Video Based Dynamic Scene Analysis and Multi-style Abstraction

TAO, Chenjun

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Thesis/Assessment Committee

Abstract of thesis entitled:
Video Based Dynamic Scene Analysis and Multi-style Abstraction
Submitted by TAO, Chenjun
for the degree of Master of Philosophy
at The Chinese University of Hong Kong in September 2008

The researches on video-based analysis and processing concentrate on presenting high-quality visual information and enhancing the visual effects. When playing high-resolution videos on mobile devices, a common, though seldom addressed problem is how to naturally condense the source video and optimally fit it into the target size with lower spatial resolution. Rendering images with an artistic style has been studied actively in non-photorealistic rendering. Rendering an image with multiple styles is still a challenge. Our objective in multi-style rendering is to transform digital images into renderings that approximate the appearance of multi-style artwork, which incorporates two or more traditional visual medias.

In video retargeting, the retargeted video should contain objects of interest and perceptibly seamless and natural. We propose a systematic approach to address the problem by automatically computing an active window set inside the input videos. Our method contributes by deriving an optimization process to compute the active pixels in videos as well as a density map, which jointly constrains the generation of the retargeted video. Our approach is general, and capable of handling videos with complex foreground motions. In multi-style image/video rendering, we develop the novel approach aimed at re-rendering real images with selective styles automatically. The rendering styles (e.g. oil-painting, watercolor) are determined by the properties of captured images. In order to avoid the over-rendering phenomena of small but important objects, layers are detected from the improved image auto-segmentation algorithm. A diffusion filter is finally employed to eliminate noise and abstract the details.

Further, we extend the work to the interactive abstract video with multiple styles (e.g., oil-painting, watercolor, chalk, ink, brush-based oil-painting, wooden) generated from a single input image. The abstract video is created by employing linear/non-linear (radial basis functions, spline, bezier) functions on different layers. Our approach aims to re-render real-world images to abstract images/videos in non-photorealistic styles selected by the user. Our work in video retargeting can be applied in window-oriented mobile devices. Rendering of abstract images/videos can be further accelerated using advanced graphics cards. The video-processing tools we have developed in dynamic and abstract medias have a wide range of applications in interactive computer games, non-photorealistic rendering, artistic desing/editing, and augmented reality systems.

摘要

目前，對於視頻信號中的動態場景的研究主要集中於高質量的視覺信息表達和視覺效果的加強兩個方面。而在圖像的非真實感繪制領域中，將圖像渲染成爲具有某種藝術風格畫作的研究也十分重要。我們的工作有兩個方面。其一，建立視頻重定位的系統模型。當在螢幕尺寸有限的手機等設備上播放高解析度的視頻時，一個普遍但較少研究的問題就是如何能夠在不失真的條件下壓縮視頻信號，使其可以最優化的方式在低解析度的設備上播放。經過重定位的視頻信號應該保留了原視頻資訊的主要物體，並且這些物體之間的過度應該自然無縫。其二，當前，將一幅圖像繪製成爲同時具有多種藝術風格的畫作的研究課題仍然具有挑戰性。而我們的目標是將這些記錄真實物體的數字圖像繪製成爲具有多種傳統藝術風格的畫作。

在關於視頻信號重定位的研究中，我們提出了一套系統的方法，可以自動地計算出截取窗口在每一個輸入幀上的位置和角度。我們這套的方法的最大貢獻是將優化演算法融入到計算信息點和深度地圖當中，並且貫穿於整個視頻信號重定位的過程中。我們提出的這套方法具有普遍性，並且可以應用於大量具有復雜運動軌跡的前景的視頻信號的分析中。

在圖像、視頻的多種藝術風格的繪制的研究中，我們開發了一種新的方法來解決將自動選取的藝術風格繪制於真實圖像上的問題。適當的藝術風格，例如油畫效果，水彩畫效果等，可以通過對拍攝到的圖像進行分析而進行選取。爲了有效的避免將一些重要的細小物體過度渲染，我們採用了改進的自動圖像分割技術對輸入圖像進行分層。並根據不同的圖層的性質決定繪制的程度。我們還應用了擴散濾波器去除噪聲和一些不重要的細節。我們還將這套圖像的繪制方法應用於通過一張圖像交互地產生同時具有多種藝術風格的視頻信號上。這些藝術風格可以是油畫，水彩，粉筆畫，水墨畫，基於筆刷式的油畫，以及木刻等風格上。風格化了的視頻是通過在不同的藝術畫作間，對不同的圖層線性和非線性插值得到的。我們的方法主要應用於將圖像繪製成具有多種非真實感的藝術風格的畫作或是視頻。用戶可以參與藝術風格的選取。

我們設計的視頻重定位的系統模型可以被廣泛的應用於基於視窗的移動設備上。而圖像、視頻的多種藝術風格的繪制系統則可以進一波那個通過 GPU 進行加速。我們提供的視頻風格化的繪制具在互動式遊戲，非真實感繪制，藝術設計、編輯，和增強現實等系統中可以有廣泛的應用。

Acknowledgements

I would like to thank my supervisors, Prof. H. Sun and Prof. J. Jia, for suggesting this interesting research topic, providing the initial direction to the solution and helping and encouraging me throughout this work.

I want to thank my friends in the work group, Mr. Zhang Fan, Mr. Zhao Chong, Mr. Sheng bin, Miss Xu Leilei, Miss Qin Guiming, Mr. Xiong Wei, Mr. Shan Qi, Mr. Chung Hinshun, Miss LIU Renting, Mr. Xu Li, Mr. Li Zhaorong and Mr. Chen Jianing for giving me all those valuable suggestions. Special thanks for Dr. Shen Jianbing for helping me to finalize the idea. It was really a great experience and pleasure to work with you all.

Finally, I want to thank my family for supporting me. Special thanks to Dr. Wong Chiwing, Mr. Dai Hongning, Dr. Shum Wingho and Dr. Li Shan.

This work is dedicated to all of my dear friends for encouraging me throughout the process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Researches in image/video based dynamic scene analysis and its applications have a long history in computer graphics and computer vision communities, concentrating on the high-quality presentation of visual information and the improved video quality on visual effects. Related topics in image/video based dynamic scene analysis include video summary, video tracking, video completion, and video stabilization. Non-photorealistic rendering, active researches in computer graphics, focuses on enabling a variety of expressive styles for digital media arts. Advanced techniques on abstraction rendering are the kernel topics in this area. In this chapter, we begin with the introduction of window-oriented video retargeting and related approaches. The topics of abstraction rendering and the objectives of our work are presented thereafter.

1.1 Window-oriented Retargeting

With the rapid growth of image/video capturing ability, it is easier and more common to capture videos with high resolution. Sharing and playing these videos on popular mobile devices is handicapped by a set of factors, one of which is the limited screen size in most of the devices. There are several methods proposed to address this ubiquitous *video migration* problem. Simply resizing the video to

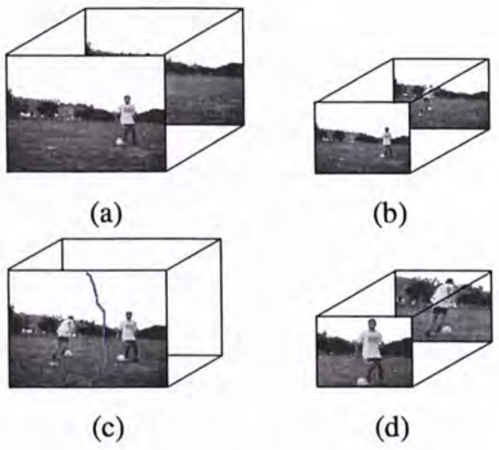


Figure 1.1: Video migration. (a) The input video. (b) The directly scaled down video. Details of the player and the football are lost. (c) The temporal compression may cause ambiguity when the objects in different frames are placed together. Two footballs appear at the same time. (d) Our method fits a set of windows by tracking most informative pixels in different frames.

fit the small screen will sacrifice most of the details. Figure 1.1 shows an example of video migration methods. In the original video, players run after a football (a). However, directly scaling the video results in the loss of most details as shown in (b). In (c), directly compressing the temporal frames by moving the interested objects in two different frames into a single one may cause large perceptual ambiguities, such as two football players appearing at the same time. In (d), when watching a video, the user is always more interested in the actions or movement of the foreground objects, and is less sensitive to the background scene. Our method fits a set of windows by tracking most informative pixels in different frames.

The problem of video migration can be regarded as one kind of the video summarization in terms of using smaller spatio-temporal space to *summarize* the original videos. However, while conventional approaches shorten the input videos in order to generate temporal segment abstracts [55, 20, 26], our approach generates a seamless video clip by satisfying the following two requirements:

1. The migrated video should naturally preserve both the temporal and the spatial distance to be faithful to the original video.
2. The migrated video should also contain as much useful information as possible in terms of object shapes and motion.

Recently, a video retargeting system was proposed [44] to produce a retargeted video to suit the target display with minimal loss of the important information. This system proved to perform well on various kinds of movies; however, it had some limitations which can be avoided by our approach. First, their judgment on the important information is based on the results of low-level feature contrast, face detection, and dominant motion direction. Therefore, if the features and faces cannot be detected well, (e.g., most of the players in sport games do not face the camera directly), the locations of the target windows will be ambiguous. Second, virtual pans and virtual cuts are utilized to make the optimization, so the orientations of the target windows are restricted to zero. In order to contain more important information on the target display without producing the ambiguity of the relative positions of the objects, our method allows the target windows to change their directions smoothly within a small range.

In our work, we introduce an automatic approach to solve the general dynamic video migration problem by optimizing an *active window* in each of the frames containing most informative pixels. There are three steps in our optimization: foreground extraction, the initialization, and dynamic active windows computation. Specifically, we propose to robustly separate the foreground objects from either static or smoothly moving background by minimizing an energy function. To avoid local minima, we introduce a clustering technique to initialize the windows, which is neatly formulated as solving a labeling problem.

1.2 Abstraction Rendering

Researches for abstracting images in styles include one-style abstraction per image and multi-style abstraction per image. The approach of rendering an image with a single style is dated to the 1990s or even earlier. For example, the technique established by [28, 58] is for generating abstract images by enhancing the visual comprehensibility. Since then, a great deal of research on simulating rendering styles, such as oil, watercolor, tempera, ink, and pencil, has been presented. Some have the computer simulating an artist by re-rendering the image in oil style [27] either automatically [31] by referencing its real counterpart, or semi-automatically [15]. Types of pen strokes are simulated in advance. The path of a stroke is determined by the gradients of the original image for automatic rendering, or by users for semi-automatic rendering.

Adobe® Photoshop is one of the best e-painting platforms, supplying a great deal of tools and rendering filters. Other systems of this type [19, 29, 51] potentially offer the artist an efficient and effective method of non-photorealistic rendering. Style-based rendering, abstracting images in a single style, focuses on simulating a certain style of artistic rendering. Such rendering is based on physical properties of real painting, e.g. the transmittance of the canvas, the properties of painting materials, optics theory, and so on. The system is usually developed for people learning to paint especially children, who enjoy drawing and learning about artistic techniques. The system presented in [73] aims at generating abstracted frames in real time, where color quantization and bilateral filters are employed as well as GPU acceleration, to produce 15 frames per second. In [6], watercolor style is applied to the input video. The purpose of the system is to render each frame in watercolor style, which must be coherent along the time domain. The system performs shape abstraction utilizing the watercolor technique of suggesting details with abstract washes of color.

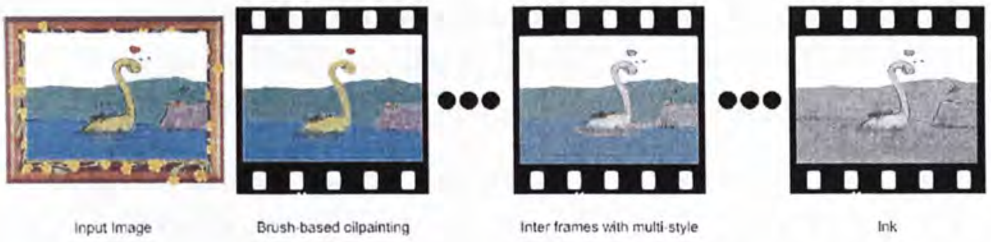


Figure 1.2: Interactive Video System using multi-style abstract images. The input image *(left)* is rendered in two styles (the first frame is rendered with brush-based oil-painting and the last frame is rendered with ink style). The intermediate frames of the video are generated by linear/nonlinear interpolations based on the grouped layers segmented by our system.

The other direction in abstraction rendering is to assign multiple styles to the single input image, where more than one traditional visual style such as oil, watercolor, or wooden is incorporated. The argument for multi-style rendering has been addressed in [21, 8, 56]. Ideas for creating dynamics within a painting, emphasizing scene features and contrasting multiple features become possible in reality. Rendering multi-style on a single image has a long and distinguished history. Recently, computer systems are developed to render multiple styles on a single image, for example in [9, 37].

Previously, most work are more focused on presenting abstraction rendering either one style or multi-style in a single image. It is subjective to specify the rendering styles to the parts of an input image. In our work, we present a novel approach for multi-style abstraction rendering. It renders the captured images based on color transaction, and pays more attention on rendering images with selective styles while preserving important information at the same time. We develop the interactive framework for rendering multi-style abstract videos. Our approach offers more flexibilities for rendering abstract images/videos, with the following unique features:

- The input image is rendered with multiple abstract styles, with the layer information automatically processed, in an unified framework.

- Multi-style abstract videos are created from the input image by employing linear/nonlinear interpolation functions, to achieve the smooth transition of the styles selected by users.
- Users interact and control layer-based styles parameters in image/video abstractions.

An abstract video example is shown in Figure 1.2, where the original image is first abstracted to two rendering styles, brush-based oil-painting and ink. The intermediate frames of the abstract video are created by linear/nonlinear interpolation functions based on the layer information automatically processed in our system.

1.3 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we outline the state-of-art technologies in image/video processing and non-photorealistic rendering. In Chapter 3, we present a systematic approach to address the video retargeting problem. The novel approach for multi-style abstract image rendering is described in Chapter 4, and presents the interactive framework for generating multi-style abstract videos is presented in Chapter 5. Lastly, conclusions are drawn in Chapter 6.

□ **End of chapter.**

Chapter 2

Related Work

Video-based dynamic scene analysis concentrates on high quality presentation of visual information and the improved video quality, based on the information obtained from groups of frames. For the pair-wise approach, hidden information is detected from two subsequent frames. Differences between related frames are used to infer the dynamics occurring in the video [61], help to extract the main object from an image sequence [69, 41], recognize human actions [53, 74, 22, 76, 18, 52, 39, 49] or result in video compression, i.e. MPEG. With the local information obtained, the approach focuses more on efficiency, and is generally used in real-time systems.

Alternatively, spatio-temporal volume [4] pays more attention to the global information by examining the video slice-based (2D) and volume-based (3D) [16] with 3-axes (u,v,t), where (u,v) specifies the pixel's position in the image and t stores time information of the video sequence. This has recently been found useful on motion analysis and motion-based video segmentation, camera work analysis and so on. Epipolar plane image analysis (EPI) is a particular case using spatio-temporal volume analysis with some constraints on camera motion path and its orientation, useful on geometric recovery of static scene structure and extracting depth information. Systems developed based on the spatio-temporal volume approach generally emphasize more the effectiveness, where special global information or effects are needed, such as video mosaics [54], detecting irregularities in video [49], human motion recognition [60] and video completion [71].

In the following, the common techniques on video-based dynamic scene analysis are reviewed in detail.

2.1 Video Migration

The problem of *video migration* is addressed by in [23, 70, 44], which need to extract important partitions from less important content using an important model. In [23, 70], the background movement is not taken as a factor, and their cropping methods may produce the ambiguity of the relative positions of the extracted objects. [44] utilizes face detection in the important model. Our system focuses on more general videos, and the orientations of the active windows can be adapted to the content of the videos.

The approaches for video summarization can be classified into two categories: key-frame-based and event-based approaches. The key-frame-based approaches [20, 48, 46, 45, 26, 77, 34, 25] typically select key-frames from the input video, and put them in the results in a chronological order. It is convenient for the user to quickly browse the video and catch all necessary information on distinct video segments rapidly.

The event-based approaches [36, 55, 13] present video abstractions by actions, such as [55] and [13], to combine several events happening in different frames into a shorter video sequence or stitched frames to achieve a better understanding of video content. These methods are effective in generating informative video abstraction. However, they are not suitable for our *video migration* problem. As the temporal frames or space distances are sampled, the motion details are lost and spatial ambiguities are caused when the summarization is played alone on mobile devices.

Our method utilizes motion separation to acquire necessarily extracted foreground, so we have reviewed most related previous work on multiple motion layers separation in videos. [64, 63] estimate the static background by modelling it with the Gaussian mixture model. In [38], stereo video sequences are required to build the background model in real time to robustly subtract background. In [65] and [17], assuming no large background motion, the foreground moving objects are segmented out from the monocular video sequences by considering the difference of gradient and color. [67] employed the optical flow to estimate the camera motion, and used a polynomial equation to model the motion. Assuming that the movements of foreground objects are independent and rigid, the presented method has difficulty tackling the problem on sports videos where players do not always move rigidly.

For the purpose of multiple-target-tracking in hockey games, [11] builds a standard hockey rink to locate the shot and eliminate the camera motion. This method can estimate the players if the stadium map is given precisely. However, this method doesn't work well if there are few cues that can be extracted from the dynamic scene to locate the shot to the prefabricated map. The algorithm in [72] is a two-step approach. First, motions of objects are estimated using feature points. Then, labels of objects are assigned and refined based on their motions and positions. This method produces good results for motion segmentation when sufficient feature points are obtained.

2.2 Video Synopsis

The power of video over still images stems from its ability to represent dynamic activities. However, with the rapid growth of video capturing ability, video browsing and retrieval has become inconvenient due to inherent spatio-temporal redundancies, where some time intervals may have no activity or have activities that occur in a small image region. Video synopsis, or video summary, aims to provide a compact video representation, while preserving the essential activities of the original video.

In [1], the authors present dynamic video synopsis, where most of the activity in the video is condensed by simultaneously showing several actions, even when they originally occurred at different times. For example, people can create a stroboscopic movie, where multiple dynamic instances of a moving object are played simultaneously.

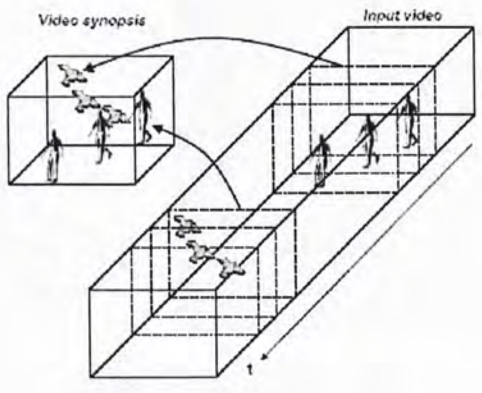


Figure 2.1: The input video shows a walking person, and after a period of inactivity displays a flying bird. A compact video synopsis can be produced by playing the bird and the person simultaneously.

Figure 2.1 shows the flowchart of the algorithm. An energy function including the part of loss in activity and the one of discontinuity across seams is well defined at first. By minimizing this energy function, a synopsis video is supposed. The experimental results illustrate well that the presented system would produce nice synopsis of the input video with various contents. However, since only the loss in activity and the discontinuity across seams are taken into consideration, the energy function cannot guarantee to generate an acceptable video synopsis when local properties such as avoiding visible seams are required.

In order to solve this problem, objects are introduced, also called "object-based synopsis." The factor of discontinuity across seams is divided into two parts: one is a penalty for occlusions between objects, and the other is the length of the synopsis, which can be predetermined or dynamically determined for lossless video synopsis. The improved method can achieve good results on video synopsis. Figure 2.2 provides one example of good results.

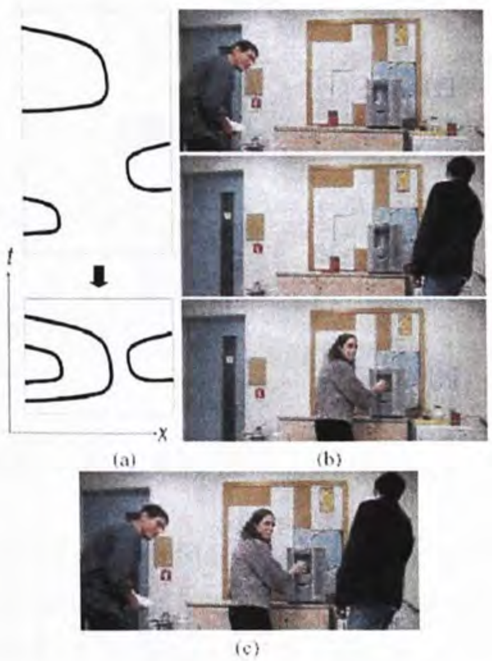


Figure 2.2: An example when a short synopsis can describe a longer sequence with no loss of activity and without the stroboscopic effect. Three objects can be time shifted to play simultaneously. (a) The schematic space-time diagram of the original video (top) and the video synopsis (bottom). (b) Three frames from original video. (c) One frame from the synopsis video.

A similar approach is presented in [33] (Figure 2.3). The problem of space-time video montage consists of three sub-problems: finding informative video portions, layer segmentation of informative video portions, and packing them in an output video volume. The informative video portions from the long input video sequence are detected as the saliency map of the input video at first. With the saliency map, where each of the video pixels is assigned a saliency value, the presented system will establish a saliency volume which

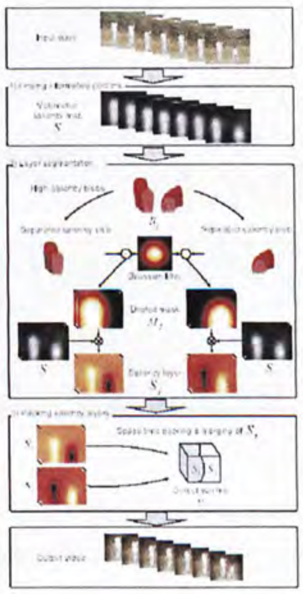


Figure 2.3: Overview of the space-time video montage.

is associated with the input video volume. The saliency volume contains a number of isolated informative portions where high saliency values are assigned. The idea of this system is to produce saliency layers to separate those informative portions. The saliency layers $S = \{S_j : j = 1, \cdots, n\}$ are extracted from the original saliency volume, where n is the number of layers. Here the notation S_j is used to represent the *j*th layer. The procedure of packing salient video portions into an output video volume is equal to making the total saliency value grow to its maximum, which can be viewed as a variant of the Knapsack problem with the following differences: input items are video volumes, each of which can have a larger volume than the output volume; every video pixel in the video volumes is associated with its importance; input items can overlap each other.

Denoting the output video volume as v_o and the associated saliency volume as S_o, serves to pack the input video volume into the output video volume v_o in such a way that S_0 contains maximal saliency. It is equivalent to **finding** the optimal space-time translations x_j of the

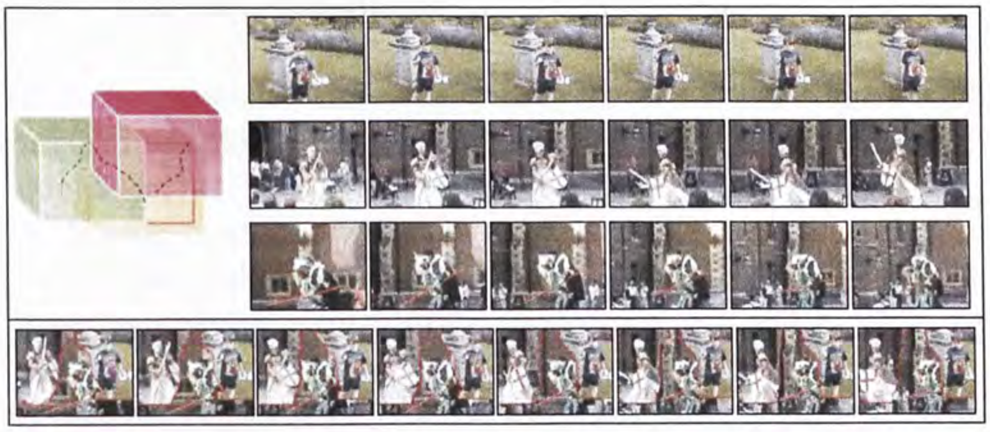


Figure 2.4: Result of fusing three different video clips. The top three rows show several frames from the input videos. The bottom row shows the video montage result.

saliency layers S_j which maximizes the following objective function:

$$\sum_{p \in S_o} f(S_j(p - x_j)), \tag{2.1}$$

where $f(\cdot)$ is the function which evaluates the saliency value for each pixel $p = (x, y, t)^T$. For instance, $f(\cdot)$ can be **defined** as $f(\cdot) = max_j(\cdot)$ which takes the maximum saliency value at a pixel where the saliency layers are overlapped. Since the saliency layers are bounded by the original input video volume, it follows $S_j(x) = 0$ if $x \notin S_j$. Once the positions x_j are determined, the color values of the output video v_o are assigned by composing the input video according to the arrangement of the saliency layers. In the case of $f(\cdot) = max_j(\cdot)$, for instance, by denoting $V(p)$ to represent the color value at the pixel p in the input video volume, a simple form of the composition can be described as

$$V_o(p) = \{V(p - x_j) : j = argmax_j(S_j(p - x_j))\}, \tag{2.2}$$

One of the results can be found in Figure 2.4.

2.3 Periodic Motion

Periodic motion is a vital cue in video repairing [35] and video tracking and matching. But, periodic motion itself is a hard task to accomplish [14].

In order to match two scenes captured by different cameras at different time and places with the same objects, spatio-temporal volume, including the time information, is created and a volumetric sequence pyramid is built. A SSD objective function is used to optimize the sequence brightness error. Two kinds of cases are discussed, which are affine transformation (without scale) and projective transformation (with scale). The results demonstrate that the efficiency of this method is good.

[40] tries to solve the problem by using Fourier Transform. First, two stabilized images, $V_{t,t-\tau}$ and $V_{t,t+\tau}$, are produced by images $I_{t-\tau}$, I_t and $I_{t+\tau}$ (the image later or before 300ms). Then the motion image can be obtained by comparing the image I_t and stabilized images $V_{t,t-\tau}, V_{t,t+\tau}$. In the motion image, the connected components are merged into objects, which are the input data for calculating the similarity plot. Auto-correlation of the similarity plot for regular textures gives out prominent peaks. By comparing these with two basic planar lattices, the periodic motion can be detected. This method is also suitable for the cases on object classification, counting people, estimating human stride and so on, as demonstrated by experiments.

2.4 Video Tracking

Video tracking is the process of locating a moving object (or several ones) in time using a camera. Such algorithms analyze the video frames and output the location of moving targets within the video frame.

The main difficulty in video tracking is to associate target locations in consecutive video frames, especially when the objects are moving fast and the frame rate is low. Video tracking systems usually employ a motion model which describes how the image of the target might change for different possible motions of the object.

In [11], a robust multi-target tracking algorithm within the application of hockey players tracking is presented. First, they built a target dynamics model by letting the computer learn the characteristics of hockey players from a database.Then by employing the Adaboost to give the initial estimations of players, mean-shift embedded particle filter is used to track various hockey players. In [12], a Bayesian Network is used to trace the objects. Both of the methods need well defined objects, obtained by machine learning or user interaction.

2.5 Video Stabilization

Video stabilization is an important video enhancement technology which aims at removing annoying shaky motion from videos. [75] proposed a practical and robust approach of video stabilization that produces full-frame stabilized videos with good visual quality. While most previous methods end up producing low resolution stabilized videos, their completion method can produce full-frame videos by naturally filling in missing image parts by locally aligning image data of neighboring frames. To achieve this, motion inpainting is proposed to enforce spatial and temporal consistency of the completion in both static and dynamic image areas. In addition, image quality in the stabilized video is enhanced with a new practical deblurring algorithm. Instead of estimating point spread functions, the proposed method transfers and interpolates sharper image pixels of neighbouring frames to increase the sharpness of the frame. The proposed video completion and deblurring methods enabled them to develop a complete video stabilizer which can naturally keep the original image quality in the stabilized videos. The effectiveness of their method is confirmed by extensive experiments over a wide variety of videos.

There are three steps in their algorithm: motion estimation and smoothing, video completion with motion inpainting, and image deblurring. Motion estimation and smoothing is needed to remove camera motions. Hierarchical model-based motion estimation [2], is employed to estimate the global motion. Unlike traditional methods to smooth out the transformation chain along the video, the authors directly compute the transformation S from a frame to the corresponding motion-compensated frame using only the neighboring transformation matrices. The indices of neighboring frames are denoted as $N_t = \{j | t - k \leq j \leq t + k\}$. If the frame I_t is located at the origin, aligned with the major axes, the position of each neighboring frame I_s relative to frame I_t can be calculated by the local displacement T_t^s . The correcting transformation S from the original frame I_t to the motion-compensated frame I_t' can be sought according to

$$S_t = \sum_{i \in N_t} T_t^i \star G(k), \tag{2.3}$$

where T_t^i denotes the coordinate transform from frame i to t. $G(k) = \frac{1}{\sqrt{2\pi}\sigma} e^{-k^2/2\sigma^2}$ is a Gaussian kernel, and the $\star$ operator represents convolution, and $\sigma = \sqrt{k}$ is used. Using the obtained matrices $S_0, \cdots S_t$, the original video frames can be warped to the motion-compensated video frames by

$$I_t'(p_t') \leftarrow I_t(S_t p_t) \tag{2.4}$$

After removal of undesired motion, missing image areas should be filled by using video completion methods. In their algorithm, motion inpainting is proposed to propagate the motion field into the missing image areas where local motion cannot be directly computed. The underlying assumption is that the local motion in the missing image areas is similar to that of adjoining image areas. The flow chart of the algorithm is illustrated in Figure 2.5. First, the local motion from the neighboring frame is estimated over the common

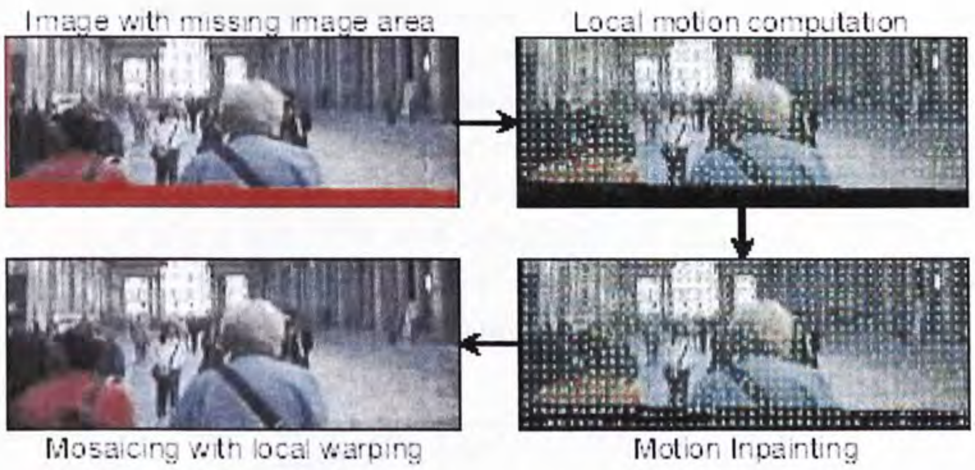


Figure 2.5: Video completion. Local motion is first computed between the current frame and a neighboring frame. Computed local motion is then propagated with motion inpainting method. The propagated motion is finally used to locally adjust mosaics.

coverage image area. The local motion field is then propagated into missing image areas. Note that unlike prior image inpainting work, this propagates local motion rather than color. Finally, the propagated local motion is used as a guide to locally warp image mosaics to achieve smooth stitching of the mosaics. Let M_t be the missing pixels, or undefined image pixels, in the frame I_t. They wish to complete M_t for every frame t while maintaining visually plausible video quality.

After stabilization, motion blur that is not associated with the new motion of the video becomes a noticeable noise that needs to be removed. It is usually difficult to obtain accurate PSFs from a free-motion camera; therefore, image deblurring using deconvolution is unsuitable for this case. In order to sharpen blurry frames without using PSFs, the authors developed a new interpolation-based deblurring method. The key idea of their method is transferring sharper image pixels from neighboring frames to corresponding blurry image pixels.

Their method first evaluates the "relative blurriness" of the image which represents how much of the high frequency component has been removed from the frame in comparison to the neighboring frames. They use the inverse of the sum of squared gradient measure to evaluate the relative blurriness, because of its robustness to image alignment error and computational efficiency. By denoting two derivative filters along the x- and y-directions by f_x and f_y respectively, the blurriness measure is defined by

$$b_t = \frac{1}{\sum_{p_t}\{((f_x \star I_t)(p_t))^2 + ((f_y \star I_t)(p_t))^2\}} \tag{2.5}$$

This blurriness measure does not give an absolute evaluation of image blurriness, but yields relative image blurriness among similar images. Therefore, they restrict the measure to be used in a limited number of neighboring frames where significant scene change is not observed. Also, the blurriness is computed using a common coverage area which is observed in all neighboring frames. Relatively blurry frames are determined by examining $b_t/b_{t'}$; $t' \in N_t$, e.g., when $b_t/b_{t'}$ is larger than 1, frame $I_{t'}$ is considered to be sharper than frame I_t.

Once relative blurriness is determined, blurry frames are sharpened by transferring and interpolating corresponding pixels from sharper frames. To reduce reliance on pixels where a moving object is observed, a weight factor which is computed by a pixel-wise alignment error $E^t_{t'}$ from $I_{t'}$ to I_t is used:

$$E^t_{t'} = |I_{t'}(T^{t'}_t p_t) - I_t(p_t)| \tag{2.6}$$

High alignment error is caused by either moving objects or error in the global transformation. Using the inverse of pixel-wise alignment error E as a weight factor for the interpolation, blurry pixels are replaced by interpolating sharper pixels. The deblurring can be described by

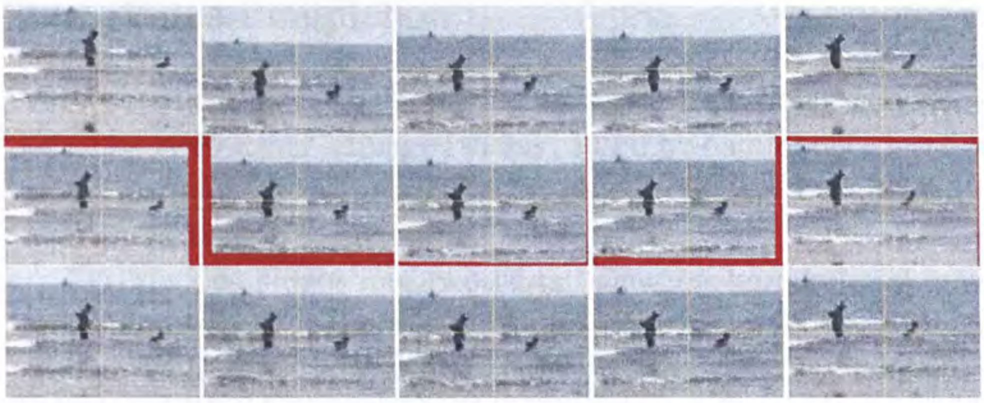


Figure 2.6: Result of video stabilization. Top row: Original input sequence. Middle row: stabilized sequence which still has missing image areas; Bottom row: stabilized and completed sequence. The grid is overlaid for better visualization.

$$\hat{I}_t(p_t) = \frac{I_t(p_t) + \sum_{t' \in N} \omega^t_{t'}(p_t) I_{t'}(T^{t'}_t p_t)}{1 + \sum_{t' \in N} \omega^t_{t'}(p_t)} \tag{2.7}$$

where ω is the weight factor which consists of the pixel-wise alignment error $E^t_{t'}$ and relative blurriness $b_t/b_{t'}$, expressed as

$$\omega^t_{t'}(p_t) = \begin{cases} 0, & if \frac{b_t}{b_{t'}} < 1, \\ \frac{b_t}{b_{t'}} \frac{\alpha}{E^t_{t'}(p_t)+\alpha}, & otherwise, \end{cases} \tag{2.8}$$

$\alpha \in [0, \infty]$ controls the sensitivity on the alignment error, e.g., by increasing α, the alignment error contributes less to the weight. As it is seen in the weighting factor defined in Eq. 2.8, the interpolation uses only frames which are sharper than the current frame.

In Figure 2.5, the result of video stabilization and completion is shown. The top row shows the original input images, and the stabilized result is in the middle row which contains a significant amount of missing image areas. The missing image areas are naturally filled in with the video completion method as shown in the bottom row.

2.6 Video Completion

Video completion is the repairing of space-time holes in videos. [35, 71] try to repair damaged video caused by natural deterioration of old celluloid films, or deliberate object removal by interactive segmentation tool, as mentioned before.

[35] divides videos into two parts: static background and moving foreground. For static background repairing, a few key frames are chosen to specify layer boundaries by mean-shift tracking algorithm, and mosaics are repaired by image repairing methods. In order to reduce boundary artifacts, misregistration among frames and to achieve better temporal coherence, homography blending is used. For moving foreground repairing, moving objects are detected first. Then, using a stationary camera and enforcing motion periodicity constraints, a video clip containing one or more cycles of the periodic motion, named movel, is chosen to fill up the holes in the moving foreground after the operations of wrapping, regularization, normalization and alignment.

[71] presents a method for repairing the large space-time holes in video sequences of complex dynamic scenes in a different way by posing the problem as a global optimization problem with a well-defined objective function. The theory is that every missed patch can be represented by other related, exiting patches. So a 3D space-temporal volume including two front frames and two back frames is established for the damaged key frame. As all the patches are classified into several categories, and the value of the missed patch is updated by the mean value of the main cluster which the missed patch belongs to. But the results shown are for very low resolution videos, the inpainted static background was different one frame from another, creating a ghost effect. Significant over-smoothing is observed as well.

□ **End of chapter.**

Chapter 3

Active Window Oriented Video Retargeting

In this chapter, a systematic approach to address the video retargeting problem is proposed, where an active window set with the predefined size is computed automatically. Our method contributes in deriving an optimization process to compute the active pixels in video as well as a density map, which jointly constrains the generation of the retargeted video. To avoid the possible local minima, we employ a robust background subtraction method to eliminate unnecessary pixels and apply clustering in initialization. Our approach is general, and capable of handling videos with complex foreground motions.

3.1 System Model

Aiming at generating spatially resized videos that contain the most informative regions naturally and seamlessly, our approach consists of the following three steps. An overview of our method is shown in Figure 3.1 .

Foreground extraction is responsible for extracting the moving objects from the background scene. It is noted that extracting moving foreground from dynamic scene has long been a great challenge in computer vision, especially when there are multiple moving ob-

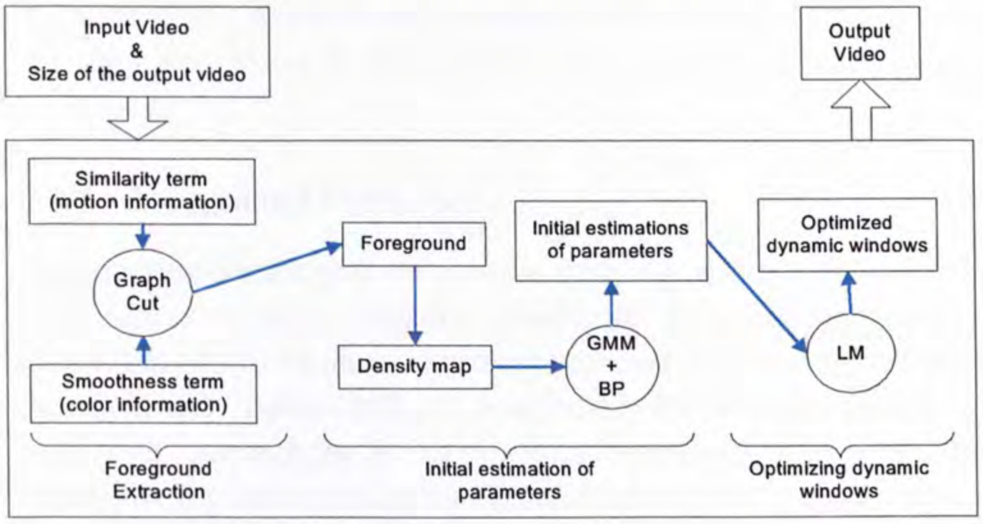

Input Video
&
Size of the output video
Output Video
Similarity term (motion information)
Graph Cut
Smoothness term (color information)
Foreground
Density map
Initial estimations of parameters
GMM + BP
Optimized dynamic windows
LM
Foreground Extraction
Initial estimation of parameters
Optimizing dynamic windows

Figure 3.1: Overview of the proposed approach.

jects. In our approach, we group motions of pixels in a frame into several bivariate normal distributions. The background is subtracted by considering both motion and color information for each pixel.

Optimizing active windows. We search and optimize the active windows with any **predefined** size in each frame from the input videos. Note that in our method, there is no need to label each foreground object and track their positions in each frame. As required, the active windows only need to contain foreground pixels as dense as possible. In our method, using the separated foreground pixels, we construct the active windows using the label masks from density maps.

Initialization of parameters. The optimization algorithm introduced above depends largely on the initial values due to the large space of optimization. We initialize parameters using cluster optimization. Each cluster represents a region of dense active pixels. We formulate the initiation as a labelling problem and solve it using Belief Propagation.

We describe the above three steps of our approach in the following subsections respectively. In the rest of this chapter, we represent the color as a vector $\mathbf{I}_p^t$ in the RGB color channels, for each pixel p in frame t.

3.1.1 Foreground Extraction

In extracting the foreground objects, applying methods to detect and track separated object motions considering the object overlapping, occlusion, and dynamic background movement can hardly get satisfactory results. Specifically, as described in the previous section, we only need to search the active pixels in foreground objects. In other words, we are interested in background subtraction in each frame.

In our approach, we propose to minimize an energy function to automatically separate foreground and background pixels in each frame. We use label $x_p = 0$ to represent that the pixel p is a background pixel, while $x_p = 1$ means that the pixel is in the foreground. To analyze motions, we initially apply the optical flow method presented in [10] to obtain the estimated motion vectors for all pixels in the video. Appendix B gives more details on optical flow. Let $\mathbf{f}_p^t$ be the 2-D optical flow vector of each pixel p in frame t. We group all the $\mathbf{f}$ in each frame into k bivariate normal distributions. Each distribution N_i^t, where $0 \leq i < k$, has the mean μ_i^t, variance Σ_i^t, and the prior weight ω_i^t.

Our energy function $E_b(X)$ is defined as

$$E_b(X) = \sum_p E_{b1}(x_p) + \lambda_1 \sum_{p,q \in N(p)} E_{b2}(x_p, x_q), \tag{3.1}$$

where $X = \{x_p\}$ is the labelling of variables, E_{b2} is a smoothness term and E_{b1} is the probability of a pixel being foreground or background, $N(\cdot)$ denotes the set of neighborhood.

Similarity term. It is noted that the previous work on background subtraction or interactive image and video segmentation [57, 68, 42] model the background and/or foreground colors using a set of clusters. In our approach, the Gaussian distributions model the motions of all pixels, without knowing which are in the foreground or background.

The motion of each pixel p has different probabilities g_p^i falling into different Gaussians i constructed below:

$$g_p^i = \omega_i^t N(\mathbf{f}_p^t; \mu_i^t, \Sigma_i^t). \tag{3.2}$$

To compute the probability that a Gaussian distribution models a background motion, we assume in general that the background scene has smooth motion mostly due to the camera movement, which implies that the background should consist of a majority of the pixels with smaller motion variances compared to the foreground objects. Considering Gaussian distributions above, a Gaussian cluster modelling background distribution should have a large weight ω_i^t and small variance $||\Sigma_i^t||^{\frac{1}{2}}$. Thus, we formulate the probability that the ith Gaussian distribution models the motions of background pixels as

$$D_b^t(i) = \tau_i^t \omega_i^t / ||\Sigma_i^t||^{\frac{1}{2}}, \tag{3.3}$$

where $\tau_i^t = (\max_i \omega_i^t / ||\Sigma_i^t||^{\frac{1}{2}})^{-1}$ is the normalization term. Similarly, we define the probability that the motions of foreground pixels are modelled by the ith Gaussians as

$$D_f^t(i) = 1 - \tau_i^t \omega_i^t / ||\Sigma_i^t||^{\frac{1}{2}}. \tag{3.4}$$

Combining the probability distribution that one pixel is in different Gaussians, we compute the sum of background and foreground motion confidence

$$O_b^t(i) = \sum_{0 \le i < k} D_b^t(i) g_p^i, \tag{3.5}$$

and

$$O_f^t(i) = \sum_{0 \le i < k} D_f^t(i) g_p^i, \tag{3.6}$$

respectively. Given Eqn. 3.5 and 3.6, the similarity energy term for the labelling of each pixel p can be written as

$$\begin{cases} E_{b1}(x_p = 0) & = \frac{O_b^t}{O_f^t + O_b^t} \\ E_{b1}(x_p = 1) & = \frac{O_f^t}{O_f^t + O_b^t} \end{cases} \tag{3.7}$$

Smoothness term. Considering the support from the neighboring pixels, we introduce a smoothness term to impose penalty on discontinuities between neighboring pixels:

$$E_{b2}(x_p, x_q) = |x_p - x_q| f(p, q), \tag{3.8}$$

where

$$f(p, q) = \frac{1}{\alpha \|\mathbf{f}_p^t - \mathbf{f}_q^t\| + \|\mathbf{I}_p^t - \mathbf{I}_q^t\| + 1}, \tag{3.9}$$

where α is a weight. Eqn. 3.9 constrains that if both the color of neighboring two pixels and their optical flow vectors are similar, the penalty of label difference of p and q is large.

Given the above energy definitions, we compute the optimal segmentation using the Graph Cut method [7] where the pixels labelling 0 are considered as the background, while the pixels labelling 1 are the active pixels. We use M^t to denote the label map in each frame t.

We compared our experimental results in foreground extraction with those proposed in [63] and [72] in Figure 3.2. Given an input video containing moving crabs and dynamic background beach with similar colors, without any user interaction to select the foreground or the background samples, our method can successfully extract the foreground objects without including many background pixels as shown in (d). The approaches proposed in [63] and [72] cannot handle videos with quite sparse features and dynamic background, and thus do not work well in our experiments, as shown in (b) and (c). Note that our foreground subtraction is the first step in our system. Without explicitly estimating the camera motion, our method has larger error tolerance than simple combination of background subtraction and video stabilization.

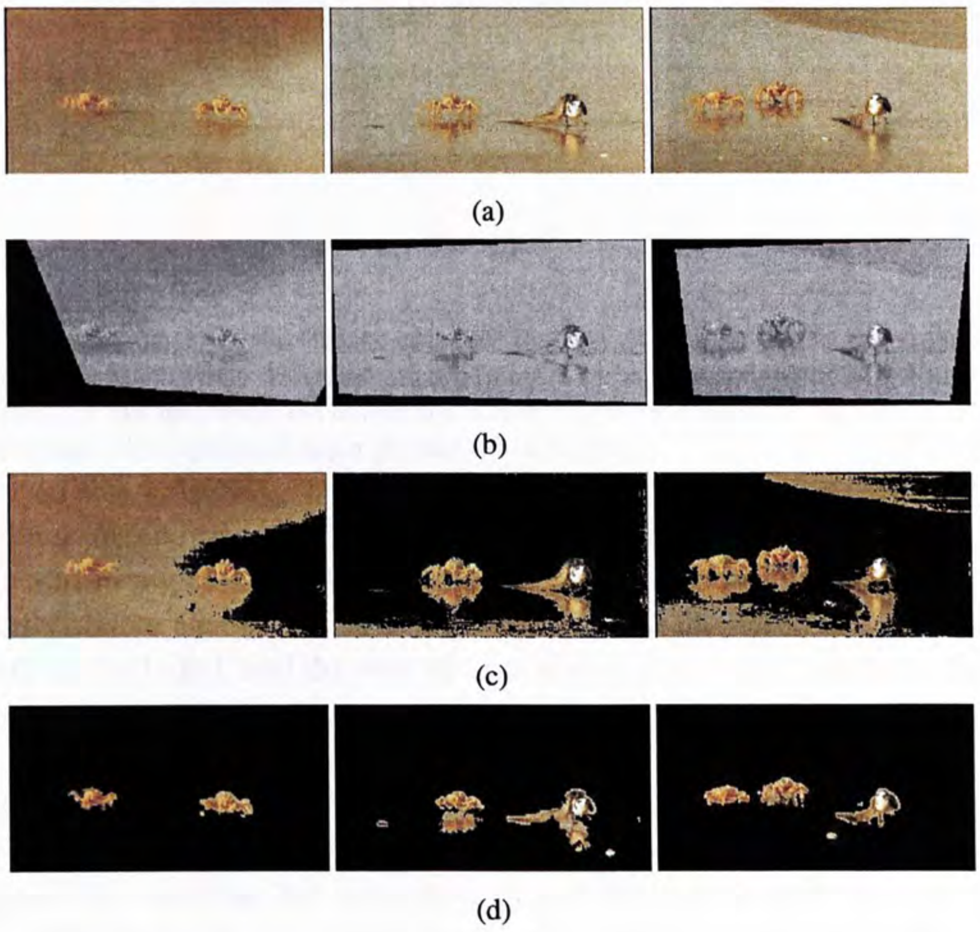


Figure 3.2: Foreground extraction comparison (shown in color). (a) shows a few frames from a video clip where two crabs are moving on the beach. Note that the color of the crabs is quite similar to the background. (b) the extracted background using the approach presented in [72]. The foreground is not extracted out since there are no sufficient feature points. (c) is the extracted foreground using the method presented in [63]. Since the backgrounds are dynamic in this example, no good background model can be established in their method. (d) shows our result. Although the foreground is similar to the background in color, our method can still successfully extract the foreground automatically.

3.1.2 Optimizing Active Windows

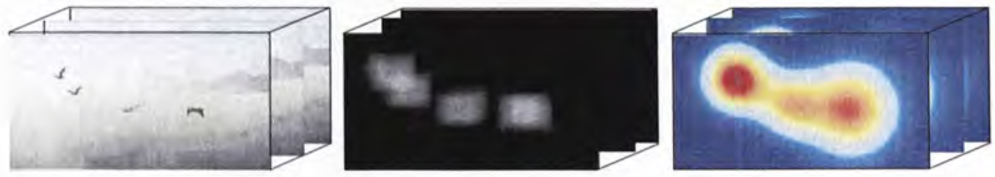


Figure 3.3: GMM on the density maps. (a) shows a set of input frames in a video. (b) is the density map computed in each frame. The background is subtracted, and thus, has low densities. We cluster the density maps into GMMs as shown in (c). Warmer color represents larger probability in GMMs.

In this section, given the extracted foreground layer containing active pixels, we optimize the active windows in the input videos to fit the target size. We describe two optimization terms, i.e., the informative energy E_{f1}, which guarantees that the dense active pixels are included, and the smoothness energy E_{f2}, which encourages temporal continuity.

E_{f1} requires that in a general video migration framework, the active window in each frame should contain most informative pixels. Obviously, it is not computationally feasible to greedily search all possible positions. We estimate it by constructing density maps.

We **defined** the parameter vector of each active window as $\mathbf{W}^t = [W^t_x, W^t_y, W^t_\theta]^T$, representing the window center in x and y coordinates, and the orientation θ of the window at frame t respectively. The window width w and height h are **predefined** values.

We denote the number of the active pixels included in each possible active window $\mathbf{W}^t$ as $d(x, y, \theta)$ in each frame. So, basically $d(x, y, \theta)$ is a function regarding all possible variables x, y and θ. If we assume $\theta = 0$, $d(x, y, 0)$ can be computed by constructing a corresponding density map using convolution

$$\begin{aligned} d(x,y,0) &= \sum_{i=x-\frac{h}{2}}^{x+\frac{h}{2}} \sum_{j=y-\frac{w}{2}}^{y+\frac{w}{2}} M(i,j) \\ &= M \otimes f, \end{aligned} \tag{3.10}$$

where f is a mean filter with size exactly the same as the active widow and M is the label map as defined in the smoothness term in Section 3.1.1. If the $\theta \neq 0$, we sample θ using a scale of $\frac{\pi}{36}$ and constrain $-\pi/6 < \theta < \pi/6$ to avoid large rotation. For each θ, we construct a new M_θ rotated on the original label map M. Then the density map $d(x,y,\theta)$ can be computed similarly in each frame as

$$d(x,y,\theta) = \sum_{i=x-\frac{h}{2}}^{x+\frac{h}{2}} \sum_{j=y-\frac{w}{2}}^{y+\frac{w}{2}} M_\theta(i,j).$$

Note that the density of the pixels around the border of each frame will be set to zero. Given the density maps computed in all frames, the energy $E^t_{f1}(x,y,\theta)$ is defined as

$$E^t_{f1}(x,y,\theta) = \frac{1}{d^t(x,y,\theta)+\epsilon}, \tag{3.11}$$

constraining that all active windows include the most dense active pixels in the original video, where ϵ is a small number.

The smoothness constraint E_{f2} requires that the center and orientation of the windows $\mathbf{W}$ crossing the frames should be similar. So we define

$$E_{f2} = |\frac{\partial^2 W_x}{\partial t^2}| + |\frac{\partial^2 W_y}{\partial t^2}| + |\frac{\partial^2 W_\theta}{\partial t^2}| \tag{3.12}$$

Combining the above two terms, we minimize

$$\begin{aligned} E_f(x,y,\theta) &= \sum_t (E^t_{f1}(x,y,\theta) + \lambda_2 E^t_{f2}(x,y,\theta)) \\ &= \sum_t (\lambda_2(|\frac{\partial^2 W_x}{\partial t^2}| + |\frac{\partial^2 W_y}{\partial t^2}| + |\frac{\partial^2 W_\theta}{\partial t^2}|) \\ &+ \frac{1}{d^t(x,y,\theta)+\epsilon}), \end{aligned} \tag{3.13}$$

by using the nonlinear Levenberg-Marquardt minimization method to iteratively optimize the parameters.

3.1.3 Initialization

In the optimization process described above, there is a large set of parameters to be estimated, which makes the optimization easily stuck in a local minimum. Thus, a good initialization of parameters $[W_x^{t(0)}, W_y^{t(0)}, W_\theta^{t(0)}]^T$ is necessary in our approach to produce a satisfactory migrated video.

In this section, we introduce a robust initialization method by first clustering the density map d using a Gaussian-Mixture model (GMM) in each frame. The corresponding EM is performed in 3-D including the 2-D image plane and an additional 1-D density value for all pixels. The output mean vector $\mu_\mathrm{i}^\mathrm{t} = [\bar{x}_i^t, \bar{y}_i^t, \bar{d}_i^t]^T$ for each Gaussian cluster G_i^t in frame t. $(\bar{x}_i^t, \bar{y}_i^t)$ is the coordinate in the image plane and $\bar{d}_i^t$ is the mean density value. The square root of the principle diagonal of the covariance matrix consists of the standard deviations $\sigma(x_i^t)$, $\sigma(y_i^t)$, and $\sigma(d_i^t)$. Figure 3.3 shows that the density maps are clustered into Gaussian clusters according to the density of the active pixels.

We then construct a single-chain graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in the input video by representing each GMM G^t in the video as one node in the vertex set $\mathcal{V}$. The nodes in immediately neighboring frames are temporally connected using edges $\mathcal{E}$, as shown in Fig. 3.4(c).

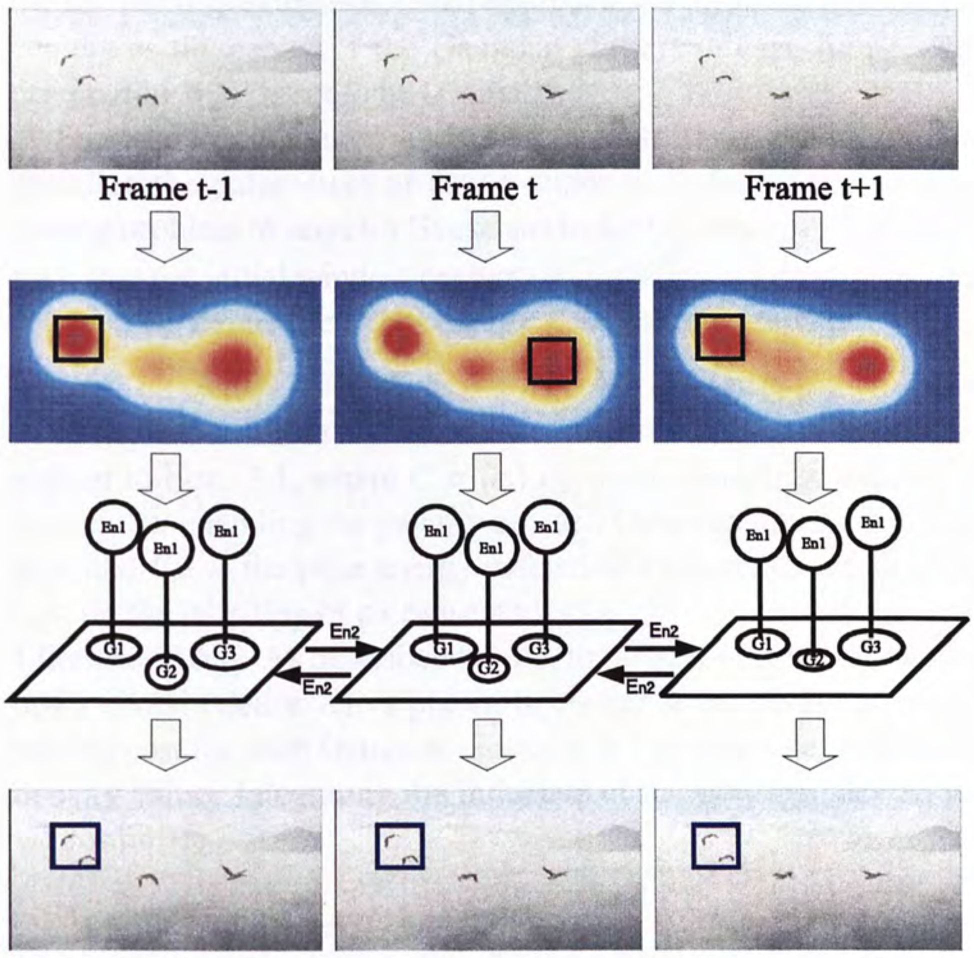


Figure 3.4: Initialization in our method. The first row shows a few consecutive frames. The Second row is the Gaussian clusters computed on the density maps. The constructed single-chain graph on all GMMs in the video is shown in the third row. Each node is a GMM in one frame. The fourth row shows the initialized windows considering smoothness and active pixel density after the Belief Propagation.

In the initialization, the active windows should be placed inside the clusters with large density means. Meanwhile, the centers of the initial windows in consecutive frames should be close to preserve the temporal smoothness. In our approach, we initialize the window centers as the center of the Gaussian clusters in each frame. The orientation $W_\theta^{t(0)}$ is set to be 0 initially.

Suppose that there are K clusters in each frame, the problem to initialize the parameters of active windows is formulated as a labelling problem to search a Gaussian cluster $G_{c_t}^t$, where $c_t \in \{1, 2, ..., K\}$, such that the initial window centers $(W_x^{t(0)}, W_y^{t(0)}) = (x_{c_t}^t, y_{c_t}^t)$. In what follows, we define the Gibbs energy $E_n(C)$ on the graph $\mathcal{G}$

$$E_n(C) = \sum_{c_t \in \mathcal{V}} E_{n1}(c_t) + \lambda_3 \sum_{(c_t, c_{t'}) \in \mathcal{E}} E_{n2}(c_t, c_{t'}), \qquad (3.14)$$

similar to Eqn. 3.1, where $C = \{c_t\}$,E_{n1} is the likelihood defined on each node, encoding the penalty on each Gaussian cluster individually, and E_{n2} is the prior energy defined on each edge, encoding the cost on the labelling of pairwise nodes.

Likelihood E_{n1}. As described before, to make the initial active windows contain dense active pixels, in cluster level, we assign the labelling cost for each Gaussian cluster $c_t = i$ proportional to its mean density value. Integrating the influence of the Gaussian deviations, we formulate

$$E_{n1}(c_t = i) = \frac{1}{\phi \bar{d}_i^t} \sqrt{\frac{\sigma_n^2(d_i^t)}{\sigma_n^2(x_i^t) + \sigma_n^2(y_i^t) + \varepsilon}}, \qquad (3.15)$$

where ε is a weight, $\sigma_n^2(d_i^t)$ is to impose larger penalty if the Gaussian cluster has large density variance, and $1/(\sigma_n^2(x_i^t) + \sigma_n^2(y_i^t))$ makes region variance large after initialization, leaving sufficient space in active windows. $\phi = \sum_i \frac{1}{\bar{d}_i^t} \sqrt{\frac{\sigma_n^2(d_i^t)}{\sigma_n^2(x_i^t)+\sigma_n^2(y_i^t)+\varepsilon}}$ is a normalization term.

Prior E_{n2}. Considering the temporally connected nodes, E_{n2} encodes the smoothness constraint

$$E_{n2}(c_t, c_{t'}) = \sqrt{(\bar{x}_{c_t}^t - \bar{x}_{c_{t'}}^{t'})^2 + (\bar{y}_{c_t}^t - \bar{y}_{c_{t'}}^{t'})^2}. \qquad (3.16)$$

Given the defined energies, the problem of minimizing $E_n(G)$ is solved using Belief Propagation [50] in an iterative message passing process. We show one example in Fig. 3.4, where a few consecutive frames are input (a), which are clustered using GMMs in each frame as shown in (b). The corresponding graph constructed in our method is shown in row (c). By solving the optimization problem, we robustly compute the initial active windows as illustrated using the rectangles in (d). It is noted that if we do not consider the temporal smoothness, the initial window parameters will only consider densities, which causes large window jump spatially in the consecutive frames as shown in the rectangle in (b). More details on belief propagation are given in Appendix C.

One may be concerned that our active window optimization over time is similar to the Bayesian filtering which is commonly applied to smoothing the video sequence. In our approach, Belief Propagation is employed to optimize window position given several discrete values. No smoothing or prediction is necessary in this step.

3.2 Experiments

We present our video retargeting examples here. In our experiments, the parameters λ_1, λ_2, and λ_3 are fixed and set to be 10, 0.1 and 10 respectively. $\epsilon = 0.001$ in Eqn. 3.11 and $\varepsilon = 0.001$ in Eqn. 3.14.

Ice hockey game video. We demon our results as one example of the ice hockey game in Figure 3.5. Several frames from the input video are shown in (a) where the players are scattered in the scene. With the defined small window, it is impossible to include all players. We highlight in (a) and (b) our computed active windows using blue rectangles given two different window sizes. In both cases, the windows are optimized to include most informative pixels of the moving athletes. (c) shows the output from our foreground extraction where the corresponding density map is computed in (d). (e) is a side-by-side comparison using the window size defined in (a). The

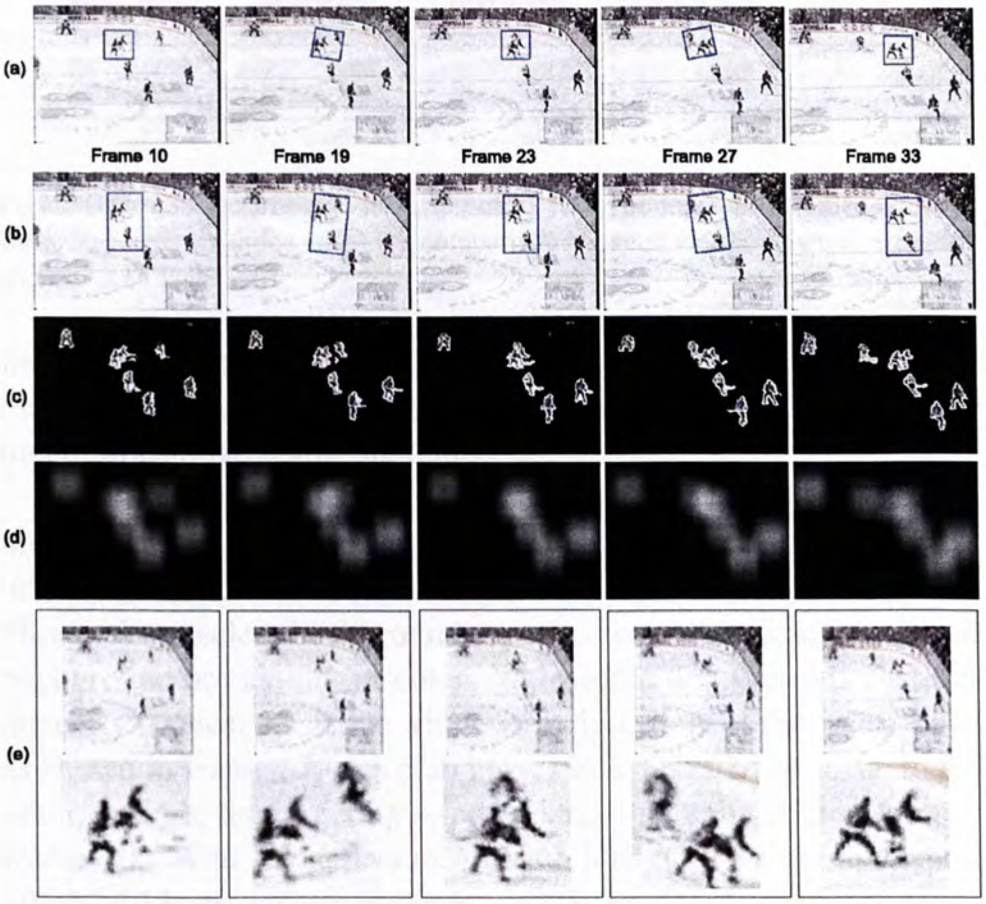


Figure 3.5: Ice hockey game example. (a) The input key-frames with the computed active windows outlined in blue. (b) Using a different size, our method can also produce an optimal output outlined in blue. (c) The extracted foreground pixels. (d) The corresponding density map. (e) Comparative results of directly resized video and our approach.

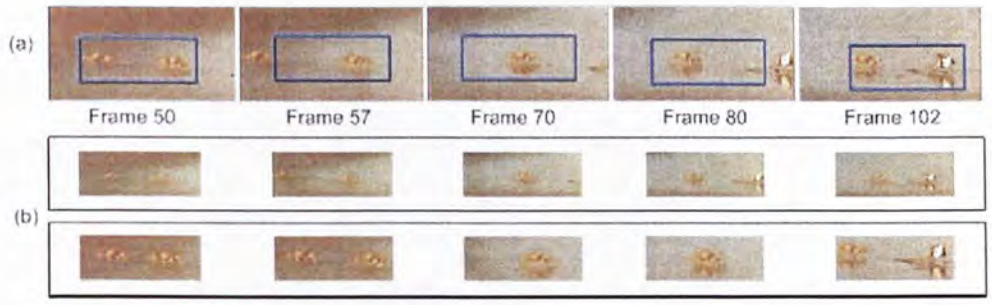


Figure 3.6: Crab example (shown in color). (a) The input key-frames with the computed active windows. (b) the comparison between the directly resized video and our approach.

first row illustrates the result from directly resizing the whole video, most details are lost. Our results are shown in the second row, the most informative pixels are included.

Crab example. In Figure 3.6, a video sequence shows two crabs moving on the beach. This video-retargeting example illustrates that our method can automatically search foreground motions while eliminating useless background regions, even when foreground and background are similar in color. We assume $\theta = 0$ in this case. In the early frames, the active windows include both of the crabs. Then as shown in frame 57, one crab moves in a direction opposite to the other, making the targeting window smoothly shift to the center in frame 70. With the appearance of the bird on the right in Frames 80 and 102, the targeting window shifts to include the informative foreground bird. (b) shows the comparative results of our method and directly resizing the video.

Figure skating example. One example of two figure skating athletes on the ice is shown in Figure 3.7. Though the motions of the players and their relative positions are changing, our method makes the active windows tracing the two players automatically. At first, the orientation of the active window is zero, as they are close enough to be displayed in the active window. From frame 65, as one of them is jumping, and the other is moving away, the orientation of the

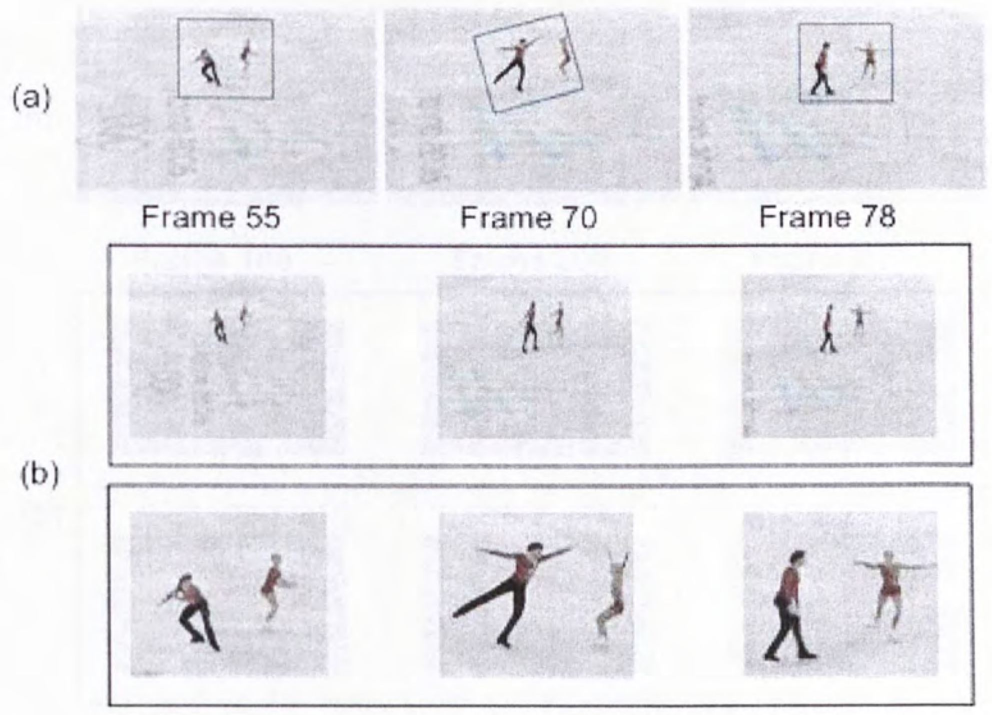


Figure 3.7: Figure skating example (shown in color). (a) The input key-frames with the computed active windows. (b) the comparison between the directly resized video and our approach.

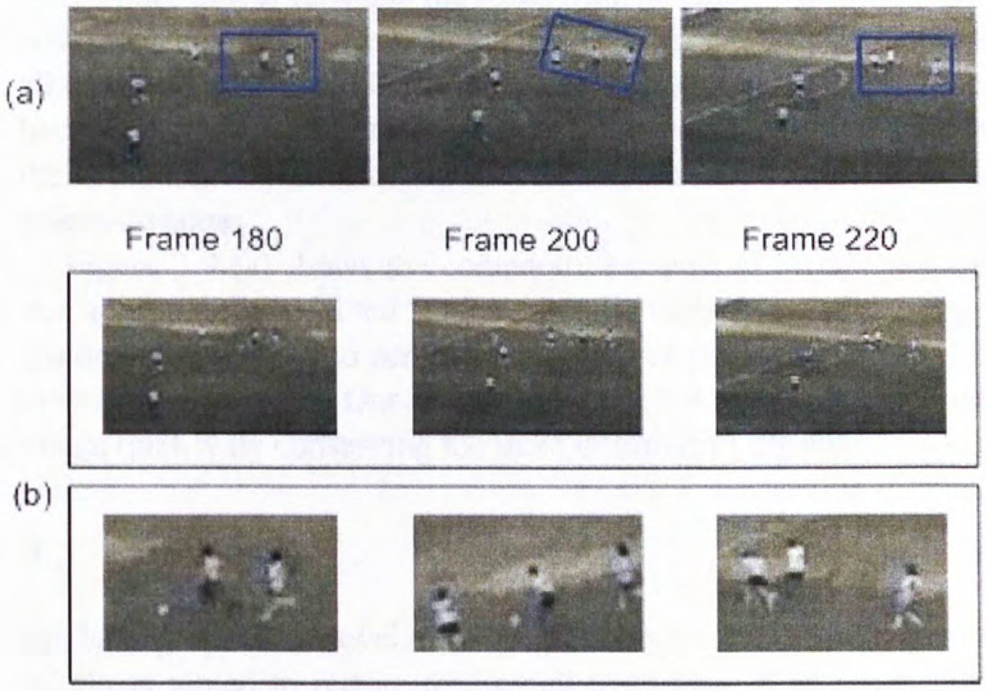


Figure 3.8: Football example (shown in color). (a) The input key-frames with the computed active windows. (b) the comparative results between the directly resized video and our approach.

active window is updated to include both of the skaters. When the skaters are closer again later, the active window rotates back to the original smoothly. (b) shows the comparative results of our method and directly resizing the video.

Football example. Figure 3.8 shows one of the most complex experiments performed. In this example, the active windows first focuses on two players who are running after the ball. From frame 190, a third player runs into the focus from distance. The active window then shifts to the center of the three players and rotate to include all of them. In frame 220, the orientation of the active window goes back to zero, when the three players run close together. (b) shows the comparative results of our method and directly resized video in selected frames.

Figure 3.9 (a) shows the comparative results of virtual pan and our approach, in selected frames. As the orientation of the target window is restricted to zero, the third player cannot be included in the viewing window. Our results in Figure 3.9 (b) shows the better visual quality by conserving the most informative objects.

3.3 Summary

We have proposed a novel video migration approach based on active windows aimed to reduce the spatial resolution of an input video without losing important motion details. Our method consists of the three processing steps: foreground extraction, active window initialization, and optimization in processes. In foreground extraction, we proposed to minimize an energy function based on the modelling of all pixel colors. Our method does not require any user interaction, and can obtain high quality foreground pixels compared to other methods. The initialization step robustly clusters density maps, in which we formulate the cluster optimization as a labelling problem, and adopt the belief propagation to compute the global minimum. The active windows are optimized considering both density of ac-

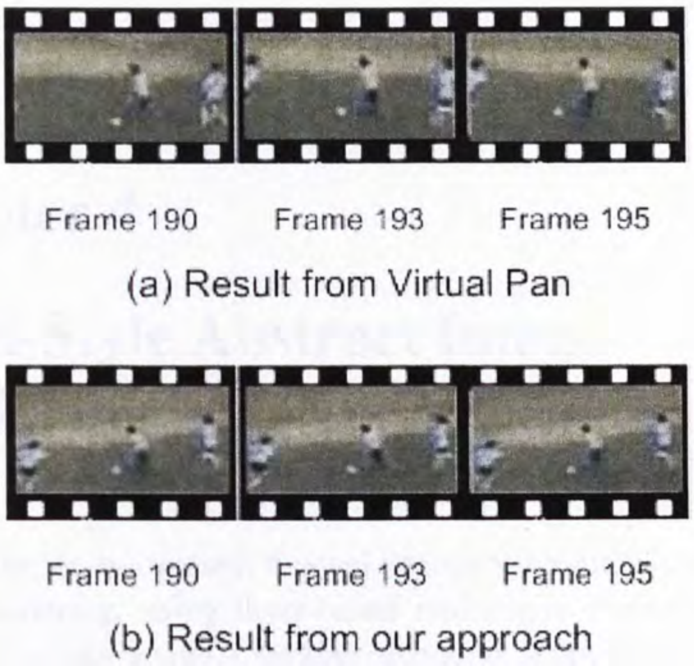


Figure 3.9: The comparative results of virtual pan (a), where the camera can move horizontally or vertically only, and our approach (b).

tive pixels and temporal smoothness, which guarantee that the output video does not shake while containing most motion information.

Our method is different from the conventional video summarization, where compressing temporal frames or abruptly reducing relative spatial distance among objects produces large ambiguities in video understanding when playing the output video alone. The goal of our approach is similar to [44]. However, as illustrated in Figure 3.9, our method performs better when new objects coming near are included. Our work is applicable for sports or surveillant video migration on mobile devices. We will further work on preserving more information without sacrificing the important motions in our current system.

□ **End of chapter.**

Chapter 4

Multi-Style Abstract Image Rendering

In this chapter, we present a novel approach for multi-style abstract image rendering, using layer-based multi-style rendering and abstract image processing. Our approach aims at re-rendering real images with selective styles automatically. The rendering styles (e.g. oil-painting, watercolor) are determined by the properties of captured images. In order to avoid the over-rendering phenomena on small but important objects, layers are detected from the improved image auto-segmentation algorithm. Different rendering weights are assigned to different layers. A diffusion filter is finally employed to eliminate noise and abstract the details.

4.1 Abstract Images

In general, Non-photorealistic rendering may be seen as any attempt to create images to convey a scene without directly rendering a physical simulation.Following the work in [28] and [58] on NPR rendering techniques, [27] summarized NPR in 2001, proposing three major NPR categories: artistic media simulation, user-assisted image creation, automatic image creation.

When it comes to rendering a real image with a certain rendering style, a particular kind of Non-Photorealistic Rendering (NPR) technique, there are generally two types of approaches presented in literature. One is to make the computer "pretending" as an artist and create the image in oils [27] either automatically [31] by referencing its real counterpart, or semi-automatically [15] which assists users drawing paintings. In this sort of system, kinds of pen strokes are simulated in advance. The path of a stroke is determined by the gradients of the original image for automatic rendering or by users for semi-automatic rendering. *Adobe®* Photoshop is one of the best e-painting platforms, supplying a great deal of tools and rendering filters. Figure 4.1 gives an example, where (a) is the original image. Users could draw a wonderful picture with the oil-painting style such as (b) by utilizing the tools from *Adobe®* Photoshop with 10 minutes or more. Image (c) is also generated by *Adobe®* Photoshop but automatically, where the filter "Paint Daubs" is used, which is the one generating the most similar style to the oil-painting style. Image (d) is the outcome of our system. Please note that (d) is generated automatically with few predefined parameters.

Unlike the "artist" approach, the other one tries to make the output image "pretending" to be rendered with a kind of artistic style. No matter how the images are processed, the system output is what that matter most. That means no matter what kinds of techniques are used, the generated image should look exactly as having been rendered with a kind of artistic style under some given conditions. For example, the color of each pixel must have some change to present a certain style. Such change should be made with physical reasons, e.g. the transmittance of the canvas, the properties of painting materials, optics theory, and so on.

Here, we primarily outline the previous work on image abstraction and its applications to video. In [31], a learning-based rendering framework is presented, in which the rendering style is learned from a pair of selected images. One is the real captured image and the other is its counterpart, rendered in a selected style. After training, the proposed approach has the ability to act as the artist, rendering

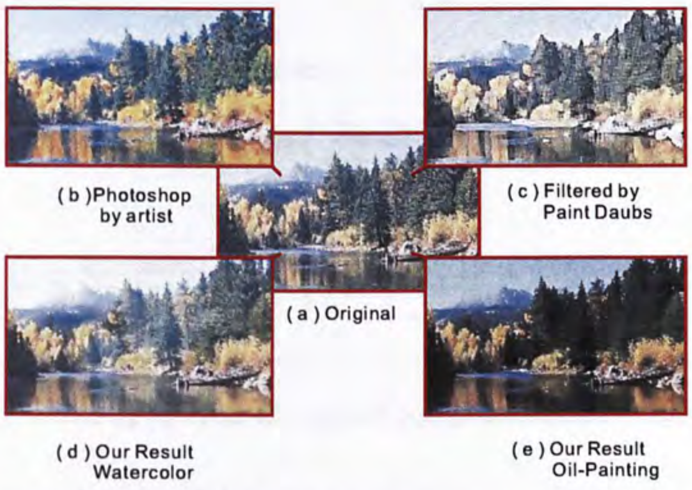

(b)Photoshop by artist
(c) Filtered by Paint Daubs
(a) Original
(d) Our Result Watercolor
(e) Our Result Oil-Painting

Figure 4.1: An example with styles painted on a captured image. (a) the original image. (b) by a senior artist using the *Adobe*® Photoshop CS2 tools. (c) by Paint Daubs Filter with the Brush Size = 7 and Sharpness = 13. (d) painted by watercolor style using our approach. (e) painted by oil-painting style, which is the output of our system.

other images in the style it has learnt. A physics-based rendering tool is presented in [15], which helps users create ink-style drawings. This model develops effective tools, making the e-ink-drawing possible.

The goal of the framework addressed in [9] is to render one image in multiple styles. This seems to confront a similar problem to that of the present study. However, since it aims at combining various styles into one image, the framework presented renders different parts of an image in different styles. How to keep the coherence of an image rendered in different styles is a big challenge. The emphasis is on rendering the most important information from the input image, and with this in mind, [19] proposed a hierarchical image representation system. Meaningful structure in an image is detected and preserved, and much of information presented by the other parts is lost after smoothing. Pixels are processed independently; nevertheless, information about less vital components of the image is lost. In [37], a layer-based multi-style rendering approach is presented, in

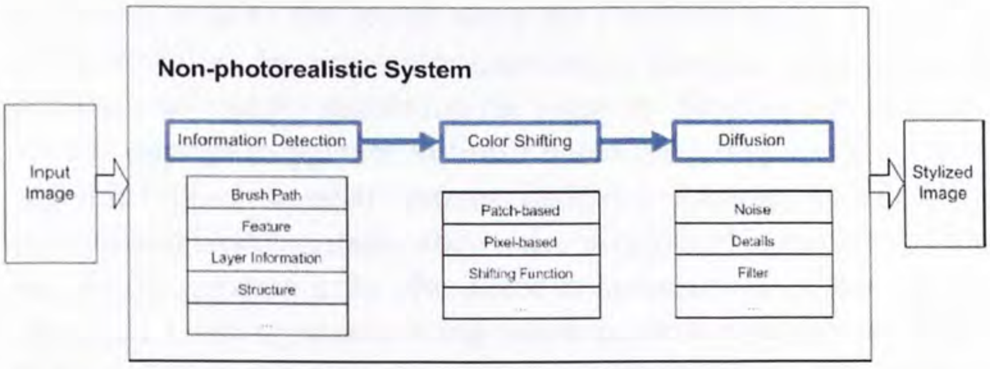


Figure 4.2: Processing pipeline of a typical non-photorealistic rendering system.

which diverse brush strokes are generated by referencing the size of related layers. The experimental results illustrate that the approach presented would produce good artwork in various styles from a captured image. However, in order to make the approach more efficient, tiny segments would be merged into prime ones. This causes the details of the merged tiny segments to be lost due to the brush strokes of its large neighbors.

There are also many studies, e.g.[43, 29, 66, 62, 30, 32, 51], that focus on simulating the paint brush and re-drawing the image automatically by following the rendering paths detected from the original image.

4.2 Multi-Style Abstract Image Rendering

Figure 4.2 gives the pipeline of the non-photorealistic rendering system. There are generally three components in such system: Information Detection, Color Shifting, and Diffusion. Rendering a image with an artistic style starts from the path on image re-drawing, which is determined by the detected image information. Take a brush-based oil-painting system for example, the vertical direction of the image gradient is recognized as the brush path in most cases. Then, the properties of the brush is important to render a non-photorealistic rendering style to the image along the rendering path. We call it Color Shifting, since the color information from the original pixels is always shifted by simulating the pigment. Shifting functions are always applied to patches in brush-based rendering systems, or to individual pixels in other systems. Diffusion is always included in a non-photorealistic system, when noise is captured in the image, and details are required to be eliminated in some artistic styles. In our approach, multi-style processing selects rendering styles with the luminance of major layers of images, to avoid the over-rendering phenomena. Noises and meaningless detail are eliminated in the final abstracted images.

The framework we developed here is a kind of layer-based abstract image rendering with multiple styles. This framework is an application of the filter-based rendering method. Rather than focusing on the procedure of drawing a painting by an artist, we concern more on the outcome itself. We have two main concerns here. One is to inject new color information to each pixel on its physical condition for a certain non-photorealistic rendering style. The other one is to eliminate noises in the input image and smooth the details. Aiming at these two concerns, our approach consists of the following three steps. Multi-style processing is responsible for determining the proper non-photorealistic rendering style for the input image from two alternatives, oil-painting or watercolor, based on the lighting conditions of major layers in the image. Small-scale rendering algorithm is used to render the input image with a selected style by rendering each pixel following the rendering function based on a layer-based rendering method where an improved auto-segmentation algorithm is utilized. Abstraction eliminates most of the noises and smooths out details from the temple result generated by the first two steps.

Multi-style Processing is responsible for determine the proper rendering style for the input image. It is noted that making the decision on which non-photorealistic rendering style is the right one for a captured image is a great challenge in NPR. In our approach, we present a mechanism on selecting one style from two alternatives, oil-painting and watercolor, based on the lighting conditions of major layers in the image. Since oil always presents colors darker, whilst one natural property of watercolor is rendering the scene brighter, we stylize the input image with the style which makes the luminance changes to the reverse way.

Layer-based Rendering is used to render the input image with a selected style by rendering each pixel following the rendering function. There are two parameters taken by our rendering function, the selected style, and how deep (the weight) should the current pixel be rendered. In order to determine the depth of rendering on each pixel, we proposed a layer-based rendering method where an improved auto-segmentation algorithm is utilized. Note that the reason that there should be layers is because that artists in the real world never draw paintings by pigments with the same concentration. However, on the contrary, Too many rendering layers lead to the lack of consistency of the rendering. Therefore, in our method, there is no need to segment images into meaningless pieces. That is to say, all the layers specified to an image should have a proper size. In our method, small pieces are avoid by automatically detecting the proper minimal size of layers for general auto-segmentation algorithms.

Abstraction eliminates most of the noises and smooth out details from the result generated by the Multi-Style Layered Image Rendering algorithm introduced above. Since paintings are generally used to express the idea of the wold, trivial details are always ignored. Bilateral filter, one of the most efficient nonlinear diffusion filter, is used to make the post-processing in our approach.

We describe all the above steps of our approach in the following sections respectively. In the rest of this paper, we represent the color as a vector $\mathbf{I}^p = (R, G, B)^T$ in RGB color channels for each pixel p.

4.2.1 Multi-style Processing

We develop an approach on selecting abstract styles by referencing images' properties and indicating users' intentions. When simulating the rendering effect of an abstraction, one of the most important factors is the thickness of rendering pigments. With a white canvas, most of the lights directed at the canvas are reflected. The observed color is the squared transmittance as the light must travel through the pigment twice. Since pigments for oil-painting are much heavier than watercolor, objects rendered by oil always lack light energy, compared with the same objects rendered by watercolor. Our system renders images with low light energy by watercolor or vice versa. We also develop an interface for users to specify the rendering styles, (e.g. rendering images with high light energy by oil is possible). All the experimental results in this chapter are generated automatically.

In our approach, light energy is calculated by the luminance of the image. The luminance $\mathbf{L}^p$ of a pixel p can be calculated by translating a true color image into a gray one:

$$\mathbf{L}^p = \frac{T \cdot \mathbf{I}^p}{255} \tag{4.1}$$

where $T = (0.3, 0.59, 0.11)$. Note that this function is exactly the one translating a true color image into a gray one.

Then, the luminance $\mathbf{L}^I$ of the input image can be defined as:

$$\mathbf{L}^I = \frac{\sum_p \mathbf{L}^p}{\mathbf{I}_w \cdot \mathbf{I}_h} \tag{4.2}$$

where $\mathbf{I}_w$ and $\mathbf{I}_h$ are the width and height of the image, respectively.

Hence, the style of the image $\mathbf{S}^I$ can be determined by:

$$\mathbf{S}^I = \begin{cases} oil-painting & \text{if } \mathbf{L}^I > W^I \\ watercolor & \text{otherwise} \end{cases} \tag{4.3}$$

where the threshold W^I is believed to be 0.5 intuitively. However, from our experiments, 0.45 is much better for most cases.

To render an image with oil-painting or watercolor style [6], convincing visual results, for computer-generated effects, are obtained. The rendered color $\widetilde{T^p}$ is generated by:

$$\widetilde{T^p} = 255 \cdot [C - (C - C^2)(P - 1)] \tag{4.4}$$

where $\widetilde{T^p}$ is the true color value of pixel p, and $C = T^p/255$. P is the rendering weight, presenting the pigment density. $P \in (1, 2]$ means the pigment is dense enough to perform oil-painting effects, while watercolor effects are generated with $P \in [0, 1)$.

However, the approach [6] has two drawbacks. First, selecting the rendering style from this method may not give satisfactory results, since each pixel has the same contribution no matter what the information it presents. The more important objects should have more impact on the results when selecting the style. Second, when some parts of the image are over-exposed, which makes other parts of the image under-exposed, neither of the two rendering styles is good enough to render the image. We introduce a method to handle the problem by utilizing layer information. Note that the term "important" in this chapter means "conspicuous". Hence, an important object is assumed to be big enough in an image.

4.2.2 Layer-based Rendering

When specifying a proper rendering style to an image, the importance of objects has already been considered, since larger objects are presented by more pixels, as well as contributing more to the luminance of the input image $\mathbf{L}^I$. However, one large object may be less important than the combination of worthless small objets, resulting in rendering the image with the improper style to the most important object.

To avoid this scenario, we introduce layers to our framework, where layers are produced by auto-segmentation algorithm first. Among the work in image segmentation [24, 42, 57], we utilize the four-step algorithm in [24] for its steady performance and the ability to determine the numbers of the segmentation automatically. The algorithm segments images by giving the smoothing term σ, 0.5 in most cases, the threshold on segmentation k, 500 generally, and the minimum size of layers, which is specified image by image. However, general parameters are needed in our system, and further interactions with users per image should be avoided. Therefore, our system joins trivial areas based on the statistical result from the primitive segmentation, where the minimum layer size *mini_size* is:

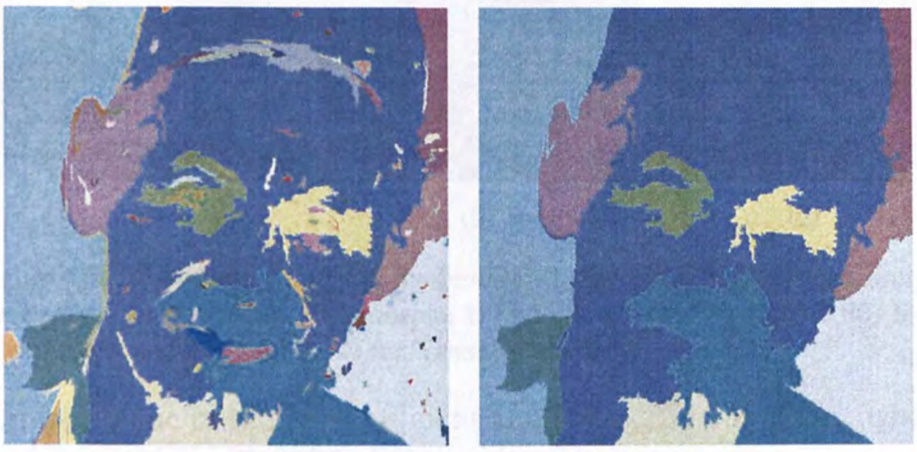
$$mini\_size = d \cdot \frac{\mathbf{I}_w \cdot \mathbf{I}_h}{\widetilde{N}} \tag{4.5}$$

where $\widetilde{N}$ is the number of segmented layers originally, d is the weight to control the minimum layer size, and 0.07 is good enough for most cases to combine trivial pieces. Figure 4.3 and Figure 4.4 give the illustration of the improved segmentation approach. Further, in order to determine the style based on the most important objects, light energy from only the third layer (the biggest one) is considered on style selection.

However, rendering all the image partitions under different lighting conditions with the same style definitely makes some parts over-rendered. Figure 4.5 gives one comparative illustration, more discussion in detail in Section 4.3. To prevent this from happening, smaller layers should be rendered less by assigning different Ps. For example, if the input image is rendered by oil-painting style, the range of rendering weight P in Function (4.4) goes to $(1, 2]$. Ps are close to 2 in larger layers, but are much smaller in small ones. From our experiments, the smallest P is always around 1.5 for oil-painting, or 0.5 for watercolor.

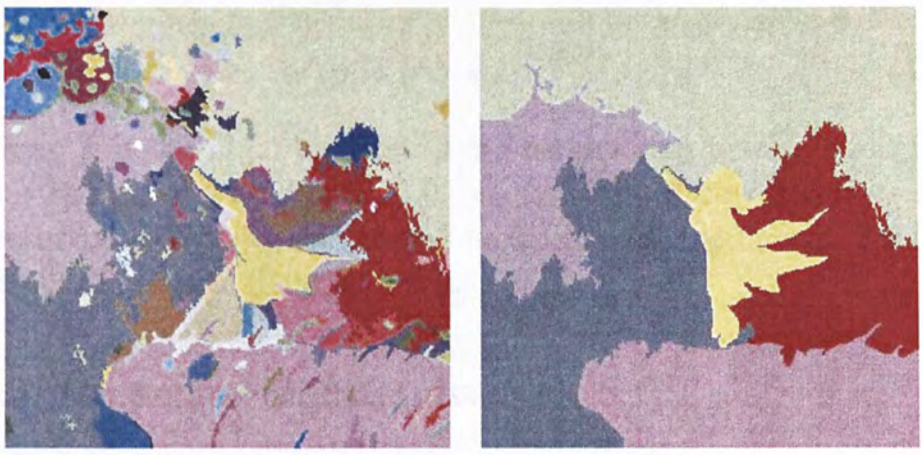
4.2.3 Abstraction

In order to eliminate most of the noises and smooth out meaningless details, abstraction is necessary. We use the bilateral filter in our



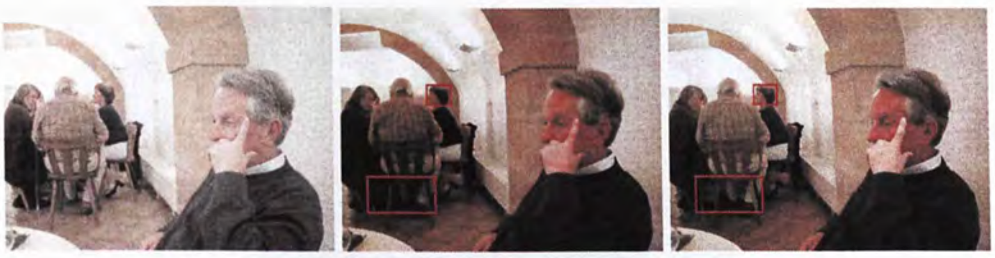
(a) (b)

Figure 4.3: (a) The result from [24], where 156 components are segmented by suggested parameter *mini_size* = 20. (b) Our result using only 11 components and the major objects of the image are well segmented.



(a) (b)

Figure 4.4: (a) The result from [24], where 189 components are segmented by suggested parameter *mini_size* = 20. (b) Our result using only 6 components and the major objects of the image are well segmented.



(a) (b) (c)

Figure 4.5: Restaurant: (a) the original image; (b) the result from uniform-weight rendering. (c) our result - Luminance $\mathbf{L}^{\prime}(0.507)$, weights P_L (1.89-2.00) for 6 large layers, and P_S (1.50-1.64) for 7 small-scale ones.

approach, one of the most efficient nonlinear diffusion filters, where the kernel of the filter $F(\cdot)$ is:

$$F(\widetilde{p}, \sigma_d, \sigma_r) = \frac{\int e^{-\frac{1}{2}(\frac{\|\widetilde{p}-p\|}{\sigma_d})} w(p, \widetilde{p}, \sigma_r) f(p) dp}{\int e^{-\frac{1}{2}(\frac{\|\widetilde{p}-p\|}{\sigma_d})} w(p, \widetilde{p}, \sigma_r) dp} \tag{4.6}$$

$$w(p, \widetilde{p}, \sigma_r) = (1 - m(\widetilde{p})) \cdot w'(p, \widetilde{p}, \sigma_r) + m(\widetilde{p}) \cdot u(\widetilde{p}) \tag{4.7}$$

$$w'(p, \widetilde{p}, \sigma_r) = e^{-\frac{1}{2}(\frac{\|\widetilde{p}-p\|}{\sigma_d})} \tag{4.8}$$

where $\widetilde{p}$ is the pixel location, p are neighboring pixels. σ_d is the blur radius, and always be 3.0. σ_r determines how contrasts are preserved or blurred, and always be 0.1 in our experiments. For the kernel of bilateral filter, $m(\cdot) = 0$.

4.3 Experimental Results

Our system is develpoed and tested on an Intel(R) Pentium(R) 4 CPU 3.20GHz with 1GB RAM, using Visual C++. Performance depends on image size and control parameters. Typical tests for a $512 * 512$ image and default parameters usually require $0.5 - 1.2$ seconds.

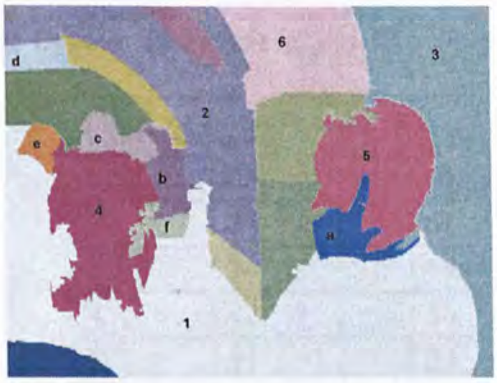


Figure 4.6: Improved auto-segmentation results. L-layers 1 ~ 6 are the major parts that determine the style of the painting (e.g. oil-painting). S-layers a~f are painted with lighter pigments to avoid the over-painting phenomena.

Figure 4.5 shows one abstract-image example using our approach. The input image is segmented into 19 components with our improved auto-segmentation algorithm. The largest 1/3 parts are used to determine the rendering style. Since the luminance $\mathbf{L}^{I}$ of this image is 0.507, oil-painting style is chosen. However, if the image is painted without small-scale information, features on the human faces and some details are lost, as shown in Figure 4.5 (b), where all the parts in the image are rendered with the same pigment density parameter $P = 2.0$. Therefore, small-scale image rendering is necessary, especially for those images whose luminance is close to the threshold of the style selection. From Figure 4.5 (c), we see that the details of the persons around the table and the features of the background are kept well, compared with Figure 4.5 (b). Finally, in the abstraction stage, bilateral **filter** is utilized to eliminate the noises and smooth the image. Figure 4.5 (c) is the abstract image rendered by our system. Two selected parts in (b) and (c) are enlarged to show the difference of rendered abstract images. Table 4.1 lists the pigment density parameters Ps, and the key layers marked in Figure 4.6. In the post-processing stage, Bilateral **filter** is utilized to eliminate the noises and smooth the image, as shown in Figure 4.5.

Table 4.1: Pigment Density Parameters of key layers used in the "Restaurant" example.

L-layer	P	S-layer	P
1	2.00	a	1.64
2	1.97	b	1.61
3	1.94	c	1.58
4	1.92	d	1.56
5	1.91	e	1.53
6	1.89	f	1.50

Figure 4.7 shows the other example that illustrates the rendering effect of our approach. The images at the top line are rendered by our multi-style rendering with fine details. The results by abstract rendering with uniform weight $P = 2.0$ are shown at the bottom line. Our results are shown on the top line, in which the details of the lady's face and the features on the background are preserved well. The system in [37] also generates layer information to paint the image with various styles. However, details in the small layers may not be preserved well, since brushes from large layers also impact their small neighbors.

The multi-style selection examples are shown in Figure 4.8, Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13, where the abstract styles are selected for the input images automatically. In Figure 4.8, since the original image Figure 4.8 (a) is captured under natural circumstances with luminance $\mathbf{L}^I = 0.314$, it is too dark to be presented in the oil-painting style shown in Figure 4.8 (b), whereas the selected watercolor style in Figure 4.8 (c) appears much better. Figure 4.9 gives another illustration where water color style is applied. The result Figure 4.9(c), which is rendered with layer information is better than the non-layer based rendering, in Figure 4.9 (b), especially on the coherence. Figure 4.10 gives the example which illustrate the oil-painting style. Since the Luminance

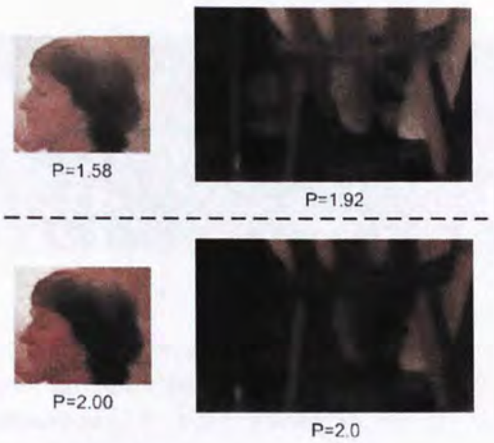
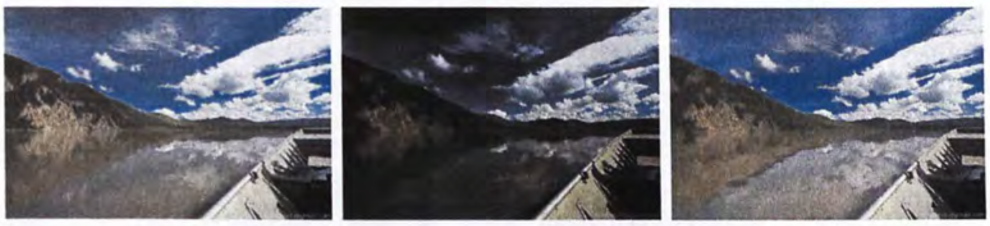


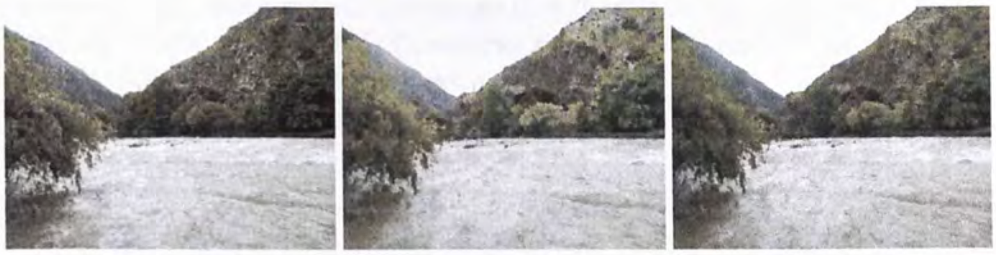
Figure 4.7: Two parts in Fig . 4.5. In the top line, part layers are rendered with weights $P(1.58)$ and $P(1.92)$ respectively. Our approach avoids over-rendering on detailed objects better, comparing with the one by uniformed weight $P(2.0)$, shown in the bottom line.

of the input image, Figure 4.10 (a), is $\mathbf{L}^I = 0.468$, the oil-painting style selected automatically in Figure 4.10 (c) is better, compared with the result from the watercolor style in Figure 4.10 (b). In Figure 4.11 (a), due to a **flash** effect, the man's face is much brighter, and the luminance is $\mathbf{L}^I = 0.513$. After applying oil-painting style, Figure 4.11 (c) shows a more visually-pleasing rendering than the watercolor style in Figure 4.11 (b). Figure 4.12 and Figure 4.13 give the selective results in comparison with [73] which focuses on the edge enhancement shown in Figure 4.12 (b) and Figure 4.13 (b). Our multi-style result in Figure 4.12 (c) and Figure 4.13 (c)focuses more on the small-scale details of the rendering, especially on the lady's cheek. Another comparison between our method and the oil-painting method presented in [29] is given in Figure 4.14. Compared with Figure 4.14 (b), our result, Figure 4.14 (c), performs better on rendering the branches and the details of the rhinoceros, which has been illustrated in Figure 4.15.



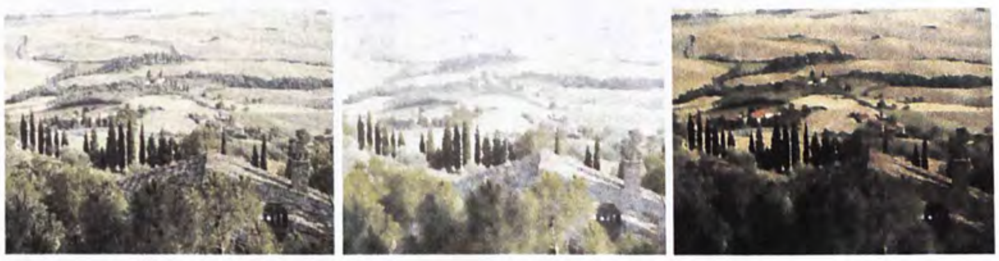
(a) (b) (c)

Figure 4.8: landscape-1: (a) the original image; (b) the result from rendering with oil-painting style, not selected; (c) our result with watercolor style selected automatically. - Luminance $\mathbf{L}^{I}(0.314)$, weights P_L (0.00-0.15) for 3 large layers, and P_S (0.25-0.50) for 5 small-scale ones.



(a) (b) (c)

Figure 4.9: Landscape-2: (a) the original image; (b) the result from uniform-weight rendering; (c) our result - Luminance $\mathbf{L}^{I}(0.397)$, weights P_L (0.00-0.14) for 3 large layers, and P_S (0.23-0.50) for 6 small-scale ones.



(a) (b) (c)

Figure 4.10: Landscape-3: (a) the original image; (b) the result from rendering with watercolor style, not selected; (c) our result - Luminance $\mathbf{L}^{I}(0.468)$, weights P_L (1.84-2.00) for 5 large layers, and P_S (1.50-1.72) for 7 small-scale ones.

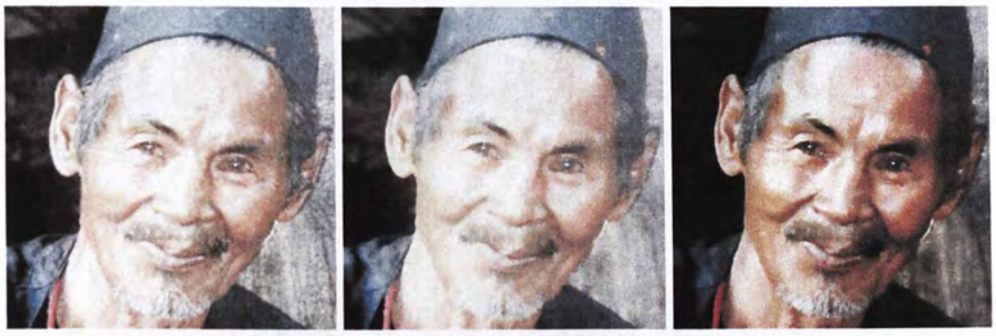

(a) (b) (c)

Figure 4.11: **Profile-1:** (a) the original image; (b) the result from rendering with watercolor style, not selected; (c) our result with oil-painting style selected automatically - Luminance $\mathbf{L}^{l}(0.513)$, weights P_L (1.86-2.00) for 3 large layers, and P_S (1.50-1.73) for 5 small-scale ones.

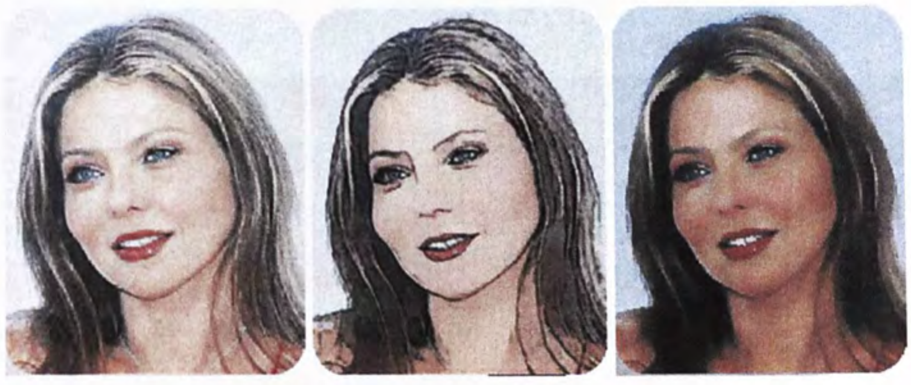
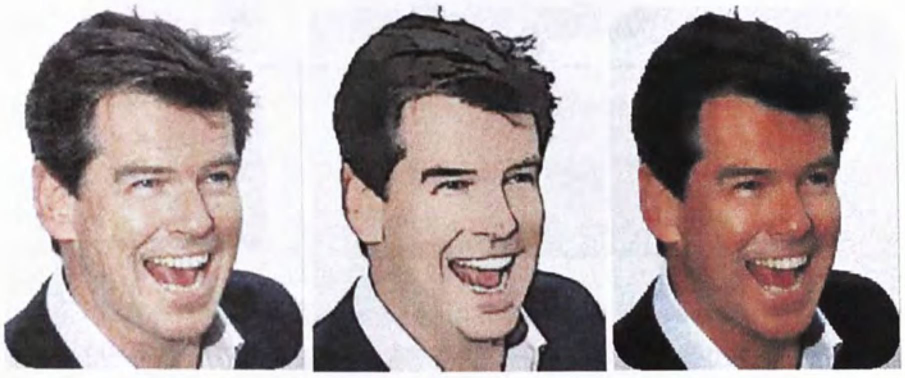
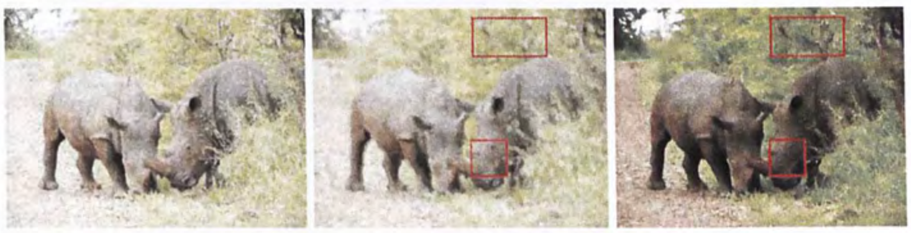

(a) (b) (c)

Figure 4.12: **Profile-2:** (a) the original image; (b) the abstract image rendered by [73]; (c) the result by our approach - Luminance $\mathbf{L}^{l}(0.501)$, weights P_L (1.89-2.00) for 3 large layers, and P_S (1.50-1.71) for 6 small-scale ones.



(a) (b) (c)

Figure 4.13: **Profile-3:** (a) the original image; (b) the abstract image rendered by [73]; (c) the result by our approach - Luminance $\mathbf{L}^l(0.503)$, weights P_L (1.83-2.00) for 3 large layers, and P_S (1.50-1.72) for 4 small-scale ones.



(a) (b) (c)

Figure 4.14: Comparison with the method presented in [29]. (a) is the original image. (b) is the oil-painting stylized image rendered by [29]. (c) is generated by our approach - Luminance $\mathbf{L}^l(0.567)$, weights P_L (1.86-2.00) for 3 large layers, and P_S (1.50-1.73) for 5 small-scale ones.

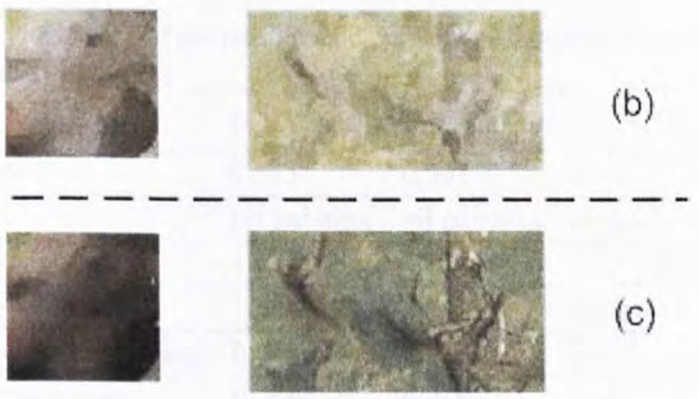


Figure 4.15: Two parts in Figure 4.14. In the bottom line, our framework renders the details better, comparing with the one by the oil-painting method presented in [29] in the top line.

Table 4.2: The statics of parameters and performance on the Multi-style Abstract Image Rendering

	Fig. 4.5	Fig. 4.8	Fig. 4.9	Fig. 4.10
Luminance	0.507	0.314	0.397	0.468
Style	oil-painting	watercolor	watercolor	oil-painting
P for L-layer	1.89-2.00	0.00-0.15	0.00-0.14	1.84-2.00
P for S-layer	1.50-1.64	0.25-0.50	0.23-0.50	1.50-1.72
Time (s) layer processing	0.235	0.331	0.718	0.330
Time (s) rendering	0.460	0.460	0.436	0.352
Time (s) in total	0.695	0.791	1.154	0.682

4.4 Summary

In this chapter, we have presented a novel approach for multi-style abstract image rendering, using selective abstract styles and the abstraction processing. In most image rendering systems, the rendering style is usually **predefined.** Users have very limited power on modifying the parameters to make rendering effects better. Furthermore, over-rendering phenomena are common since all the pixels are rendered with the same rule. Using layer-based process-

Table 4.3: The statics of parameters and performance on the Multi-style Abstract Image Rendering (cont.)

	Fig. 4.11	Fig. 4.12	Fig. 4.13	Fig. 4.14
Luminance	0.513	0.501	0.503	0.567
Style	oil-painting	oil-painting	oil-painting	oil-painting
P for L-layer	1.86-2.00	1.89-2.00	1.83-2.00	1.86-2.00
P for S-layer	1.50-1.73	1.50-1.71	1.50-1.72	1.50-1.73
Time (s) layer processing	0.646	0.216	0.220	0.341
Time (s) rendering	0.289	0.316	0.307	0.314
Time (s) in total	0.935	0.532	0.527	0.655

ing, our approach can render images with selective styles and avoid over-rendering important detailed objects by assigning more variable weights. Our work is generally applicable for multi-style image re-rendering to preserve small but vital objects well in a unified framework. We will further extend the current work to GPU-accelerated processing, and to other styles of illustration such as charcoal or pastel that has not been implemented in computer graphics.

□ **End of chapter.**

Chapter 5

Interactive Abstract Videos

Further from our work in abstract images, we introduce a novel approach for interactive abstract videos. A sequence of images with smooth styles transition is created from a single input image, as follows: First, the painting styles are specified by the user to render the input image. Users can select painting styles from the filter-based painting styles of oil-painting, watercolor, chalk, ink; the brush-based painting style of oil-painting; the wooden style. The layer information from the input image is studied from the improved algorithm of image auto-segmentation. The output video is produced through a smooth and controllable translation of the input image from one style to the other, employing different interpolation functions on different grouped layers. This approach aims to re-render real-world images or handy images in non-photorealistic styles selected by the user. The approach is capable of handling images with various contents, and can be applied as an interactive form of digital-media edutainment.

5.1 Abstract Videos

Recently, two applications based on this method have been developed for video rendering [73, 6]. Since the video must be abstracted in real time in [73], the approach has to perform at speed and this eliminates most of the details, and enhances the differences in colors. In order to achieve its goals, this model uses a bilateral filter to first smooth the image. Colors are then quantized into several levels. To strengthen the color variation, selected edge information from the original image is preserved in the output and bold-faced. Finally, the author's algorithm is implemented with GPU techniques, producing 15 frames per second. In [6], a watercolor style is imposed on the input video. The purpose of the approach in that study is to make each frame conform to the watercolor style, whilst keeping the modification coherent along the time domain. The approach also performs shape abstraction utilizing the ability of watercolor to suggest details with abstract washes of color.

However, there are problems with the previously mentioned approaches. Each approach usually generates one particular style. Users have limited power to modify the parameters and improve the results, also they cannot change the style into another. Failure is common when the input images do not match with the predefined style. Moreover, once the parameters have been set, pixels in the image are processed with the same rule, no matter what the pixels represent. Small but important objects in the input images always suffer from this problem.

5.2 Multi-Style Abstract Video

TWe propose interactive abstract video approach aimed at generating multi-style videos from a single input image. This abstract-video approach is novel for the following concerns. Generally speaking, one abstracted image is usually generated from the input image by the current approaches. Our work is motivated to create abstract videos from the existing non-photorealistic rendering styles. Users can interactively control the rendering parameters as wee as the interpolation functions, and visualize the styles smooth transition based on the grouped layers.

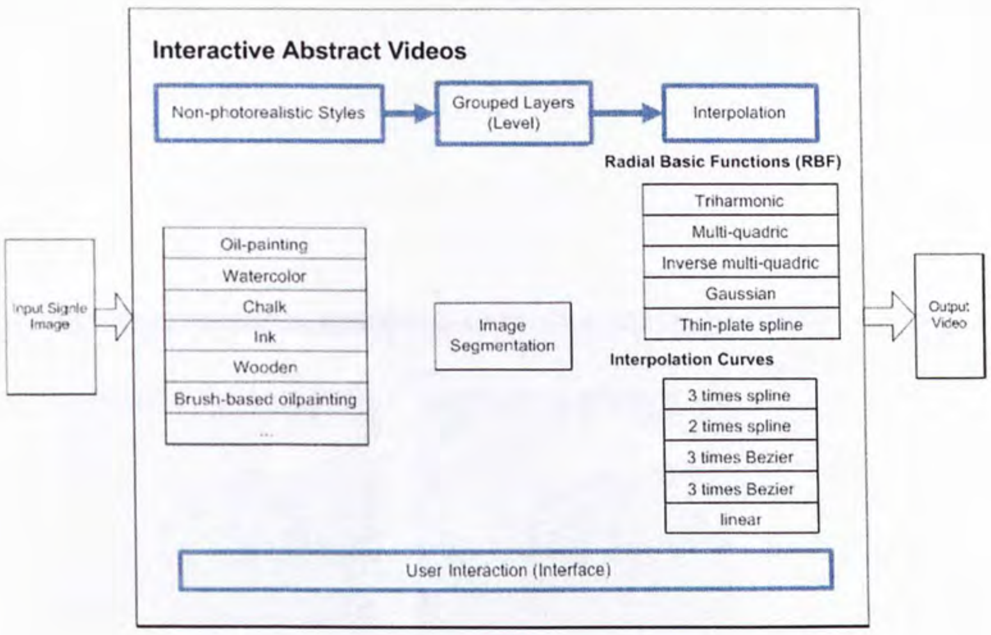
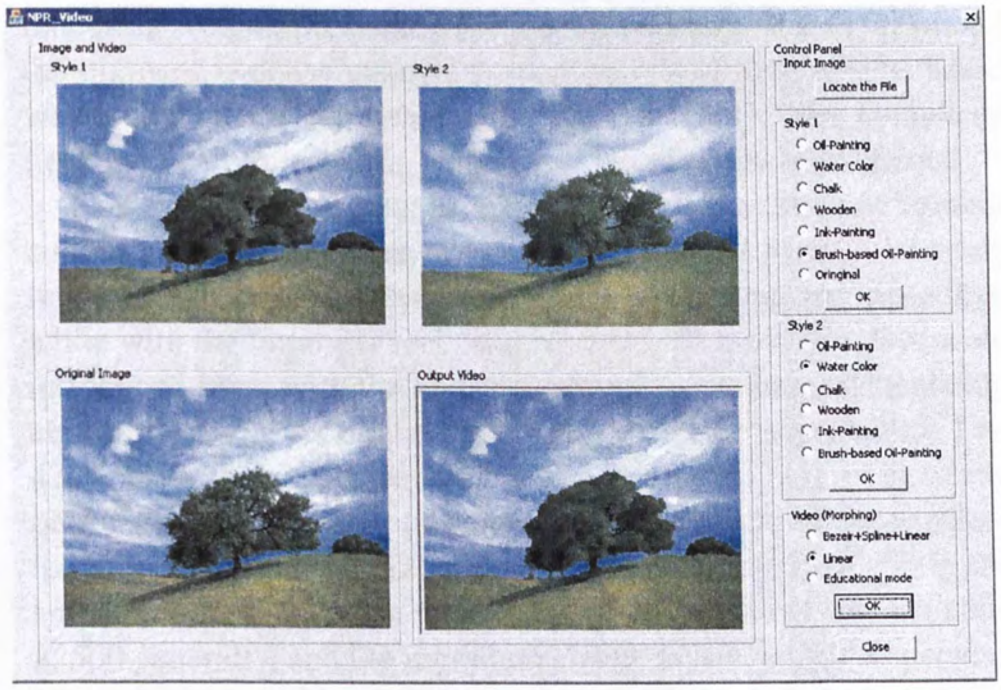


Figure 5.1: System components of interactive videos.

Figure 5.1 gives the components of the system on interactive abstract videos, and the interface of the system is shown in Figure 5.2. After specifying the abstract rendering styles to the input image, users interact with the processing of video morphing by controlling the parameters on interpolation functions. Educational mode is for users to specify the transition on one certain level (grouped layers).

5.2.1 Abstract Images

We have worked on the abstraction styles including oil-painting, watercolor, chalk, ink, wooden and brush-based oil-painting in our system. More abstraction styles can be flexibly included in our abstract rendering framework.



Style 1: Oil-Painting, Water Color, Chalk, Wooden, Ink-Painting, [✓ Brush-based Oil-Painting], Oringinal

Style 2: Oil-Painting, [✓ Water Color], Chalk, Wooden, Ink-Painting, Brush-based Oil-Painting

Video (Morphing): Bezeir+Spline+Linear, [✓ Linear], Educational mode

Figure 5.2: The multi-style video system interface. On the left is the display window. In the top row, two selected abstract styles are displayed. The input image and the generated multi-style abstract video are showed in the bottom row. On the right is the control panel, where users can specify rendering styles, and control parameters for the abstract video morphing.

Oil-painting / Watercolor Styles

To render the input image with oil-painting or watercolor style [5], we generate the rendered color $\widetilde{T^p}$ as follows:

$$\widetilde{T^p} = 255 \cdot [C - (C - C^2)(P - 1)] \tag{5.1}$$

where $\widetilde{T^p}$ is the true color value of pixel p, and $C = T^p/255$. P is the rendering weight, presenting the pigment density. $P \in (1, 2]$ means the pigment is dense enough to produce oil-painting effects, while watercolor effects are generated with $P \in [0, 1)$. For computer-generated painting effects, convincing visual results are obtained.

However, rendering the image partitions under different lighting conditions with the same style may cause some parts to be over-rendered, which means that some pixels may be changed to black or white with the improper rendering weights. To avoid this, layers are introduced here, with the layers produced by an auto-segmentation algorithm. There are studies that discussed image segmentation [24, 42, 57], our system utilizes the four-step algorithm [24] for its steady performance and ability to determine the numbers of the segmentation automatically. The system segments images by giving the smoothing term σ, 0.5 in most cases, the threshold of segmentation k, 500 generally, and the minimum size of layers, which is specified image-by-image. However, general parameters are needed, and further interactions with users per image should be avoided. Therefore, the system investigates the work based on the statistical results from the primitive segmentation, where the minimum layer size *mini_size* is 7% of the average size of the segments.

Smaller layers are rendered less by assigning different Ps. For example, if the input image is rendered in oil-painting style, the range of rendering weight P in Function (5.1) is $(1, 2]$. Ps are close to 2 in larger layers, but much smaller in small-scale layers. In our experiences, the smallest P is around 1.5 for oil-painting, or 0.5 for watercolor.

In order to eliminate most of the noise and smooth out meaningless details, abstraction is necessary. The bilateral filter was used as one of the most efficient nonlinear diffusion filters, where the kernel of the filter $F(\cdot)$ is:

$$F(\widetilde{p}, \sigma_d, \sigma_r) = \frac{\int e^{-\frac{1}{2}(\frac{\|\widetilde{p}-p\|}{\sigma_d})} w(p, \widetilde{p}, \sigma_r) f(p) dp}{\int e^{-\frac{1}{2}(\frac{\|\widetilde{p}-p\|}{\sigma_d})} w(p, \widetilde{p}, \sigma_r) dp} \tag{5.2}$$

$$w(p, \widetilde{p}, \sigma_r) = (1 - m(\widetilde{p})) \cdot w'(p, \widetilde{p}, \sigma_r) + m(\widetilde{p}) \cdot u(\widetilde{p}) \tag{5.3}$$

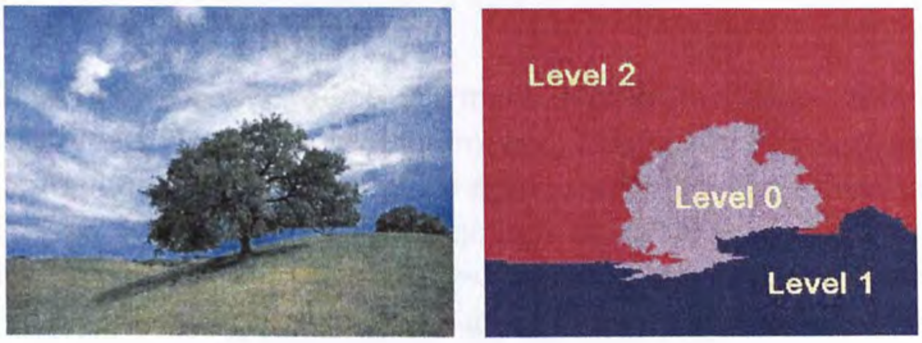
$$w'(p, \widetilde{p}, \sigma_r) = e^{-\frac{1}{2}(\frac{\|\widetilde{p}-p\|}{\sigma_d})} \tag{5.4}$$

where $\widetilde{p}$ the pixel location, p is the neighboring pixel. σ_d is the blur radius; always 3.0. σ_r determines how contrasts are preserved or blurred, and this is always 0.1. For the kernel of the bilateral filter, $m(\cdot) = 0$.

Chalk / Ink Styles

With the rendering approach introduced in the previous section, chalk style is rendered by making Function (5.1) work on the hue value in the HSV color space instead of the RGB color space for our approach. P locates at $[0, 1)$. HSV stands for hue, saturation, value, which describe colors as points in a cylinder whose central axis ranges from black at the bottom to white at the top with neutral colors between them, where angle around the axis corresponds to "hue", distance from the axis corresponds to "saturation", and distance along the axis corresponds to "lightness", "value", or "brightness". Unlike the oil-painting or watercolor styles, the simulating chalk style focuses more on the decrease of the value on the "hue" channel.

There are approaches presented on simulating ink style for non-photorealistic rendering [15]. In the interactive abstract videos, the ink effect in our approach is simply produced by changing the RGB color space into gray color space in Function (5.1), where $P \in [0, 1)$.



(a) (b)

Figure 5.3: The proposed approach segments the input image into three grouped layers (levels). (a) the input image. (b) the three layers generated by the auto-segmentation algorithm.

Brush-Based Oil-Painting / Wooden Styles

The brush-based oil-painting style algorithm presented in [29] is employed by our approach. The image is rendered with a series of spline brush strokes. Layers are detected from the input image. Rendering starts with a rough sketch drawn with a large brush. The sketch is rendered over with progressively smaller brushes in areas where the sketch differs from the blurred source image. The curves of the strokes are aligned to normals of image gradients.

The wooden style is generated by rendering the pixels white or black based on the results of

$$\widetilde{W^p} = \begin{cases} white & if(\widetilde{G^p} > 128) \\ Black & if(\widetilde{G^p} < 128) \end{cases} \tag{5.5}$$

where $\widetilde{G^p}$ is the gray color of pixel p, and $\widetilde{W^p}$ is the color value (white/black) in the wooden style.

Note that the painting framework of our system is open to other non-photorealistic rendering, so it is possible to abstract the input images with more abstract rendering styles.

5.2.2 Video Morphing

In order to generate in-between multi-style abstract images from the specified styles, the approach produces a sequence of images translating the abstract image in one style into another. Therefore, more abstract images rendered with multiple styles are generated in video for users to control and visualize. The most straightforward way to realize the idea is to interpolate the abstract styles linearly.

$$\widetilde{S^p}(t) = (1-t)\widetilde{S^p}(0) + t\widetilde{S^p}(1) \quad t \in [0,1] \tag{5.6}$$

where $\widetilde{S^p}(t)$ is the style of pixel p at time t. $\widetilde{S^p}(0)$ is the first style selected by the user, and $\widetilde{S^p}(1)$ is the second style specified.

The linear interpolating of styles is good enough to illustrate the possibility of generating more multi-style images in smooth transition, where different rendering styles contribute to the output respectively. However, this method ignores the properties of layers in the images, where the pixels from different partitions in the same image are rendered under the same rule. We have further worked on the approach to generate the multi-style abstracted frames based on layer information. Each layer follows the interpolation rule specified by the user, where

$$InterFun = f(layer^p) \tag{5.7}$$

In this approach, we employed the same segmentation algorithm in Section 5.2.1 to detect layers of the input image. The layers are further grouped into three levels, each of them has the specified interpolation function. Figure 5.3 gives an illustration of the grouped layer information generated for the abstract video morphing.

Based on the levels (grouped layers) information, radial basis functions (RBF), which are the natural generalization of coarse coding to continuous-valued features, can be utilized in our system to generate the morphing output video, where

$$\widetilde{S^p}(t) = (1 - \phi(t))\widetilde{S^p}(0) + \phi(t)\widetilde{S^p}(1) \quad t \in [0, 1] \tag{5.8}$$

The value of radial basis function $\phi(t)$ depends only on the distance from the origin, $\phi(t) = ||t||$; or alternatively on the distance from some other point c, called a center, so that $\phi(t, c) = ||t - c||$. Usually, the popular basis functions are:

- triharmonic, where $\phi(t) = t^3$
- multi-quadric, where $\phi(t) = \sqrt{t^2 + c^2}$
- inverse multi-quadric, $\phi(t) = \frac{1}{\sqrt{t^2+c^2}}$
- Gaussian, $\phi(t) = e^{-ct^2}$
- Thin-plate spline, $\phi(t) = t^2 log(t)$

where c is a constance specified in advance. With different function $\phi(t)$s, different levels transit from one rendering style to another with different speed. Therefore, in-between frames are generated in a controllable manner. Figure 5.5 illustrate one example of the approach. In our system, the Gaussian function is employed to Level 0, and the triharmonic function works on Level 1. Level 2 uses the simplest linear interpolation function. The pseudo code of the video morphing using radial basis functions on grouped layers is listed in VMW Processing 1.

When it comes to interpolating from a curve, the most used interpolation functions are the spline interpolation [59], Bezier curve [3], or the linear interpolation. There are other forms of interpolation, such as interpolation by rational functions, trigonometric polynomials or wavelets, which can be similarly employed in abstract video system. In the approach presented, the cubic spline, the cubic Bezier curve, and the linear interpolation are used for layer-based transition of abstract videos. Therefore,

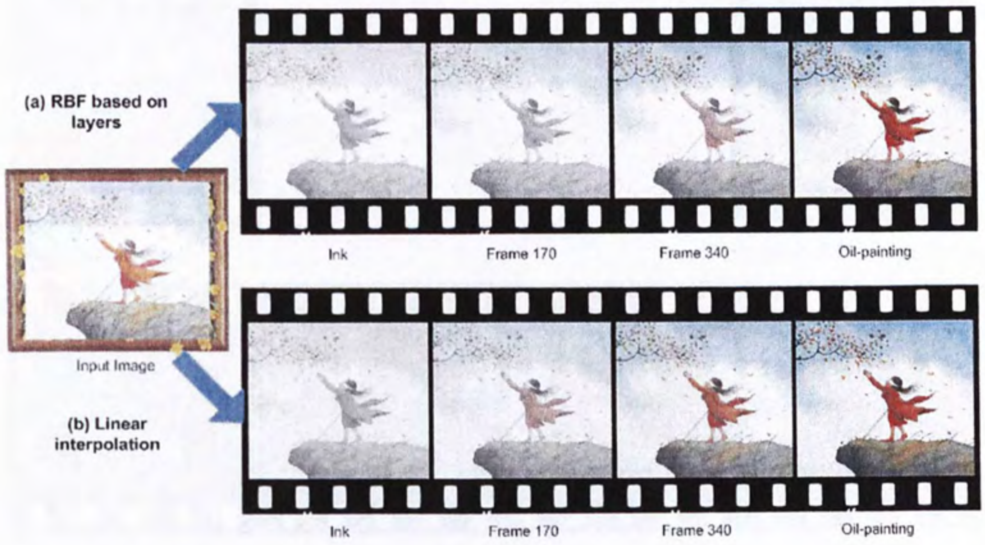


Figure 5.4: Girl: video morphing from ink style to oil-painting style, selected intermediate frames 170 and 340. (a) RBF based on grouped layers (Level 0 - Gaussian function, Level 1 - triharmonic, Level 2 - linear) (b) linear interpolation of the whole image.

VMN Processing 1 *Video_Morphing_Nonlinear_RBF(Input)*

```
$Style_1 \Leftarrow RenderingStyle(Input)$
$Style_2 \Leftarrow RenderingStyle(Input)$
$Layer \Leftarrow Segmentation(Input)$
$Create\_Video(Output\_Video)$
$Time \Leftarrow 0$
while $Time < Output\_Video\_Length$ do
  for $EachPixelPinInput$ do
    if $Layer(P) = 0$ then
      $Frame(P) \Leftarrow Guassion(Style_1(P), Style_2(P))$
    else if $Layer(P) = 1$ then
      $Frame(P) \Leftarrow Triharmonic(Style_1(P), Style_2(P))$
    else
      $Frame(P) \Leftarrow Linear(Style_1(P), Style_2(P))$
    end if
  end for
end while
```

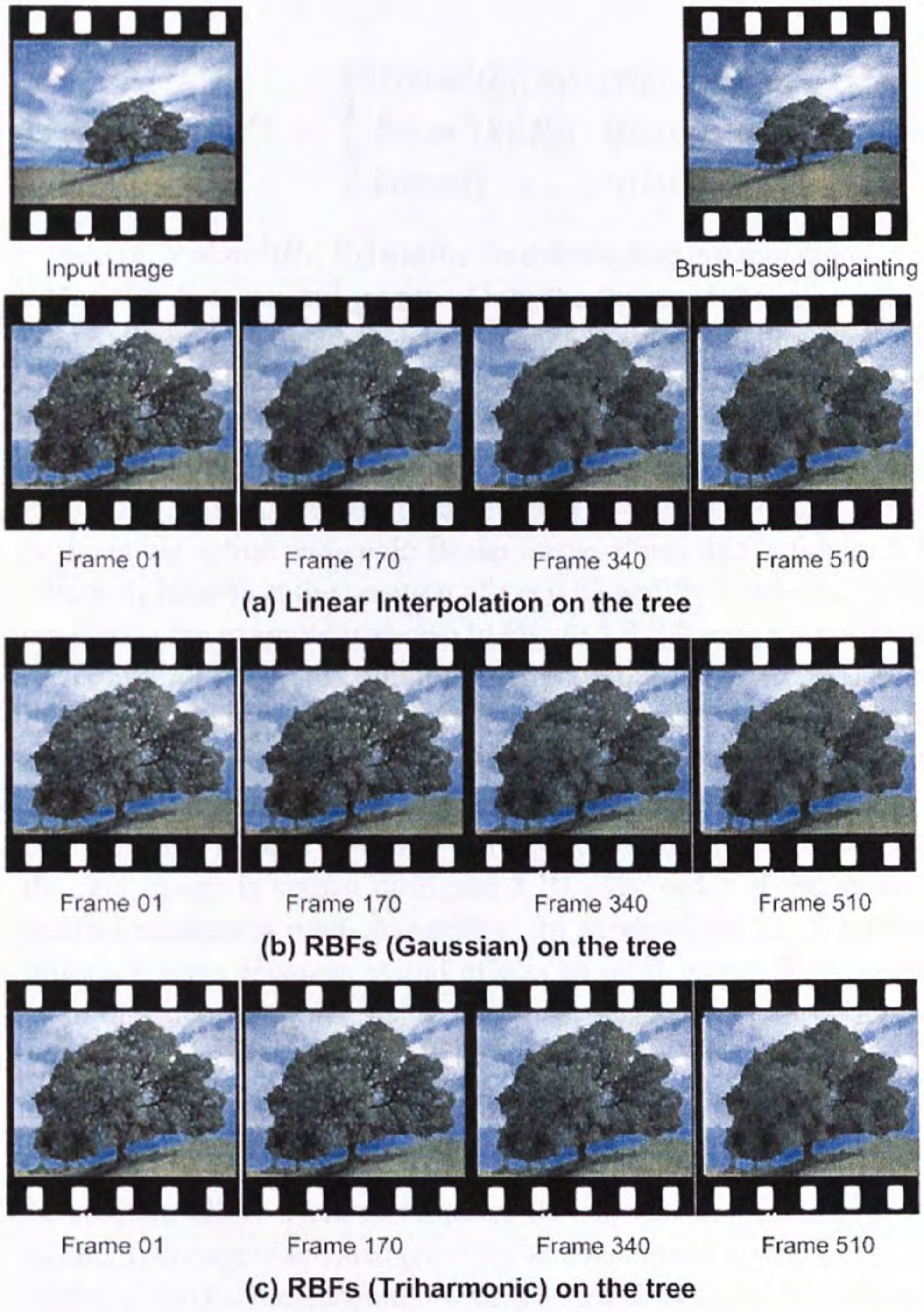


Figure 5.5: Tree: video morphing on level 0 (the tree) from the input image to brush based oil-painting style, selected intermediate frames 170 and 340. (a) the linear interpolation on the tree. (b) the Gaussian RBF is employed to the tree. (c) the Triharmonic RBF is utilized to the tree.

$$f(layer^p) = \begin{cases} Spline^3(R_1, R_2) & if(layer^p = 0) \\ Bezier^3(R_1, R_2) & if(layer^p = 1) \\ Linear() & if(layer^p = 2) \end{cases} \quad (5.9)$$

where, $Spline^3(R_1, R_2)$ means the cubic spline interpolation functions with four control points. Only the intermediate two control points R_1 and R_2 need to be determined, since the start and end control points are from the abstracted images specified by the user. $Bezier(R_1, R_2)$ represents the cubic Bezier curve with two unspecified control points R_1 and R_2. The function *Linear*() is the same with Function (5.6). As default parameters, $R_1 = \widetilde{S^p}(1)$ and $R_2 = \widetilde{S^p}(0)$ in the cubic spline and cubic Bezier curve, illustrated in Figure 5.7, where R_1 locates at the position of $t = 0.33$ and R_2 at the position of $t = 0.67$. One example is shown in Figure 5.8. Moving the positions of R_1 and R_2 generates different interpolation functions, illustrated in Figure 5.9, where R_1 and R_2 are given different positions with the same value in the cubic spline interpolation function. In our examples, the positions of R_1 and R_2 are set to be the default ones, where $t = 0.33$ for R_1 and $t = 0.67$ for R_2. One example on rendering the real image is shown in Figure 5.10. The order of the interpolation functions is inter-changeable. In experiments, the suggested ordering gives pleasure visual effects to most users. The pseudo code of video morphing using non-linear functions (spline, Bezier) on grouped layers is listed in VMW Processing 2.

5.2.3 Interactive System

In order to allow users to visualize the smooth transition in styles in detail, the approach also provides an educational mode, where the output video can be generated with selected abstract level(s) individually or collectively. The interactive controls are designed for users to study the variations in the abstract progressively styles. In Figure

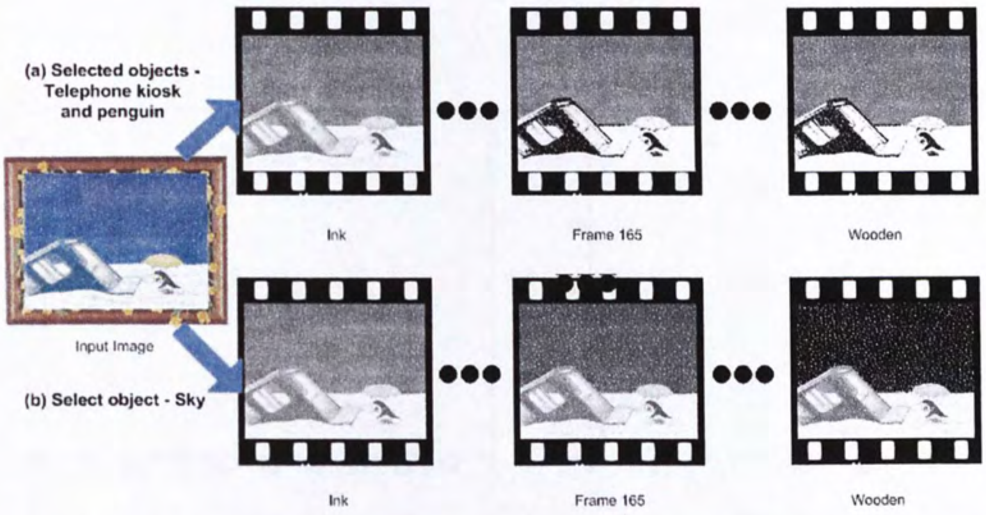
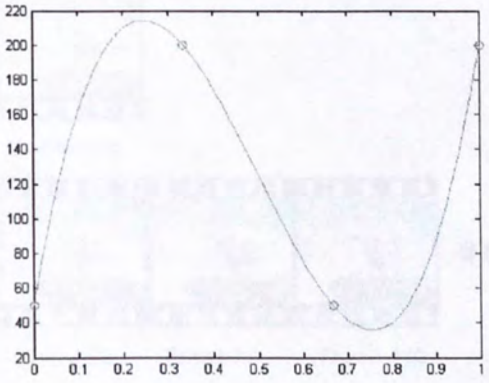
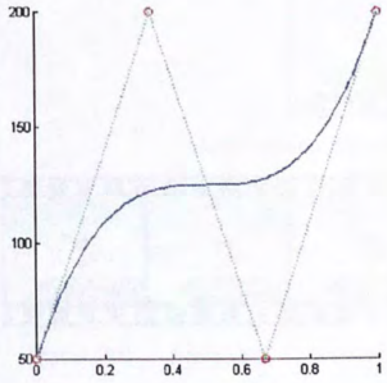


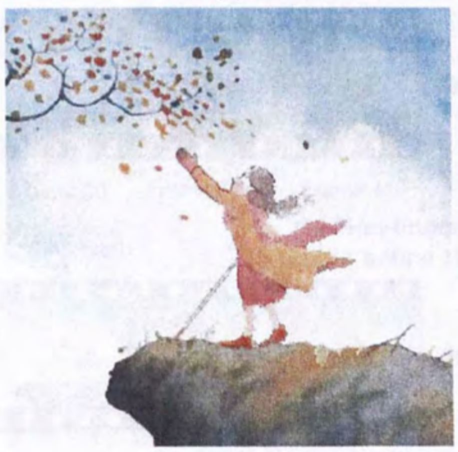
Figure 5.6: Video morphing on selected objects from ink style to wooden style, and intermediate frames 165. (a) telephone kiosk and penguin are specified (cubic spline). (b) the sky is selected (linear interpolation).

VMN Processing 2 *Video_Morphing_Nonlinear_Curve(Input)*

```
$Style_1 \Leftarrow RenderingStyle(Input)$
$Style_2 \Leftarrow RenderingStyle(Input)$
$Layer \Leftarrow Segmentation(Input)$
$Create\_Video(Output\_Video)$
$Time \Leftarrow 0$
while $Time < Output\_Video\_Length$ do
   for $EachPixelPinInput$ do
      if $Layer(P) = 0$ then
         $Frame(P) \Leftarrow CubicSpline(Style_1(P), Style_2(P))$
      else if $Layer(P) = 1$ then
         $Frame(P) \Leftarrow CubicBezier(Style_1(P), Style_2(P))$
      else
         $Frame(P) \Leftarrow Linear(Style_1(P), Style_2(P))$
      end if
   end for
end while
```


(a)


(b)


(c)

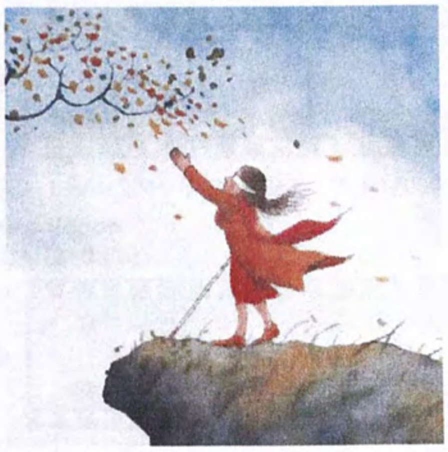

(d)

Figure 5.7: Illustration on the interpolation functions. With the pixel value of 50 in the first style image, and 200 in the second style, $R_1 = 200$, $t = 0.33$ and $R_2 = 50$, $t = 0.67$ in both of the two nonlinear interpolation curves. (a) the result by cubic spline; (b) the result by cubic Bezier curve. Note that in our approach, the interpolation functions are utilized on RGB channel respectively. Take Figure 5.8 for example, (c) is the frame at $t = 0.17$ following the parameter settings of (a), and (d) is the frame at $t = 0.17$ following the parameter settings of (b).

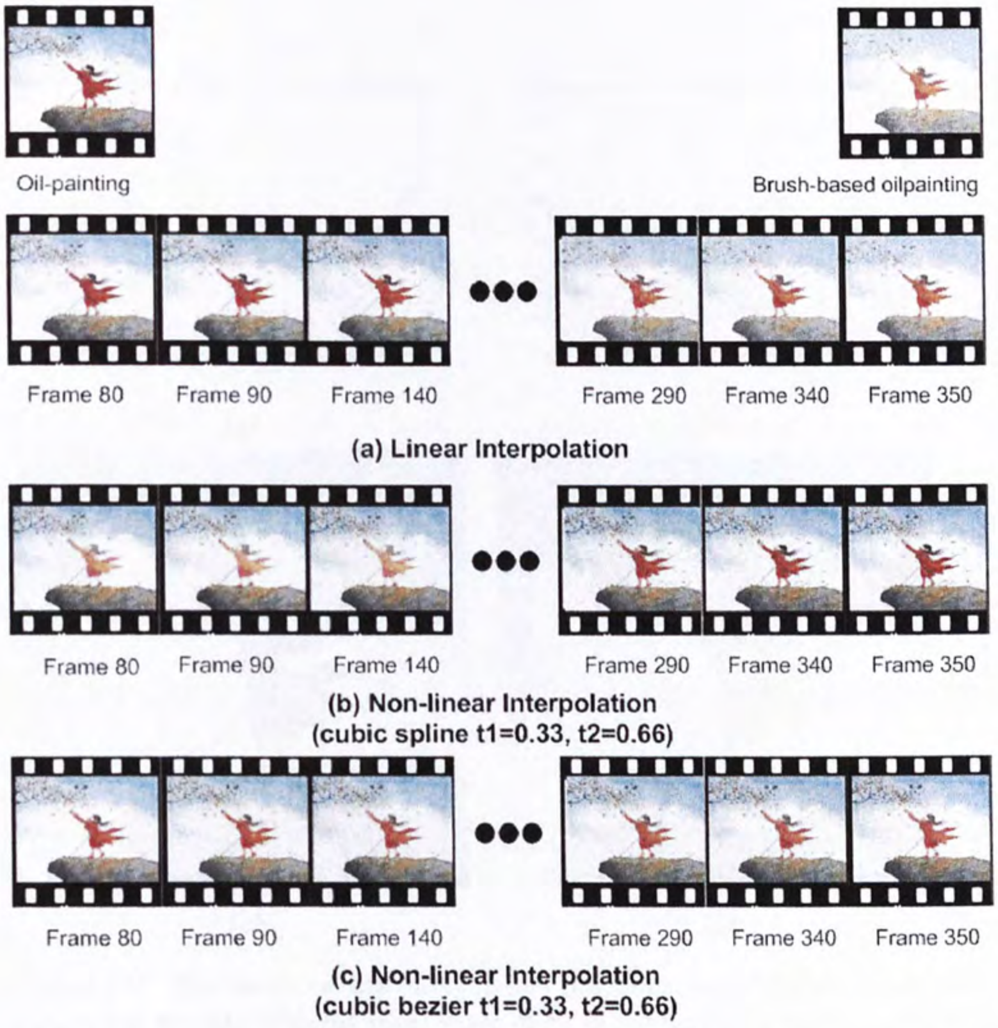
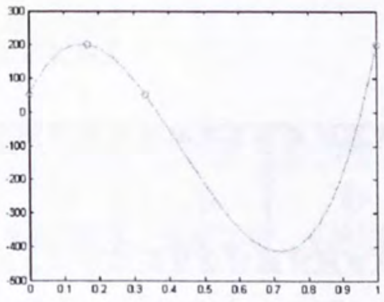
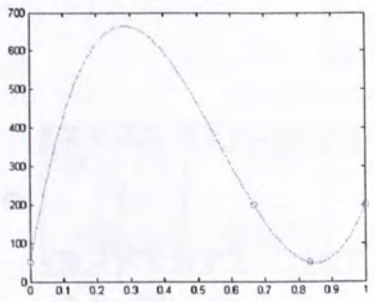


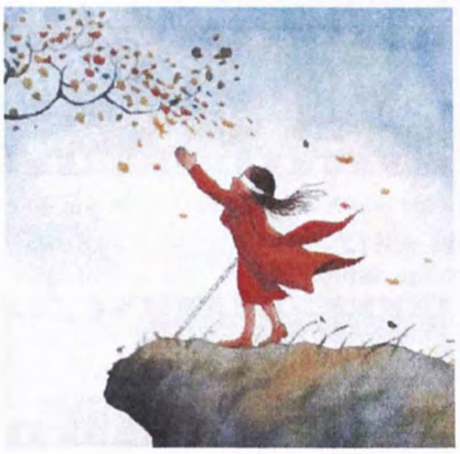
Figure 5.8: Example on the interpolation functions. Between the oil-painting style and brush-based oilpainting style on level 0 - the girl, (a) frames are generated based on the linear interpolation function only, (b) cubic spline function, where $sw_1 = 1$ in function 5.10 at the position $t_1 = 0.33$ and $sw_2 = 0$ in function 5.11 at the position $t_2 = 0.66$, and (c) cubic bezier function, where $bw_1 = 1$ at the position $t_1 = 0.33$, and $bw_2 = 0$ at the position $t_2 = 0.66$ to the control points.



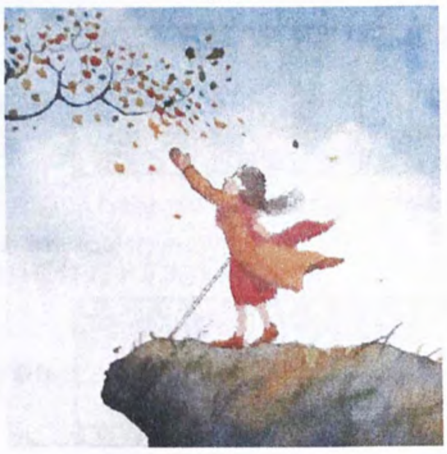
(a)



(b)



(c)



(d)

Figure 5.9: Illustration on the interpolation functions with different positions of the control points. With the pixel value of 50 in the first style image, and 200 in the second style, the result by cubic spline in (a), where $R_1 = 200$, $t = 0.17$ and $R_2 = 50$, $t_1 = 0.33$, is different from the result in (b), where $R_1 = 200$, $t_2 = 0.67$ and $R_2 = 50$, $t = 0.83$. Take Figure 5.10 for example, (c) is the frame at $t = 0.17$ following the parameter settings of (a), and (d) is the frame at $t = 0.17$ following the parameter settings of (b).

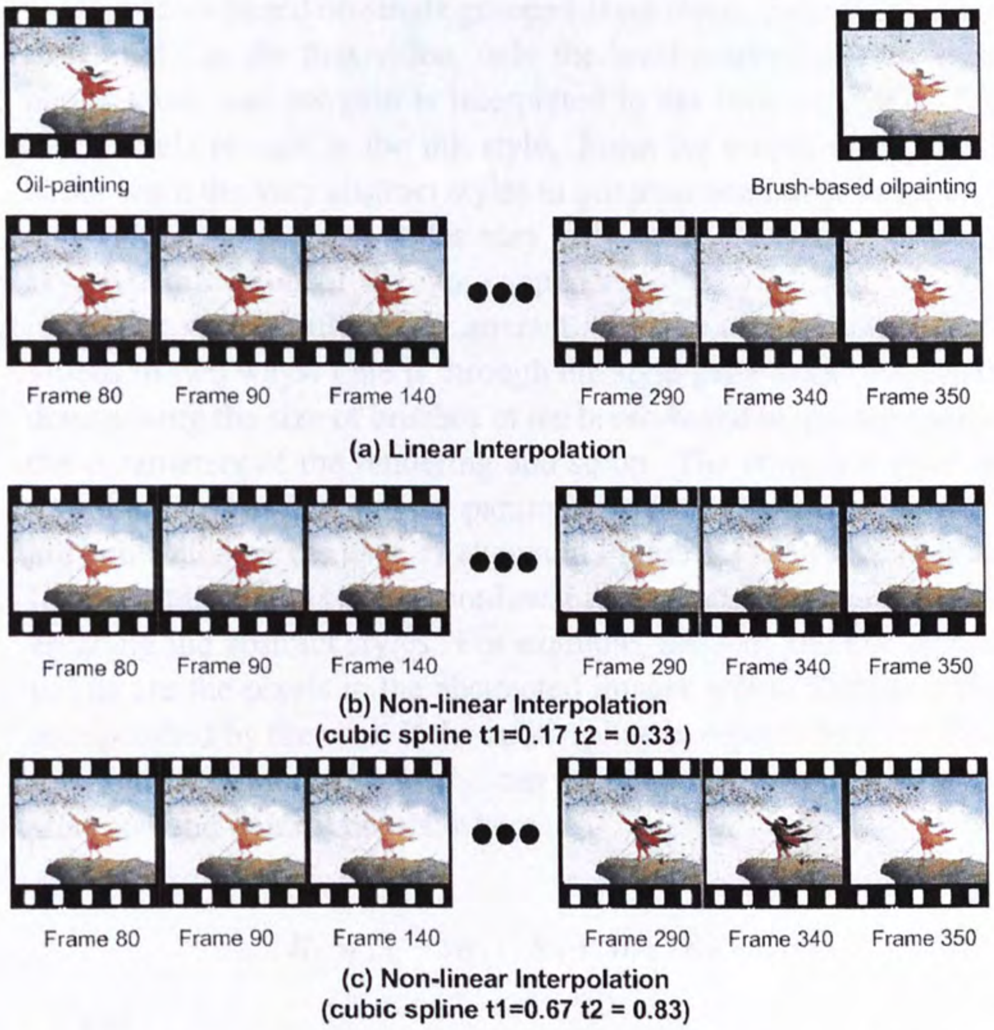


Figure 5.10: Example on the interpolation functions with different positions of the control points. Between the oil-painting style and brush-based oilpainting style on level 0 - the girl, (a) frames are generated based on the linear interpolation function only, (b) cubic spline function, where $sw_1 = 1$ in function 5.10 at the position $t_1 = 0.17$ and $sw_2 = 0$ in function 5.11 the position $t_1 = 0.17$, and (c) cubic spline function, $sw_1 = 1$ in function 5.10 at the position $t_1 = 0.66$ and $sw_2 = 0$ in function 5.11 the position $t_1 = 0.83$.

5.6, an example of the educational mode of the system is given . The ink and wooden styles are first selected by users. In the education mode, videos based on single grouped-layer (level) interpolation are generated. In the first video, only the level representing the telephone kiosk and penguin is interpreted in the abstract styles. The other levels remain in the ink style. From the output video, users better learn the vary abstract styles in progress smoothly. In the second video, the abstract styles vary only the level representing the sky, dynamic effect of sky more apparent.

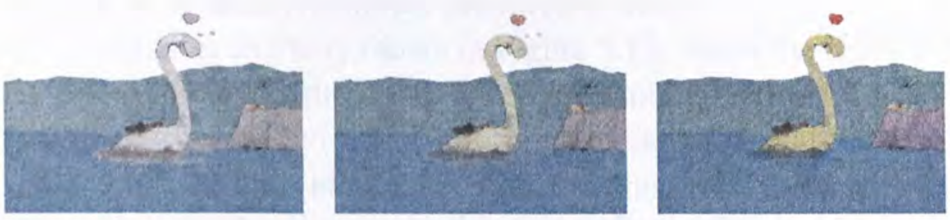
In our system, users can interact with the creation of abstract videos in two ways. One is through the style-generation process, in determining the size of brushes in the brush-based oil painting style, the parameters of the rendering and so on. The other is the video-morphing process, where the parameters for interpolation functions are controlled by the user, as shown in Figure 5.11. Users can specify the control points for the nonlinear interpolation functions by referencing the abstract styles. For example, the start and end control points are the pixels in the abstracted images whose abstract styles are specified by the user. If the cubic spline interpolation is selected, the control points, R_1 and R_2, can be determined by referring the start and end control points, where

$$R_1 = (1 - sw_1) \cdot R_0 + sw_1 \cdot R_3 \tag{5.10}$$

and

$$R_2 = (1 - sw_2) \cdot R_0 + sw_2 \cdot R_3 \tag{5.11}$$

where the weights sw_1 and sw_2 are set by users, and $sw_1 = 1$, $sw_2 = 0$ by default. Similarly when the cubic Bezier curve is selected, where the weights for the two unspecified control points are represented by bw_1 and bw_2, and $bw_1 = 1$, $bw_2 = 0$ by default.



(a) (b) (c)

Figure 5.11: User interactions affect the rendering result on video morphing. With the same input image and rendering styles (brush-based oil-painting style for the first frame, and ink style for the last frame), different multi-style abstract frames are produced. (a) with the user specified parameters $sw_1 = 0.8$, $sw_2 = 0.2$, $bw_1 = 0.8$, and $bw_2 = 0.2$; (b) in-between, user specified parameters $sw_1 = 0.5$, $sw_2 = 0.5$, $bw_1 = 0.5$, and $bw_2 = 0.5$; (c) with the user specified parameters $sw_1 = 0.2$, $sw_2 = 0.8$, $bw_1 = 0.2$, and $bw_2 = 0.8$;

5.3 Interactive Videos

Our interactive videos system is developed on an Intel(R) Pentium(R) 4 CPU 3.20GHz with 1GB RAM, Windows XP, using Visual C++ .Net 2008. Performance depends on image size and the selected abstract styles. Typical time for a $640 * 480$ image and default parameters of an abstraction selected is about $3 - 4$ seconds. In our experiments, the parameters for the oil-painting and watercolor rendering are set as following: for the image segmentation, the smoothing term $\sigma = 0.1$, the threshold on segmentation $k = 500$, the weight to control the minimum size of the grouped layers $d = 0.1$; for the bilateral filter in the abstraction, the blur radius $\sigma_d = 3.0$, and $\sigma_r = 0.1$, which determines how contrasts are preserved or blurred.

Figure 5.12 shows one interactive video example to illustrate the approach. The input image is rendered with the ink style first, then the linear/nonlinear interpolations, combined with the cubic spline and cubic Bezier Curve for grouped layers are applied for more emphasis on the smooth transition from the input image to an abstract image style such as ink. Note that in Frame 32, Frame 187 and Frame 357, the hat, neckerchief and overcoat are rendered in the transition of linear/nonlinear interpreted videos.

Another example is shown in Figure 5.13, where the input image is first rendered in the brush-based oil-painting and the watercolor styles. In the video using linear interpolation, the styles change smoothly. In the video using linear/nonlinear layer-based output, the tree is rendered more as the watercolor in Frame 170, and the sky is more abstracted in the oil-painting style. In Frame 340, when the sky is more abstracted into the watercolor style, the oil-painting style is more apparent on the tree. The multi-style images in abstract video is pleasure to visualize and flexible to control by users. This example shows a good illustration of how the system offers interactive control to users in multi-style abstraction rendering.

The illustration of the interactive videos is also shown in Figure 5.14, where the ink and brush-based oil-painting styles are first specified. The weights of control points R_1 and R_2 in the cubic spline interpolation are set at $sw_1 = 0.25$ and $sw_2 = 0.75$. The weights of the control points R_1 and R_2 in the cubic Bezier curve interpolation are $bw_1 = 0.75$ and $bw_2 = 0.25$. From the output videos, we can see that the rendering effect from the linear interpolation is largely different from the result of interactive video, especially on the hair of the boy. Figure 5.14 gives another illustration on the controllable parameters, where the cubic spline interpolation are set at $sw_1 = 0.75$ and $sw_2 = 0.75$, and the weights of the control points R_1 and R_2 in the cubic Bezier curve interpolation are $bw_1 = 0.25$ and $bw_2 = 0.25$. Figure 5.15 gives the comparison with the method presented in [8], where $sw_1 = 1$, $sw_2 = 0$, $bw_1 = 1$, and $bw_2 = 0$ in video morphing. Our result is good enough to generate multi-style images as well as the results from the state of art.

5.4 Summary

In this chapter, we have developed the interactive system for creating multi-style abstract videos. Our system supports multi-style

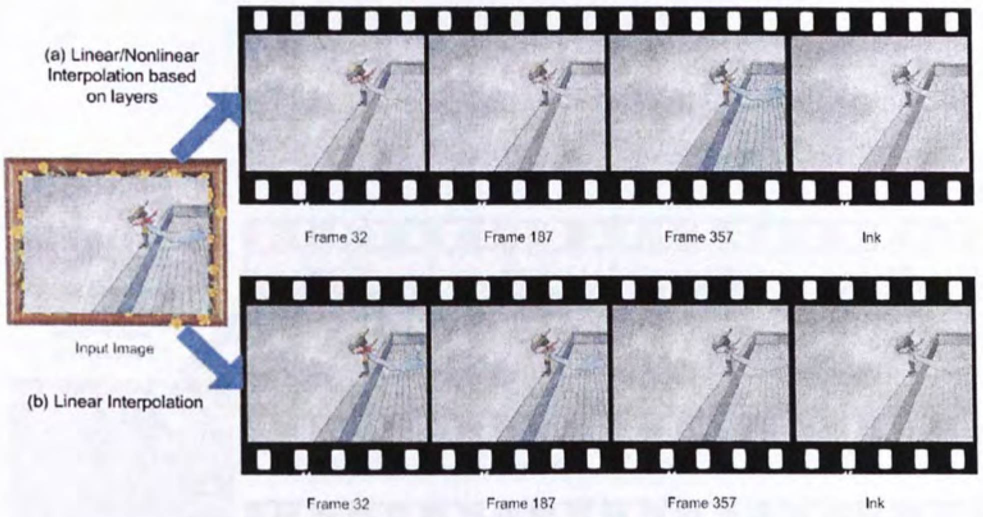


Figure 5.12: Wendy: video morphing from the input image to ink style, selected intermediate frames 32, 187, 357. (a) linear/nonlinear interpolation based on grouped layers (Level 0 - cubic spline, Level 1 - cubic Bezier, Level 2 - linear). (b) linear interpolation of the whole image.

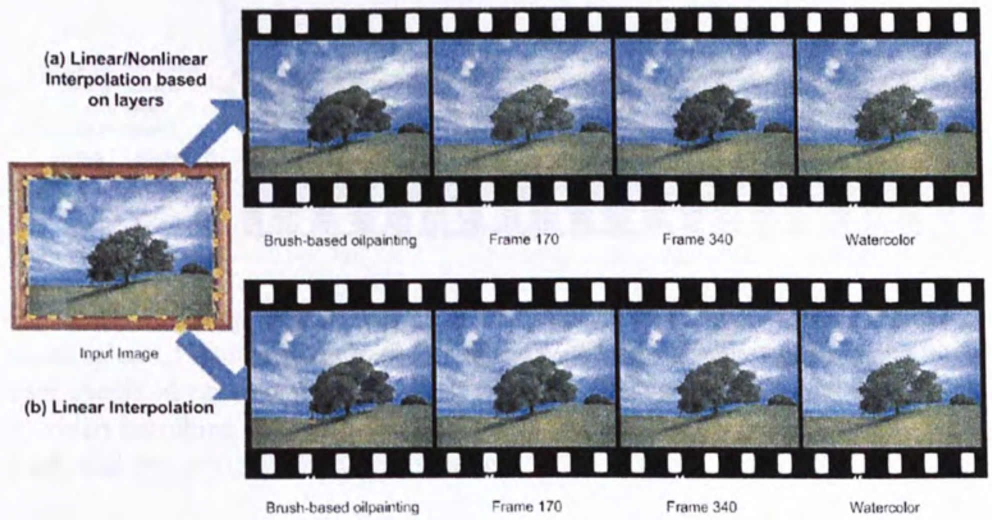


Figure 5.13: Tree on the grass: video morphing from brush-based oil-painting style to watercolor style, selected intermediate frames 170 and 340. (a) linear/nonlinear interpolation based on grouped layers (Level 0 - cubic spline, Level 1 - cubic Bezier, Level 2 - linear) (b) linear interpolation of the whole image.

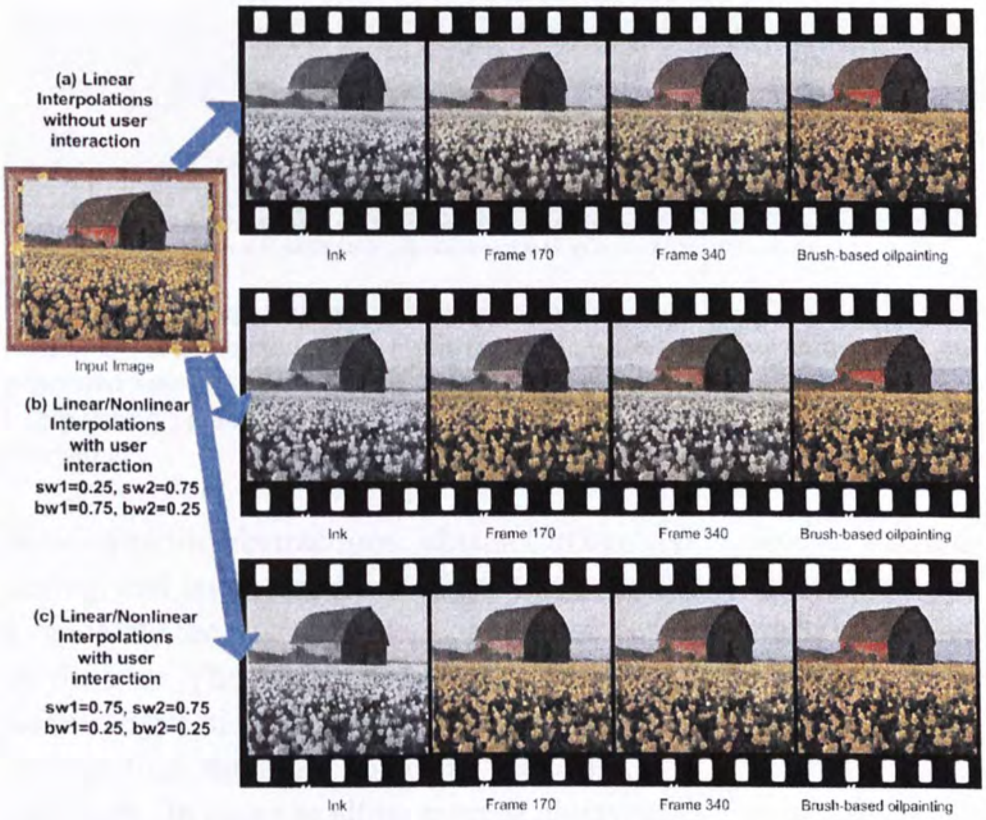

(a) Linear Interpolations without user interaction
Input Image
(b) Linear/Nonlinear Interpolations with user interaction
sw1=0.25, sw2=0.75
bw1=0.75, bw2=0.25
(c) Linear/Nonlinear Interpolations with user interaction
sw1=0.75, sw2=0.75
bw1=0.25, bw2=0.25
Ink
Frame 170
Frame 340
Brush-based oilpainting

Figure 5.14: House: video morphing from ink style to brush-based oil-painting style, selected intermediate frames 170 and, 340. (a) without user interaction. (b) user **specified** parameters $sw_1 = 0.25$, $sw_2 = 0.75$, $bw_1 = 0.75$, and $bw_2 = 0.25$ in video morphing. (c) user **specified** parameters $sw_1 = 0.75$, $sw_2 = 0.75$, $bw_1 = 0.25$, and $bw_2 = 0.25$ in video morphing.

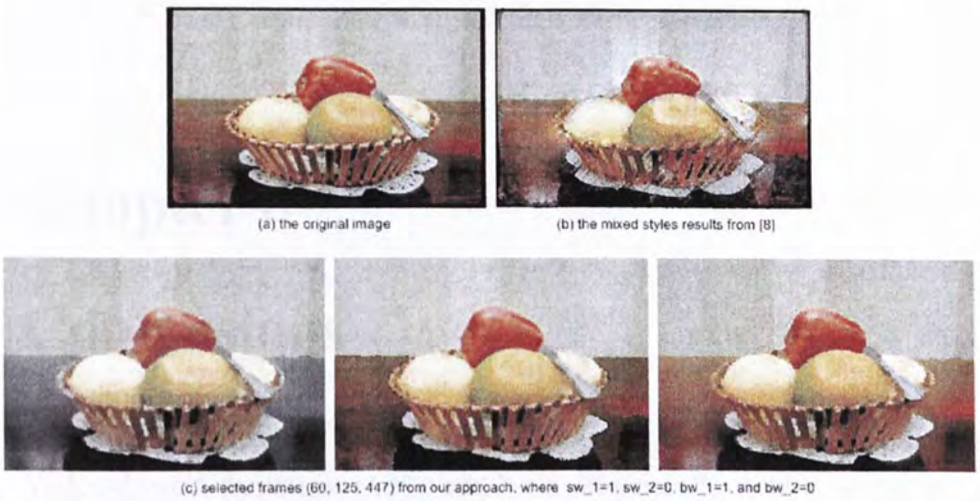

(a) the original image
(b) the mixed styles results from [8]
(c) selected frames (60, 125, 447) from our approach, where sw_1=1, sw_2=0, bw_1=1, and bw_2=0

Figure 5.15: Comparison with the method presented in [8]. (a) is the original image. (b) is the mixed styles result from [8]. (c) list selected frames from our generated video, where ink watercolor and oil-painting style are rendered. The interpolation parameters $sw_1 = 1$, $sw_2 = 0$, $bw_1 = 1$, and $bw_2 = 0$ in video morphing.

stitic/dynamic abstractions: abstract image style selection and rendering, and layer-based linear/non-linear interpolations of the styles to generate the abstract videos, which can be interactively controlled by the user. The goal of our system is different from that of conventional abstraction rendering systems, where more multi-style renderings from the input image are generated as abstract videos in our approach. In order to allow more static/dynamic abstract styles, we will extend the current work to other non-photorealistic rendering styles, such as charcoal or pastel, and seamlessly fuse more abstract styles into the input image or video clips.

□ **End of chapter.**

Chapter 6

Conclusions

Video-based dynamic scene analysis, especially video retargeting, and abstraction rendering are of great importance in interactive game development, non-photorealistic rendering, virtual reality, and augmented image/video processing. Video retargeting systems focus on narually condense the source video and optimally fit it into the target window by directly scaling the video, or using the technique of virtual pan. In most image rendering systems, the rendering style is predefined. Users have very limited power to modify the parameters to make rendering effects better. In addition, over-rendering phenomena are common since all the pixels are rendered with the same rule.

In this thesis, we have explored the state-of-art methodologies in video-based dynamic scene analysis and abstraction rendering. We presented a thorough study on the recent advances in the related research topics. We proposed a novel video migration approach based on active windows, aiming to reduce the spatial resolution of an input video without losing important motion details. Our approach consists of the three integrated steps: foreground extraction, active window initialization, and optimization using these processes. The retargeted videos from the experimental results illustrate that our work is effective and widely applicable for sports or surveillant video migration on mobile devices.

Further, we have worked out a novel approach for multi-style abstract image rendering, using fine-scale selective styles and the abstraction processing. Using the layer-based processing, our approach can render images with selective styles and avoid over rendering the important detailed objects by assigning more variable weights on layer-based information. Our approach is generally applicable for image rendering to preserve small but vital objects well. Moreover, we have developed a systematic approach for interactive multi-style images/videos. Our system supports the static/dynamic abstractions: abstract styles selection and rendering, and layer-based abstract videos of selected styles to generate the multi-style abstraction. The goal of our system is different from that of conventional multi-style image rendering systems, where in-between multi-style renderings from the input image are generated as videos, which can be interactively controlled by users. The experimental results showed that our interactive framework is flexible to control, and generally applicable for image re-rendering to preserve the small but vital objects well, and also provide an interactive form of dynamic-media edutainment.

We will further work on video retargeting applications using window oriented mobile devices, such as mobile phone, PDA, palm computer, and the like. GPU-based image/video abstraction is another interesting research topic. When the rendering parameters are set, the in-between frames can be produced at the same time, so that it is possible for the real-time abstract video system. We will further investigate to extend our system to more non-photorealistic rendering styles.

□ **End of chapter.**

Appendix A

List of Publications

1. Chenjun TAO, Jianbin Shen, Hanqiu SUN, Jiaya JIA, Interactive Multi-Style Abstract Video. Submitted to the Fourth International Conference on Games Research and Development 2008 (CyberGames 2008).
2. Chenjun TAO, Jianbin Shen, Hanqiu SUN, Jiaya JIA, Multi-Style Abstract Image Rendering. Computer Graphics International 2008.
3. Chenjun TAO, Jiaya JIA, Hanqiu SUN, Active Window Oriented Dynamic Video Re-targeting. Workshop on dynamical vision at ICCV 2007, (lecture notes in CS), 2007.
4. Dynamic Color Texture Synthesis in the YUV Color Sapce, Leilei Xu, Hanqiu Sun, Jiaya Jia, Chenjun Tao, In Proceedings of the International Conference on Entertainment Computing 2007 (ICEC07), Lecture Notes in Computer Science, Springer, 2007.
5. Generalized Linear Perspective Shadow Map Reparameterization, Fan Zhang, Leilei Xu, Chenjun Tao, Hanqiu Sun, in Proceedings of ACM/SIGGRAPH VRCIA 2006, Published by ACM SIGGRAPH, 14-17 June, 2006.

□ **End of chapter.**

Appendix B

Optical flow

Optical flow is the velocity field which warps one image into another (usually very similar) image. From an image sequence, the optical flow is defined as follows: for every pixel, a velocity vector $V = (u, v)$ is found which says [47]:
(1) how quickly whatever is in that pixel is moving across the image,
(2) direction it is moving.
With assumptions that brightness constancy, spatial coherence and temporal persistence, the optical flow can be calculated as follows:
Let $I^{t+1}(x, y)$ be the intensity of the pixel (x, y) in frame $t + 1$, and I^t be the one in frame t. u and v are the displacements of the pixel in two directions.
Then, by the Taylor series expansion of I,

$$I^{t+1}(x, y) = I^t(x + u, y + v) \approx I^t(x, y) + I_x u + I_y v \tag{B.1}$$

which is an equivalence relation with

$$0 = I_t(x, y) + I_x u + I_y v = I_t(x, y) + \nabla I(P_i)[u, v] = I_t(P_i) + \nabla I(P_i)[u, v] \tag{B.2}$$

where $P_i = (x_i, y_i)$
As different pixels have diverse displacements (u,v), it is impossible to calculate B.2 without additional constrains.
One of the methods on calculating the optical flow of an image sequence is Lukas-Kanade flow, which was proposed by Lukas and

Kanade in 1981. Some constraints are added to solve the aperture problem and get more equations for a pixel such as

- most commonly to assume that the flow field is smooth locally.
- pretend the pixel's neighbors have the same displacement (u,v).

If the window size is 5×5, then there are 25 equations to each pixel for gray scale image.

$$\underbrace{\begin{bmatrix} I_x(P_1) & I_y(P_1) \\ I_x(P_2) & I_y(P_2) \\ \vdots & \vdots \\ I_x(P_25) & I_y(P_25) \end{bmatrix}}_{A_{25\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d_{2\times 1}} = - \underbrace{\begin{bmatrix} I_t(P_1) \\ I_t(P_2) \\ \vdots \\ I_t(P_25) \end{bmatrix}}_{b_{25\times 1}} \tag{B.3}$$

Then, by solving the function

$$\underbrace{(A^T A)}_{2\times 2} \underbrace{d}_{2\times 1} = \underbrace{A^T b}_{2\times 1} \tag{B.4}$$

which is

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A)} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{d} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b} \tag{B.5}$$

the optical flow over all pixels in the $K \times K$ window can be calculated.
Optical flow is always used as a tool for segmentation in image processing or video processing, i.e. [22], which aims at recognition of human actions.

□ **End of chapter.**

Appendix C

Belief Propagation

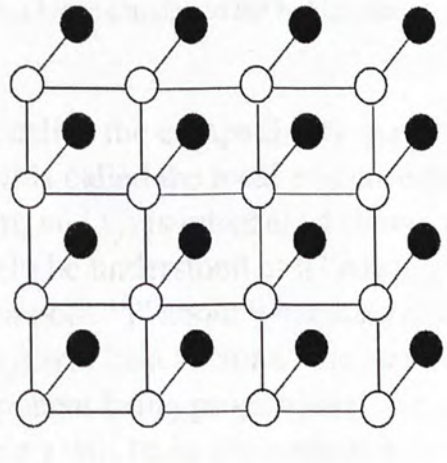


Figure C.1: A square lattice pairwise Markov Random Field.

Belief Propagation is a dynamic programming approach to answering conditional probability queries in a graphical model, pairwise Markov network, as shown in Figure C.1. White nodes x_i are hidden variables and black nodes y_i are observed variables. The joint probability distribution for the unknown variables x_i under this graph model would be:

$$p(\{x\}) = \frac{1}{Z} \prod_{ij} \psi_{ij}(x_i, x_j) \prod_{i} \phi_i(x_i, y_i) \qquad \text{(C.1)}$$

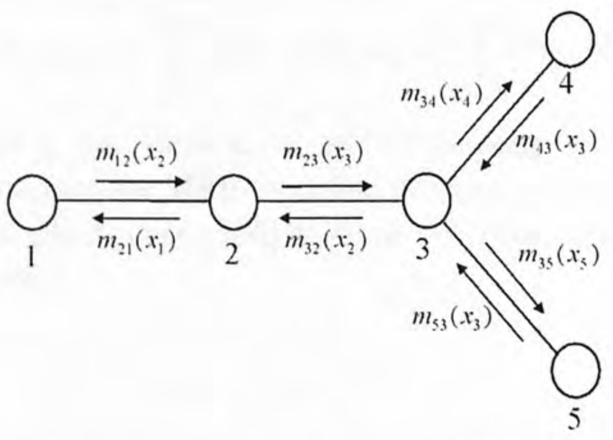


Figure C.2: An illustration of the messages passed in BP. $m_{ij}(x_j)$ can be thought of as a "message" from a hidden node i to the hidden node j about what state node "j" should be in.

where $\psi_{ij}(x_i, x_j)$ is called the compatibility matrix between nodes x_i and x_j, and $\phi_i(x_i, y_i)$ is called the local evidence for node x_i.
In the BP algorithm, $m_{ij}(x_j)$ is introduced shown in Figure C.2, which can intuitively be understood as a "message" from a hidden node i to the hidden node "j" about what state node j should be in. The message $m_{ij}(x_j)$ will be a vector of the same dimensionality as x_j, with each component being proportional to how likely node i thinks it is that node j will be in the corresponding state. In the BP algorithm, the belief at a node i is proportional to the product of the local evidence at that node $\phi_i(x_i, y_i)$, and all the messages coming into node i:

$$b_i(x_i) = k\phi_i(x_i, y_i) \prod_{j \in N(i)} m_{ji}(x_i) \tag{C.2}$$

where k is a normalization constant (the beliefs must sum to 1) and $N(i)$ denotes the nodes neighboring i. The messages are determined self-consistently by the message update rules:

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i, y_i)\psi_{ij}(x_i, x_j) \prod_{k\in N(i)\backslash j} m_{ki}(x_i) \tag{C.3}$$

Finally, starting with some initial set of messages, the BP algorithm could approximate the MRF-modelled problem in polynomial time. It has been applied successfully to some decoding algorithms and vision problems.

□ **End of chapter.**

Bibliography

[1] S. P. Alex Rav-Acha, Yael Pritch. Making a Long Video Short: Dynamic Video Synopsis. *Computer Vision and Pattern Recognition, 2006. CVPR 2006. Proceedings of the 2006 IEEE Computer Society Conference on*, 2006.

[2] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. *European Conference on Computer Vision*, 588:237–252, 1992.

[3] P. Bézier. *Numerical Control; Mathematics and Applications*. John Wiley & Sons, 1972.

[4] R. Bolles, H. Baker, and D. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1(1):7–55, 1987.

[5] A. Bousseau, M. Kaplan, J. Thollot, and F. Sillion. Interactive watercolor rendering with temporal coherence and abstraction. *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 141–149, 2006.

[6] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin. Video watercolorization using bidirectional texture advection. *ACM Transactions on Graphics (TOG)*, 26(3), 2007.

[7] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV (1)*, pages 377–384, 1999.

[8] M. Braunstein. *Painting Various Subjects with Mixed Media.* Barrons Educational Series, 1996.

[9] S. Brooks. Mixed-Media Painting and Portraiture. *IEEE Transactions on Visualization and Computer Graphics*, pages 1041–1054, 2007.

[10] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *ECCV*, 3024:25–36, 2004.

[11] Y. Cai, N. de Freitas, and J. J. Little. Robust visual tracking for multiple targets. *ECCV*, 2006.

[12] P. N. J. S. S. Carlsson. Multi-Target Tracking C Linking Identities using Bayesian Network Inference. *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2006.

[13] Y. Caspi, A. Axelrod, Y. Matsushita, and A. Gamliel. Dynamic stills and clip trailers. In *Pacific Graphics*, 2006.

[14] Y. Caspi and M. Irani. A Step towards sequence-to-sequence alignment. *PROC IEEE COMPUT SOC CONF COMPUT VISION PATTERN RECOGNIT*, 2:682–689, 2000.

[15] N. Chu. MoXi: real-time ink dispersion in absorbent paper. *ACM Transactions on Graphics (TOG)*, pages 504–511, 2005.

[16] T. Collins. Analysing video sequences using the spatio-temporal volume. In *MSc Informatics Research Review*, 2004.

[17] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. *CVPR*, pages 53–60, 2006.

[18] J. Davis and H. Gao. Recognizing human action efforts: An adaptive three-mode PCA framework. *Proceedings of IEEE*

International Conference on Computer Vision, pages 1463–1469, 2003.

[19] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. *ACM Transactions on Graphics (TOG)*, 21(3):769–776, 2002.

[20] N. Doulamis, A. Doulamis, Y. Avrithis, and S. Kollias. Video content representation using optimal extraction of frames andscenes. *ICIP*, 1, 1998.

[21] S. Dye. *The Mixed Media Source Book: Techniques for Successfully Combining Painting and Drawing Mediums*. Watson-Guptill, 2004.

[22] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 726–733, 2003.

[23] X. Fan, X. Xie, H.-Q. Zhou, and W.-Y. Ma. Looking into video frames on small displays. *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 247–250, 2003.

[24] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[25] D. B. Goldman, B. Curless, S. M. Seitz, and D. Salesin. Schematic storyboarding for video visualization and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), 2006.

[26] Y. Gong and X. Liu. Video summarization using singular value decomposition. *CVPR*, 2:174–180, 2000.

[27] B. Gooch and A. Gooch. *Non-Photorealistic Rendering*. AK Peters, Ltd. Natick, MA, USA, 2001.

[28] P. Haeberli. Paint by numbers: abstract image representations. *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 207–214, 1990.

[29] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, 1998.

[30] A. Hertzmann. Fast paint texture. *Proceedings of the 2nd international syumposium on Non-photorealistic animation and rendering*, pages 91–96, 2004.

[31] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. *International Conference on Computer Graphics and Interactive Techniques: Proceedings of the 28 th annual conference on Computer graphics and interactive techniques*, 2001:327–340, 2001.

[32] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 7–12, 2000.

[33] X. T.-X.-Q. C. Hong-Wen Kang, Yasuyuki Matsushita. Space-Time Video Montage . *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2:1331–1338, 2006.

[34] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. Efficient representations of video sequences and their applications. *Signal Processing: Image Communication*, 8(4):327–351, 1996.

[35] J. Jia, T. Wu, Y. Tai, and C. Tang. Video repairing: Inference of foreground and background under severe occlusion. *Computer Vision and Pattern Recognition, 2004. CVPR 2004.*

Proceedings of the 2004 IEEE Computer Society Conference on, 1, 2004.

[36] H.-W. Kang, X.-Q. Chen, Y. Matsushita, and X. Tang. Space-time video montage. In *CVPR*, pages 1331–1338, 2006.

[37] A. Kasao and K. Miyata. Algorithmic Painter: a NPR method to generate various styles of painting. *The Visual Computer*, 22(1):14–27, 2006.

[38] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. *CVPR*, pages 407–414, 2005.

[39] M. Kumar, P. Torr, and A. Zisserman. Learning Layered Motion Segmentations of Video. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 1, 2005.

[40] R. C. L. Davis. Real-time periodic motion detection, analysis, and applications. In *IEEE Transactions on pattern analysis and machine intelligence*, page 781, 2000.

[41] Y. Li, J. Sun, and H. Shum. Video object cut and paste. *Proceedings of ACM SIGGRAPH 2005*, 24(3):595–600, 2005.

[42] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 23, 2004.

[43] P. Litwinowicz. Processing images and video for an impressionist effect. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 407–414, 1997.

[44] F. Liu and M. Gleicher. Video retargeting: automating pan and scan. *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 241–250, 2006.

[45] H. Luo and J. Fan. Concept-oriented video skimming via semantic video classification. *ACM international conference on Multimedia*, pages 760–761, 2004.

[46] Y.-F. Ma, X.-S. Hua, L. Lu, and H.-J. Zhang. A generic framework of user attention model and its application in video summarization. *IEEE Transactions on Multimedia*, 7(5):907–919, 2005.

[47] A. Marshall. Optical flow. In *Vision Systems*, 1994.

[48] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang. Automatic video summarization by graph modeling. In *ICCV*, page 104, 2003.

[49] M. I. Oren Boiman. Detecting irregularities in images and in video. In *ICCV05*, 2005.

[50] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, California, 1988.

[51] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[52] D. Ramanan, D. Forsyth, and A. Zisserman. Strike a pose: tracking people by finding stylized poses. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1, 2005.

[53] C. Rao, A. Yilmaz, and M. Shah. View-Invariant Representation and Recognition of Actions. *International Journal of Computer Vision*, 50(2):203–226, 2002.

[54] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg. Dynamosaics: Video mosaics with non-chronological time. *Proceedings of CVPR*, 2005.

[55] A. Rav-Acha, Y. Pritch, and S. Peleg. Making a long video short: Dynamic video synopsis. In *CVPR*, pages 435–441, 2006.

[56] S. Ross. *Paint Radiant Realism in Watercolor, Ink and Colored Pencil.* North Light Books, 2000.

[57] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cut. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, pages 309–314, 2004.

[58] T. Saito and T. Takahashi. Comprehensible Rendering of 3-D Shapes. *Computer Graphics*, 24(4), 1990.

[59] L. Schumaker. *Spline functions: basic theory.* Wiley, New York, 1981.

[60] E. Shechtman and M. Irani. Space-Time Behavior Based Correlation. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. Proceedings of the 2005 IEEE Computer Society Conference on*, 2005.

[61] J. Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[62] M. Shiraishi and Y. Yamaguchi. An algorithm for automatic painterly rendering based on local source image approximation. *Non-Photorealistic Animation and Rendering: Proceedings of the 1 st international symposium on Non-photorealistic animation and rendering*, 5(07):53–58, 2000.

[63] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *CVPR*, 2:246–252, 1999.

[64] J. Sullivan and S. Carlsson. Tracking and labelling of interacting multiple targets. *ECCV*, 2006.

[65] J. Sun, W. Zhang, X. Tang, and H.-Y. Shum. Background cut. *ECCV*, pages 628–641, 2006.

[66] S. Treavett and M. Chen. Statistical Techniques for the Automated Synthesis of Non-Photorealistic Images. *Proc. 15th Eurographics UK Conference*, 1997.

[67] R. Vidal and D. Singaraju. A closed form solution to direct motion segmentation. *CVPR*, 2005.

[68] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. Cohen. Interactive video cutout. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2005.

[69] J. Wang, P. Bhat, R. Colburn, M. Agrawala, and M. Cohen. Interactive video cutout. *Proceedings of ACM SIGGRAPH 2005*, 24(3):585–594, 2005.

[70] J. Wang, M. J. Reinders, R. L. Lagendijk, J. Lindenberg, and M. S. Kankanhalli. Video content representation on tiny devices. *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, pages 1711– 1714 Vol.3, 2004.

[71] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 1, 2004.

[72] J. Wills, S. Agarwal, and S. Belongie. What went where. In *CVPR*, pages 37–44, 2003.

[73] H. Winnemöller, S. Olsen, and B. Gooch. Real-time video abstraction. *ACM Transactions on Graphics (TOG)*, 25(3):1221–1226, 2006.

[74] M. Yang, N. Ahuja, and M. Tabb. Extraction of 2D motion trajectories and its application to hand gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(8):1061–1074, 2002.

[75] X. T. H.-Y. S. Yasuyuki Matsushita, Eyal Ofek. Full-time video stabilization. *Computer Vision and Pattern Recognition, 2006. CVPR 2005. Proceedings of the 2005 IEEE Computer Society Conference on*, 2006.

[76] A. Yilmaz and M. Shah. Recognizing Human Actions in Videos Acquired by Uncalibrated Moving Cameras. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 1, 2005.

[77] X. Zhu, X. Wu, J. Fan, A. K. Elmagarmid, and W. G. Aref. Exploring video content structure for hierarchical summarization. *Multimedia Syst.*, 10(2):98–115, 2004.