently we assume rigid bodies and so search for the best homography to account for the shape change of the background. The worst case outcome of poor motion modelling is a decrease in the temporal coherence of any markings or brush strokes within the interiors of objects. Other artistic styles (such as

sketchy outlines or cartoon-style rendering) do not use the homography data in the IR, and so are unaffected. As a work-around we allow the user to set the motion models of video objects to be "stationary" if they deform in an undesirable manner. This single "point and click" corrective interaction is necessary to introduce additional knowledge into an under-constrained system, and is in line with the high level of creative interactive we desire with the animator. Future work might examine whether the planar surface assumption could be replaced by an improved model; perhaps a triangulated mesh, or replacement of the linear bases which form the plane with curvilinear bases (adapting the recent "kernel PCA" technique of [137]). However, many of the video sequences we have presented contain distinctly non-planar surfaces which nevertheless create aesthetically acceptable animations, exhibiting superior levels of temporal coherence than the current state of the art. We therefore question whether the additional effort in fitting more complex models would pay off in terms of rendering quality.

We did not set out to produce a fully automated system — not only do we desire interaction with the Video Paintbox for creative reasons (setting high level parameters, etc.) but also, rarely, for the correction of the Computer Vision algorithms in the front end. The general segmentation problem precludes the possibility of segmenting any given video into semantically meaningfully parts. However we have kept the burden of correction low (Section 8.5). Users need only click on video objects once, for example to merge two over-segmented feature sub-volumes in the video, and those changes are propagated throughout the spatiotemporal video volume automatically. In practical terms, user correction is often unnecessary, but when needed takes no more than a couple of minutes of user time. This is in contrast to the hundreds of man hours required to correct the optical flow motion fields of contemporary video driven AR techniques [61].

A selection of source and rendered video clips have been included in Appendix C.

Part IV

Conclusions

Chapter 9

Conclusions and Further Work

In this chapter we summarise the contributions of the thesis, and discuss how the results of the algorithms we have developed support our central argument for higher level spatiotemporal analysis in image-space AR. We suggest possible avenues for the future development of our work.

9.1 Summary of Contributions

This thesis addressed the problem of image-space AR; proposing novel algorithms for the artistic rendering of real images and post-production video.

We performed a comprehensive review of image-space AR, and argued that the low-level spatiotemporal operations, used to drive the rendering process in existing automatic algorithms, are a limiting factor in their development of image-space AR. In Section 1.2, we identified three key areas where this was the case:

1. Aesthetic quality of rendering, specifically:
    1a. Control of level of detail (emphasis)
    1b. Temporal coherence of animations
2. Diversity of style

We have shown that application of Computer Vision methods, to perform higher level spatiotemporal analysis of 2D source content, proves beneficial in terms of relaxing the limitations identified in these areas. To support this argument we developed several novel AR algorithms which operate at a higher spatial (Part II: Chapters 3 and 4) and temporal (Part III: Chapters 5, 6, 7 and 8) level to render images and video sequences. In order of appearance these are:

1. A single-pass, salience adaptive painting algorithm capable of rendering pointillist-style paintings from photographs (later extended to video).

2. An algorithm for creating Cubist style compositions from sets of images, or from video, with minimal interaction (and in certain cases, full automation)

3. A salience adaptive, genetic algorithm based relaxation scheme for creating impasto style oil paintings from photographs (building upon our pilot algorithm of (1.)).

4. A framework for inserting a range of visual motion cues into video to emphasise motion, for example streak-lines, ghosting and deformations.

5. A framework for inserting "time and pose" cues into video, to create animation timing effects for motion emphasis, for example anticipation and motion exaggeration.

6. A framework for creating a range of temporally coherent shading effects in video. These included sketchy, painterly and cartoon shaded rendering styles, as well as novel temporal effects unique to our approach. We unified rotoscoping, matting and stroke based rendering styles in a single framework, and extended our earlier painterly rendering technique (1.) to video.

7. A means to integrate frameworks (4–6) to create a single "Video Paintbox", capable of creating artistic animations directly from video with a high degree of automation.

9.2 Conclusions

We now examine the algorithms that we have developed, and conclude as to how their results contributed to our central argument for higher level analysis in AR. Specifically we identify the improvements gained in each of the three areas (ability to control level of emphasis in artistic renderings, temporal coherence of AR animations, and diversity of AR style).

9.2.1 Control over Level of Detail in Renderings (aesthetic quality)

In Chapter 3 we described a novel pointillist-style painterly rendering algorithm which automatically controlled visual emphasis by varying the level of detail in regions of the painting, according to the salience or "importance" of those regions in the original image. The resulting paintings exhibited a conservation of salient detail, and the abstraction of non-salient detail — so mimicking the work of human artists. We also demonstrated how automatic tonal variation could be created to draw attention toward salient regions of a rendering, for example the eyes in a portrait (see Figure 9-2, left).

Figure 9-1 Recalling a result from Chapter 4 (full painting in Figure 4-14) illustrating the improved aesthetics of our global salience adaptive painterly technique (Chapter 4), over locally driven approaches. Left: Section of the original photograph exhibiting non-salient background texture (shrubbery) and salient foreground (sign-post). Middle: All fine detail is emphasised using existing automatic approaches [103], which place strokes using only spatially local, low-level information (in this case Sobel edges). In this image, the high frequency detail of the background leaf texture has caused strokes to be clipped at edges, tending the process back toward photorealism. However attempts to mitigate this effect, by reducing the edge threshold for clipping, will further degrade salient detail on the sign. Right: Using our adaptive approach, salient detail is conserved, and non-salient detail (as determined by our global rarity based measure) is abstracted away, yielding improved aesthetics.

In Chapter 4 we presented an algorithm for creating impasto style painterly renderings from photographs, which built upon our previous single-pass, salience adaptive pointillist technique. The new algorithm placed strokes using an iterative relaxation strategy — explicitly searching for an "optimal" painting in which there is maximum correlation between the salience map of the original image and the level of detail present in the rendered painting. The resulting paintings exhibited improved aesthetics over those of Chapter 3 — curved spline brush strokes were tightly fitted to salient contours in the painting, and level of stroke detail closely correlated with the presence of salient detail in the original image.

A salience adaptive approach to image-space painting is unique to our work, and demands a global (rather than local, low-level) image analysis to determine relative salience over the source image. By contrast, existing techniques operate by processing small regions of pixels into brush strokes; each pixel region is treated independently, and using only locally determined measures. The renderings of current automatic algorithms do not exhibit variation in emphasis due to relative importance in the scene, as with our approach. Rather, all fine detail is treated equally regardless of its salience (see Figure 9-1). Interestingly, some recent publications [37, 149] have begun to consider the problem of static object-space AR from a similar viewpoint to our work; determining which silhouette edges or cusps of a 3D model to draw and which to omit

Figure 9-2 Temporal coherence and diversity of AR style. Left: A portrait generated automatically by the Cubist rendering algorithm described in Chapter 3, which uses high level, salient features to create abstract compositions. Middle: Still from an animation produced by the framework of Chapter 6, capable of automatically synthesising motion cues. Right: Still from a temporally coherent, sketchy cartoon generated directly from video shot with a hand-held DV camera.

when rendering line drawings. Although such techniques may be viewed as similar in spirit to our argument for salience driven artwork, the object-space and image-space branches of AR continue to address very different problem domains, demanding distinct approaches to their solution.

9.2.2 Temporal Coherence of Animations (aesthetic quality)

In Chapter 8 we introduced a novel spatiotemporal framework for rendering video into a range of artistic animation styles. Existing video driven AR techniques operate by painting brush strokes on the first frame of the video, then translating those brush strokes from frame to frame using an estimate of optical flow. The higher level spatiotemporal approach of our framework allowed us to coherently segment video, smoothly vary visual attributes such as region or stroke colour over time, and create improved motion estimates of objects in the video. The animations generated by our framework exhibit both quantitative and qualitative improvements in temporal coherence over the current state of the art in video painting (for example, exhibiting around half an order of magnitude less stroke flicker). These improvements are a direct consequence of our higher level spatiotemporal approach to automatic video driven AR.

9.2.3 Diversity of AR Style

Many artistic styles are not characterised by spatially low-level characteristics such stroke placement, but by the composition and arrangement of features in the scene — for example eyes, or mouths in a portrait. In Chapter 3, we described a novel AR algorithm which produced compositions in a Cubist style, using high level, salient features identified across an image set. Control of the AR process was specified at the compositional, rather than the stroke based level. We were able to create renderings in

styles inaccessible through the low-level stroke based rendering paradigm of automatic image-space AR.

Existing video driven AR techniques process video on a per frame sequential basis. Although we observed that this behaviour imposes limitations on temporal coherence, there are also a large number of artistic effects whose synthesis demands a higher level temporal analysis of video. In Chapter 6 we described how animators use visual motion cues to emphasise the historic motion of objects; capturing and depicting the essence of that motion using tools such as streak-lines and deformations (much as a sketch captures the salient elements of a scene). Performing a higher level temporal analysis of the video is a prerequisite to characterising this motion, and also allows us to deal with problematic Computer Vision issues such as occlusion and depth ordering in the scene. In Chapter 7, we presented a framework for synthesising "time and pose" cues; a further class of motion cue which affects the timing of an animation. Specifically, we describe how to generate cartoon "anticipation" effects and motion cartooning (exaggeration). The former effect appears to "predict", or anticipate motion prior to its execution in the animation. Clearly processing video at a temporally low-level (per frame sequential basis) precludes the analysis of future events in the video. Similarly, the identification of the patterns and characteristics of a subject's motion over time are a prerequisite to exaggerating those motions within the video sequence.

The character of an animation is influenced not only by the artistic style in which it is drawn, but also by the manner in which the animated objects appear to move. However, until now this latter consideration has been overlooked by video driven AR research. Through the successful insertion of motion cues into video we have demonstrated a diversification of style in video driven AR that could not have been achieved within the constraints of the temporally low-level, per frame processing paradigm of automated video driven AR.

9.3 Discussion

Artistic renderings such as drawings or paintings (be they static or animated) are stylised representations of the salient spatiotemporal information present in a scene. The process of deriving artwork from a photograph or scene therefore demands the visual interpretation, or "perception" of that original content. We have argued that such a goal is challenging enough to warrant application of higher level image analysis techniques, implying interesting new application areas for Computer Vision (and motivating new Computer Vision research as a result). This thesis has demonstrated improvements in several key areas of image-space AR, achieved by breaking away from

the current low-level spatiotemporal filtering paradigm of automatic image-space AR techniques.

We can draw parallels between the early bottom-up development of Computer Vision, and the current low-level processing approach of image-space AR — a field very much in its youth. The majority of Computer Vision research, during the eighties and early nineties, followed the paradigm of "bottom-up" processing originally proposed by Marr [109]. For example, to determine the position of a car licence plate one might locate low-level features such as edges or corners, then examine the spatial relationships between such artifacts to form candidate rectangles for the licence plate, and so on. This bottom-up processing paradigm produced a number of a successful applications, but was found to be limiting by the Vision community which has largely shifted its focus towards model fitting or "top-down" approaches. Top-down approaches tend to be more robust and more generally applicable, often because they do not rely upon the accuracy of a battery of earlier feature extraction steps. Returning to our licence plate recognition example, in a top-down approach a model of a typical licence plate would be encoded as *a priori* information, and a search of a configuration space performed to find the optimal fit for that model within the source image.

Similarly, most automated examples of image-space AR approach the problem of synthesising artwork using simplistic image-processing operators such as Sobel edge detection. Some more recent methods attempt to piece together local neighbouring fragments of edge, to form curved spline brush strokes [71]; a practice similar to the bottom-up processing strategy of Marr. However existing image-space AR techniques remain characterised by their low-level, highly localised approach to spatiotemporal signal processing; local pixel regions are independently transformed into collections of strokes without regard to the remainder of the image, and frames of animation rendered without regard to the wider temporal extent of the clip.

The objectives of both image-space AR and Computer Vision are, to some extent, synonymous; both desire extraction of information from visual content. Our argument for a higher level of analysis in image-space AR is similar to the approach of top-down Computer Vision, in that we have begun to impose externally defined models to govern the AR process. In most cases these models have been "mid level" models designed for generality. For example, our rarity based salience measure in Chapter 3, the objective function used to "fit" paintings to photographs in Chapter 4, the LCAT motion model for tracking features and the heuristics driving motion emphasis in both Chapters 6 and 7, and the spatiotemporal segmentation process and Stroke Surfaces in Chapter 8. The choice of these models reflects our original motivation; to produce a series of tools

for artists animators which are both general and highly automatic.

The per frame, or per pixel neighbourhood, models employed by automatic image-space AR are arguably among the lowest level form of representation for digital image-space content — yet such representational forms do hold certain advantages. All image-space content conforms to this low-level model, thus the associated AR methods exhibit high generality and are also often simple to implement. However disadvantages arise from the same properties; the lack of a structured model in low-level spatiotemporal processing admits unfavourable situations where, for example, temporal incoherence may arise. This is the case in optical flow based video renderers, where each stroke is permitted to translate independently of its immediate neighbours (when we would intuitively expect behaviour to conform to rules, for example, imposing local spatial coherence between strokes). A higher level spatiotemporal model constrains the range of possible solutions to fall within a more acceptable range, but potentially at the cost of generality. For example, consider our video shading solution (Chapter 8), whose painterly animations exhibit significant improvements in temporal coherence over the state of the art. These improvements are primarily due to the grouping of pixels into segmented regions, and the attachment of rigid planar reference frames to those regions which move coherently over time. In this case our solution to determining a suitable "level of abstraction" was to choose a mid-level description of video which was general enough to represent most classes of video content, but simultaneously was sufficiently constrained to permit generation of temporally coherent animations. Consequently, our model does not admit some classes of source footage (that which can not be easily segmented, for example water or crowd scenes), but the footage that can be processed exhibits improved temporal coherence relative to optical flow driven techniques.

A concern throughout this thesis has been to balance the desire for a high degree of automation in our algorithms, with that of working at a higher level of abstraction. In many cases it has been the level of technology in contemporary Computer Vision that has forced interaction. This point is best illustrated by the Cubist rendering algorithm, which creates compositions from salient features identified over several images. We were able to automate the task of isolating these salient features for the case of faces, which are well-studied in Computer Vision. However, producing a general solution to the automatic segmentation problem remains a significant challenge to contemporary Computer Vision and we were forced to defer to an interactive scheme in the general case. Similarly, in Chapter 8 we allowed for interactive correction where video objects became over-segmented in our video shading framework. In both cases the automated system provided an objective basis for segmentation, so as to minimise user load during interaction to a few mouse clicks. As new Computer Vision technologies develop,

we may be able to reduce the level of interactive correction required in automatic AR systems. The fields of image-space AR and Computer Vision appear destined to become highly interdependent in the future. However, it is likely that some level of human creative direction will always be desirable.

9.4 Further Work

The techniques we have proposed in this thesis have raised a number of interesting possibilities for future work. Many of these address specific incremental improvements which could be made to particular algorithms, and were best discussed in the conclusions sections of the relevant chapters. Here we highlight the directions for development which appear to hold the greatest potential.

We have described two novel techniques for creating painterly renderings from photographs. These approaches are unique in that the focus, or emphasis, in the resulting paintings is automatically controlled using a global measure of image salience. It is clear that the notion of salience is highly subjective and task specific. In choosing our rarity based models of salience (Section 3.2, Section 4.3) we are approaching the problem from a somewhat simplistic point of view — though one that has been shown to give surprising good results, for example abstracting away repetitive fine background texture in favour of rarer foreground edge structures (Figure 3-3). The development of alternative salience measures is an area for future work, and one which continues to be investigated both at Bath and elsewhere.

Our GA relaxation based painting algorithm currently searches for paintings which minimise an objective function, measuring the discrepancy between level of detail in the painting and the salience map of the original photograph. This objective function forms a salience dependent model of painting, but there is no reason for us to be restricted one such model. In Chapter 4 we suggested a further investigation in which various styles of painting might be studied, and a library of style specific models built. This "paint by model fitting" approach could potentially yield a single versatile system, capable of generating many artistic styles. Other model driven approaches to AR could be explored. For example, in Chapter 3 we suggested borrowing global search techniques, such as the Hough transform, from Computer Vision to identify target shapes within an image taken from a user defined "shape library". The subsequent rendering of those shapes might provide a basis for synthesising alternative abstract artistic styles to complement our Cubist rendering algorithm.

The video driven AR algorithms presented in this thesis encompass a range of artistic

styles, the majority of which are novel to image-space AR. Further exploitation of the coherent reference frames attached to video objects in Chapter 8 should facilitate the extension of many other existing static stroke based AR styles to video (for example [71, 140, 159]). Perhaps the most significant criticism of this aspect of the work is that the homography, used to attach reference frames, assumes both planar motion and rigid objects. However many artistic styles (such as cartoon shading, and sketchy rendering) do not use this aspect of the system. Ideally our video shading subsystem would be extended to increase generality of all artistic styles: three-dimensional descriptions of objects, or curvilinear (rather than linear) bases [137] are two possible directions for development.

Although we have consulted with artists and animators during the course of this project, we have yet to explore the many interaction issues that are likely to arise from our proposed frameworks. For example, tools are known to influence the artists that use them. As the tools that we have described are developed and applied in the field, it will be interesting to observe the effect that they have on the creative process, and what restrictions they may impose. Equally, it will be interesting to see if AR tools such as the Video Paintbox will encourage experimentation in new forms of dynamic art.

Throughout our work we have strived to strike a balance between generality and our desire for higher level content analysis, leading to the imposition of mid-level models on the AR process. However by disposing of this desire for generality, it may be possible to impose yet higher level models to driven the AR process. An example might be a high level 3D facial model which could be registered to images or video using Computer Vision techniques [104, 183], as a means to a producing a specific AR system for portrait rendering. As regards animation, we might be able to impose behavioural models on motion in videos, to extract semantic information about the actions being performed; perhaps drawing upon the automated video summarisation systems currently being developed, for example to analyse sports footage [87]. This information could be used to augment the results of our existing motion cue framework, for example adding cartoon captions such as “Crash!”, “Zap!”, or maybe even sound effects, depending on the semantic context of the action being performed rather than solely its trajectory.

Neither our work, nor existing AR research, has considered the influence of non-visual stimuli upon the artistic process. One example of such a stimulus in human artwork is emotional state. Modelling emotional state is an active research field within Artificial Intelligence, and number of continuous “emotion spaces” have been developed (such as the 2D space of [132], and 3D space of [110]). Points within these spaces correspond to specific emotions such as happiness or despair. An interesting direction for future work

might consider the creation of associations between regions in these spaces and various painting styles. An application might be an AR driven "interactive digital painting", the style of which could be manipulated by stating one's mood.

9.5 Closing Remarks

The algorithms and solutions that we have developed form the first steps toward a new paradigm for automated image-space AR, in which images and video clips are parsed, analysed and then rendered to produce artwork — rather than being subjected to local non-linear filtering operations. Consequently, there are many avenues open for future work. These include not only the development of new algorithms for content analysis and artistic rendering, but also for cross-fertilisation of ideas between Computer Vision and AR communities — a convergence area which is only just starting to be explored.

Part V

Appendices

Appendix A

Miscellaneous Algorithms

We now give details of a number of algorithms whose explanations were deferred for clarity. The first algorithm, applied to the Cubist work of Chapter 3, concerns the adaptation of the Hough transform to detect superquadrics — in particular, a novel implementation strategy is described to help cater for the large (6D) parameter space of this class of shapes. The second algorithm is a P-time approximation to the graph colouring problem, borrowed from [15], also used in Chapter 3. The third algorithm is a sub-pixel accurate refinement of the Harris corner detector [68], used for camera motion compensation and feature tracking in Chapter 6. The fourth algorithm is a simple technique for identifying coloured physical markers for tracking, when Harris corner detection fails. The fifth algorithm is an illustrative example of a Kalman filter based tracker, as used in the feature tracker of Chapter 6. Finally, we have included a short proof at the end of this appendix regarding pivot point recovery in Chapter 7.

A.1 A Sparse Accumulator Representation for the Hough Transform

On two occasions in Chapter 3 our algorithms call for a method for automatically fitting superquadrics to edge data. The first occasion arises when fitting to a single feature boundary during deformation (Section 3.4.2), and the second arises when locating candidate superquadrics amongst edge pixels for the identification of candidate facial regions during portrait registration (Section 3.5.1). In this section we describe a novel implementation strategy for fitting superquadrics using the Hough transform, which makes use of a sparse representation to encode the accumulator space.

A.1.1 Introduction to the Hough Transform

The Hough transform [79] is a well studied technique for the location of geometric forms within an image. The transformation has the benefit of being a global method,

examining artifacts over the whole image to identify a target geometric structure. Local noise and artifacts tend to be disregarded in favour of the global structure in the image. The core principal behind the Hough transform is the mapping between parameter ("Hough") space, and image space. Each point in parameter space maps uniquely to a shape in image space; for example, long lines are often fitted using the polar representation $x\cos\theta + y\sin\theta = r$. In this case the Hough space is two-dimensional; the two dimensions map to free parameters θ and r respectively.

The classical Hough transform operates upon a thresholded, edge detected image. Each edge pixel in the image contributes a certain amount of evidence toward the presence of a target shape; specifically, the presence of an edge pixel contributes a small amount of evidence toward the presence of all configurations of the target shape whose boundaries intersect that edge pixel. Thus by processing all edge pixels, one may gather evidence for the presence of the target within the whole image. This evidence is stored in an "accumulator array"; each location in the high dimensional parameter (Hough) space has an associated scalar which counts the number of times that evidence has been found for the shape configuration to which that point corresponds. This evidence gathering process, often termed "voting", forms the first stage of the the Hough transform. Finding maxima in the accumulator allows us to isolate the most likely locations of shapes in the image. This search forms the second and final stage of the Hough transform.

The principal disadvantage of the Hough transform is that it does not scale well to geometric figures with a large number of free parameters. Simple lines have two free parameters, circles have three, and more complex shapes a greater number still; this results in an exponential time and space requirement for exploration of the parameter space, rendering the approach impractical to locate complex shapes. Accordingly, much of the research in extending the Hough transform has concentrated upon heuristic methods to refine the search of the parameter space. The Fast Hough transform [101] recursively divides the accumulator space into quadrants, and concentrates on the quadrant exhibiting the greatest evidence. Other localisation strategies in accumulator space include the Randomized Hough Transform [176], which uses a combination of pyramid techniques and stochastic sampling, and the Adaptive Hough Transform [83] which uses a small, fixed size accumulator space to iteratively examine potential maxima in the space. Other application specific performance improvements involve mathematical analysis of the target shape geometry to collapse the number of dimensions needed for representation in a parameter space. An excellent review of such analysis for common shapes can be found in Nixon and Aguado [116].

A.1.2 Finding superquadrics with the Hough Transform

Given the set of thresholded edge pixels in an image, we wish to locate superquadrics in that image. In the case of Section 3.4.2, we are only interested in the "best fit" i.e. maximum likelihood superquadric. In the case of Section 3.5.1 we are interested in locating candidate superquadrics, and select a fraction of superquadrics for which there is the strongest evidence.

Recall from equation 3.2 that an origin centred superquadric has the Cartesian form:

$$\left(\frac{x}{a}\right)^{\frac{2}{\alpha}} + \left(\frac{y}{b}\right)^{\frac{2}{\alpha}} = r^{\frac{2}{\alpha}} \tag{A.1}$$

and hence has three free parameters, a, r, α (note that due to the constraint $a+b=1$, b is dependent on a). These parameters correspond to eccentricity, scale, and form factor respectively. We also made use of a parametric form of this equation:

$$U_x(s) = \frac{r\cos(s)}{\left(|\cos(s)/a|^{\frac{2}{\alpha}} + |\sin(s)/b|^{\frac{2}{\alpha}}\right)^{\frac{\alpha}{2}}} \tag{A.2}$$

$$U_y(s) = \frac{r\sin(s)}{\left(|\cos(s)/a|^{\frac{2}{\alpha}} + |\sin(s)/b|^{\frac{2}{\alpha}}\right)^{\frac{\alpha}{2}}} \tag{A.3}$$

The parameter s ranges from $[0, 2\pi]$ allowing us to iterate around the perimeter; note this is not a degree of freedom defining the shape, so does not constitute a dimension of the Hough space. The superquadric may, of course, be centred anywhere in the image, and at any orientation; thus there exist three further free parameters C_x, C_y, θ, corresponding to 2D centroid location and orientation.

$$x(s) = C_x + (U_x(s)cos(\theta) - U_y(s)sin(\theta)) \tag{A.4}$$

$$y(s) = C_y + (U_x(s)sin(\theta) + U_y(s)cos(\theta)) \tag{A.5}$$

The Hough parameter space for superquadrics is therefore six dimensional, and this presents a number of practical problems. To search for a reasonable range of target shape configurations requires examination of a reasonable range of parameters at suitable level of discretisation; typical values are given in table A.1. Clearly this would require a large number of accumulator cells; for the typical example of table A.1, we would require:

$$100 \times 20 \times 180 \times 360 \times 288 \times 15 = 5.6 \times 10^{11} \, cells \tag{A.6}$$

Assuming the use of standard 32 bit integer counters for votes, we would require ap-

Parameter	Min.	Max.	Step	Interval required
r	100	200	1	100 pixels
a	0.1	0.5	0.025	20
θ	0°	180°	180°	180°
C_x	0	360	1	360 pixels
C_y	0	288	1	288 pixels
α	0.5	7	0.5	15

Table A.1 Reasonable search intervals for each of the six free parameters, when fitting superquadrics to edge pixels. A half-size PAL video frame of size 360x288 is assumed in this example, which corresponds to the size of images processed in Chapter 3.

proximately 220GB of storage for the 6D accumulator array. Although innovations such as virtual memory might make this target attainable on very high end machines, the widely distributed memory access patterns across the accumulator array can slow processing down severely due to paging, and existing Hough based accumulator approaches become impractical.

A.1.3 Our Sparse Representation Strategy

Although the distribution of votes in accumulator space is dependent on the source image, it is often the case that huge swathes of accumulator space typically gather zero or very few votes. These correspond to highly unlikely shapes, which are typically discarded when thresholding the accumulator array to find the best solutions. Our own experimentation on stock photo images using a 3D accumulator for detecting circles, and a 2D accumulator for detecting lines, has shown that on average 98 to 99 percent of accumulator space entries contain vote totals of less than a couple of percent of the maximum vote. It is intuitive that the storage required by this invalid space will increase exponentially with the dimensionality of the Hough space, and so large chunks of valuable memory are allocated, but are effectively redundant.

We therefore propose that a sparse accumulator space representation is preferable in cases of high dimensional search, and have implemented such a system to search the 6D parameter space for superquadrics. We have modified the classical Hough transform approach in two ways. First, as votes are gathered we automatically determine those solutions which are likely to be poor fits (that is, will accumulate few votes in the long term), and refrain from contributing them to the accumulator array. Second, the accumulator array is represented sparsely using a hash-table, which allows our implementation to search the 6D parameter space using several orders of magnitude less storage; typically only a few megabytes of memory.

Evidence Gathering

For each edge pixel in the image, we identify which superquadrics (i.e. points in the discrete 6D parameter space) produce boundaries which intersect that edge pixel. Our approach is to exhaustively compute all combinations of a, r, α, and θ. For each 4-tuple $[a, r, \alpha, \theta]$ tested, we determine the values $[C_x, C_y]$ that produce superquadrics whose boundary intersects the edge pixel. Given the location of the edge point $[P_x, P_y]$, we can compute this by rearranging equations A.4 and A.5.

$$C_x(s) = P_x - (U_x(s)cos(\theta) - U_y(s)sin(\theta)) \quad \text{(A.7)}$$

$$C_y(s) = P_y - (U_x(s)sin(\theta) + U_y(s)cos(\theta)) \quad \text{(A.8)}$$

Recall that s is a dummy parameter in range $[0, 2\pi]$, which iterates along the perimeter of the superquadric. We can convert s to an approximate arc-length parameterisation (which we write as $s = C(s'), s' = [0, 1]$) by linear approximation; unfortunately, an exact parameterisation in closed-form is known to be impossible in the general case. The level at which we discretise s' determines the resolution at which we can produce centroids; we use a ratio of one step interval to each perimeter pixel, usually requiring around 200 to 300 discrete intervals of s'. Thus around 200 to 300 potential centroids are generated.

Typically we obtain around 5 to 10 million parameter 6-tuples (potential superquadrics) for each edge pixel (again, referring to our typical values in Table A.1). Although this is a large number of solutions the majority are immediately discarded by a computationally trivial sifting process, which we now describe.

Culling Poor Solutions

Most of these potential superquadrics are poor fits. To mitigate against the large storage requirements needed to record these solutions in the accumulator, we disregard poor superquadric solutions by determining the quality of their fit to the image data. First, superquadrics whose centres fall outside the bounds of the image are culled. Second, we plot each potential superquadric in image space, and count how many edge pixels intersect its boundary. We set a heuristic stating that if less than K percent of the edge pixels covered by the superquadric are lit, then that solution is to be disregarded. We use $K = 20$. The remaining superquadric solutions are added to the sparse accumulator, in a process we now describe.

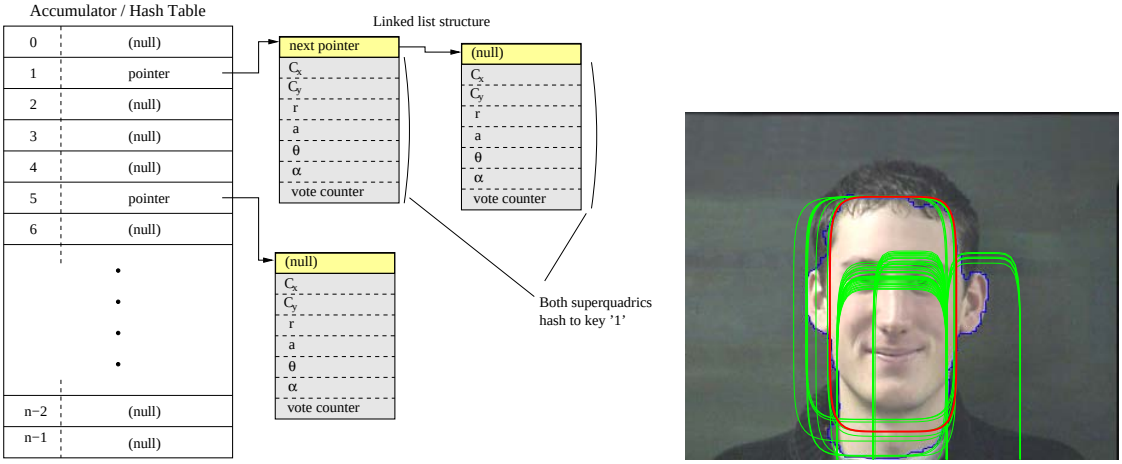


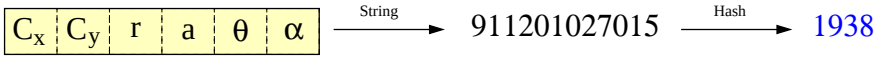
Figure A-1 Left: Illustrating the hash table and linked list structures used to store the sparse accumulator array. Right: Example source image, with a fraction of the most likely superquadrics detected by our method overlaid.

Sparse Accumulator and Hash Table

The accumulator array is stored using a sparse representation in our implementation. We initialise a hash-table prior to processing, with a predefined number of slots n (we discuss choice of n momentarily). Each slot in the hash-table contains a linked-list of 7-tuples; each contains the superquadric parameter 6-tuple plus an integer count which records the number of times that superquadric has been voted for by the evidence gathering process (Figure A-1, left).

When a superquadric vote is to be registered in the accumulator during processing, an integer hash key is created from that superquadric's parameters $[C_x, C_y, \alpha, r, a, \theta]$. We then traverse the linked list in the slot of the hash-table indexed by the hash key, to determine if that superquadric has been previously stored. If it has, we increment its vote count. Otherwise we append the superquadric to the end of the linked list, and initialise its vote count to one.

The hash key is created by a hash function, which operates as follows. All superquadric parameters are rounded to their integer values (some parameters such as a have a fractional range < 1, and so are scaled by a power of ten) and these integer values are concatenated as a string.



This string is converted back into an integer value modulo n to produce the hash function. As Knuth points out [94], it is important to choose a prime hash-table size

(n) when using modulo based hash functions, to promote uniform distribution of the hash function over its range.

Performance

Our Pentium IV, 2GHz based implementation processes one edge pixel every 30 seconds, over the full interval range of Table A.1. A typical deinterlaced PAL image (Table A.1, Figure A-1, right) may contain around 5000 Canny edge pixels, requiring 150000 seconds, or almost two days for the evidence gathering process to complete. With the relaxed memory requirements of our sparse representation this is an attainable albeit clearly unacceptable amount of processing time. However this falls quickly if broad estimates are used to constrain the intervals of potential parameters, as we now describe.

In the case of Section 3.4.2 we wish to fit a single superquadric to a set of boundary edge points. We can obtain good estimates for a, θ and r through a PCA of points distributed evenly inside the shape boundary. For a we use the ratio between eigenvalues of the principal axes, r the magnitudes of both eigenvalues, and θ takes into account principal axis orientation. A narrow search window for each of these parameters replaces the broad ranges in Table A.1, yielding a fit in about one second.

In the case of Section 3.5.1 we wish to return a collection of superquadrics most likely to be present with the edge data. Again, estimates can be used to constrain this search. The application of Section 3.5 is that of face location within commercial photo booth, which is a controlled environment. Orientation θ ($\frac{\pi}{2}$) and scale r ([100,150]) fall within reasonably tight constraints, as does eccentricity a which may assumed to be $0.5 \leq a \leq 0.8$. The form factor of the superquadric may be assumed to vary between $[1, 6]$. These constraints on the search space reduce execution time to less than a minute using unoptimised code.

As regards storage, the hash-table is rarely written to (relative to the number of solutions processed and discarded) due to the success of the culling step, and often contains no more than around 10000 individual superquadrics. This gives a guide as to typical values for n; in our implementation $n = 10007$ (a prime number in the neighbourhood of 10000).

Results

Once evidence gathering is complete, it is a trivial matter to extract the relatively small number of superquadric solutions from the hash table, and to sort them in order of their vote count. The best solution (Chapter 3, Section 3.4.2) or a fraction of the best

solutions (Chapter 3, Section 3.5.1) can be readily obtained from this list. Figure A-1, right shows a number of identified superquadrics in a typical image.

A.1.4 Summary and Conclusion

We have described a simple implementation strategy for the Hough transform which enables the location of geometric figures exhibiting a large number of free parameters. Our approach centres around the implementation of the Hough space as a sparse data field, which is indexed using a hash-table structured coupled with a secondary linear search. This approach enables us to search the 6D parameter space of superquadrics using a quantity of storage several orders of magnitude lower than required by existing Hough based methods, which are impractical for application to this problem space. Although computational overhead is still too high for practical exhaustive searches of this space, sensible estimates can be used to reduce processing time to practical levels for the specific application of Chapter 3 (Cubist rendering).

There is an underlying assumption behind our strategy, namely that the majority of edge information for a superquadric will be present in the image. In order for a superquadric to survive the culling process, at least K percent of its boundary must be evidenced by edge pixels in the source image. If the image is very noisy (causing much of the edge detail to be lost), then a classical Hough transform could eventually recover shapes by slowly accumulating poorly evidenced solutions, whereas our approach could not. This could be mitigated by reducing K, but this would require increased storage, and ultimately when $K = 0$, the method degenerates into the classical Hough transform. Therefore the choice of K value must be decreased in proportion to the amount of edge loss expected in the source image (for example, due to occlusion or noise). It would be interesting to further study how the performance of this method deteriorates against the classical Hough transform, as edge corrupting noise increases. However this work falls outside the remit of this thesis and empirical investigation on matters such as robustness, and choice of K are deferred for future investigation.

A.2 A P-time Approximation to the Graph Colouring Problem

We give details of the P-time approximation used to colour our region adjacency graph in Section 3.4.3. The algorithm is guaranteed to return a graph where no two directly connected nodes are assigned the same colour. The algorithm may use a greater *total number* of colours than the minimum possible solution. The algorithm trades accuracy for speed of execution; a precise solution is known to be NP-hard. However, the heuristics implicit in this approximation have been empirically shown to guide the algorithm

toward a close approximation to the minimum colouring. In our application, adjacency graphs are often encoded using eight or fewer colours. The algorithm originates in machine-language compilation literature, where a limited set of registers are allocated to potential larger set of program variables using a graph colouring approach [15].

Require: graph G of n nodes denoted by G_i $[1 \leq i \leq n]$
1: Integer $maxcols \leftarrow n$
2: Integer $mincols \leftarrow 4$
3: **for** $totalcols = mincols$ to $maxcols$ **do**
4: Graph $W \leftarrow G$
5: Ordered List $L \leftarrow \emptyset$
6: **repeat**
7: $R \leftarrow$ index of node W_R with greatest count of immediate neighbours, that count not exceeding $totalcols - 1$
8: **if** $R \neq \emptyset$ **then**
9: remove node R from graph W
10: add node R to list L
11: **end if**
12: **until** $W == \emptyset \| R == \emptyset$
13: **if** $W == \emptyset$ **then**
14: Define empty sets C_i where $[1 \leq i \leq totalcols]$
15: **for** $idx = n$ to 1 **do**
16: **for** $i = 1$ to $totalcols$ **do**
17: **if** G_{idx} is not a direct neighbour of any nodes in C_i **then**
18: $C_i \leftarrow C_i \cup G_{idx}$
19: **end if**
20: **end for**
21: **end for**
22: **end if**
23: **end for**

The algorithm terminates with all members of C_i being assigned some colour i, such that two directly connected nodes do not share the same i. The algorithm operates by repeatedly "loosening" the graph; each iteration of the inner *repeat* loop eliminating the colourable node with highest connectivity. This is the heuristic that guides the algorithm, and seems intuitively correct. Were we to pursue the consequences of eliminating each node at each iteration of the inner *repeat* loop, one branch of the resulting decision tree would terminate at a minimal solution. However this would produce a combinatorial search of all possible colourings.

A.3 The Harris Corner Detector

We make use of the Harris corner detector in our motion emphasis work of Chapter 6, and give details here of a refined detector which operates with sub-pixel accuracy.

A.3.1 Basic Algorithm

In Chapter 6 we identify interest points within an image using the corner detector proposed by Harris and Stephens [68], which is an extension of a previous feature detection technique due to Moravec [113]. Although the operator is dubbed a "corner" detector, it is more accurately a means of robustly locating interest points which remain relatively stable over viewpoint change (i.e. if a "corner" is detected on an world point imaged from one point of view, then that world point is likely to be detected as a "corner" when imaged from a second point of view).

Corners are identified in an image $I(x, y)$ by performing a local template operation upon the horizontal and vertical first derivatives of the luminance channel. The operator is based upon the Moravec operator which performs an auto-correlation of the image; this may be written as a sum of squares:

$$\partial I(\partial x, \partial y) = \sum_{i,j \subset template} |I(i + \partial x, j + \partial y) - I(i, j)|^2 \tag{A.9}$$

The horizontal and vertical first derivatives are each generated by simple differencing template operations; the size of the template effectively governs the scale at which artifacts are detected, and so the level of low-pass filtering that occurs. Others [158] have found a window width of 3 to be a good trade off between resolution and susceptibility to noise, and we accept this value. The Harris detector is based upon the Taylor expansion of equation A.9, from which is obtained:

$$\partial I(\partial x, \partial y) \quad = \quad (\partial x, \partial y) \underline{\underline{N}}(\partial x, \partial y)^T \tag{A.10}$$

Where matrix $\underline{\underline{N}}(x, y)$ characterises the signal in that window:

$$\underline{\underline{N}}(x, y) \quad = \quad \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{A.11}$$

The eigenvalues of $\underline{\underline{N}}(.)$ are proportional its principal curvatures, and Harris and Stephens determined that analysis of $\underline{\underline{N}}(x, y)$ may be used to decided if a "corner" is present at point (x, y). Specifically, they observe that the trace of the matrix is large when an edge is present, and the determinant of the matrix is large when an edge or corner is present. Thus by subtracting these two signals, they propose a reasonable

measure of "cornerness" to be:

$$C(x,y) = |\underline{\underline{N}}(x,y)| - \kappa T(\underline{\underline{N}}(x,y)) \tag{A.12}$$

where $T(.)$ returns the trace of the matrix, and commonly the constant $\kappa = 0.04$.

For a given window, the pixel with maximum "cornerness" is picked as the interest point within the template; the value of this pixel is the likelihood of that point being a "corner". Since the function $C(.)$ is largely image dependent, the strongest n corners are often isolated, rather than specifying a global threshold.

A.3.2 Extension to Sub-pixel Accuracy

Since the Harris process identifies the pixel with maximum "cornerness" within a small template, the measure is accurate only to the level of one pixel. Several authors have reported extension of this technique to operate with sub-pixel accuracy; simple approaches improve resolution by scaling up the image prior to applying the Harris operator. We have developed a closed form solution to obtain sub-pixel accuracy within the 3×3 neighbourhood of an identified corner $C(x,y)$. Although we do not claim novelty in describing a sub-pixel Harris operator, we have observed a distinct lack of literature documenting a closed-form refinement, and include details of our approach here for reference. We fit a quadratic surface to the 3×3 neighbourhood centred upon $C(x,y)$, which we write as $C'(x,y)$.

$$C'(x,y) = ax^2 + by^2 + cxy + dx + ey + f \tag{A.13}$$

The coefficients a–f are obtained through solution of the following homogeneous linear system:

$$0 = \underline{\underline{A}}[a\ b\ c\ d\ e\ f\ 1]^T \tag{A.14}$$

$$\underline{\underline{A}} = \begin{bmatrix} x^2 & y^2 & xy & x & y & 1 & -C(x,y) \\ (x-1)^2 & (y-1)^2 & (x-1)(y-1) & x-1 & y-1 & 1 & -C(y-1,x-1) \\ (x+1)^2 & (y-1)^2 & (x+1)(y-1) & x+1 & y-1 & 1 & -C(y-1,x+1) \\ (x-1)^2 & (y+1)^2 & (x-1)(y+1) & x-1 & y+1 & 1 & -C(y+1,x-1) \\ (x+1)^2 & (y+1)^2 & (x+1)(y+1) & x+1 & y+1 & 1 & -C(y+1,x+1) \\ x^2 & (y+1)^2 & x(y+1) & x & y+1 & 1 & -C(y+1,x) \\ x^2 & (y-1)^2 & x(y-1) & x & y-1 & 1 & -C(y-1,x) \\ (x-1)^2 & y^2 & y(x-1) & x-1 & y & 1 & -C(y,x-1) \\ (x+1)^2 & y^2 & y(x+1) & x+1 & y & 1 & -C(y,x+1) \end{bmatrix}$$

We obtain an algebraic least squares fit via SVD of A. Once the coefficients have been

fitted, the exact coordinates of the turning point $(x', y')^T$ corresponding to the interest point, may be obtained by solving a further linear system:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2a & c \\ c & 2b \end{bmatrix}^{-1} \begin{bmatrix} -d \\ -e \end{bmatrix} \tag{A.15}$$

Thus we obtain an exact pair of coordinates for the corner $(x', y')^T$ and an exact value for its "cornerness" from $C'(x', y')$ as defined in equation A.13.

A.4 A 1D Illustrative Example of a Kalman Tracker

We give an illustrative example of tracking using the Kalman Filter [90]. This technique was used in the Computer Vision component of Chapter 6 to track features through video clips. In this example we consider a particle moving in 1D, modelled by a motion equation of 2nd order. This is representative of the system used in Section 6.3.2, where a point is tracked within the 4D parameter space of the LCAT under a 2nd order motion model. In that case, four independent Kalman filters were used; one for each dimension of the space.

Considering our 1D example, the state of the particle (at position p in the 1D space) may be written as a vector:

$$\underline{\hat{x}} = [p \;\; \dot{p} \;\; \ddot{p}]^T \tag{A.16}$$

We will model the evolution of the particle's state over equal, discrete time intervals, each of length dt. Let $\underline{\underline{A}}$ be the "state transition" matrix. Later, this matrix will be multiplied with the state vector to update that state over the time interval dt. The matrix thus encodes our 2nd order motion model:

$$\underline{\underline{A}} = \begin{bmatrix} 1 & dt & \frac{dt^2}{2} \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix} \tag{A.17}$$

Process noise, the error due to motion being inaccurately described by our motion model, is written as q. We assume the expectation value of this noise to be zero. $\underline{\underline{Q}}$ is the process noise covariance:

$$\underline{\underline{Q}} = q^2 \begin{bmatrix} \frac{dt^6}{36} & \frac{dt^5}{12} & \frac{dt^4}{6} \\ \frac{dt^5}{12} & \frac{dt^4}{4} & \frac{dt^3}{2} \\ \frac{dt^4}{6} & \frac{dt^3}{2} & dt^2 \end{bmatrix} \tag{A.18}$$

Finally, let $\underline{c} = [1 \;\; 0 \;\; 0]$ be the "measurement vector" which is a projection describing

which part of $\underline{x}$ corresponds to the p we are observing i.e. $p = \underline{cx}$.

A.4.1 Initialisation

The internal state of the Kalman filter at time t is written as $\underline{\hat{x}}_t$. We typically initialise the first iteration $\underline{\hat{x}}_1$ with a measurement of position from the tracker at the first frame. In Chapter 6 we specified estimates of zero for initial velocity and acceleration; these gave reasonable results for our application.

The covariance of prediction error for $\underline{\hat{x}}_t$, at time t, is written as $\underline{\underline{P}}_t$. $\underline{\underline{P}}_t$ represents the filter's confidence in its internal state. Initially we set $\underline{\underline{P}}_1 = \underline{\underline{Q}}$.

A.4.2 Iterative Process

The Kalman tracker runs as an iterative "predict–correct" process, tracking motion from time t to time $t + dt$ as follows:

1. Let $\underline{x}^+$ be our advance prediction for the state of the system at time $t + dt$:

$$\underline{x}^+ = \underline{\underline{A}}\underline{\hat{x}}_t \tag{A.19}$$

2. Let $\underline{\underline{P}}^+$ be our advance prediction for the covariance of prediction error of the system at time $t + dt$:

$$\underline{\underline{P}}^+_{t+dt} = \underline{\underline{A}}\underline{\underline{P}}_t\underline{\underline{A}}^T\underline{\underline{Q}} \tag{A.20}$$

3. Let z be the noisy measurement of p made from the world at time $t + dt$. Let r be the variance (i.e. our un-confidence) in this measurement at time $t + dt$. In Chapter 6 our estimate for p was determined using RANSAC, and r was a function of the MSE between the feature's known appearance and the image region bounded by the RANSAC-supplied contour.

4. Let s be the covariance of our prediction, taking into account r:

$$s = \underline{c}\underline{\underline{P}}^+\underline{c}^T + r^2 \tag{A.21}$$

5. Let the Kalman gain $\underline{k}$ be:

$$\underline{k} = \underline{\underline{P}}^+\underline{c}^T s^{-1} \tag{A.22}$$

6. The updated covariance of prediction error $\underline{\underline{P}}_{t+dt}$ for this iteration is:

$$\underline{\underline{P}}_{t+dt} = (\underline{\underline{I}} - \underline{k}\underline{c})\underline{\underline{P}}^{+} \tag{A.23}$$

7. The updated state estimate $\underline{\hat{x}}_{t+dt}$ for this iteration is:

$$\underline{\hat{x}}_{t+dt} = \underline{\hat{x}}^{+} + \underline{k}(z - \underline{c}\underline{\hat{x}}^{+}) \tag{A.24}$$

   At this stage $\underline{\hat{x}}_{t+dt}$ is our best estimate for time $t + dt$, and the position of the particle $p = \underline{c}\underline{\hat{x}}$ may be extracted and used as a result of the tracking process at this instant.

8. Repeat from step (1.) until all frames are tracked.

A.5 Identification of Markers for Substitution in Tracking

In Section 6.3.2, we describe the tracker within the Computer Vision component of the motion emphasis subsystem. The default operation of the tracker is to identify Harris interest points [68] within features, and by creating correspondence between points in adjacent frames, produce an estimate for the motion of the feature over that particular time interval. However, in more complex cases where point correspondences for tracking can not be found (perhaps due to signal flatness, small feature area, or similarity between closely neighbouring features), distinctively coloured markers may be physically attached to the subject and later removed digitally (see *STAIRS*, Figure 6-2). In these cases the Harris interest points are substituted for points generated by analysis of the colour distribution in a frame. We now describe the marker identification process.

We first train a colour model for each marker type. Several frames of the stationary subject are captured and an eigenmodel fitted to manually segmented observations of each marker. In practice we make use of only the hue and saturation components. Each marker's pixels form distinct, tight clusters in HS space, justifying our use of a single Gaussian based model (Figure A-2, left).

We apply our trained colour model to a novel image sequence containing the moving subject, captured in similar lighting conditions. To locate markers of a particular colour within a video frame, we compute the Mahalanobis distance of every pixel to the mean of the relevant colour eigenmodel (specified by a mean $\underline{\mu}$, eigenvectors $\underline{\underline{U}}$

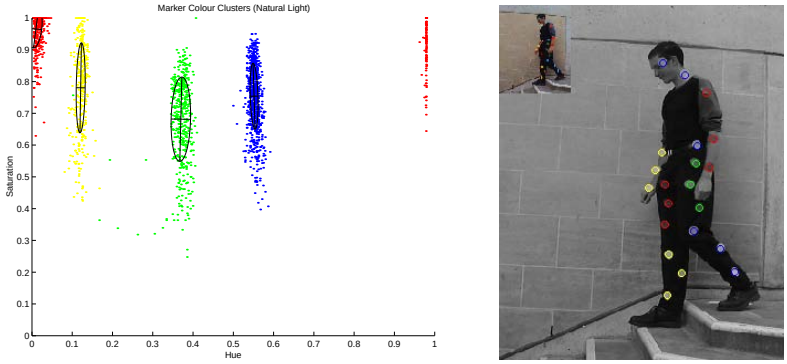


Figure A-2 Marker based capture. Left: Four distinct marker colour clusters in HS space; ellipses denote one standard deviation. Note the hue dimension is cyclic (modulo 1), thus the graph should be interpreted as cylindrical. Right: Markers identified within a single video frame.

and eigenvalues $\underline{\underline{\Lambda}}$). The likelihood of a particular pixel at location $\underline{x} = (i, j)^T$, and of colour $\underline{c} = (h, s)^T$ belonging to a marker m may be given by:

$$L(\underline{c}, m) = \left[(\underline{c} - \underline{\mu}_m)^T \underline{\underline{U}}_m \underline{\underline{\Lambda}}_m \underline{\underline{U}}_m^T (\underline{c} - \underline{\mu}_m) \right]^{\frac{1}{2}} \tag{A.25}$$

As with the skin colour clustering of Section 3.5 we refine this expression to take into account $c_h = c_h \mod 1$ (because hue is cyclic):

$$L'(\underline{c}, m) = \min(L(\underline{c}, m), L((c_h + 0.5, c_s)^T, m)) \tag{A.26}$$

Simply thresholding this function can produce disjoint regions about a marker, and does not yet take into account pixel location. We therefore refine our model to include a spatial smoothness assumption; if a pixel is likely to belong to a marker, there is an increased probability that its immediate neighbours do too. We model this influence as a 2D Gaussian of constant deviation σ, which we implement via a convolution over the likelihood field generated by equation A.26 for each pixel in the image:

$$L''(\underline{x}, \underline{c}, m) = L'(\underline{c}, m) * G(\underline{x}, \sigma); \tag{A.27}$$

We threshold the modified function, and perform morphological closure [145] within connected regions to produce a binary map of markers for that frame. Potential markers are identified at the centroid of each connected region (Figure A-2b), and form a substitute for the Harris points as input to the LCAT estimator described in Section 6.3.2.

A.6 On the Effects of Pivot Motion during Rotation in 2D

In Chapter 7 we address the problem of recovering the location of the pivot point between two polygons, exhibiting motion in the plane. We give a short proof demonstrating that any translation of the pivot in the plane will cause an orthogonal translation in the motion field generated by rotation about that pivot.

Consider a point $\underline{x}(t)$ at time t, rotating about pivot $\underline{p}$ at a constant radius ρ. If ω is the angular velocity of that rotation, then:

$$|\dot{\underline{x}}(t)| = \rho\omega \tag{A.28}$$

When $\underline{p}$ is coincident with the origin, we may write:

$$\dot{\underline{x}}(t) = \omega(\underline{r}(t) \times \hat{\underline{n}}) \tag{A.29}$$

where $\underline{r}(t) = \underline{x}(t) - \underline{p}$ is a position vector of length ρ, and $\hat{n}$ is the vector normal to the plane (i.e. parallel to the axis of rotation). If the pivot $\underline{p}$ undergoes translation, and is no longer coincident with the origin, the system must now be written as:

$$\begin{aligned} \dot{\underline{x}}(t) &= \omega((\underline{r}(t) - \underline{p}) \times \hat{\underline{n}}) \\ &= \omega(\underline{r}(t) \times \hat{\underline{n}}) - \omega(\underline{p} \times \hat{\underline{n}}) \end{aligned} \tag{A.30}$$

Thus the velocity of $\underline{x}(t)$ is the sum of two velocity vectors. The first term is angular velocity about the origin, and invariant to pivot motion. The second term is a change in velocity due to $\underline{p}$, and is orthogonal to both the plane normal $\hat{\underline{n}}$ and to $\underline{p}$ itself. Thus translation of the pivot $\underline{p}$ generates a change in $\dot{\underline{x}}(t)$ orthogonal to that translation. If the pivot point translation is unknown, and we seek the pivot point (which is itself unknown) then the sum of the two vector terms can not be uniquely decomposed.

Appendix B

Points of Definition

Appendix C

Supplementary Material: Images, Papers and Videos (Electronic)

The enclosed DVD-ROM (affixed to the back cover) contains a collection of paintings, papers and videos which support this thesis.

C.1 Paintings

High resolution versions of paintings presented in this thesis are contained in the `/paintings` directory:

`bathabbey.tif`	Bath Abbey (Figure 4-21)
`chilepickup.tif`	Pickup Truck (Figure 4-14)
`dragon_best.tif`	Chinese dragon post-relaxation (Figure 4-1)
`dragon_gen1.tif`	Chinese dragon after 1 iteration of GA relaxation
`dragon_gen30.tif`	Chinese dragon after 30 iterations of GA relaxation
`dragon_gen70.tif`	Chinese dragon after 70 iterations of GA relaxation
`modelface.tif`	Man on a rock (detail on face before sharpening)
`relaxation.mpg`	Video of the GA relaxation process for dragon
`rock.tif`	Man on a rock (Figure 4-13)
`still-life.tif`	Still-life, kitchen (Figure 4-8)
`sunflowers.tif`	Sunflowers (Figure 4-15)
`cubist/CharlesClark.pdf`	Front page of the THES feat. Cubist portrait (Figure 3-18)
`cubist/guitar.tif`	Cubist still-life (guitar) (Figure 3-17d)
`cubist/portrait_jpc.tif`	Cubist portrait (of author) (Figure 3-17b)
`cubist/portrait_pmh.tif`	Cubist portrait (of supervisor) (Figure 3-21)

C.2 Papers

Copies of publications arising from this thesis [22, 23, 24, 25, 26, 27] and the state of the art video painting algorithm used in the comparison of Chapter 8 [103] are included in the `/papers` directory.

C.3 Videos

The following source videos and animations are included in the `/videos` directory:

`ballet_streak.avi`	*BALLET* sequence with streak-lines, demonstrating depth ordering.
`basket_rendered.avi`	*BASKETBALL* sequence with augmentation and deformation cues, demonstrating occlusion handling.
`basket_source.avi`	Source *BASKETBALL* video sequence.
`bounce_deformationonly.avi`	*BOUNCE* sequence exhibiting only squash and stretch deformation, illustrating collision handling.
`bounce_flatshade.avi`	*BOUNCE* sequence, flat shaded with no outlines. Exhibits squash and stretch deformation.
`bounce_fullcartoon.avi`	*BOUNCE* sequence, full cartoon flat shaded with sketchy outlines. Exhibits both augmentation and deformation motion cues.
`bounce_gradshade.avi`	*BOUNCE* sequence, gradient shaded with no lines.
`bounce_mixedmedia.avi`	*BOUNCE* sequence, mixed media effect — actor photorealistic, background sketchy. Exhibits both augmentation and deformation cues.
`bounce_motiononly.avi`	*BOUNCE* sequence exhibiting both augmentation and deformation cues, but no shading.
`bounce_painterly_ourmethod.avi`	Our painterly video algorithm (Chapter 8) applied to *BOUNCE*.
`bounce_painterly_SoA-.avi`	State of the art painterly video algorithm [103] (without stroke density regulation) applied to *BOUNCE*.

bounce_painterly_SoA.avi	State of the art painterly video algorithm [103] applied to *BOUNCE*.
bounce_refframe.avi	Demonstrating the attachment of a rigid reference frame for painting in *BOUNCE*.
bounce_source.avi	Source *BOUNCE* video sequence.
bounce_watercolourwash.avi	*BOUNCE* sequence with watercolour effect.
contraption.avi	Tracked *CONTRAPTION* sequence, used to test pivot point recovery algorithms. Here, pivot points have been identified and articulated pose recovered automatically (pose vector on right hand side of frame).
cricket_mblur.avi	*CRICKET* sequence exhibiting motion blur (tightly packed ghosting lines).
cricket_source.avi	Source *CRICKET* video sequence.
cricket_streakghost.avi	*CRICKET* sequence exhibiting streak lines and ghosting.
metro_anticipate.avi	*METRONOME* sequence exhibiting "time and pose" cues (specifically, anticipation).
metro_qmapped.avi	*METRONOME* sequence exhibiting both augmentation cues and coherent application of Q-mapped textures [65].
metro_source.avi	Source *METRONOME* video sequence.
metro_streaks.avi	*METRONOME* sequence exhibiting streaklines and ghosting.
metro_warp_accel.avi	*METRONOME* sequence exhibiting emphasised inertia by non-linear deformation.
metro_warp_anticipate.avi	*METRONOME* sequence exhibiting both anticipation, and deformation motion cues.
metro_warp_veloc.avi	*METRONOME* sequence exhibiting emphasised drag by non-linear deformation.
panorama.avi	Demonstrating how video frames are registered to one another via homography, so producing a camera motion compensated sequence (uses *VOLLEY* footage).
pooh_angrybear.avi	Example of rotoscoping (an illustration is rotoscoped onto the head in *POOHBEAR* sequence).

pooh_cartoon.avi	*POOHBEAR* sequence, cartoon flat shaded with solid brush lines.
pooh_coherentshade.avi	*POOHBEAR* sequence, demonstrating coherent setting of interior region attributes (colour). See Figure 8-13.
pooh_flatshade.avi	*POOHBEAR* sequence, flat shaded no lines.
pooh_gradshade.avi	*POOHBEAR* sequence, gradient shaded no lines.
pooh_incoherentshade.avi	*POOHBEAR* sequence, demonstrating simplistic (incoherent) setting of interior region attributes (colour). See Figure 8-13.
pooh_painterly_falsecolour.avi	*POOHBEAR* painterly rendering of head, in false colour to demonstrate stroke coherence.
pooh_painterly_truecolour.avi	*POOHBEAR* painterly rendering of head in true colour.
pooh_refframe.mpg	Demonstrating the attachment of a rigid reference frame for painting in the *POOHBEAR* sequence.
pooh_source.avi	Source *POOHBEAR* video sequence.
pooh_watercolourwash.avi	*POOHBEAR* sequence with watercolour wash effect.
pooh_wobblyflatshade.avi	Introducing controlled incoherence into the Stroke Surfaces in *POOHBEAR* (flat shaded).
pooh_wobblygradshade.avi	Introducing controlled incoherence into the Stroke Surfaces in *POOHBEAR* (gradient shaded).
sheep_flatsegment.avi	Cartoon flat-shaded *SHEEP* sequence (no lines).
sheep_painterly_falsecolour.avi	*SHEEP* painterly rendering of sheep in false colour to demonstrate stroke coherence.
sheep_painterly_SoA-.avi	State of the art painterly video algorithm (without stroke density regulation) applied to *SHEEP*.
sheep_painterly_SoA.avi	State of the art painterly video algorithm as described in [103] applied to *SHEEP*.

sheep_painterly_truecolour.avi	*SHEEP* painterly rendering of sheep in true colour.
sheep_rotomatte.avi	Demonstrating rotoscoping and video matting in the *SHEEP* sequence.
sheep_sketchcartoon.avi	*SHEEP* sequence, flat shaded with sketchy lines.
sheep_source.avi	Source *SHEEP* video sequence.
sheep_wobblycartoon.avi	Introducing controlled incoherence into the holding line in the *SHEEP* sequence, but maintaining coherence in interior regions (flat shaded with solid brush lines).
spheres_painterly_ourmethod.avi	Our proposed painterly video algorithm applied to *SPHERES*.
spheres_painterly_SoA-.avi	State of the art painterly video algorithm (without stroke density regulation) applied to *SPHERES*.
spheres_painterly_SoA.avi	State of the art painterly video algorithm as described in [103] applied to *SPHERES*.
spheres_sketchywash.avi	*SPHERES* with watercolour wash effect and sketchy lines.
spheres_source.avi	source *SPHERES* video sequence (synthetic test sequence used in Chapter 8).
spheres_wobblyflatshade.avi	*SPHERES* sequence, flat shaded and after introduction of controlled incoherence into the region boundary.
stairs_exaggerate.avi	*STAIRS* sequence with motion exaggeration (filled feature polygons only)
stairs_exaggerate_polys.avi	*STAIRS* sequence with motion exaggeration, textured to produce Monty Python style animation.
stairs_source.avi	Source *STAIRS* video sequence.
volley_motiononly.avi	*VOLLEY* sequence with augmentation and deformation cues, demonstrates camera motion compensation.
volley_source.avi	Source *VOLLEY* video sequence, exhibits large scale camera motion.
vpshowreel.avi	Video Paintbox show-reel (uncompressed RGB — very large, but full quality).

`vpshowreel_lowres_divx5.avi`	Video Paintbox show-reel (highly compressed, 52Mb approx DIVX-5 compression).
`wand_cartoon.avi`	*WAND* sequence, begins with original footage, then adds augmentation cues, then adds deformations, then adds video shading (flat shaded cartoon with sketchy lines).
`wave_sketchonly.avi`	*WAVE* sequence, sketchy lines only.
`wave_sketchycartoon.avi`	*WAVE* sequence, full cartoon flat shaded with sketchy lines.
`wave_source.avi`	Source *WAVE* video sequence.
`wave_thicklinecartoon.avi`	*WAVE* sequence, cartoon flat shaded with solid brush lines.

Bibliography

[1] Agarwala, A. (2002, June). Snaketoonz: A semi-automatic approach to creating cel animation from video. In *Proc.* 2nd *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 139–147.

[2] Armstrong, A. and J. Jiang (2002, June). Direct DCT indexing using genetic algorithm concepts. In *Proc.* 20th *Eurographics UK Conference*, pp. 61–66.

[3] Bangham, J. A., S. E. Gibson, and R. Harvey (2003, September). The art of scale-space. In *Proc.* 14^{th} *British Machine Vision Conference (BMVC)*, Volume 1, pp. 569–578.

[4] Baxter, B., V. Scheib, M. C. Line, and D. Manocha (2001). DAB: Interactive haptic painting with 3D virtual brushes. In *Proc.* 28th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 461–468.

[5] Belongie, S., C. Carson, H. Greenspan, and J. Malik (1998). Color and texture-based segmentation using EM and its application to content-based image retrieval. In *Proc. of Intl. Conference on Computer Vision (ICCV)*, pp. 675–682.

[6] Bennett, E. P. and L. McMillan (2003, November). Proscenium: A framework for spatio-temporal video editing. In *Proc. ACM Multimedia Systems*, pp. 177–183.

[7] Bergen, J. R. and R. Hingorani (1990). Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Centre, Princeton, N. J.

[8] Bregler, C., L. Loeb, E. Chuang, and H. Deshpande (2002). Turning to the masters: motion capturing cartoons. In *Proc.* 29th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, July, pp. 399–407.

[9] Buchanan, J. W. (1996). Special effects with half-toning. In *Proc. Computer Graphics Forum (Eurographics)*, pp. C97–108.

[10] Cai, Q. and J. K. Aggarwal (1996, August). Tracking human motion using multiple cameras. In *Proc. Intl. Conference on Pattern Recognition (ICPR)*, Vienna, Austria, pp. 68–72.

[11] Cai, Q., A. Mitiche, and J. K. Aggarwal (1995, October). Tracking human motion in an indoor environment. In *Proc. Intl. Conference on Image Processing*, Washington, D.C., pp. 215–218.

[12] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 6*(8), 679–698.

[13] Capel, D. (2001). *Image Mosaicing and Super-resolution*. Ph. D. thesis, Oxford University, Dept. of Engineering Science.

[14] Catmull, E. E. and R. J. Rom (1974). A class of local interpolating splines. *Computer Aided Geometric Design*, 317–326.

[15] Chaitin, G. J., M. A. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins, and P. W. Markstein (1981). Register allocation via colouring. *Computer Languages 6*, 47–57.

[16] Chen, A., K. Knudtzon, J. L. Stumpfel, and J. K. Hodgins (2000). Artistic rendering of dynamic motion. In *Proc. Computer Graphics (ACM SIGGRAPH Sketches)*, pp. 100.

[17] Chen, Z. and H. J. Lee (1992). Knowledge-guided visual perception of 3D human gait from a single image sequence. *IEEE Transactions on Systems, Man and Cybernetics 22*(2), 336–342.

[18] Chenney, S., M. Pingel, R. Iverson, and M. Szymanski (2002, June). Simulating cartoon style animation. In *Proc.* 2nd *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 133–138.

[19] Christoudias, C., B. Georgescu, and P. Meer (2002, August). Synergism in low level vision. In 16th *Intl. Conference on Pattern Recognition (ICPR)*, Volume 4, Quebec City, Canada, pp. 150–155.

[20] Cockshott, T., J. Patterson, and D. England (1992, September). Modelling the texture of paint. *Computer Graphics Forum 11*(3), 217–226.

[21] Cohen, I., L. D. Cohen, and N. Ayache (1992, September). Using deformable surfaces to segment 3-d images and infer differential structures. *Journal on Graphical Models and Image Processing (CVGIP) 56*(2), 242–263.

[22] Collomosse, J. P. and P. M. Hall (2002, June). Painterly rendering using image salience. In *Proc. 20th Eurographics UK Conference*, pp. 122–128.

[23] Collomosse, J. P. and P. M. Hall. (2003, October). Cubist style rendering from photographs. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 4*(9), 443–453.

[24] Collomosse, J. P. and P. M. Hall (2003, October). Genetic painting: A salience adaptive relaxation technique for painterly rendering. Technical Report CSBU 2003–02, University of Bath, UK.

[25] Collomosse, J. P., D. Rowntree, and P. M. Hall (2003a, July). Cartoon-style rendering of motion from video. In *Proc.* 1st *Intl. Conference on Vision, Video and Graphics (VVG)*, pp. 117–124.

[26] Collomosse, J. P., D. Rowntree, and P. M. Hall (2003b, June). Stroke surfaces: A spatio-temporal framework for temporally coherent non-photorealistic animations. Technical Report CSBU 2003–01, University of Bath, UK.

[27] Collomosse, J. P., D. Rowntree, and P. M. Hall (2003c, September). Video analysis for cartoon-like special effects. In *Proc.* 14^th *British Machine Vision Conference*, Volume 2, Norwich, pp. 749–758.

[28] Collomosse, J. P., D. Rowntree, and P. M. Hall (2004). Automatic rendering of cartoon-style motion cues in post-production video. *Journal on Graphical Models and Image Processing (CVGIP)*. Submitted.

[29] Comanicu, D. and P. Meer (2002, May). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 24*(5), 603–619.

[30] Cook, R., L. Porter, and L. Carpenter (1984). Distributed ray tracing. In *Proc.* 11^{th} *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 137–145.

[31] Cosgriff, R. L. (1960). Identification of shape. Technical Report 820-11, ASTIA AD 254792, Ohio State Univ. Research Foundation, Columbus, Ohio USA.

[32] Curtis, C. (1999). Loose and sketchy animation. In *Proc. Computer Graphics (ACM SIGGRAPH Sketches)*, pp. 317.

[33] Curtis, C., S. Anderson, J. Seims, K. Fleischer, and D. H. Salesin (1997). Computer-generated watercolor. In *Proc.* 24^{th} *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 421–430.

[34] Daniel, G. and M. Chen (2003, July). Visualising video sequences using direct volume rendering. In *Proc.* 1^{st} *Intl. Conference on Vision, Video and Graphics (VVG)*, pp. 103–110.

[35] Daniels, E. (1999). Deep canvas in disney's tarzan. In *Proc. Computer Graphics (ACM SIGGRAPH, Abstracts and Applications)*, pp. 200.

[36] de Jong, K. (1988). Learning with genetic algorithms. *Machine Learning 3*, 121–138.

[37] DeCarlo, D., A. Finkelstein, S. Rusinkiewicz, and A. Santella (2003, July). Suggestive contours for conveying shape. In *Proc.* 30^{th} *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 22, pp. 848–855.

[38] DeCarlo, D. and A. Santella (2002). Abstracted painterly renderings using eye-tracking data. In *Proc.* 29^{th} *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 769–776.

[39] Delignon, Y., A. Marzouki, and W. Pieczynski (1997). Estimation of generalized mixtures and its application in image segmentation. *IEEE Transactions on Image Processing 6*(10), 1364–1376.

[40] DeMenthon, D. (2002). Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Proc. Statistical Methods in Video Processing (SMVP) Workshop at ECCV*, Copenhagen, Denmark.

[41] Dempster, A. P., N. M. Laird, and D. B. Rubin (1997). Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society Series B 39*, 1–38.

[42] Deng, Y. and B. S. Manjunath (2001). Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 23*(8), 800–810.

[43] Deussen, O., S. Hiller, C. van Overveld, and T. Strothotte (2000). Floating points: A method for computing stipple drawings. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 19, pp. 41–50.

[44] Elber, G. (1998, January). Line art illustrations of parametric and implicit forms. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 1*(4).

[45] Elber, G. (1999, September). Interactive line art rendering of freeform surfaces. *Computer Graphics Forum 3*(18), 1–12.

[46] Fekete, J., E. Bizouarn, T. Galas, and F. Taillefer (1995). Tictactoon: A paperless system for professional 2D animation. In *Proc.* 22nd *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 79–90.

[47] Fischler, M. A. and R. C. Bolles (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*(6), 381–395.

[48] Fitzpatrick, K. (Ed.) (2002). *Art: An A-Z Guide to the World's Greatest Artists and their Paintings*. Flame Tree Publishing. ISBN: 1-903-81789-7.

[49] Fleck, M. M., D. A. Forsyth, and C. Bregler (1996). Finding naked people. In *Proc. European Conference on Computer Vision (ECCV)*, Volume 2, pp. 593–602.

[50] Fogel, D. B., K. Chellapilla, and P. J. Angeline (1999). Inductive reasoning and bounded rationality reconsidered. *IEEE Transactions on Evolutionary Computation 2*(3), 142–146.

[51] Foley, J., A. van Dam, S. Feiner, and J. Hughes (1992). *Computer Graphics* (2nd ed.). Reading, Massachusetts: Addison Wesley. ISBN: 0-201-84840-6.

[52] Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IEEE Transactions on the Electronic Computer EC-10*, 260–268.

[53] Fuchs, H., Z. M. Kedmen, and S. P. Uselton (1977). Optimal surface reconstruction from planar contours. *Communications of the ACM 10*(20), 693–702.

[54] Galvin, B., B. McCane, K. Novins, D. Mason, and S. Mills (1998). Recovering motion fields: An evaluation of eight optical flow algorithms. In *Proc.* 9th *British Machine Vision Conference (BMVC)*, Volume 1, pp. 195–204.

[55] Ganapathy, S. and T. G. Dennehey (1982). A new general triangulation method for planar contours. In *Proc.* 9th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 16, pp. 69–75.

[56] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley. ISBN: 0-201-15767-5.

[57] Gooch, A. (1998). Interactive non-photorealistic technical illustration. Master's thesis, Department of Computer Science, University of Utah.

[58] Gooch, B., G. Coombe, and P. Shirley (2002, June). Artistic vision: Painterly rendering using computer vision techniques. In *Proc.* 2nd *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 83–90.

[59] Gooch, G. and A. Gooch (2001, July). *Non-photorealistic rendering.* A. K. Peters, U.S.A. ISBN: 1-568-81133-0.

[60] Gray, H. and H. Carter (1994). *Gray's anatomy* (20th century ed.). London: Longmans, Green, and Co. ISBN: 1-859-58018-1.

[61] Green, S., D. Salesin, S. Schofield, A. Hertzmann, and P. Litwinowicz (1999). Non-photorealistic rendering. In *ACM SIGGRAPH '99 Non-Photorealistic Rendering Course Notes.*

[62] Haeberli, P. (1990). Paint by numbers: abstract image representations. In *Proc.* 17th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 4, pp. 207–214.

[63] Haggerty, M. (1991, November). Almost automatic computer painting. *IEEE Computer Graphics and Applications 11*(6), 11–12.

[64] Halasz, G. (1949). *Camera in Paris (quotation taken from pp.56).* Focal Press.

[65] Hall, P. (1999). Non-photorealistic rendering by Q–mapping. *Computer Graphics Forum 1*(18), 27–39.

[66] Hall, P. M., M. Owen, and J. P. Collomosse (2004). A trainable low-level feature detector. In *Proc. Intl. Conference on Pattern Recognition (ICPR).* To appear.

[67] Hanrahan, P. and P. Haeberli (1990, August). Direct WYSIWYG painting and texturing on 3D shapes. In *Proc.* 17th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 24, pp. 215–223.

[68] Harris, C. J. and M. Stephens (1988). A combined corner and edge detector. In *Proc.* 4th *Alvey Vision Conference*, Manchester, pp. 147–151.

[69] Harrison, H. (1996). *How to Paint and Draw: A complete course on practical and creative techniques.* Select Editions. ISBN: 1-840-38524-3.

[70] Heitkoetter, J. and D. Beasley (1995, March). comp.ai.genetic faq. USENET. http://www-2.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic.

[71] Hertzmann, A. (1998). Painterly rendering with curved brush strokes of multiple sizes. In *Proc.* 25th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 453–460.

[72] Hertzmann, A. (2001, May). Paint by relaxation TR2001–801. Technical report, New York University.

[73] Hertzmann, A. (2002, June). Fast paint texture. In *Proc.* 2nd *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 91–96.

[74] Hertzmann, A., C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin (2001). Image analogies. In *Proc.* 28th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 327–340.

[75] Hertzmann, A. and K. Perlin (2000). Painterly rendering for video and interaction. In *Proc.* 1st *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 7–12.

[76] Hicks, Y., P. M. Hall, and D. Marshall (2002). Tracking people in three dimensions using a hierarchical model of dynamics. *Image and Vision Computing 20*, 691–700.

[77] Holland, J. (1975). *Adaptation in Natural and Artificial Systems. An introductory analysis with applications to biology, control, and artificial intelligence* (1st ed.). Univ. Michigan Press. ISBN: 0-472-08460-7.

[78] Horowitz, S. L. and T. Pavlidis (1976). Picture segmentation by a tree traversal algorithm. *Journal of the ACM 23*, 368–388.

[79] Hough, P. V. C. (1962). Methods and means for recognizing complex patterns. U.S. Patent 3,069,654.

[80] Hsu, S. C. and I. H. H. Lee (1994). Drawing and animation using skeletal strokes. In *Proc.* 21st *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Orlando, USA, pp. 109–118.

[81] Hughes, R. (1991). *The Shock of the New. Art and the Century of Change*. Thames and Hudson. ISBN: 0-563-20906-2.

[82] Igarashi, T., S. Matsuoka, and H. Tanaka (1999, August). Teddy: A sketching interface for 3D freeform design. In *Proc.* 26th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 409–416.

[83] Illingworth, J. and J. Kitler (1987). The adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 5*(9), 690–697.

[84] Isard, M. and A. Blake (1996). Contour tracking by stochastic propagation of conditional density. In B. Buxton and R. Cipolla (Eds.), *Proc. European Conference on Computer Vision (ECCV)*, pp. 343–356.

[85] Isard, M. and A. Blake (1998). Condensation – conditional density propagation for visual tracking. *Intl. Journal of Computer Vision (IJCV) 29*(1), 5–28.

[86] Jähne, B. (1993). *Spatio-temporal image processing*, Volume 751 of *Lecture Notes in Computer Science*. Springer-Verlag. ISBN: 3-540-57418-2.

[87] Jaser, E., J. Kittler, and W. Christmas (2003, June). Building classifier ensembles for automatic sports classification. In R. F. Windeatt T (Ed.), *Proc.* 4th *Intl. Workshop on Multiple Clasifier Systems*, Volume 2709 of *Lecture Notes in Computer Science*, pp. 366–374. Springer-Verlag.

[88] Kakadiaris, I. A. and D. Metaxas (1995). 3D human body model acquisition from multiple views. In *Proc. Intl. Conference on Computer Vision (ICCV)*, pp. 618–623.

[89] Kakadiaris, I. A. and D. Metaxas (1996). Model based estimation of 3D human motion with occlusion based on active multi-viewpoint selection. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 81–87.

[90] Kalman, R. E. (1960, March). A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering 82*, 35–45.

[91] Kass, M., A. Witkin, and D. Terzopoulos (1987). Active contour models. *Intl. Journal of Computer Vision (IJCV) 1*(4), 321–331.

[92] Keppel, E. (1975, January). Approximating complex surfaces by traingulation of contour lines. *IBM Journal of Research and Development 19*, 1–96.

[93] Klein, A. W., P. J. Sloan, R. A. Colburn, A. Finkelstein, and M. F. Cohen (2001, May). Video cubism. Technical report, Microsoft Research MSR-TR-2001-45. http://research.microsoft.com/research/downloads/#video+cube.html.

[94] Knuth, D. (1998). *The Art of Computer Programming (Sorting and Searching)*, Volume 3. Addision-Wesley. ISBN: 0-201-89685-0.

[95] Koffka, K. (1935). *Principles of Gestalt Psychology*. New York: Harcourt Brace.

[96] Kovacs, L. and T. Sziranyi (2002). Creating video animations combining stochastic paintbrush transformation and motion detection. In *Proc.* 16th *Intl. Conference on Pattern Recognition (ICPR)*, Volume II, pp. 1090–1093.

[97] Lansdown, J. and S. Schofield (1995, May). Expressive rendering: a review of nonphotorealistic rendering techniques. *IEEE Computer Graphics and Applications 15*(3), 29–37.

[98] Lasseter, J. (1987, July). Principles of traditional animation applied to 3D computer animation. In *Proc.* 13th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 21, pp. 35–44.

[99] Lee, J. (1997). Physically-based modeling of brush painting. *Computer Networks and ISDN Systems 29*, 1571–1756.

[100] Leister, W. (1994, March). Computer generated copper plates. *Computer Graphics Forum 13*(1), 69–77.

[101] Li, H. and M. A. Lavin (1986). Fast hough transform: a hierarchical approach. *Journal on Graphical Models and Image Processing (CVGIP)* (36), 139–161.

[102] Li, Y., M. Gleicher, Y. Xu, and H. Shum (2003). Stylizing motion with drawings. In *Proc. ACM Symposium on Computer Animation*, pp. 309–319.

[103] Litwinowicz, P. (1997). Processing images and video for an impressionist effect. In *Proc.* 24th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Los Angeles, USA, pp. 407–414. A copy of this paper is included in Appendix C on DVD-ROM.

[104] Lu, L., X. Dai, and G. Hager (2004). A particle filter without dynamics for robust 3D face tracking. In *Proc. IEEE Workshop on Face Processing in Video (at CVPR)*.

[105] Lu, Z. (2001). *Perceptually Realistic Flower Generation*. Ph. D. thesis, Department of Computer Science, University of Bath, U.K.

[106] Maio, D. and D. Maltoni (2000, September). Real-time face location on gray-scale static images. *Pattern Recognition 33*(9), 1525–1539.

[107] Markosian, L., M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes (1997). Real-time nonphotorealistic rendering. In *Proc. Computer Graphics (ACM SIGGRAPH)*, Los Angeles, USA, pp. 415–420.

[108] Markosian, L. and J. D. Northrup (2000). Artistic silhouettes: a hybrid approach. In *Proc.* 1st *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 31–37.

[109] Marr, D. (1982). *Vision — A Computational Investigation into the Human Representation and Processing of Visual Information*. New York: Freeman. ISBN: 0-716-71284-9.

[110] Mehrabian, A. (1996). Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology: Developmental, Learning, Personality, Social 14*, 261–292.

[111] Meier, B. (1996). Painterly rendering for animation. In *Proc.* 23th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 447–484.

[112] Mohr, A. and M. Gleicher (2002). Hijackgl: Reconstructing from streams for stylized rendering. In *Proc.* 2nd *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 13–20.

[113] Moravec, H. (1980, September). Obstacle avoidance and navigation in the real world by a seeing robot rover CMU-RI-TR-3. Technical report, Carnegie-Mellon University, Robotics Institute.

[114] Nelder, J. and R. Mead (1965). A simplex method for function minimization. *Computer Journal 7*, 308–313.

[115] Ngo, J. and J. Marks (1993). Space-time constraints revisited. In *Proc.* 20th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 343–350.

[116] Nixon, M. S. and A. S. Aguado (2002). *Feature Extraction and Image Processing* (1st ed.). Reed Elsevier plc. ISBN: 0-750-65078-8.

[117] Ostromoukhov, V. (1999, August). Digital facial engraving. In *Proc.* 26th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 417–424.

[118] Pan, J.-Y. and C. Faloutsos (2002, July). Videocube: A novel tool for video mining and classification. In *Proc.* 5th *Intl. Conference on Asian Digital Libraries (ICADL)*, pp. 11–14.

[119] Parker, J. R. (1994). *Algorithms for Image Processing and Computer Vision*. Wiley and Sons, USA. ISBN: 0-471-14056-2.

[120] Perlin, K. and L. Velho (1995). Live paint: Painting with procedural multiscale textures. In *Proc.* 22nd *Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 153–160.

[121] Pham, B. (1991, January). Expressive brush strokes. *Journal on Graphical Models and Image Processing (CVGIP) 1*(53), 1–6.

[122] Praun, E., H. Hoppe, M. Webb, and A. Finkelstein (2001). Real-time hatching. In *Proc.* 28th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 581–586.

[123] Press, W. H. (1992). *Numerical Recipes in C: the art of scientific computing*. Cambridge University Press. ISBN: 0-521-43108-5.

[124] Pudet, T. (1994). Real time fitting of hand-sketched pressure brushstrokes. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 13, pp. 227–292.

[125] R-L. Hsu, M. Abdel-Mottaleb, A. K. J. (2002, May). Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 24*(5), 696–706.

[126] Rana, A. and A. Zalzala (1995). An evolutionary algorithm for collision free motion planning of multi-arm robots. In *Proc. Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 123–130.

[127] Rehrmann, V. (1994). *Stabile, echtzeitfähige Farbbildauswertung*. Ph. D. thesis, Dept. Computer Science, University of Koblenz, Germany.

[128] Reinhart, M. The TX-transform. World-wide Web.

[129] Reynolds, C. (1987). Flocks, herds, and schools: A distributed behavioural model. In *Proc.* 14th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 25–34.

[130] Rowntree, D. (2002). Personal correspondence.

[131] Rudolph, G. (1997). *Handbook of Evolutionary Computation*, Chapter Evolution Strategies, pp. B1.3:1–6. Oxford University Press. ISBN: 0-750-30392-1.

[132] Russell, J. A. (1997). Reading emotion from and into faces: Resurrecting a dimensional-contextual perspective. In J. A. Russel and J. M. Fernández-Dols (Eds.), *The Psychology of Facial Expression*, pp. 295–320. Cambridge University Press.

[133] Saito, T. and T. Takahashi (1990). Comprehensible rendering of 3-d shapes. In *Proc.* 17th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 24, pp. 197–206.

[134] Salisbury, M. P., S. E. Anderson, R. Barzel, and D. H. Salesin (1994). Interactive pen-and-ink illustration. In *Proc.* 21st *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Florida, USA, pp. 101–108.

[135] Salisbury, M. P., M. T. Wong, J. F. Hughes, and D. H. Salesin (1997). Orientable textures for image-based pen-and-ink illustration. In *Proc.* 24th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Los Angeles, USA, pp. 401–406.

[136] Samaria, F. and A. Harter (1994, December). Parameterisation of a stochastic model for human face identification. In *Proc.* 2nd *IEEE Workshop on Applications of Computer Vision*, Sarasota FL.

[137] Scholkopf, B., A. Smola, and K. Muller (1998). Non-linear component analysis as a kernel eigenvalue problem. *Neural Computation 10*(5), 1299–1319.

[138] Schumann, J., T. Strothotte, and S. Laser (1996, April). Assessing the effect of non-photorealistic rendered images in CAD. In *Proc. Human Factors in Computer Systems (CHI)*. http://www.acm.org/sigchi/chi96/proceedings/papers/Schumann/chi96fi.html.

[139] Seitz, S. (2003, July). Frontiers in 3D photography: Reflectance and motion. In *Proc.* 1st *Intl. Conference on Vision, Video and Graphics (VVG)*, pp. 7.

[140] Shiraishi, M. and Y. Yamaguchi (2000). An algorithm for automatic painterly rendering based on local source image approximation. In *Proc.* 1st *ACM Symposium on Non-photorealistic Animation and Rendering*, pp. 53–58.

[141] Sims, K. (1994). Evolving virtual creatures. In *Proc.* 21st *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 15–22.

[142] Small, D. (1991, February). Simulating watercolor by modeling diffusion, pigment and paper fibres. In *Proc. SPIE*, pp. 70–76.

[143] Smith, A. R. (1984). Plants, fractals and formal languages. In *Proc.* 11th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 1–10.

[144] Smith, S. M. and J. M. Brady (1997, May). SUSAN - a new approach to low level image processing. *Intl. Journal of Computer Vision (IJCV) 1*(23), 45–78.

[145] Sonka, M., V. Hlavac, and R. Boyle (1999). *Image Processing, Analysis, and Machine Vision* (2nd ed.). P.W.S. Publishing. ISBN: 0-534-95393-X.

[146] Sousa, M. C. and J. W. Buchanan (1999a). Computer-generated graphite pencil rendering of 3D polygonal models. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 3, pp. 195–208.

[147] Sousa, M. C. and J. W. Buchanan (1999b, June). Observational models of blenders and erasers in computer-generated pencil rendering. In *Proc. Graphics Interface*, pp. 157–166.

[148] Sousa, M. C. and J. W. Buchanan (2000, March). Observational models of graphite pencil materials. *Computer Graphics Forum 1*(19), 27–49.

[149] Sousa, M. C. and P. Prusinkiewicz (2003, September). A few good lines: Suggestive drawing of 3D models. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 22, pp. 381–390.

[150] Strassmann, S. (1986). Hairy brushes. In *Proc.* 13$^{\text{th}}$ *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Volume 20, pp. 225–232.

[151] Strothotte, T., B. Preim, A. Raab, J. Schumann, and D. R. Forsey (1994). How to render frames and influence people. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 13, Oslo, Norway, pp. C455–C466.

[152] (Student), W. S. G. (1908). On the probable error of a mean. *Biometrika 1*(6).

[153] Sutherland, I. (1963). SKETCHPAD: A man-machine graphics communication system. In *Proc. AFIPS Spring Joint Computer Conference*, pp. 329–346.

[154] Szeliski, R. (1994). Image mosaicing for tele-reality applications. Technical report, Digital Equipment Corporation.

[155] Sziranyi, T. and Z. Tath (2000). Random paintbrush transformation. In *Proc.* 15$^{\text{th}}$ *Intl. Conference on Pattern Recognition (ICPR)*, Volume 3, Barcelona, pp. 155–158.

[156] Takagi, S., M. Nakajima, and I. Fujishiro (1999, October). Volumetric modeling of colored pencil drawing. In *Proc.* 7$^{\text{th}}$ *Pacific Conference on Computer Graphics and Applications*, Seoul, Korea, pp. 250.

[157] Tang, W. and T. R. Wan (2002, June). Intelligent self-learning characters for computer games. In *Proc.* 20$^{\text{th}}$ *Eurographics UK Conference*, pp. 51–58.

[158] Torr, P. H. S. (1995). *Motion segmentation and outlier detection*. Ph. D. thesis, University of Oxford.

[159] Treavett, S. and M. Chen (1997, March). Statistical techniques for the automated synthesis of non-photorealistic images. In *Proc.* 15$^{\text{th}}$ *Eurographics UK Conference*, pp. 201–210.

[160] Tu, X. and D. Terzopoulos (1994). Artificial fishes: Physics, locomotion, perception, behaviour. In *Proc.* 21$^{\text{st}}$ *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 43–50.

[161] van Bakergem, W. D. and G. Obata (1991). Free hand plotting: Is it life or is it digital? *CAAD-Futures*, 567–582.

[162] Veryovka, O. and J. Buchanan (1999, September). Comprehensive halftoning of 3D scenes. *Computer Graphics Forum 3*(18), 13–22.

[163] Walker, K. N., T. F. Cootes, and C. J. Taylor (1998). Locating salient object features. In *Proc.* 9th *British Machine Vision Conference (BMVC)*, Volume 2, pp. 557–567.

[164] Wang, J.-P. (1994). Stochastic relaxation on partitions with connected components and its application to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 20*(6), 619–636.

[165] Watt, A. H. (1999). *3D Computer Graphics* (3rd ed.). ISBN: 0-201-39855-9.

[166] Whitted, T. (1983, July). Anti-aliased line drawing using brush extrusion. In *Proc.* 10th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Detroit, Michigan, pp. 151–156.

[167] Williams, D. and M. Shah (1992). A fast algorithm for active contours and curvature estimation. *Journal on Graphical Models and Image Processing (CVGIP) 55*(1), 14–26.

[168] Williams, L. (1983). Pyramidal parametrics. In *Proc.* 20th *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 1–11.

[169] Williams, R. (2001). *The Animator's Survival Kit.* Faber and Faber Ltd. ISBN: 0-571-21268-9.

[170] Winkenbach, G. and D. H. Salesin (1994). Computer-generated pen-and-ink illustration. In *Proc.* 21st *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Orlando, USA, pp. 91–100.

[171] Winkenbach, G. and D. H. Salesin (1996, August). Rendering parametric surfaces in pen and ink. In *Proc.* 23rd *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 469–476.

[172] Wong, E. and N. Weisstein (1984). Flicker induces depth: Spatial and temporal factors in the perceptual segregation of flickering and nonflickering regions in depth. *Perception & Psycophysics 35*(3), 229–236.

[173] Woodring, J. and H. Shen (2003). Chronovolumes: A direct rendering technique for visualizing tie varying data. In I. Fujishiro and K. Mueller (Eds.), *Proc. Volume Graphics.*

[174] Wu, X. (1992, October). Colour quantization by dynamic programming and principal analysis. *ACM Transactions on Graphics 4*(11), 348–372.

[175] Wyszecki, G. and W. S. Stiles (1982). *Color Science: Concepts and Methods, Quantitative Data and Formulae* (2nd ed.). John Wiley and Sons. ISBN: 0-471-02106-7.

[176] Xu, L., E. Oja, and P. Kultanen (1990). A new curve detection method: Randomised hough transform. *Pattern Recognition Letters* (11), 331–338.

[177] Xu, S., F. C. M. Lau, F. Tang, and Y. Pan (2003, September). Advanced design for a realistic virtual brush. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 3, Grenada, Spain, pp. 533–542.

[178] Xu, S., F. Tang, F. C. M. Lau, and Y. Pan (2002, September). A solid model based virtual hairy brush. In *Proc. Computer Graphics Forum (Eurographics)*, Volume 21, pp. 299–308.

[179] Zakaria, M. N. (2001). Interactive evolutionary approach to character animation. In *Proc. Winter School of Computer Graphics (WSCG)*.

[180] Zeleznik, R. C., K. P. Hemdon, and J. F. Hughes (1996). Sketch: An interface for sketching 3D scenes. In *Proc.* 23rd *Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, pp. 163–170.

[181] Zhang, H., A. Kankanhalli, and S. W. Smoliar (1993, June). Automatic partitioning of full-motion video. In *Proc. ACM Multimedia Systems*, Volume 1, pp. 10–28.

[182] Zhang, Y. Y. and C. Y. Suen (1985). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM 3*, 236–239.

[183] Zhu, Z. and Q. Ji (2004). Real time 3D face pose tracking from an uncalibrated camera. In *Proc. IEEE Workshop on Face Processing in Video (at CVPR)*.