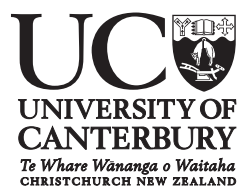


Artistic Content Representation and Modelling based on Visual Style Features

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Doctor of Philosophy
by
Philip Buchanan



University of Canterbury
2013

Abstract

This thesis aims to understand visual style in the context of computer science, using traditionally intangible artistic properties to enhance existing content manipulation algorithms and develop new content creation methods. The developed algorithms can be used to apply extracted properties to other drawings automatically; transfer a selected style; categorise images based upon perceived style; build 3D models using style features from concept artwork; and other style-based actions that change our perception of an object without changing our ability to recognise it. The research in this thesis aims to provide the style manipulation abilities that are missing from modern digital art creation pipelines.

Table of Contents

Abstract	i
Acknowledgments	iv
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Research Goals	2
1.3 Contributions	3
1.4 Defining Style	4
1.5 Vector Notation	6
1.6 Software Implementation	7
1.7 Rationale	8
1.7.1 Consistency	10
1.7.2 Concept Art	12
1.7.3 User Generated Content	13
Chapter 2: Background Research	16
2.1 Introduction	16
2.2 Cognitive Approaches	16
2.3 Shape Abstraction	19
2.4 Statistical Classification	21
2.5 Animation Style	22
Chapter 3: Line Style Verification	23
3.1 Introduction	23
3.2 Existing Studies	24
3.3 Categorisation	26
3.4 Transfer	30
3.5 Conclusion	35

Chapter 4:	Structural Vectorization	37
4.1	Introduction	37
4.2	Existing Studies	38
4.2.1	Thick Line Vectorization	43
4.2.2	Structural Vectorization	48
4.3	Results	57
4.4	Form Preserving Scaling	58
Chapter 5:	Characteristic Proportions	62
5.1	Introduction	62
5.2	Existing Studies	63
5.2.1	Geons	64
5.2.2	Image Subdivision	67
5.3	Evaluating Geons	70
5.4	Algorithm Overview	76
5.5	Prototype Subdivision	78
5.6	Image Extraction	81
5.7	Database Analysis	84
5.8	Proportion Transfer	88
5.9	Results	91
5.10	Conclusion	96
Chapter 6:	Model Reconstruction	98
6.1	Introduction	98
6.2	Previous Work	100
6.2.1	Overview	100
6.2.2	Core Techniques	102
6.2.3	Artist Assisted Systems	104
6.2.4	CAD-based Systems	110
6.2.5	Rescaled Base Meshes	112
6.2.6	Alternative Approaches	116
6.3	Mesh Rescaling	118
6.3.1	Implementation	119
6.3.2	Rescaling Results	126

6.3.3	Rescaling Discussion	129
6.4	Model Generation from Concept Artwork	130
6.4.1	Algorithm Overview	130
6.4.2	Concept Image Preparation	132
6.4.3	Skeletonisation	133
6.4.4	Mesh Construction	141
6.4.5	Mesh Refinement	151
6.5	Results	164
6.6	Evaluation	164
6.6.1	Method Comparison	167
6.6.2	Bredth of Input	168
6.6.3	Discussion	170
6.7	Conclusion	171
Chapter 7:	Conclusion	179
7.1	Limitations and Future Work	179
7.2	Conclusion	182
7.3	Attribution	184
7.3.1	Line Art	184
7.3.2	Concept Artwork	184
7.3.3	3D Models	186
References		188

Acknowledgments

To Dr. R Mukundan for your support, your insights, and for having the faith to let my research run off at all kinds of tangents. Thank you for proof-reading countless documents and double-checking countless equations. It was a pleasure to work with you.

Thank you to the Lund University Graphics Group for hosting me, I had a wonderful time! Thanks especially to Mike Doggett for such strong support during my stay, your experience and insight into the peer review process was invaluable, and our discussions about my research always helped to keep everything focused. This dissertation subsumes and extends the work that appeared in our two coauthored papers.

To Stickmen Studios, Wil McLellan and Brooklyn Waters; Carly Wheeler and the Foundation for Research, Science and Technology (now the Ministry of Business, Innovation and Employment). Thank you for the funding. I appreciate the leeway you gave me when I changed topics, the acceptance when my reports went missing, and for understanding that research doesn't always go in a straight line. This research would never have happened without the technology fellowship.

To Margaret and Malcolm, who have given me the most amazing support throughout the whole project and are quite simply the world's best parents. Thank you.

Thank you to the faculty and staff from the Computer Science and Software Engineering department at Canterbury University. To Richard Green, Andy Cockburn and Tanja Mitrovic for accepting a somewhat unorthodox and open ended research proposal; and to Gillian, Alex, Phil, Pete and Joff for making the department such a wonderful place to work.

Finally, thanks to the many other people who have lent me their time, their skills, their patience, and their artwork.

Chapter I

Introduction

1.1 Introduction

Content creation in modern entertainment is now one of the most time consuming components of a project. Extensive commercial toolsets allow artists to streamline this process by assisting with sketching, painting, modelling and animation. While these tools assist with the mechanical tasks of drawing, the core of artistic creation – an image’s visual style – remains in the realm of the artist. Style is what makes something appear cute, or creepy; it’s what differentiates two artists who draw using the same media. More concisely, style is a mechanism that changes our perception of an object without changing our ability to recognise it. This thesis aims to understand visual style in the context of a computational and algorithmic framework, using these traditionally intangible artistic properties to enhance existing content manipulation algorithms and develop new content creation methods. Successful research in this area has the potential to facilitate and reduce the workload of content creators.

Specifically, this thesis investigates the properties of visual style, developing metrics that allow stylistic classification, replication and generation of graphics. It also investigates the indirect application of these metrics in developing smarter algorithms. Overall this research was a success, producing viable results that would be applicable in a commercial setting. Two self-contained and different approaches were developed, using artistic style features in the context of digital manipulation. Several related research areas also resulted in new algorithms and image processing techniques.

Direction manipulation of visual style is explored in Chapter 5. *Characteristic Proportions* is a technique that can extract certain properties of

an individual artist’s style and apply those properties to other drawings. In concert with existing line-style techniques, the algorithm can be used to ensure consistent artistic style across a project by automatically transferring a selected style between hand drawn images. Characteristic proportions can also be used to categorise drawings by different artists, and re-scale unknown input images to match a target style.

Chapter 6 shows the strength of using visual style properties to enhance existing techniques. *Single-View Concept Modelling* is the second advanced technique developed as part of this research. It allows a single piece of hand drawn 2D character concept artwork to be automatically analysed and converted into a computer generated 3D model without the need for user or artistic guidance. The constructed 3D model includes standard technical properties such as UV texture coordinates and bone influence values that mean this low-to-mid resolution 3D content is especially useful for rapid prototyping or in applications such as mobile games.

The research in this thesis deals primarily with hand-drawn cartoon and concept images. The outcomes provide style manipulation abilities that are missing from modern digital art creation pipelines, and the potential applications reach beyond the scope of video games and electronic entertainment. Understanding and manipulating visual style at the software level could lead to automatic solutions or tools that address this issue, as well as faster and more coherent integration of generic art assets or user generated content. This thesis is an attempt to expand on the complicated and relatively unexplored area of visual style, and hopefully opens up further opportunities in stylistic manipulation.

1.2 Research Goals

The initial aim of this research is to develop metrics that allow for quantitative measurement of style. Such metrics are based upon measurable image properties and should be representative of artistic style as opposed to image content.

With industry funding and backing, this research is directed toward the digital entertainment sector. After successfully developing or understanding

style metrics, the goal of this thesis is to use visual style properties to create and enhance software for content creation.

Visual style properties could be used for direct style interaction, including methods for the understanding, categorisation, and manipulation of visual style. Style properties could also be used indirectly to enhance existing software or automate currently manual procedures in the content creation pipeline. The goal of this research is to speed up the development of large digital media projects by making content creation faster, easier, or by putting it in the hands of users.

1.3 Contributions

This thesis makes a number of significant contributions in the areas of visual style manipulation and 3D model generation. Traditionally intangible artistic properties are measured and used throughout this thesis to enhance existing content manipulation algorithms and develop new content creation methods.

Chapter 3 successfully reproduces existing research, and confirms that line metrics can be representative of the artistic style and not only of the content of an image. This research led directly to the development of a vectorization technique specifically for line based cartoon content. This was published at IVCNZ in 2012:

BUCHANAN, P., DOGETT, M., AND MUKUNDAN, R.
Structural vectorization of raster images. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand* (2012), ACM, pp. 319–324

A new method of image content analysis has been developed, also making possible a number of new style manipulation methods. *Characteristic Proportions* define the relationships within a set of similar shaped objects by the same artist. Using a database system these can be compared and used to distinguish artwork based upon the author, or transfer visual style between two stylistically different images. This research was presented at Computer Graphics International:

BUCHANAN, P., DOGGETT, M., AND MUKUNDAN, R.
Transferring characteristic proportions to modify the artistic style of cartoons. In *Proceedings of the 30th Computer Graphics International Conference* (jun 2012)

Two automatic model creation methods are presented for use in rapid prototyping and content generation pipelines. Using image metrics, a single piece of 2D concept artwork is used to produce a fully textured and rigged 3D model without user intervention. This work advances the field of Sketch-Based Interfaces and Modeling:

BUCHANAN, P., MUKUNDAN, R., AND DOGGETT, M. Automatic single-view character model reconstruction. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2013), SBIM '13, ACM, pp. 5–14

Overall, the research in this thesis provides a number of important contributions to the field of visual style and automatic modelling.

1.4 Defining Style

Defining style is an implicitly difficult topic due to the amount of personal interpretation involved. People from different cultural backgrounds understand and classify image style in different ways, as do people from technical backgrounds or fine art backgrounds. What we perceive as "style" is a function of our hugely complex visual system, combined with lifelong environmental experience. Section 2.2 touches upon the difficulty of understanding style in a measurable context. However, to develop computer systems in this area requires that we develop a concrete definition of style. Many different terms are used to describe the appearance of an image. There is no single standard, and previous literature in this area often uses terms interchangeably or in different contexts. Common terms include Visual and Artistic Style, as well as Image Dimensions, Features, Metrics, and Geons. This section attempts

to define the common terms used throughout this thesis.

The Oxford English Dictionary gives two definitions for style:

1. A way of painting, writing, composing, building, etc., characteristic of a particular period, place, person, or movement.
2. A distinctive appearance, typically determined by the principles according to which something is designed.

These both encompass a wide range of different ideas, and a more concise definition of style is needed. In fact, we want to separate the different types of style that we deal with in the thesis:

In this thesis *Visual Style* refers to the appearance of an object or drawing. Classified under the moniker of visual style are all components and visual features of the image that do not directly contribute to the substance of the image. More precisely, visual style indicates any and all properties of the image that can be changed in a consistent manner without affecting a viewer's understanding of the image contents themselves. Examples are the type of drawing media, the stroke properties, or the colour palette.

Artistic Style is a subset of visual style, and encompasses image features that have been chosen specifically by and are unique to an artist or school of art. While the appearance of individual features may be shared between artists, the values of multiple features are often unique to an artist. As opposed to emotional style, artistic style is viewer independent and therefore a good candidate for algorithmic representation and manipulation. In the context of this research we focus upon individual artists, because schools of art for example, "modernism" - are often too broad of a category to classify using statistical methods. It's worth noting however that individual artists can drift or change their style over time, and in these cases the style may also become too broad to easily classify.

Emotional Style is also a subset of visual style, and while this thesis rarely explores this area it is important to understand the distinction. Emotional style encompasses image features that have either been chosen or have occurred during image creation with the purpose of contributing to the emotional response of a viewer toward the image. As opposed to artistic style,

these features are independent of the artist and often occur in groups. "Cute" or "creepy" would be examples classifying these emotional style groups, although it is worth noting that emotional style classification is viewer dependant.

To measure style, an underlying representation must be used. While this is developed as a concept in Chapter 5, the following terms are used throughout the thesis.

A *Feature* is a meaningful or semantic part of an image. There is no set size, shape or classification, however a feature is always a subset of the object or scene being portrayed. Features have two functions, the first being its contribution to the object so as to increase detail and aid understanding, and the second being the contribution to the visual style of the image.

A *Metric* is the appearance or measurement of a feature that can be stored as a single, often scale-independent, value. Examples would be the angle of line curve or the size ratio of a Geon. Specifically, the use of 'metric' in this thesis refers to the common use in Computer Science, where it refers to a measure of some property or specification.

Line *Curve* and line *Camber* are often used interchangeably in literature, however in this thesis they are given distinct meanings. The line curve is a first-order metric, that is to say the average angle of a line defined in Equation 3.1; whilst the line camber is a second-order metric representing the rate of change of the line curve over its length.

1.5 Vector Notation

Within the field of engineering and computer graphics, several different styles of vector notation are commonly used. This thesis uses the following notations for calculations.

Single and double pipes are used to find length. This is the absolute value of a scalar and the length of a vector respectively:

$$\begin{aligned} |-2| &= 2 \\ \|(\overleftarrow{3}, -4)\| &= 5 \end{aligned}$$

The *norm* of a vector, or vector normalization, is represented by a hat and is performed as following:

$$\hat{v} = \frac{\overleftarrow{v}}{\|v\|}$$

In cases where this notation would be ambiguous due to formatting, single pipes around a vector term are used to denote the norm:

$$|v| = \hat{v}$$

1.6 *Software Implementation*

A number of software components were developed for the thesis, ranging from stand-alone programs to a full automatic modelling suite and toolset.

For the algorithms outlined in Chapters 3 to 5, individual programs were developed using a common framework and file format. These programs were written in C++ and used a command line interface to interact with the datafiles. A separate visualisation program was used to view the dataset and record positional input if needed. This format meant that it was easy to adapt and implement new features as standalone applications because they did not interfere with already working components. The final list of components is : Line Extraction, Line Modification, Geon Database Analysis, Geon

Substitution, Geon Extraction (Brute Force), Expansion & Smart Scaling, Colour Analysis, Artist Classification.

For the 3D components in Chapter 6, a different approach was taken. A single web interface was developed in Javascript and HTML5, using DAT.GUI and WebGL. Because the majority of algorithms are not designed to be interactive, the software has a broad array of initial setup and options, followed by a series of steps that can be run automatically or manually to allow the intermediate steps to be seen.

The use of javascript and run-time languages allowed faster and easier implementation of designed methods, and real-time interaction could be used to view changes and improve the methods. The Web interface is designed to allow easy distributed use within a studio or commercial environment.

1.7 Rationale

The aim of this research is to allow classification and replication of style. This has many applications, but due to its basis in an industry partnership through the New Zealand Foundation for Research Science and Technology (now the Ministry of Business, Innovation and Employment), this research focuses primarily on the digital entertainment industry. The ability to manipulate graphical content in new ways has the potential to save a large amount of effort in many projects, as well as opening up new possibilities for content presentation. Sections 1.7.1 to 1.7.3 show the motivation and need for this type of research.

Chapter 2 outlines some of the research that has already been done in this area, and what the authors of that research consider to be potential research targets and ideas for future work. While these give a good idea of knowledge lacking in the technical space, it is also important to ensure the research is solving problems that actually exist in the artistic space. The following section outlines some of the issues faced by content creators.

TV series and big budget video games typically have large art teams and enormous amounts of content. Ensuring stylistic consistency is difficult, and is something our algorithm aims to assist. In contrast, video games for the social space - such as facebook and the iPhone - typically have fast

development cycles with small teams. Consistency within the team is rarely an issue, however content is often sourced from external locations. Providing tools to deal with artistic style will allow easier, faster and more coherent integration of generic art assets or user generated content.

The research in this thesis focuses primarily on the visual style of characters – including human forms, monsters, and anthropomorphic objects. While most of the methods can also be applied to other inputs, this focus is important for a number of reasons. Due to the lack of prior research, a constrained input type provides the best balance between complexity and scope. A standard character is complex enough that the algorithms must be robust and flexible, yet constrained enough that it does not have to deal with input data with an unexpected or unknown context. Characters are usually at the forefront of the user experience in both computer games and animated movies, and provide a high impact for a small computational focus. The ubiquity of characters in animated media also offers a large dataset to draw from and analyse, in comparison to many other object classes. Finally, line style experiments in Chapter 3 found that characters displayed more stylistic variety and greater correspondence with unique artists than background objects did.

It is also worth noting that visual style in real-time media can have a significant impact on other areas of the production. In video games, art direction can influence the form of the gameplay [124] [63] and vice versa [171]. In animated films the visual style can be a feature of the narrative, or just as easily detract from it [73]. There has even been a case where the artist wanted to change the emotional impact by changing the style of an image while it was being viewed [154]. This introduces complexity to the problem of analysing and understanding visual style, as there may not always be a single technical or artistic reason underpinning it.

Modern video games aim for a greater empathy with the player. Several recent titles have experimented with dynamically changing the graphical style of the environment to reflect changes in the game or player state. In the case of Spicy Horse Games’ Grimm [48], supporting this technology required every environment asset in the game to be drawn in two different styles. Performing this process automatically would have halved the production workload, as

well as allowing for more than two stylistic states.

Overall, the ability to categorise and replicate visual style is an invaluable tool in a digital art creation pipeline.

1.7.1 Consistency

Project teams in the entertainment industry often reach into the hundreds of personnel, so that retaining stylistic consistency across all assets is difficult. Artists Hyung-Tae Kim and Saul Marchese explain:

“ When we hire people, I try to recruit artists who naturally can understand and draw the kind of image I need. [...] I’ve had everyone practice drawing these characters and images in a certain style. [...] It’s all basically an attempt to maintain a consistent style. ”

*Hyung-Tae Kim - Lead Artist, Blade & Soul;
NCSoft; 29-03-2010 [153]*

“ When you have a large group of artists working on a game it can be difficult to achieve a consistent style. ”

*Saul Marchese - Lead Environment Artist;
Climax Studios; 21-03-2007 [116]*

Character art is not the only field where this is an issue. In the full interview [116], Saul Marchese talks about the issues surrounding quality control and how translating level designers ideas to art can cause problems. During production, Saul had to act as a final check to ensure consistency for every element in the levels. Concept artist and comic author Jason Brukaker ran

into similar problems when outsourcing artwork to save himself time:

“ One time I actually hired some freelancers from India and Japan to color a test page to see if they could match my style and save me precious time. I gave them specific reference of how I wanted it to look and even examples of the color pallet. I even gave them my homemade texture files. When I got the pages back they looked ... well, lets just say I deleted the files even after revisions so that I wouldnt be influenced in any way by what I saw. Dont get me wrong, they did a great job coloring the pages but it wasnt right for my project. ”

*Jason Brubaker - Writer & Artist, reMIND;
23-11-2009 [32]*

In the study 'Visual-Style Variation as a Narrative Device in Animated Productions' [73] a number of modern animated films were analysed for stylistic inconsistencies in their art content. Particularly, the study focused upon productions that used different visual styles within a single frame. It attempted to draw a link between theoretical cognitive frameworks in terms of an emotive or narrative context and the use of multiple visual styles. A number of features and shorts were analysed, and while in select cases stylistic inconsistencies were present for a narrative or storytelling purpose, the conclusion reached was that visual style variation in animated films is often due to technical limitations.

This paper is important because it supports a need for the research outlined in this thesis. While style variation is desired in some contexts, Gupta found several cases where a technical limitation was the cause of visual style variation. If a technical solution existed to help develop style consistency, several of these media would better have achieved their aims. Gupta also performs a depth-first qualitative assessment, quoting several film directors and artists who desire visual style consistency in their work. These people and products would benefit from better style analysis and manipulation tools.

Using the proposed research, software could compare assets against a standard, alerting the artist to problems, or automatically fixing any issues. Additionally, such software used in a production pipeline could be used to integrate existing assets into a new game, reducing the art workload.

1.7.2 Concept Art

Stylistic inconsistencies can also manifest themselves indirectly. A common content creation pipeline for 3D media has several steps. Initial designs are created by concept artists in 2D, and then passed on to a 3D artist who creates the model. A texture artist paints the details onto the model and if the model is animated, a rigger sets up the internal skeleton and controls for the animator. All together half a dozen people may have influenced the style of the final model, creating a problem where the final 3D artwork often isn't representative of the initial sketch.

In some cases, this is such a large problem that 3D is avoided altogether:

“ For example if you look at the Pokmon on the package you can really see how cool it looks as a 2D illustration and in the games [...]. If we were able to take that style we have now and have it translate into 3D with no problems; that is definitely something we would be into.

”

*Junichi Masuda - Producer, Pokmon Black
and White 2; 17-09-2012 [80]*

In other cases, this extra workload means that turning 2d concept art into 3d models takes significantly longer than drawing the 2d art in the first place. When developing the Role Playing Game *Torchlight*, Jason Beck had to balance unique content against a tight schedule:

“ What's the value in sacrificing our desires to do unique playable character meshes worth the simpler, more schedule-friendly approach? [...] We began to contemplate ways we could create unique meshes for our playable characters without a pipeline that would jeopardise our timetable.

”

*Jason Beck - Art Director, Runic Games; 03-
09-2009 [20]*

Understanding style features in the concept art allows for easier development of higher level systems that solve problems such as this, with the problem of rapid model creation covered in Chapter 6.

1.7.3 User Generated Content

Rapid and automatic model creation processes are also important in the context of user generated content for video games. To cope with the large content workload and to accommodate user demands for more interaction, user generated content is playing a more important role in modern games. Current approaches to handling the wealth of new data involve numerous trade-offs, and tend to take two main approaches that are outlined below.

1.7.3.1 Free Content Generation

With free content generation, users are given tools that allow them to create whatever they want. Often external modelling packages can be used to export data into the game. Two major problems commonly arise with this method. If the player is not a technically trained game artist the resulting model is often not suitable for use in a real-time game engine. Linden Labs' *Second Life* allows users to freely edit the world at any time, however this limits the amount of graphical optimisation that can be performed and there are many locations in the Second Life world where the game exhibits bad lag and stuttering. Second Life content creator Penny Patton explains:

“ Most of Second Life’s problems can be blamed entirely on this disconnect between Linden Lab’s development teams and the realities of how Second Life works. [...] Linden Lab bears the brunt of the responsibility here. They do absolutely nothing to prevent people from creating content this way. ”

*Penny Patton - Second Life content creator;
28-02-2013 [146]*

Additionally, allowing for free-form data creation creates opportunity for inappropriate or game-breaking content. Sony Entertainment allows user generated content for several online games through their Player Studio tool [1] and solve this issue by screening all user generated content before it is inte-

grated into the game.

“ Through the use of standard third party art tools, [users] learn how to develop, design and personalise items of their own. [... They] submit it to SOE for review and possible inclusion in the SOE Marketplace.

”

Sony Online Entertainment; 2013 [1]

1.7.3.2 Restricted Content Generation

One solution to these issues is to allow user content creation within a preset framework. Simple examples are changing the colour of the player's clothing, and complex examples extend all the way to EA Games' *Spore*, where players have complete control over creature and structure creation at an abstract level but are limited at the technical level. This allows for a wide variety of content within the given aesthetic:

“ No matter how customised, the whole game has an inescapably Sporish look to it - and its bulbous, cartoony lines won't be to all tastes - but that's better than the hideous mishmash that would be born of total aesthetic freedom.

”

Oli Welsh - Deputy Editor, EuroGamer Magazine; 13-02-2008 [179]

This design restriction makes content creation easier for the players, but often means that the tool design is difficult. In the case of *Spore*, a large programming team was required to research new technologies to implement the system.

“ The editor was more of a design challenge - how do we make something that is as powerful as Maya, but that the average Sims player can use?

”

Will Wright - Spore Lead Designer; 03-08-2006 [29]

Changing the visual style or allowing for different aesthetics within this framework requires additional work that the player cannot influence. In the case of Spore, EA Games released extra content packs such as the 'cute and creepy pack' [2] that were developed by in-house artists. This gives the required level of control over the users interactions so that the produced content remains consistent. This was a concern to *Little Big Planet* developer Media Molecule:

“ We wanted players to feel comfortable making constructions in the game, without being put off by talk of level editors and poly-counts. [... It was] also a stylistic decision - However, we didnt want these avatars to exist as blank canvases; they still needed to live within the same stylised world. ”

*Rex Crowle - Art Designer, Media Molecule;
01-01-2010 [49]*

Understanding visual style better could allow for more flexibility in developing options for user generated content. Limited input ability could be expanded upon without reducing stylistic consistency, or stylistic understanding could be used to determine the important display features from an object and therefore be of use in automatically optimising models for real-time use. Overall, the ability to deal with visual style features could increase the options for and viability of user generated content.

Chapter II

Background Research

2.1 Introduction

This section covers research relevant to the problem of quantifying visual style, covering visual perception and previous attempts at creating entire systems. Existing studies that concern only parts of the subsystem or technical details are discussed in their respective chapters.

Traditionally, visual style research in computer science has fallen into two categories. One common area of research is the recognition of brush strokes and painting styles in traditional media, including methods to classify the renaissance masters [151]. This research rarely transfers across to different types of media. Another common research area is non-photo-realistic rendering techniques for 3D models, such as cell shading or pencil hatching [53]. Other papers approach the topic from a psychological perspective, exploring the cognitive processes our brains use in object and style recognition [141]. Little research appears to have been done regarding analysis of artistic style in non-traditional media, nor in regards to an artist’s overall visual style. In this section, the relationship between cognitive models and software systems is explored with the aim of better understanding of the state of the art as well as uncovering relevant and sound principles of cognitive perception upon which to base the research in this thesis.

2.2 Cognitive Approaches

Central to visual style research are several papers outlining the core aspects of visual cognition, line style, object component separation, and structural

re-mapping.

Palmeri & Gauthier [141] performed an extensive literature review of visual object understanding, summarising and categorising 174 visual cognition research papers. They found two dominant schools of thought when it comes to designing a perception model: ‘Object Recognition’ and ‘Perceptual Categorisation’. Whilst this paper does not primarily deal with visual style, performing any type of stylistic manipulation requires object recognition to be preserved.

On the other end of the spectrum, psychologists have tried to model the human understanding of art. Works in this area often focus on art as an object of appreciation and the characteristics most prominent in increasing its appeal. Several papers have performed empirical studies in this area, including Hagtvedt et al. in ‘The Perception and Evaluation of Visual Art’ [74] and the earlier ‘The Psychology of Art Appreciation’ by Funch [66]. These studies found that in general, viewers had distinct scales of appreciation for artistic merits and technical merits. This indicates that it should be possible to change the technical content of a drawing without influencing the viewer’s perception of the artistic components.

As outlined in ‘Visual object understanding’ [141], Object Recognition theory assumes that the brain resembles a large database of stored images, from different angles and at different sizes. Recognition comes down to fast visual recall and indexing. On the other hand, Perceptual Categorisation asserts that recognition is achieved through calculation and associative decisions, matching the view of an object in a probability space with all other previously seen views of objects. The first stage of image translation and understanding in computer science is fundamentally one of recognition, and from a cognitive approach, one of perception.

The two perception models appear to align with the two fundamental computer science techniques for style and animation translation. Object Recognition is a form of re-mapping, whilst Perceptual Categorisation is a form of re-generation. Recognition at a subordinate level is considered easier than recognition at the entry or basic level (see pages 6 - 7). This aligns with the computer science findings that context-free (subordinate level) style classification and remapping is easier than performing the same task contextually

(entry level) [50]. Making this link allows us to better understand the advantages and disadvantages of each method by increasing the understanding behind how they actually work.

Perceptual Categorisation is farther explored in ‘Multidimensional Models of Perception and Cognition’ [11] and attempts to produce digital models of this system are often implemented using Geons, small primitives that, combined with their spatial relationship, create a full image. This is a parallel to the described context-free image recognition, combined with contextual meta-data. Context free data can be transformed in subordinate style, but context is required to transform basic level style.

The relationship between context-free data structures and the ability to generate contextual information through programming was explored in ‘Structure vs. Style’ [76]. Often, context-free metrics cannot capture enough information to be useful, while contextual metrics are almost unexplored due to their overwhelming complexity. Jupp & Gero [94] attempt to tackle this complexity by implementing a categorisation system based upon self-organising maps (SOMs). The program operates at both the context-free level, via extracted features, and the context level through the SOMs.

Deciding upon which properties of an image are style-based metrics and which are related to the object contents is a difficult task. One approach is to look at how properties are processed by the human perception system, a research area covered by de Beeck et al. [52] who carried out empirical research to confirm theoretical models defining the boundary between shape dimensions. Category learning experiments showed that recognition of high-level objects could be trained, while low-level metrics remained independent. Similar results were found by Goldstone [70] who explored the development of high-level models from low-level components. This lends weight to the argument that low-level metrics individually represent style, as changing these is independent and does not adversely affect recognition of higher-level objects.

The paper ‘Visual Explanations’ [54] looks at the crossover between abstract rendering techniques and formal cognitive representations, and is in part a discussion about style reduction for better representation and in part an attempt at defining computer science methods in terms of perceptual science. The ideas in this paper aim to promote a higher level of understanding

when developing technical methods in the area of style and perception. While the contents seem too ill-defined to successfully draw parallels between the fields, the importance of using research from the cognitive sciences to support technical developments is well communicated and has influenced the approach taken for many of the techniques outlined in this thesis.

Like Visual Explanations, the paper ‘Perception and Artistic Style’ [145] looks at the relationship between visual perception and traditional art. Parker & Deregowski allege that the fundamental visual processes, such as those outlined by Ashby et al [11], contribute to and in some cases cause the major stylistic features in works of art. While compelling arguments are made for the strong link between technical perception and artistic style, the lack of quantifiable data to support the link would make it difficult to develop a software system based on these principles.

2.3 Shape Abstraction

These cognitive models are useful when considering papers such as ‘Abstraction of 2D Shapes in Terms of Parts’ [122]. This research describes shape abstraction, “using a new synthesis of holistic features” which creates simplified shapes that retain important features. The holistic features can be explained by the cognitive models which show the importance of corner-invariant detail scaling in relation to the perception of style. The ability of their algorithm to perform context-sensitive part division is important when considering the need for context when creating metrics for subordinate level manipulation of style.

The paper ‘Separating Style and Content with Bilinear Models’ [165] focuses on copying the function of perceptual systems in separating content from style. Whilst this thesis deals primarily with artistic style, Tenenbaum focuses on the general problem domain, giving several other examples such as words spoken in an unfamiliar accent, identifying a font or handwriting style across letters, or recognising a familiar face or object seen under unfamiliar viewing conditions. The paper separates style manipulation functions into three parts - classification, extrapolation, and translation - although it states that the essential challenge in these tasks is the same. A bi-linear model is

then used to order a large series of input into a modular form, such as a 2D or 3D table. Each dimension represents a property of style and by interpolating or extrapolating on each continuum new style variants can be established. Although limited in scope the results are important because they show that it is possible to automatically separate style and substance with little overlap. This can be improved upon and is an essential step toward achieving the goal of stylistic manipulation.

Both Theobalt et al. [166] and Li et al. [107] present methods for decomposition of complex shapes. Although both have different target applications and apply primarily to 3D meshes, they each develop important structural representations of context.

While much of the literature around object categorisation is based around the idea of geons (defined features that have a spatial relationship to create an object) there are few papers that explore exactly what these features need to be. ‘The development of features in Object Concepts’ [148] looks at the relative importance of each geon within an object. It forms the hypothesis that different object categories have different orders of geon importance, and that geons differ in importance and form between people.

Hummel & Stankiewicz [85] explore a structural description model, where Geons within a shape are identified and stored in a relational pattern. This is implemented in MetriCat, a program designed to model the brain’s object classification approach. Whilst the program also models the debilitating factors our brain introduces into classification, it still proves categorisation via the use of shape abstraction is both possible and effective.

Ullman et al. [169] also look at Geons, finding that visual features of intermediate complexity are of best use in image classification. A training set of cars and faces was used to test an image recognition algorithm. Geons of different sizes and complexity were used in the recognition process, which was based upon standard recognition techniques. When looking at this research in the context of visual style, keeping Geons at an intermediate complexity should produce the highest quality remapping.

2.4 Statistical Classification

A brief study by Wayner [176] attempts to classify visual style using two descriptors; line length and line camber. It uses these descriptors to create a histogram of an image’s line length, and then match this histogram with a database of authors. It was found that using only a line length classification, 95% accuracy could be achieved in the categorisation of black and white cartoon strips between 7 authors.

This research is important because it shows, definitively, that it is possible to mathematically classify and recognise artistic and visual style based upon simple metrics. Interestingly, the introduction to the paper mentions line camber and curve as classification descriptors. However, the algorithm showcased does not take curve into account, and there is no reference to this descriptor for the rest of the paper. Given a 95% accuracy rate with only one descriptor, this could potentially be increased dramatically with the inclusion of further descriptors.

There are also examples of research into style classification outside of the typical computer graphics arena. Fischer et al. [60] outline a system for classifying film genres based upon editing properties. Attributes such as the length between cuts, the type of camera movement, and the audio can be used to build templates that represent the editing style of each genre. It is interesting and important to note that many of the same underlying principles are used in regards to what makes a style metric and how the classification is performed. This is reinforced by the success of Tenenbaum & Freeman’s multi-use system [165] and suggests that the technical approach to style recognition remains roughly the same irrespective of the style content itself.

Zhang et al. [191] use shape metrics to extract ‘semantically meaningful layers’ from animated cartoon videos. Despite the claim that each layer represents a different motion style the result does not appear significantly different to standard segmentation techniques. This shows the difficulty of separating style from content, and highlights the necessity in both defining these terms and evaluating results in terms of the initial aims.

Kalogerakis et al. [95] skip the classification step and use probabilistic

models to directly generate new Geon configurations that statistically match an existing dataset. The probability model uses relationships between properties of shape components, and assumes that these are related to or caused by the underlying structure within the dataset. Variability within these properties allows generation of new models, and it is this type of database and Geon approach that is taken in Chapter 5 to transform 2D images.

2.5 Animation Style

Chris Hecker [77] describes the procedural animation system used in the computer game Spore. This paper explores an animation system where the action is abstracted from the skeleton. It was found that categorising structural components allowed for much better generalisation of the animation, something that standard animation re-mapping did not do well. Instead of directly re-mapping style, the ability to decompose style into a descriptive structure, and then re-generate the animation using the style descriptors meant that the final result could create context-sensitive actions. This may be necessary when performing style re-mapping, as illustrated with the issues encountered by Freeman et al. [64].

Database-driven style remapping for animation sequences has been proposed in papers such as ‘Style-Based Inverse Kinematics’ [72] and ‘Style Machines’ [30]. In both cases, a probability function derived from a large library of Motion Capture data was used to find the appropriate body pose for the specified end-points of the animation sections. Grockhow additionally used inverse kinematics to allow for better re-targeting to the new poses. Earlier research by Unuma et al. [170] also attached an additional emotion channel that was tied to the style of each action, while Shapiro et al. [152] use Independent Component Analysis with exemplars to extract style tracks from animations. In all of these cases, it was possible to change the style of the animation without affecting the understanding or actions of the animation.

Chapter III

Line Style Verification

3.1 Introduction

Visual style analysis can happen at multiple levels. This thesis focuses primarily on high level style features, such as Characteristic Proportions in Chapter 5 and 3D shape in Chapter 6. However these high level features rely on accurate knowledge of low level features such as line properties, and knowledge of how these affect and interact with artistic style.

It is therefore important to explore and validate the results shown by some of the core research in this area. Studies by researchers Freeman et al. [64] and Hertzmann et al. [79] have outlined methods for style transformation at the line level. More importantly, research such as that by Wayner [176] shows that artist identification and categorisation can be performed reliably using only basic line metrics.

Two major verification steps have been performed that validate the results of these papers. This chapter explores stylistic categorisation based upon the properties of lines within an image, which is referred to as the line style. Section 5.9 confirms that stylistic categorisation can also be performed at a higher level by using Geons. The Geon system does not modify properties of the image relating to brush technique. Style perception by the viewer is a complex subject but is based in part on the line style and therefore in addition to Geon techniques, style translation needs to be performed at the line level.

By validating previous research, and confirming that line style can be cleanly manipulated, the requirement of also performing line style translation during high-level techniques is satisfied and can be integrated into the characteristic proportion system to allow for a more complete style change.

Likewise, with artistic recognition, it should be possible to augment the recognition using line style metrics. To this end, a qualitative survey is performed to evaluate line style transfer.

Knowledge of line style was also important when developing the line extraction technique, Structural Vectorization, in Chapter 4. The line property research in this chapter is also used in Section 4.4 as the basis of intelligent line scaling.

3.2 Existing Studies

Two comprehensive papers describe techniques for the style translation of lines within 2D drawings.

In 'Learning Style Translation for the Lines of a Drawing', Freeman et al. [64] outline two similar matching techniques used in transferring line style between a training data-set and a basic line drawing. Line style in this paper refers to *the length, width, curvature and corner angles of vector line segments within an image*. The authors explore two methods for line-matching the original drawing with the style database. This is done by comparing the drawn line with every line in the database using a custom matching algorithm. They found that the nearest-neighbour match produced the best style translation, but the image became distorted. A linear combination (using a least-squares matching) produced the best image translation, but the style does not transition well. Much of the paper is spent looking at these two methods, and introducing a k-N mapping where the nearest neighbour (N) algorithm is used first, and adjusted by the least-squares combination up to (k) number of lines. This produces a translation algorithm that preserves the image and the visual fidelity of the style.

One interesting style-interpolation technique that appears to have been introduced first in this paper is the uneven point distribution for the style descriptors. In previous papers, style descriptors have been features such as end-points, corners, or junctions. In this paper, lines are represented by series of points picked from along the line and encoded as [X, Y, thickness] coordinates. They are picked in an uneven distribution depending upon the

curve's *interest value*, which is a combination of several line properties. More points cluster around end-points, corners and junctions, but are not exclusively restricted to these areas. This allows much better linear interpolation between two lines of similar shape, preserving the interest points.

In contrast to a database system, Hertzmann et al. [79] explore the creation of 'curve analogies'. By using a texture-matching, scale and rotation invariant transform, their algorithm can analyse a single line, stroke, or image, and transfer the style properties from one drawing onto a second drawing. A major advancement shown in this paper is the ability to perform this transplant between two very different images.

While the paper talks about style in the context of visual style, this definition extends only as far as the shape variations within a single curve. It directly transplants the original image to the secondary image without concern for the underlying structure. This reduces algorithmic complexity, but means that extra drawing information is required for the original image. Two images (the drawing, and the underlying curve structure) are required instead of simply one. One issue not explored or explained in the paper is the case where no corresponding mapping can be found (i.e., the curve for one line cannot be found in the underlying structure of the other, and hence no segment can be found to perform the transfer).

Both papers successfully use re-mapping techniques, with manually created databases or input. The style mapping examples by Freeman et al. [64] are of high quality and show that this technique works well. However, the limitation of line style re-mapping shows in areas such as the human face where modifications are stylistically consistent, but create unrealistic facial features.

As mentioned in Chapter 2, a 1991 study by Wayner [176] used line length and line camber to classify visual style. It uses these descriptors to create a histogram of an image's line lengths, and then matches this histogram with a database of authors. It was found that using only a line length classification, 95% accuracy could be achieved in the categorisation of black and white cartoon strips between 7 authors.

This research is important because it shows that it is possible to mathematically classify and recognise artistic and visual style based upon simple metrics.

3.3 Categorisation

The Structural Vectorization process outlined in Chapter 4 is used to extract line information from source images. Basic line properties of width, length, and taper are measured. Additionally, the first and second order line angle properties (*curve* and *camber*, respectively) are recorded. Curve is measured according to Equation 3.1 which finds the average angle between adjacent points in a line segment. This equation depends on point density and can only be used to compare lines with the same density measure. The vectorization process used to generate the lines in this section produces results with the same densities, and therefore lines can be safely compared between images and artists. Camber represents the rate of change of the line curve over its length. Figure 3.1 shows two input images representative of the two artists analysed in this section. Ten images were analysed from each artist to give around 2,000 line segments. 17 outliers were manually removed and the resulting datapoints used produce the figures in this section.

The fairest method of comparison would be to use the same comic drawn by multiple artists. In practice, this is difficult to obtain because you need a large body of work. Therefore, the images in this section were selected as best as possible to have comparable content. They were selected from webcomics and cartoon strips with similar layouts. In addition, only object-internal measurements are saved, so there is no relation or influence from the background or surrounding objects. While this could potentially be an issue when looking at the visual style of entire scenes, the advantage is that it produces a fairer comparison between characters and objects within the strips.

$$c = \frac{\pi}{2(n-1)} \sum_{i=1}^{n-1} |(V_i - V_{i-1}) \cdot (V_{i+1} - V_i) - 1| \quad (3.1)$$

where V is a set of $0..n$ vertices.

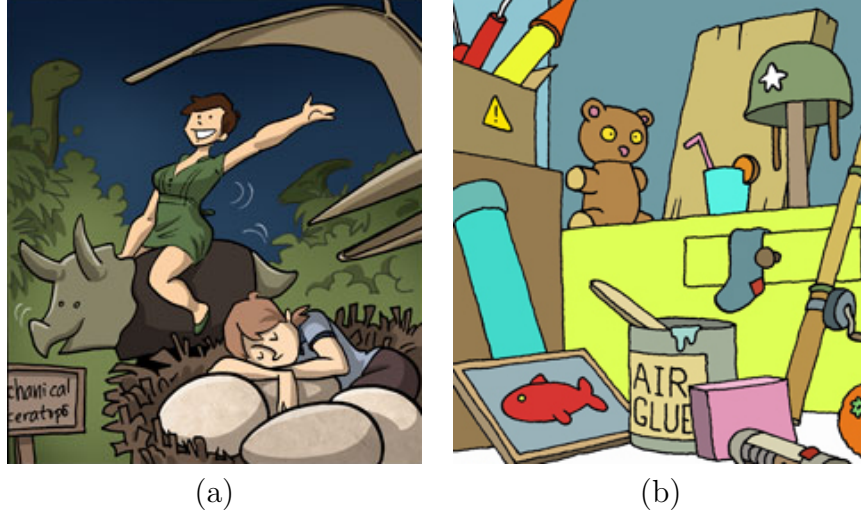


Figure 3.1: The two source images used in this section. *Girls With Slingshots* (a) is hand-drawn by Danielle Corsetto, while *YodaBlog* (b) is drawn digitally by Thierry Vivien

The statistical analysis by Wayner [176] used only one metric — line length — to classify images. This was successfully repeated with our source images and metrics recorded through the vectorization. Figure 3.2 shows the line lengths where the horizontal axis is the length as a proportion of the image and the vertical axis is the frequency. The difference can be clearly seen, with the lines used to draw *YodaBlog* consistently longer than those used to draw *Girls with Slingshots*. Wayner used line length to successfully classify images, and this would be possible in this case. However, there are several other line metrics that produce clearer distinctions between artists.

Significant differences can be seen in the line width plotted in Figure 3.3. *YodaBlog* is drawn digitally with a single line width, which can be clearly seen in the single spike that occurs just under the 1-pixel width. The line is not precisely 1 pixel wide due to antialiasing, measurement error, and cases of overdraw where lines were corrected by the artist. *Girls With Slingshots* is drawn by hand before being digitised, and the width of lines is significantly more spread. Significant peaks can be seen in the width frequency, which suggest a discrete number of line widths are used in the creation of the

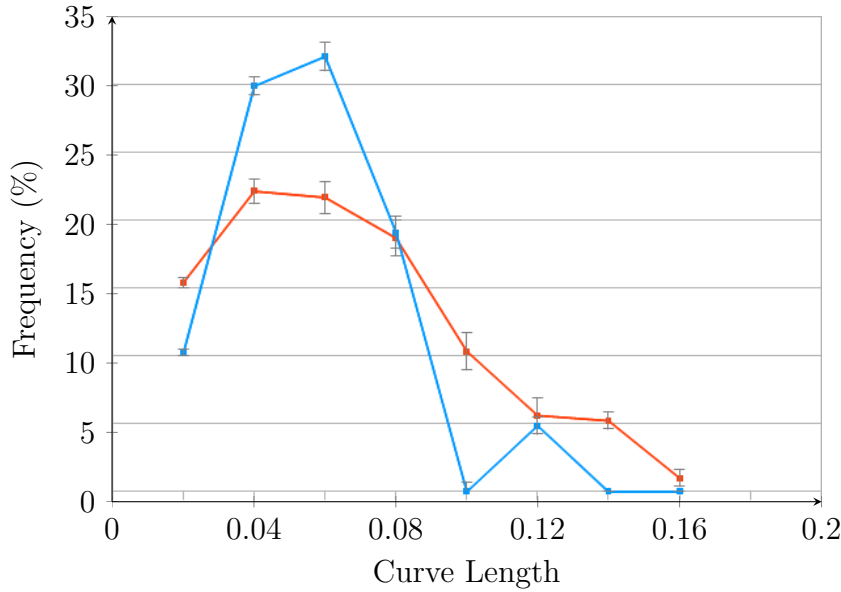


Figure 3.2: This graph shows the distribution of line lengths in YodaBlog (orange) and Girls with Slingshots (blue). The horizontal axis is the length of the curve relative to image size, where 1 is the length of the image diagonal. The vertical axis shows the frequency of each length (at intervals measuring 2% of the image size).

artwork. This may be due to the artist using different pens or different drawing styles for different elements of the image. Categorisation using this metric is accurate in the case of these two cartoons because the line-width metric has less overlap than line length.

Figure 3.4 shows the distribution and clear relationship between line width and length. This can be analysed using a separability test, where the distribution of points orthogonal to the principal component is measured. For the combined dataset the principal axis is in the direction $(200.3, 1.174)$ with an offset of $(-132.8, 0.901)$. Because each dataset has a different source they also have unequal sizes and variances, and so Welch's T-Test is used to analyse the point distribution. This gives a T-Value of 2.07 and a two-tailed P-Value of 0.0394. This is less than the threshold value $p = 0.05$, which confirms that the difference between the two artists is statistically significant. However this is a relatively high P-Value compared to other artist metrics such as the Geons outlined in Chapter 5. The concentration of points at low

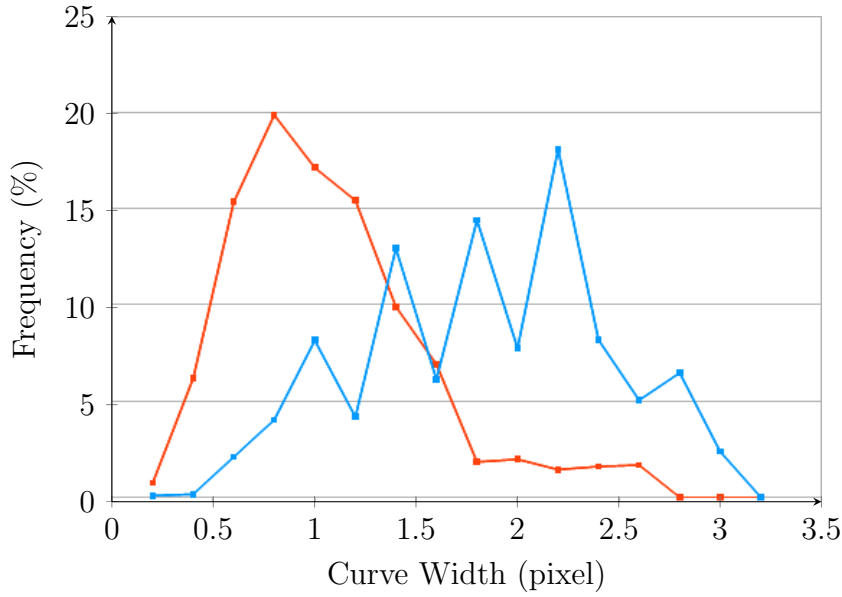


Figure 3.3: This graph shows the distribution of line width in YodaBlog (orange) and Girls with Slingshots (blue). The horizontal axis is the width of the curve in pixels and the vertical axis shows the frequency of each width (grouped to 0.2 pixels). Input image size was normalised before analysis.

widths means that automatic separation for artist classification using this metric would be difficult.

There are several distinguishing characteristics in this dataset worth noting. Both sets of image have numerous short lines, which is to be expected as these are fundamental in adding detail and form to any object. However, these short lines differ in their distribution. Below a width of 0.5 pixels the widths are stratified due to the vectorization algorithm and shows the need for a better sub-pixel algorithm for low resolution images. An improved vectorization algorithm is outlined in Chapter 4. Above 0.5 pixels the line width in YodaBlog is more evenly distributed compared to Girls with Slingshots. This is likely to be a property of the visual style as explained for Figure 3.3.

At higher line lengths and widths there is an obvious stylistic choice that separates the two artists. Because of this clear distinction, a grouping algorithm such as k-means clustering can be used to categorise images by artist based purely on these two metrics.

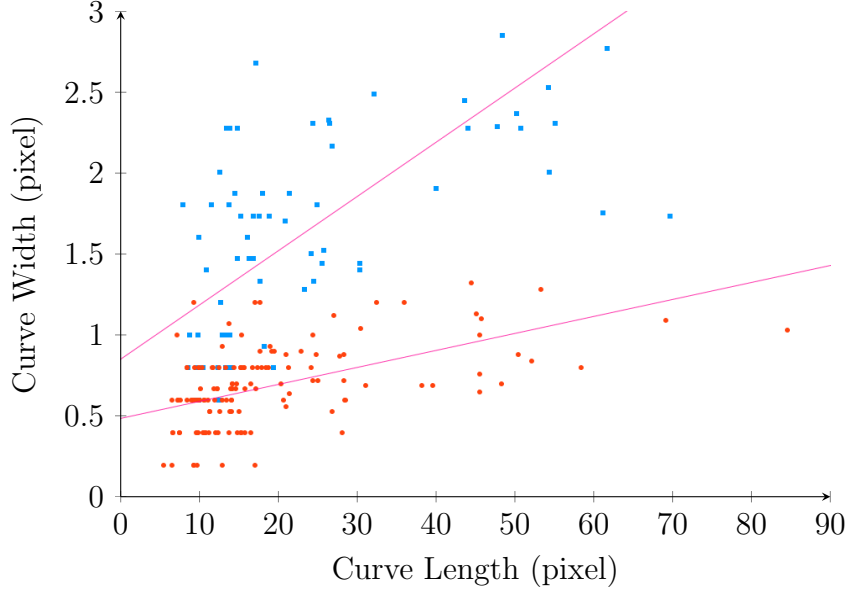


Figure 3.4: This plot shows the width and length of individual line segments from YodaBlog (orange) and Girls with Slingshots (blue). The horizontal axis is the length of the curve in pixels and the vertical axis is the width in the same units. Input image size was normalised before analysis. The principal axis (best fit line) for each data source is shown in purple.

Figure 3.5 shows the line curvature, with the angle on the horizontal axis and frequency of occurrence on the vertical. Girls with Slingshots is characterised by sweeping curves, which can be seen in the high frequency of curves across the entire range of angles. YodaBlog has in comparison straighter lines with sharper corners, which is clearly reflected in the frequency spike at high angles. As with the line width and length metrics, this provides an opportunity for style recognition.

Data from the line taper showed no clear distinction between artists, primarily because both sets of images have lines with consistent thickness.

3.4 Transfer

Line style difference between artists has shown to be statistically significant, and categorisation based upon line style is possible. The final step in validating these low-level style properties is to use the information to modify

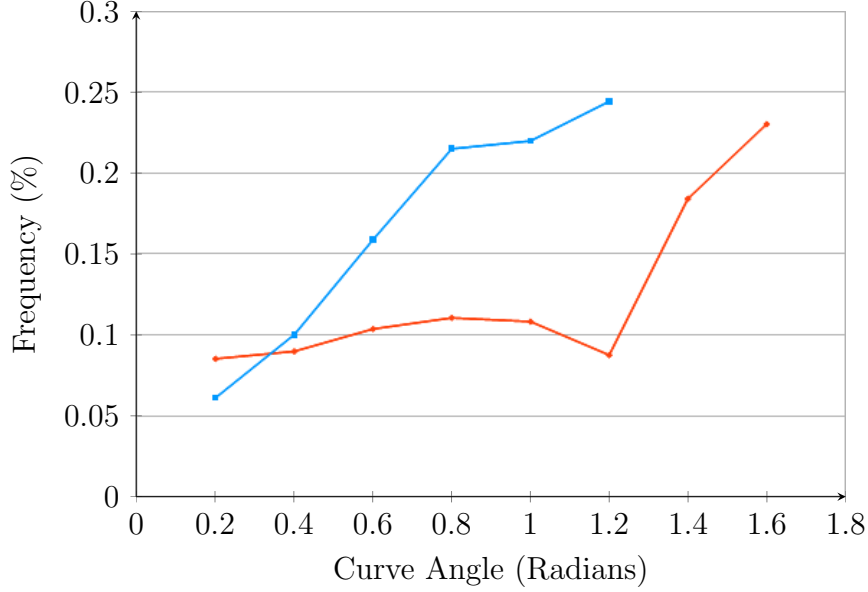


Figure 3.5: This graph shows the distribution of curvature in YodaBlog (orange) and Girls with Slingshots (blue). The horizontal axis is the average angle of curvature of each line segment in radians. The vertical axis shows the frequency of each angle (grouped to the nearest 0.2 radians). The orange line represents data from YodaBlog, whilst blue is Girls with Slingshots. The sharpest curves in each dataset are 1.2 and 1.6 radians respectively.

the style of an existing image.

A manually-assisted approach is taken to perform line style transfer. An approach is taken based in part upon the database system outlined by Freeman, and the idea of replacing line properties in-place outlined by Hertzmann. The mechanics of the algorithm are simple. First a target dataset is created from the property distribution histogram with the number of points matching the number of line segments in the source image. The contents of this dataset now contain a set of widths and curve angles that represent the average target image. Lines in the source image are iteratively changed to match the target dataset, starting with the values that have highest spread as these are less likely to find exact matches. A visualisation of this process is shown using a single metric (size) and simplified images in Figure 3.6.

Selecting the target values is one of the most important steps. For each column in the histogram the target value is selected via Equation 3.2, which

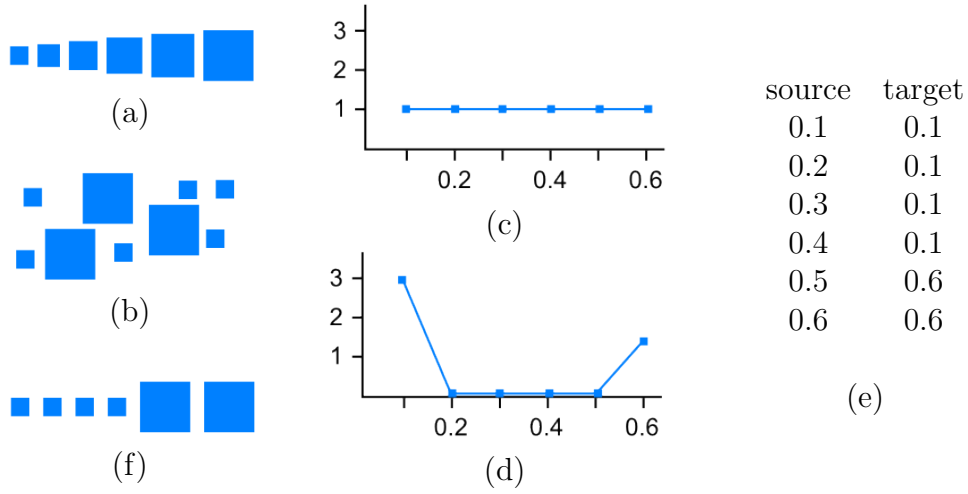


Figure 3.6: A simplified example showing style transfer with a single metric, the size of the square. The source image (a) and a number of target images represented by (b) are used to create frequency charts (c) and (d) respectively. Table (e) shows all size values for the input, and the corresponding target metrics selected from (d). The result (f) retains the image layout but better reflects the spread of sizes in the target style.

gives for example in Figure 3.6(d) 4 points at 0.1 and 2 points at 0.6. For each datapoint the untransformed source metric closest to the target value is selected and assigned the target datapoint. This works well because it preserves relative changes. For example, a line that is more heavily curved in relation to a second line will always remain more curved, even while the global distribution of line curvature is changed.

$$value = c_t \frac{f}{c_s} \quad (3.2)$$

where f is the frequency, c_s and c_t are the number of measurements in the source image and target datasets respectively.

The metrics used for transformation are different to those used for recognition. Line segment length is not changed because this distorts the image so significantly that it impacts upon recognition. Even line curvature changes cause problems when they become too flat or too curved. This can be seen in

the characters' sweater in Figure 3.8(c) where the horizontal pattern is misaligned. Small gaps and overlaps can also be seen around the forehead and the arms. This is a common problem and is addressed in Chapter 4.4 with the smart scale technique. This preserves detail ratios and line connections while changing line curvature. Line width is easily changed and presents fewer problems, while line camber is not adjusted in this trial.

Additionally in this trial the number of lines in the source image is reduced to match the average in the target images. YodaBlog has significantly less detail than Girls with Slingshots, and this is reflected in the transformed image as the smaller and assumedly less important lines are removed from the input. This works well in the case of the sweater, although causes issues such as around the underarm where line segments are small yet important. Note that this detail reduction would however be difficult to do in reverse if transferring style from Girls with Slingshots to YodaBlog.

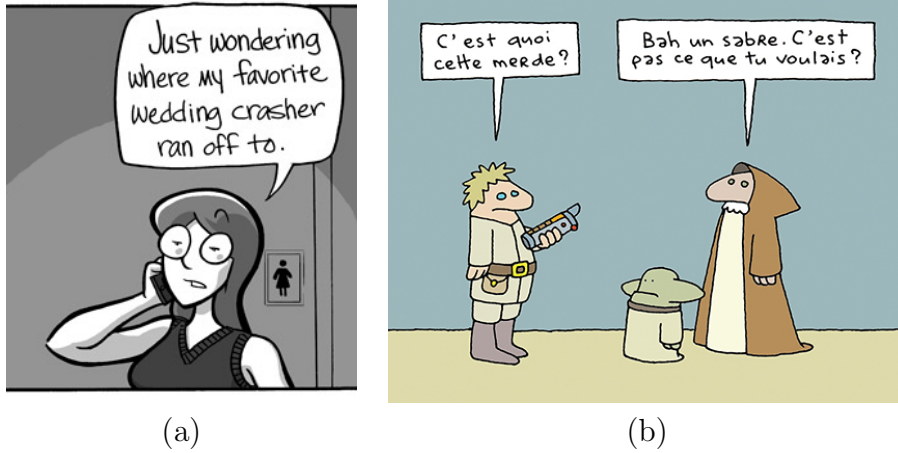


Figure 3.7: The input image (a) from Girls with Slingshots, and an an example (b) from YodaBlog of the target style.

To complete the evaluation of this style transfer method, a trial was carried out using two established artists. Figure 3.7 (a) shows a single image from Girls with Slingshots that was not in the set used to create the line style database in Section 3.3. Figure 3.7 (b) shows an image from YodaBlog exemplifying the target style. Lines from the input image are extracted using the same technique as explained in Section 3.3, with additional manual

intervention to fix small alignment issues and ensure line intersection points are correctly labelled. Colours and shading are ignored, and caption text is removed as vectorizing letters creates a style bias and font style translation would require a different approach.

An important point to note is that this transfer is the best alternative of two different approaches, and doesn't always perform well. Changing line curvature without changing the end points guarantees a correct topology but can disrupt the structure. This was found to perform better than guaranteeing structure without topology because while each individual part may overlap, the lines are still joined at corners and present a more cohesive picture.

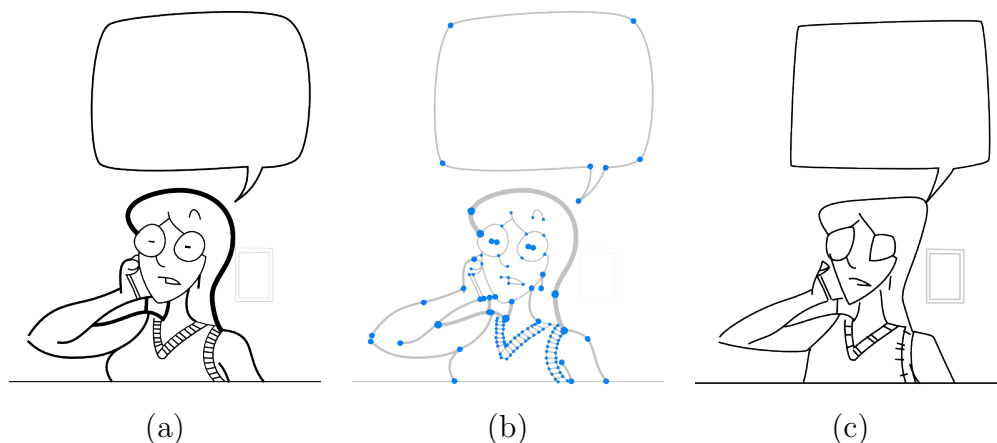


Figure 3.8: A vectorized version (a) of a character from Girls With Slingshots is divided (b) into segments and has its line curvature changed (c) in an attempt to match that of YodaBlog.

Figure 3.8 (a) shows the vectorized input image, with Figure 3.8 (b) showing the adjusted points that denote line segments. Style translation is performed by changing the distribution of line properties in the source image to equal that of the target style. Because the source image is in a vector format and lines can be represented as spline segments, both the curvature and the width of lines in the image can be easily changed.

Overall the style change is significant. The image is clearly the same content, yet lines are still slightly curved and reflect the messier nature of

YodaBlog. The images in Girls With Slingshots have flowing curves, while YodaBlog has sharper curves and this is most clearly illustrated in the head where the character’s hairstyle now matches that of YodaBlog characters (the hood and the yoda ears).

3.5 Conclusion

The evaluation performed in this section successfully supports the findings of previous researchers. Line style differs between artists, and this property can be both measured and adjusted.

The statistical analysis by Wayner [176] was successfully reproduced, with the distribution of measured line length in a drawing proving to be different between artists and a valid statistic to classify images. Further statistics were evaluated and similar results found in line width as well as curvature. A clear relationship was found between line width and length, with specific characteristics distinguishing each artist. At higher line lengths and widths there is also an obvious stylistic choice that separates artists. Because of this clear distinction, a grouping algorithm such as k-means clustering can be used to categorise images by artist based purely on these two metrics.

Moreover, these distributions are representative of the artistic style and not of the content of the image. When multiple images from the same artist were analysed, the histogram remained similar in each case. An unexpected feature in the analysis was the ability to also make educated guesses about an artist’s drawing media.

With the successful categorisation of artistic style based upon line metrics, modification of line properties was attempted based on these statistics. The qualitative survey performed showed a significant style change was possible. The resulting image contains clearly the same content, yet the lines reflected the nature of the target image.

Despite some positive results, there remain numerous problems and assumptions with the techniques outlined in this section. The results are based upon only two sets of data that were chosen due to content availability and permission, and it’s possible that other combinations of artists would not

work as well or at all. To perform a wider ranging survey would require analysis of a high numbers of images and therefore the algorithm would have to be made completely automatic, a research topic that lies outside the scope of this thesis.

Additionally, metrics such as line length were highly dependant on the vectorization algorithm, and a different choice of corner location would result in different results. The line style metrics do not take into account brush properties such as those outlined by Hertzmann et al. or the situation dependant differences that would be apparent using a lower level database system as proposed by Freeman et al. Overall, however, the results in this section are close enough to prior research results that line metrics can be considered a valid and viable method to manipulate low level style. With both prior approaches and the simple transfer outlined here, there exists a solid base from which to approach the rest of this thesis.

Chapter IV

Structural Vectorization

4.1 Introduction

Traditionally, image analysis is performed on raster images based upon global or local features. However some types of algorithm such as the style and stroke analysis outlined in Chapter 3 perform better or must be performed on vector data. Vector data always contains a line topology made from line position and connectivity data, and may also include width, colour, and border properties. While modern tools allow rapid drawing directly into vector formats, many artists and studios still use raster images for cartoon work. Additionally, older artwork only exists in raster format, which must be vectorized before it can be used. Storing image data in a vector format has the added benefit that it is highly efficient in comparison to the source image.

The fidelity of the extracted lines has a direct bearing on the quality of the operations that can be performed using the vector data. When approaching line style evaluation in Chapter 3 several existing vectorization approaches were tried, however difficulties extracting the required data and accuracy problems at low resolutions meant that none of these were ideal. Due to this lack of existing solutions, this chapter deals with the improvement of line-data extraction algorithms.

The method outlined differs significantly from prior work due to the format of the output data. Existing solutions usually aim for visual fidelity at the cost of topological information. This information is important for the metrics in Chapter 3, where the focus is to detect lines in a way that enables easy segmentation and accurate measurements. Section 4.4 shows how this information can be used to perform vector scaling in a way that standard

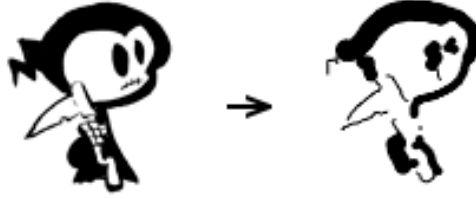


Figure 4.1: Structural vectorization is performed on the original image (left), with the result (right) showing the image represented as lines of varying width. This structural representation trades loss of visual fidelity with ease of processing due to the clearly defined structure. The loss of visual fidelity is acceptable when performing operations that rely on topological or other structural information.

vector data is unsuitable for.

The Structural Vectorization method outlined in Section 4.2.2 is computationally expensive, but extracts line centres even if the line has an irregular profile or the image has unusual topology.

4.2 Existing Studies

Vectorization is a common analysis problem, and many solutions exist. Early research focused primarily on CAD and technical drawings, while later vectorization techniques have been tailored to handwriting recognition, skeletonisation, and more recently digitisation of older artwork. Skeleton data was seen as a better storage format than raster images as early as 1987 [115].

Two algorithms with similar results were proposed at around the same time by Elliman [58] and Dori et al. [55]. While Elliman uses edges extracted from a binary image to find the line direction and Dori et al. use linear segments to approximate the underlying structure, both are suited to technical drawings with straight lines and were developed primarily for speed. Both techniques have problems tracing irregular shapes. Research specific to cartoon drawings has recently gained a higher profile, with Cheng [38] and Zhang [192] releasing vectorization papers that deal with irregular shapes. Cheng et al. [38] provide the better algorithm due to their accurate stroke

segmentation and image complexity reduction. However, both papers require even line widths and must be tuned for specific line profiles. Hand-drawn cartoon input rarely has even line widths, and so any vectorization algorithm must cope with width variation.

Research released by Huang et al. [84] presents a stroke extraction algorithm that does not rely on even line widths. They provide a robust stroke extraction algorithm but unfortunately stop short of vectorization. Our focus upon cartoon imagery means that source imagery already has clearly defined strokes, and unless the range of input images is extended, stroke extraction is not needed. Recent advances by Noris et al. [133] also solve the width variation problem by using gradient-based pixel clustering. As opposed to Cheng and Zhang, the input must be a clean mid- to high-resolution line image, as lower resolutions don't provide adequate data. A maximum line width is however established to ensure correct joint detection, and so this method does not work for blocks of colour or uncharacteristically wide lines. While the vectorization in this chapter is required to work on low-resolution inputs, the junction-detection techniques in this paper could be incorporated in the algorithm design to better cope with line splits and areas where the lines are ill defined.

As opposed to Huang et al. the research by Houle et al. [82] uses an adapted Freeman Chain to group sections into strokes after vectorization of handwriting. The final strokes have no width data, however this is used as an intermediate step and could potentially be extracted. Zhen et al. [193] also tackle the problem of separating individual pen strokes. Their successful results were enabled by limiting the problem domain to sketched tree and flow diagrams. Given that Structural Vectorization is designed primarily for stylistic manipulation of character artwork, it may be possible to use properties unique to this problem domain to increase the accuracy of the results.

Once an image has been reduced to black and white strokes, a common vectorization method is the Potrace algorithm by Selinger [150]. This method produces graphically accurate representations of an image by treating it as a series of geometric shapes. This is useful for preserving fidelity, but makes structural processing difficult due to the lack of line centre and width infor-

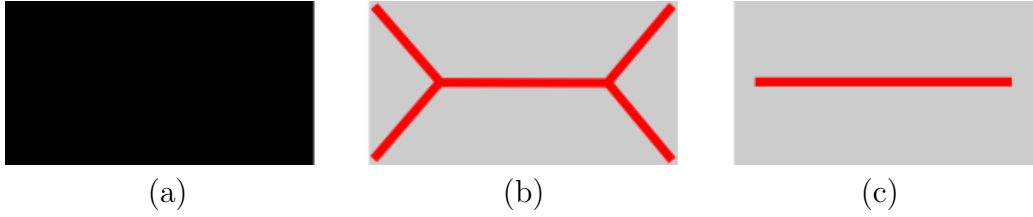


Figure 4.2: Morphological skeletonization operators such as the medial axis transform provide a geometric decomposition (b) of a shape that even when thinned does not always represent the human recognised structure the proposed algorithm extracts (c).

mation. These are the two of the most important properties an algorithm requires when processing line style.

In addition to stroke extraction and recognition, many morphological and topological skeletonisation algorithms exist that produce outputs ranging from unconnected point clouds [56] to β -skeletons [15] that contain the topology in a connected graph. An algorithm tailored to visual style features should reach a compromise that allows for disconnected elements but strives to join line vertices when possible.

Medial transforms are perhaps the most well established method for skeletonisation, having been proposed in 1967 [27] and tweaked in various different ways up until the present [103] [126] to solve problems such as the influence of surface noise on branching. Another method with the same result but a different approach is joining the centres of bi-tangent circles or maximal discs within a shape [10]. The medial transform produces geometric skeletons, however as can be seen in Figure 4.2 even simple shapes can produce a skeleton that does not correspond logically to the underlying structure.

Modern techniques based on morphological skeletons often focus on improving branch pruning to reduce extraneous limbs and reduce the effect of noise on the results. Suh & Kim [158] use the results of a pixel thinning algorithm directly, whilst Olsen & Samavati [135] followed this with a directional tracing algorithm to extract strokes. The strokes are then classified according to their function. This type of extra classification, seen also in content-based vectorization systems [75], could be used to determine what type of information needed to be stored for each stroke. In addition, a range

of thinning algorithms are explored by Lam et al. [104] and given that many produce the same results this is a good source for locating simple and effective approaches to problems such as vector creation from centrepoints and junction pruning.

This problem arises even when different approaches are taken [118], while papers that retrieve a clean structural topology do so by limiting images to a specific domain such as handwriting recognition [99]. In addition, Lam, Lee & Suen found that most skeletonisation algorithms do not store width or colour data [104].

High quality vectorization that aims to produce images identical to the input. These often use colour fields or large number of vector patches. Battiato et al. [18] present a technique to convert raster images into a vector format using Data Dependent Triangulation (DDT). DDT approximates local pixel neighbourhoods by subdividing them into a grid represented by relative triangulation. Xia et al. [185] also use triangulation but allow a greater flexibility in patch size to represent underlying gradients. These high quality methods focus primarily on photorealism, and while they produce reasonable results for monochrome line drawings the data format is not easy to parse for object and line understanding.

Orzan et al. [137] introduce image partitioning diffusion curves, a vectorization technique that stores different colours on both sides of the curve. In the work, they also describe an algorithm for automatically extracting this representation from a raster image. Much like the triangulation technique described by Xia et al, this extraction technique relies on Canny edge detection and therefore does not work well at small scales and pixel-width lines. Another argument against vision techniques can be inferred from the results by Nguyen et al. [131] who test six popular robotics and vision algorithms for line detection on real-world data. The data is low resolution and noisy, and the results are significantly less accurate than other vectorization techniques evaluated in this section while also lacking any underlying vector data. When developing the vectorization technique outlined in this section, pixel-based vision techniques were avoided and sub-pixel detection techniques explored in greater depth.

One filter-based technique worth mentioning is the vectorization process

outlined by Williams & Green [181] due to the stroke-extraction algorithm, which is based on a triangle-area heuristic. This is mathematically equivalent to the distance-angle metric outlined in this chapter and while the thresholds are calculated differently Williams' comments in Section III C on improving accuracy and limitations through tuning the Cost Functions were applicable to this system.

One approach to low-resolution vectorization is to bypass the line extraction process and produce a cell-based vector image from the input. Numerous systems have been implemented and improved over time, including Veroni-based solutions and the pixel table look-up system hq4x [157]. A system with high quality results was outlined in 'Depixelizing Pixel Art' by Kopf, J and Lischinski, D [102] which modifies Veroni cells based upon a number of heuristics. This type of post-processing makes a large difference in the final result, and in the context of line-based extraction it is possible that existing research could be improved solely through a post-processing step if viable heuristics are found. A cell-based system however is infeasible for Structural Vectorization as it does not support the main aim of creating easily understandable data.

Steger [156] outlines a subpixel line detection algorithm for use in aerial photography annotation. The technique is primarily designed for identifying borders between different coloured and textured segments, however the subpixel techniques used could potentially be modified to find centrelines in strokes, and outlined techniques such as subpixel stroke biasing could be applied to centreline realignment. Many of the subpixel issues identified in this paper will need to be considered and addressed when designing any future sub-pixel vectorization algorithm.

It is also worth looking at other models for edge extraction, such as Active Contours which were introduced by Kass et al. [97] in 1988. Also called Snakes, contours are fitted to object boundaries using a number of constraints created from image contents. Recent advances have adapted and improved the algorithm, including self initialisation and increased accuracy based on image gradients [8]. While this method only creates a single outline, it guarantees a connected vector and works well on low resolution images and edges with discontinuities caused by noise. Templated approaches that allow for

complex shapes can be found in Active Shape Models [45] [46] [25], and the more advanced Active Appearance Models [44]. These are both based on the principles of Active Contours, and require a manually defined vector template as well as a training step to initialize the template difference tolerances. However, the idea of using morphological gradients to increase accuracy was key to developing the gradient fields used in this chapter.

A more mathematical approach to the vectorization problem was taken by Lin et al. [109] who outline a B-spline fitting algorithm that runs on extracted point-cloud data. A series of rectangles are constructed subdividing the point cloud, with each one containing a spline representing the centreline of the underlying data. While the accuracy of this method is high it does not cope well with line junctions, although the results can be improved by combining line fitting with corner detection [100].

After vectorization, the data must be stored in a form that can be parsed for image reconstruction, the accuracy of the reconstruction is a good indicator of the quality of the vectorization. Mestetskii, L [121]. outlines a mathematical representation of so-called "fat curves" and a rendering technique for converting this to a raster display using a series of positioned discs. Structural Vectorization contains similar data and these techniques were a basis for the reconstruction techniques used in this chapter, substituting the discs with linear cross-sections to be able to better represent sharp corners.

The method outlined in this chapter is computationally expensive, but finds line centres accurately even if the line has an irregular profile and varying width. It is designed primarily for low to medium resolution images and produces usable results for a wide range of input, from long and thin lines to lines that have almost the same width and length.

4.2.1 Thick Line Vectorization

To enable the type of style manipulation performed in Chapter 3, cartoon images in raster format must be analysed for line style data. One way to do this is to vectorize the images and extract the properties. To this end an existing vectorization algorithm is implemented and extended to accommodate the required line properties. The most effective algorithms for line style

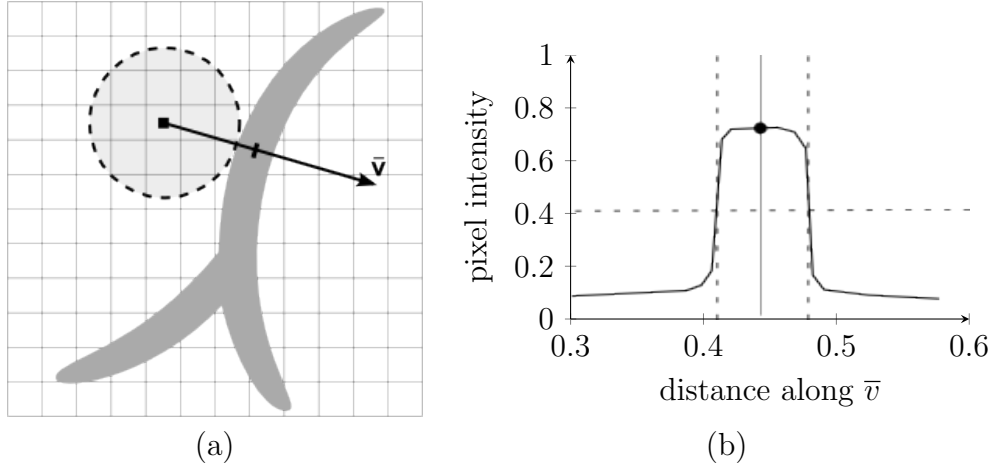


Figure 4.3: Finding a line centre for use in vectorization. Diagram (a) shows how a vector (\bar{v}) is drawn between each pixel and the closest line edge, after which the intensity values along the vector (b) are used to find the line centre to a sub-pixel accuracy.

research are those that can subdivide strokes. While most line extraction methods use pixel filters and similar techniques, Huang et al. [84] use a type of circular mask that collects clouds of line centrepoints that can be more easily vectorized. This section is based on their approach.

The circular mask method uses a pixel search algorithm to find points that lie in the middle of the lines. Each point has an associated direction, which is used to recreate the strokes. Additional processing, such as projected rays or angle comparisons, can be used to find sharp corners and junctions.

The method is based upon line cross sections. A vector representing the line cross-section is shown in Figure 4.3(a) and is comprised of a point on the image and a direction that is approximately orthogonal to the normal of the image line. Image intensity values along the vector are used to find data such as the line centre. There are numerous ways to find this cross-sectional vector, for example the gradient bands used by Noris et al. [133] or the basic area search outlined by Huang et al. Due to ease of implementation, the basic search is used in this section and performed for each non-line pixel in the image to find the closest pixel that lies on a line. Points within a line are not used as two reference points are required to calculate the line normal. The direction of the search vector is taken to be the line normal.

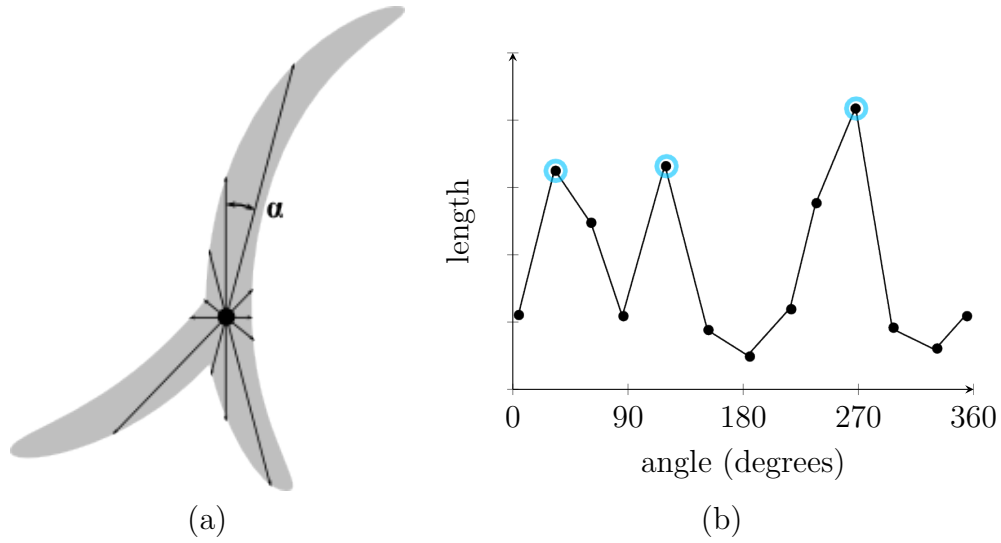


Figure 4.4: Finding line connectivity information via centrepoint projection. Internal line distances are measured outwards from the centrepoint (a) and plotted by length and angle. The number of discrete maxima (b) show the directions and number of connecting lines.

This normal does not always fall at right angles to the line tangent, which is important later in the process because it results in point clusters with wide angle spreads. This aids continuity when converting to splines.

Finding the centre of the line is possible by using the line normal. Figure 4.3(b) shows a graph of the image intensity along the length of the intersecting normal. An intensity threshold of 50% is chosen and the centrepoint calculated to be the average of the crossing points. The high number of created points functions as an antialiasing method because the centrepoints are not aligned to the pixel grid. The width data can also be recorded at this point, calculated as the distance between the crossing points. An additional line property is also the 'definition' or 'clarity' of the line. If the line has fuzzy edges and blends with the background, then the cross-section will be round, whereas a well-defined black line on a white background will have an almost square cross-section.

Running this centrepoint detection process across the image results in a large number of clustered points that conform to the shape of the lines. To finish the vectorization process, these must be joined together. Additional

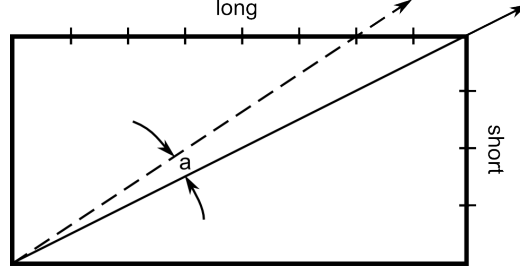


Figure 4.5: Illustration of Equation 4.1 showing the smallest possible divergent angle for two pixel-width lines. Each mark indicates one pixel.

data is required for this step, and Figure 4.4 shows raycasting within the line to detect corners and joints. Starting at the centre point of the line, 2D rays are cast out from the point until they hit the edge of the line (a). The lengths of these rays are then plotted against their angle (b) and the local maxima used to find joint information. The maximum number of angle samples needed to ensure full coverage of all possible branches in an image can be calculated using Equation 4.1. If no α value is given by the user, this calculated value is used by default. The ability to override α is given because a high number of angles is computationally expensive and depending on the complexity of the image a smaller number of angles can often be used without sacrificing quality.

$$\alpha = \tan^{-1} \left(\frac{l}{s} \right) - \tan^{-1} \left(\frac{l-2}{s} \right) \quad (4.1)$$

where l and s are the long and short dimensions of the image in pixels, shown in Figure 4.5

If the centrepoint lies at the end of a line, then there will only be one maxima on the plot. If it is in the middle of a line, then there will be two maxima (one in each direction that the line extends). In the case of the diagram, the core lies at an intersection, and there are three maxima. The distribution of the maxima can also be used to find the type of intersection. In this case, two maxima are evenly spaced apart by half of the width of the

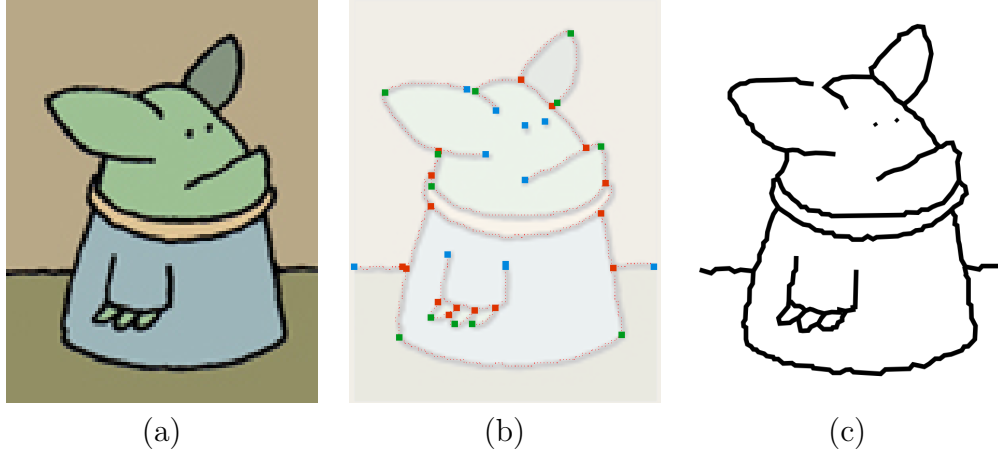


Figure 4.6: Example showing an image (a) and the associated pointcloud after thinning and with highlighted connectivity information (b). Blue represents the end of a line segment, green is a sharp or distinct corner, and orange is a multiply-connected joint. An unsmoothed vector representation is shown in (c).

graph. This shows that they lie on a straight or gently curved line. The third maxima therefore shows a single line that joins halfway along the straight or gently curved segment.

Figure 4.6 shows this process in action. Nearest-neighbour thinning is used to reduce display data and consolidate joint information. This is performed by iteratively combining points that lie closer together than a given threshold, resulting in a single point at the average location. The threshold used was 1 pixel. The results show accurate line tracing and joint type detection. Minor errors can be seen in Figure 4.6(b) where several line segments have unnecessary corners detected close to existing multiply-connected joints. Overall, however, enough information exists to create a reliable and accurate vector representation.

This vectorization method was used to obtain the results in Chapter 3, unfortunately a number of problems became evident and are unable to be solved without major changes to the algorithm. The method of projecting radial distances from line centrepoints fails for lines with thin widths – especially those less than 2 pixels in diameter. It was found that a number of

artists still work at very low resolutions, even when preparing their source material and older artwork digitised at low resolutions commonly contains lines of this width or thinner.

A second problem is pointcloud accuracy around areas that have closely located convex corners. Because of the nearest-neighbour search used within each circular mask, this type of line feature receives fewer points and can result in small gaps in detected line segments. Huang et al. outlined similar problems in their research, and so an improvement on this method is introduced in the next section, Structural Vectorization.

Currently, lines are defined with a large number of points. An improvement on this is to reduce these to control points for Bezier curves or Splines. This makes it easier to modify line curvature and is important in techniques such as the Smart Scaling outlined in Section 4.4. Using all datapoints to draw the lines results in a vector that conforms accurately to the original drawing, however it can be seen in Figure 4.6 that this is not necessarily the best visual representation because the lines follow pixel borders too closely. A tradeoff between visual and mathematical accuracy is explored alongside improvements to the circular mask and additional line data representation in the next section, Section 4.2.2.

4.2.2 Structural Vectorization

Improving the vectorization process while specifically allowing for line style manipulation introduces a number of additional requirements. Current algorithms that vectorize images while preserving visual quality often do so at the cost of either stroke or topological information. This section outlines an improved vectorization algorithm that extracts line data in a format that allows for easy access and use in further analysis. Figures 4.7, 4.2 and 4.14 show the advantages over edge vectorization and morphological skeletonization respectively.

This algorithm vectorizes an image using three main stages based on those outlined in the previous section. A resolution independent gradient map is generated for the image, containing vectors orthogonal to the lines and replacing the function of circular masks; the line centres are found with

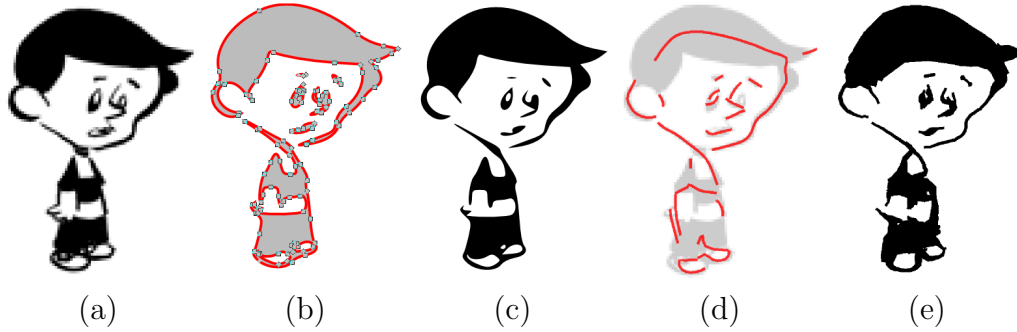


Figure 4.7: Comparison between Structural Vectorization and a polygon based vectorization technique [150], both taken from the same source image (a). Polygon based vectorization produces images with high visual fidelity (c), but results in complex topography (b). Structural vectorization extracts dominant strokes (d), and line width information. While not as visually accurate as polygon vectorization, strokes and widths are nevertheless able to represent the source image (e), and are of more use in image processing because they possess more cohesive information about topology and stroke data.

subpixel accuracy by analysing cross sections aligned to the gradient field; and the vectors are created using a weighted nearest-neighbour algorithm to join the centres. Section 4.3 shows the output from this process and compares it to existing methods. Figure 4.8 shows the stages in more detail. Overall, the algorithm preserves structural information such as topology and connectedness, and is able to process complex images such as the one shown in Figure 4.7(a) where relevant lines may not be obvious.

The process begins by identifying line centres. The existing circular mask method based on research by Huang et al. is computationally expensive. Several recent researchers, such as Olsen et al. [135], propose a morphological thinning step, which uses erosion as a core step to identify line centres. However if lines within the image have different widths, using an erosion step can lead to distorted or entirely incorrect identification of line centres.

To avoid this issue, our improved algorithm finds midpoints by matching points on the boundaries on opposite sides of a stroke using a method similar to the gradient bands implemented by Noris et al. [133]. The gradient filter applied in this section is however independent of image scale which enables the system to cope with large variations in line width. For the purpose of this

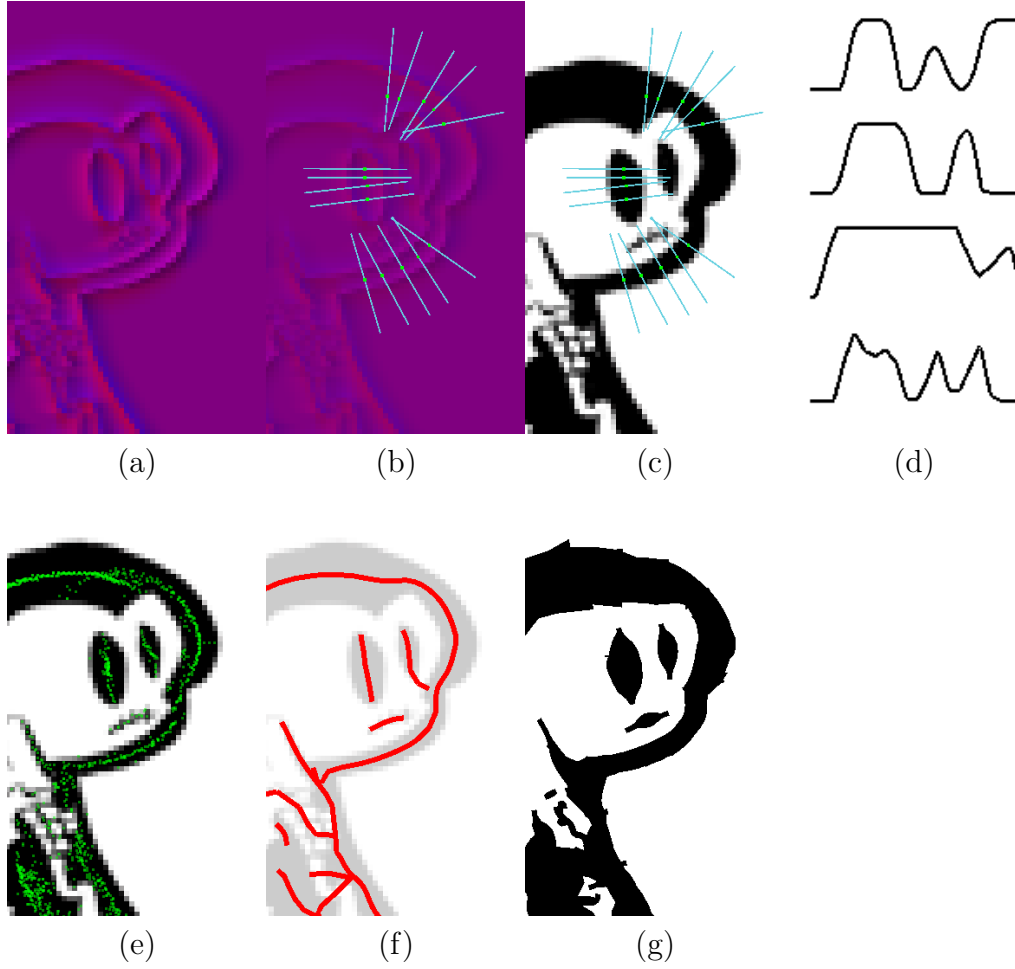


Figure 4.8: Raster images are vectorised to preserve structural information about line centres and widths at the cost of visual fidelity. A vector field is calculated for the source image and shown here in the Blue and Red colour channels for X and Y respectively (a). This is used to slice along the shortest path from each pixel to the nearest edge (b). When values are taken from the greyscale image (c), these slices measure the value profile (d) and are subsequently used to place control points at the local maxima (e) which represents the line centre. Joining these with a nearest-neighbour algorithm creates a structural representation of the image (f) that together with line width information is enough to store a representation of the input image (g).

algorithm, line edges are considered to be sharp changes in intensity, and for a line to be detected it must have two opposing edges. These opposing edges are detected by measuring the intensity of the image along a line perpendicular

to the line. To find the direction of this line, a distance map is calculated for the image and the gradient — calculated using the difference of intensity within a given radius — is used to find the direction to the nearest line. The distance map is calculated using the following weighting kernel K :

$$K(x, y) = \frac{1}{\left(\frac{1}{2\sigma}\sqrt{x^2 + y^2} + 1\right)^2} \quad (4.2)$$

where $\sigma = \frac{\sqrt{w^2 + h^2}}{4}$

where σ is the spread of the weighting kernel and w & h are the image width and height. A typical value for σ that produces good results was found to be a quarter of the image diagonal.

This kernel is then convoluted with the image. During the convolution pass a maximum operator is used to preserve sharp edges and ensure that line values are not diluted:

$$I'_{(x,y)} = \max \left[I(x + x', y + y') K(x', y') \right]_{\substack{-\sigma < x' < \sigma \\ -\sigma < y' < \sigma}} \quad (4.3)$$

where I is the original 2D image.

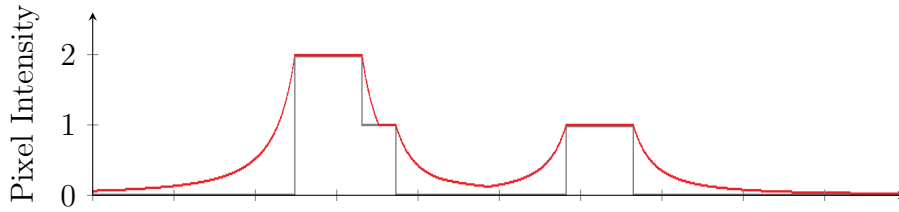


Figure 4.9: A 2D representation of the kernel applied in Equation 4.3. The black step functions represent greyscale lines, and the red line shows the result of Equation 4.3. This function provides several advantages over both Gaussian kernels and a linear distance map because it preserves the line intensity and the nonlinear falloff means that lines of low intensity are not obscured.

This intensity map has a number of unique properties. Figure 4.9 shows a

2D representation of the kernel being run on pixel data. This shows how the function has a sharp falloff but preserves the values of the lines themselves. When using linear falloff maps, greyscale lines that have values below the falloff gradient are often obscured. Morphological distance maps avoid this problem, but are often more computationally expensive to calculate. In addition, this function provides an advantage over a smoothing kernel because it preserves the initial line intensity.

The gradient at a given point on the image is calculated by Equation 4.4. \hat{G} gives a 2D vector at (x, y) orthogonal to the closest line or edge structure in the raster image:

$$\hat{G}(x, y) = \sum_{\substack{-\sigma < x' < \sigma \\ -\sigma < y' < \sigma}} \left| \overrightarrow{(x', y')} \right| \cdot I'(x + x', y + y') \cdot S(x', y') \quad (4.4)$$

where S is a standard 2D Gaussian kernel using $A = 1$, $\mu_{x,y} = 0$ and $\sigma_{x,y} = 2$.

This 2D gradient field provides more information than the computationally faster single-convolution operators such as the Laplacian, Prewitt, Canny and Sobel. The 2D gradient field is shown in Figure 4.8(a), with the x and y components of the vector field represented by the *red* and *blue* channels respectively. The resulting vector field makes it possible to determine the direction to the closest edge from any pixel and the tangent and normal vectors for a line before it is vectorized. This allows the centre of the line to be found with sub-pixel accuracy.

The centre of the line in the raster image is found by searching each pixel within the same colour block. The gradient map approximates the line normal, and is used to place a cross-sectional cut, or slice, across the line, as in Figures 4.8(c) and 4.10(a). This allows accurate measurement of the line width and centre at that point, as shown with the line profile in Figure 4.8(d).

In cases where the image is antialiased and the profile does not change sharply between black to white, the relative intensity values of the neighbouring pixels are used to adjust the centrepoint. For each non-white pixel at the edge of the line, the centrepoint is moved by $\frac{\text{intensity}}{2}$ units along the slice as shown in Figure 4.10(b).

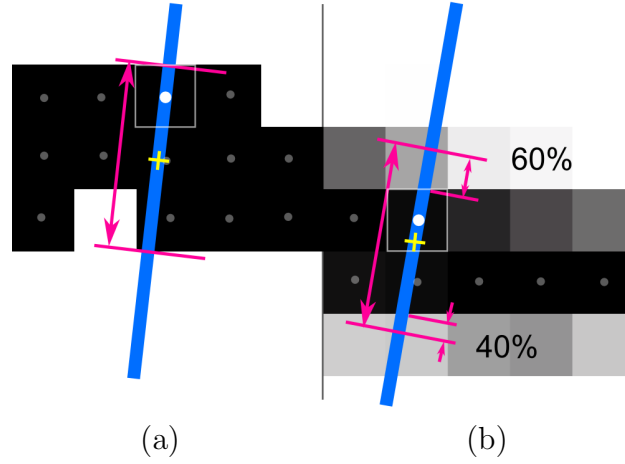


Figure 4.10: Subpixel accuracy can be obtained by taking advantage of cues such as feathering. Image (a) shows centrepoint placement for a monochrome image, while (b) shows this extended to antialiased pixels. Grey points represent pixel centres, with the white dots and square outlines indicating the pixel being evaluated. The blue slice line has been placed based upon the image gradient from Equation 4.4

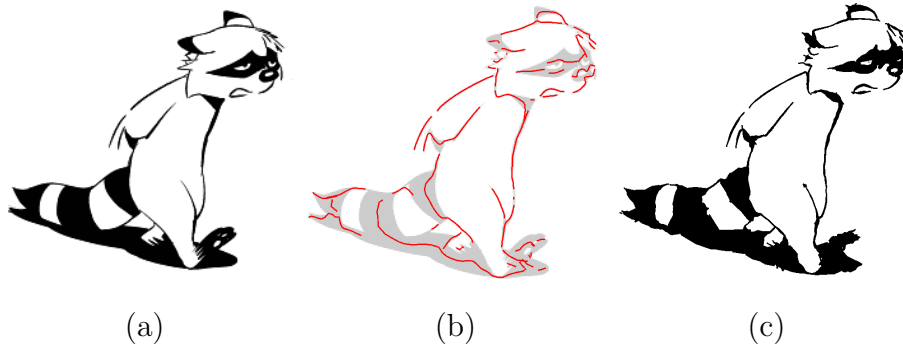


Figure 4.11: Structural Vectorization performed on a low-resolution complex object (a). Significant lines are extracted from the image (b) along with line width information. This combination allows easy processing for image analysis or modification, while still retaining sufficient data (c) to represent the original image.

A control point is then created at the centre of the line, and the process repeated for the next pixel. Figure 4.8(e) shows the result after all pixels have been processed, with green dots representing the control points. Important

lines in an image tend to be isolated or wide and therefore receive the thickest control point clusters, while noise, detail and insignificant lines receive few points. These control points are joined using a best-influence algorithm that attempts to match points along a curve as shown in Equation 4.5.

$$\begin{aligned}
a &= l_n - l_{n-10} \\
b &= v_x - l_n \\
l_{x+1} &= \sum_0^N \begin{cases} v_x & \text{if } (\hat{a} \cdot \hat{b} > -\frac{1}{\sqrt{2}}) \text{ and } (\|b\| < 1.5) \\ 0 & \text{otherwise} \end{cases} \quad (4.5)
\end{aligned}$$

where l_x are the vertices in a line segment with n vertices, and v_x are a point-cloud of size N .

This calculation is only performed on point clusters with more than 10 datapoints, as typically anything more sparse than this did not produce viable results. In general, this algorithm finds the nearest neighbour to each point in terms of line influence. This is a combination of distance, similar to a typical greedy algorithm, and angle. The angular comparison term $-\frac{1}{\sqrt{2}}$ is balanced against the pixel distance of 1.5 to ensure lines do not contain acute angles and do not skip over intermediate pixels.

Any points remaining after the lines have been created are considered outliers and culled. Points lying within the same pixel are considered to be duplicate vertices and are also removed. Figure 4.8(f) shows the result of point joining and culling. Line widths at each vertex are calculated based upon the previously measured line profile and stored per vertex. The distance threshold value of 1.5 units was chosen because datapoints are spaced on average 1 unit apart in the tangent direction. Small differences can be caused due to antialiasing, but the total value of antialiased pixels will never exceed 1 and therefore the maximum distance a point can be dislodged is 0.5 units. The angle threshold value is not as easy to calculate, and the best value depends upon image complexity. 45° was found to work best across our sample dataset, as higher values produce disconnects at intersections and smaller values cause disconnects at corners. This value could be adjusted by

the user if necessary.

Lines are smoothed and jitter removed by removing unnecessary control points. Reducing the number of control points also simplifies the data for processing after vectorization. This is performed on a point-by-point basis by iteratively removing control points until the new segment no longer approximates the correct shape sufficiently well. As in Section 4.2.1 pairs of points with the shortest distance between them are removed first and replaced by a single point at the average location. The error between the simplified line and the original line is considered to be the total area between the two, measured using Equation 4.6:

$$error = \int_0^1 (v(t) - s(t)) dt \quad (4.6)$$

where $v(t)$ is a point on the original line and $s(t)$ on the simplified line.

The simplification process continues until the error rises over a user-specified threshold, after which the end vertices are fixed and the process run on the next segment of the line.

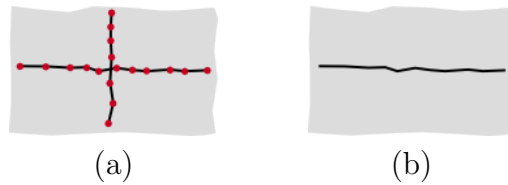


Figure 4.12: Centreline culling is typically needed for thick objects where orthogonal sides generate two possible alternatives (a) for a centreline. After ensuring both lines are viable candidates, the shorter line is discarded (b), resulting in a single centreline.

In the case of thick lines or shapes, it is possible that two orthogonal centrelines are found and an extra culling step must be performed. Figure 4.12 shows a typical situation where centreline culling is needed. All intersecting lines are compared for overlap and culled according to the conditions in Equation 4.7, which checks that the lines are approximately straight, orthogonal,

and centred.

$$c(v) = \frac{1}{n} \sum_{x=0}^n v(x)$$

$$\begin{aligned} \text{cull } v_s \text{ if} \quad & |c(v_s) - c(v_l)| < t_c \\ \text{and} \quad & |c(v_s) - v_s(0)| \cdot |v_s(n_s) - c(v_s)| > 1 - t_\alpha \\ \text{and} \quad & |c(v_l) - v_l(0)| \cdot |v_l(n_l) - c(v_l)| > 1 - t_\alpha \\ \text{and} \quad & |(v_s(n_s) - v_s(0)) \cdot (v_l(n_s) - v_l(n_l))| < t_\alpha \end{aligned} \quad (4.7)$$

where $c(v)$ calculates the geometric centre of a line with n vertices; $v_l(0..n_l)$ and $v_s(0..n_s)$ are sets of vertices in the longer and shorter lines from an intersecting pair; and t_c and t_α represent threshold values with default values of 1 and 0.1 respectively.

Likewise, extremely short lines are culled to reduce noise. A user-defined threshold is used, and can be adjusted based upon the image being vectorized. The default value is defined to be the average line width in the image, and lines with lengths falling under this threshold are often able to be removed without impacting the overall results.

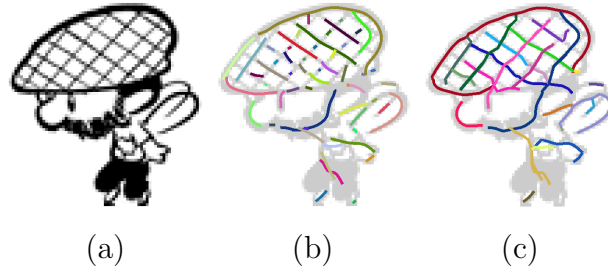


Figure 4.13: Disconnects can be caused in detected lines (b) by light pixel values, noise, or patterns in the source bitmap (a). Short pieces of line with co-linear ends are therefore joined so as to better represent the source image (c).

The remaining lines follow the centre of the rasterised contours correctly, with one vectorized line per contour. However in many cases, the lines are

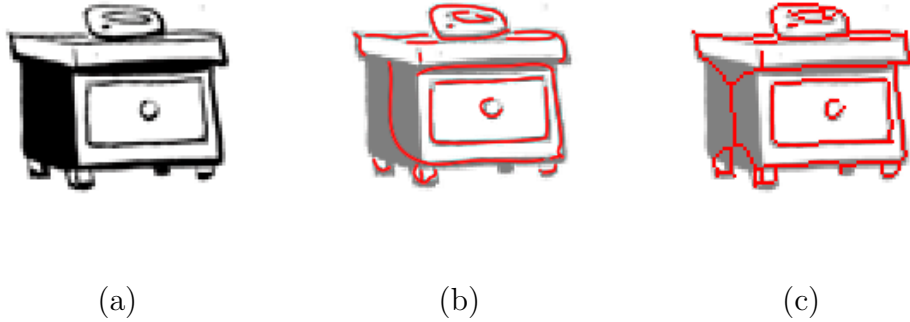


Figure 4.14: The topology generated by the outlined structural vectorization algorithm (b) differs from typical morphological algorithms that produce geometric centers (c). Note how the side of the table is represented by fewer lines in the structural model.

short and therefore not representative of the image structure. Figure 4.13(b) shows an example of this. To provide more coherent data, lines are therefore joined in the cases where multiple lines terminate within a given threshold and the ends are co-linear. The effect of this can be seen in Figure 4.13(c).

While the result in Figure 4.8(g) is not a perfect reconstruction of the image, vectorization is possible even for very thin lines and the underlying line-based vector structure provides more accessible information than polygon based deconstruction.

4.3 Results

Figure 4.11 shows the input and output stages of the Structural Vectorization process performed on a complex object. The underlying structure of the image is extracted without significant distortion or interference, and without overlapping lines. Given the structure and the line widths, the original image can be reconstructed without the need for polygon outline information.

Structural Vectorization provides a different set of information to other common vectorization techniques. Figure 4.7 uses a typical cartoon image to illustrate the trade-off between visual quality and the underlying data. Figure 4.15 shows the algorithms performance on an engineering drawing, where the structure of the vector data is typically considered the most important

aspect. Figure 4.14 shows how the generated structural topology differs from morphological algorithms.

Complex images also provide vectorization challenges that would require context awareness to solve. Specifically, the algorithm has difficulty distinguishing independent shapes that are obscured by objects of the same colour. In Figure 4.11, this causes the middle section of the tail to become joined to the shadow. Note that the tip of the tail is sufficiently unique to be extracted as a separate stroke.

There is no single correct solution for the vectorization of strokes in a cartoon image, and therefore this algorithm attempts to provide a general solution that results in usable data across a range of input images. User tunable parameters also provide additional control over results. Tuned correctly, the output line data is useful for a range of applications. Analysis of line properties such as line length and camber can be used to identify and classify the author of an image, and vector graphics using line centres provides better time-stability than polygons when working with animation. In analysing hand drawn cartoon images, the topological line data produced by the algorithm shows that line based structural vectorization does not need to be limited to fixed-width strokes or technical drawings.

4.4 Form Preserving Scaling

Structural Vectorization results in vector data that contains line property information in a format that is easy to process. This section shows a form preserving scaling method that takes advantage of the available line data to scale images such that their overall form remains as undistorted as possible. When linearly scaling objects or images, features easily become distorted and this has an impact on the visual style. For example, scaling a picture of a car in one direction will distort the shape of the wheels so that they are no longer circles. If the intent of the scaling is to make a shorter, longer, or taller looking car then an ideal solution would keep the wheels circular while scaling the body appropriately.

Looking at the important characteristics of visual style in regards to lines, it can be inferred that the best way to preserve the appearance is to ensure

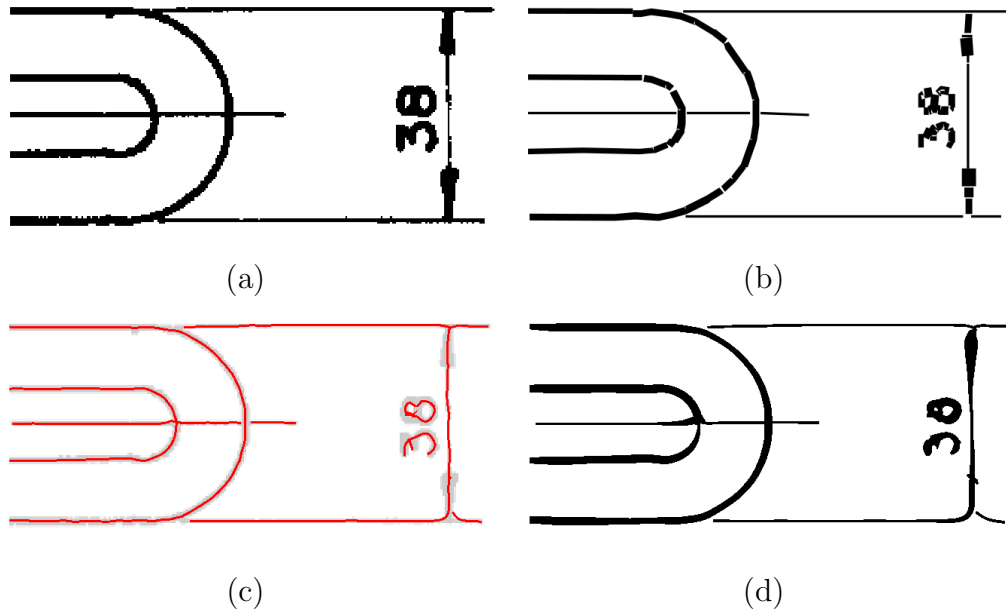


Figure 4.15: Principally designed for organic and cartoon images, structural vectorization can also be applied to architectural or engineering drawings (a). Compared with a basic line-based vectorization method (b) [55], structural vectorization produces comparable structural data (c), and performs better on curved lines (d).

properties such as curve, camber, thickness and relative position remain the same. It is not possible to keep all properties identical, but preserving the correct properties can result in a better form than linear scaling. This section outlines a simple *form preserving scaling* method that preserves curvature at the cost of relative position, and does a better job at retaining the original visual style.

The core of the algorithm is that images should be scaled only at places with minimal image complexity. Areas with high complexity are left as close to untransformed as possible to reduce distortion, while lines in the image close to the point of scaling have their endpoints scaled but the curvature modified so as to best preserve the form. This type of content-aware scaling was first proposed by Kwatra et al. [174] in 2003, with their paper about *Graph Cuts*. While numerous improvements and alternatives such as seam carving [13] have been suggested by other researchers, these techniques are primarily focused on photos and raster images and are not applicable to

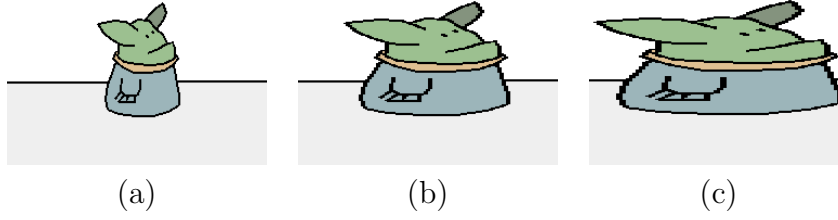


Figure 4.16: Standard horizontal stretching, with the source image (a) shown at two (b) and three (c) times larger.

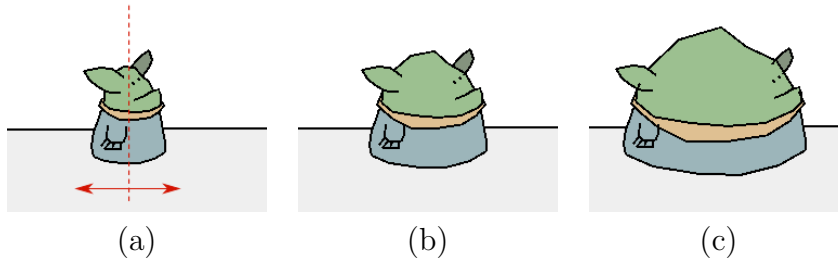
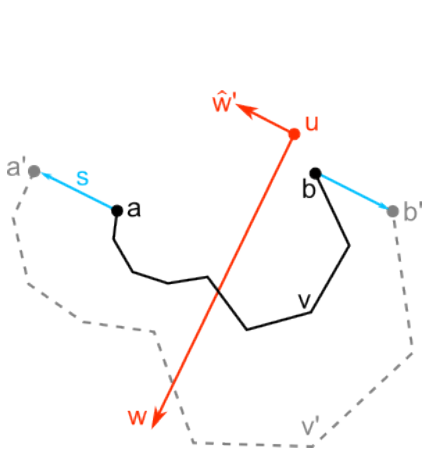


Figure 4.17: An example of complexity adjusted scaling that preserves the features from the original image (a) even when it is scaled horizontally (b & c). The scaling direction is shown by the red arrow, and the dotted red line is the orthogonal bisection line.

vector graphics.

The place in the image with minimal complexity is defined to be the line that bisects the image with the least number of crossings. This is multiplied by the smallest distance from the image centre to the bisection line, so as to reduce the bias toward the borders of the image. The bisection line is always a line orthogonal to the scaling direction specified by the user. While a raster-based transform approach using grids is shown in Chapter 5, a vector-based solution provides better quality in applicable situations. Each line in the image that crosses this bisection is scaled using Equation 4.9 so as to preserve the curve while still keeping the endpoints connected.

Figure 4.16 shows a standard scaling applied to a character, while Figure 4.17 shows the results of curvature-preserving scaling using our algorithm. The bisection line is shown in red.



$$\begin{aligned}
 \hat{w}' &= \frac{w^\perp}{|w|} \\
 a' &= a + s |(a - u) \cdot \hat{w}'| \\
 b' &= b + s |(b - u) \cdot \hat{w}'| \\
 R &= \text{rotation} \left[(b - a)^\perp \cdot (b' - a') \right] \\
 S &= \text{scale} \left[\frac{\|b' - a'\|}{\|b - a\|} \right] \\
 T &= \text{translate} \left[\frac{(a' + b') - (a + b)}{2} \right]
 \end{aligned}$$

$$v'(x) = v(x) \cdot R \cdot S \cdot T \quad (4.8)$$

$x=0 \rightarrow n$

where the bisection line is defined by point u and direction w , and the distance to scale is s . The set of points $v(0..n)$ defines the line being scaled, with endpoints $a = v(0)$ and $b = v(n)$. R , S , and T are 2D transformation matrices.

This type of scaling is useful in the context of visual style because it modifies the appearance of an object in an intuitive way without disrupting the underlying form or distorting important details.

Chapter V

Characteristic Proportions

5.1 Introduction

In this chapter, a new method is outlined for automatically extracting an individual artist's style based upon the characteristic proportions of an image. This can then be applied to other drawings. Unlike previous work which typically transfers artistic elements such as line style, characteristics are captured and transferred with spatial context. When two different artists draw stylised versions of the same object, the exaggerations and shapes within the image are often significantly different. By measuring these properties across a large body of the artists work, it is possible to discover which proportions characterise the artist, and which are inherently part of the object.

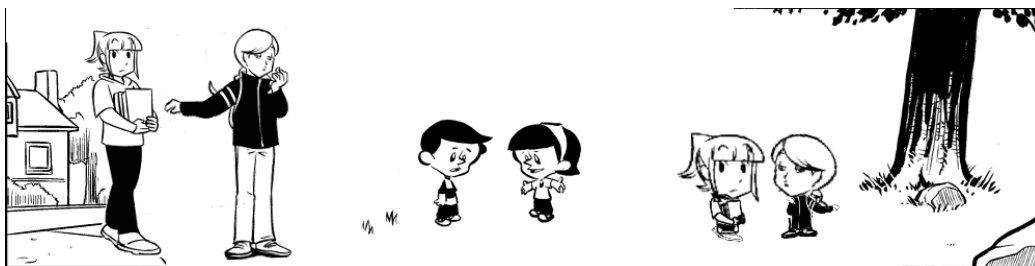


Figure 5.1: Characters from the comic strip ‘Sandra and Woo’ have the artist style from ‘Misery Loves Sherman’ transferred to them. Left and Middle: Original characters. Measurements that are specific to an artists’ style are automatically differentiated from measurements that belong to the object. Right: Left characters using the Middle character’s style.

Characteristic proportions define the relationships within a set of similar shaped objects by the same artist. Figure 5.1 shows how the style of an

object can be changed by altering its characteristic proportions to match the characteristic proportions of an image in the target style.

For example; a character with a large head and small body will have different characteristics to a character with a large head and large body, even if their overall height and width are the same. In this respect, objects of the same type drawn by the same artist have very similar characteristic proportions, whereas objects of the same type drawn by different artists often have different characteristic proportions.

Hertzmann et al. [79] found a number of issues when transforming line style, and in their words “limitations would be expected with any low-level approach to style translation that treats a drawing as just a set of lines with no additional structure”. Characteristic Proportions and a style transformation approach using Geons is an attempt to provide this additional higher-level structure that can be used to perform style translation without changing image features past the point of recognition.

The algorithm outlined in this chapter automatically analyses the images, and creates a database with the characteristic proportions of a single artist. Given an input image and the appropriate Geon subdivision, the algorithm finds the most likely target proportions and automatically resizes the source image to fit them. Using the complexity sensitive scaling algorithm in Section 4.4 prevents character details from being distorted. The transfer algorithm does not deal with additional properties, and so the results are constrained to line drawn cartoon images to reduce the influence of rendering technique in the portrayal of style, focusing upon the proportions alone.

The results show that analysis of object proportions can be used practically to ensure consistency for objects with the same Geon structure but drawn by different artists.

5.2 Existing Studies

The idea that Characteristic Proportions can be used to characterise part of the style of an object is based upon a number of well-supported theories in cognitive psychology [11] [81], and has been implemented to some degree and with varying degrees of success in several existing research papers. Various

parameters used in the Geon system in this chapter were determined using principles from the following research and from evaluating the outcomes of previous attempts.

5.2.1 *Geons*

The word ‘Geon’ was first coined by Binford in 1971 [24] and was used in the description of *geometric ions*, so-called generalised cones formed by the volume swept out by a cross section moving along an axis. After occasional usage, the term was popularised by Irving Beiderman in the paper ‘Recognition-by-Components: A Theory of Human Image Understanding’ [23], where ‘Geon’ was used to denote simple geometric primitives. In most work, the word Geon is used to mean a feature extracted from a 2D image, however a number of authors also use ‘Geon’ to refer to 3D features extracted from a model.

Wu et al. [184] extract 3D Geons from noisy data by using parametric models. A surface-error metric is used to measure the distance energy between a point cloud of rough data and parametric functions representing seven of the most common modelling primitives. Because there is only a limited number of ground shapes the result may not be an accurate match, however the algorithm returns consistent results with very few failure cases. This is ideal for systems such as the one outlined in this chapter that must deal with a wide range of dissimilar input data, and so despite the inflexibility a 2D parametric system is worth exploring in this context.

Similar systems are outlined by Osada et al. [138] and Funkhouser & Kazhdan [67]. These use non-parametric distance metrics to match similar 3D shapes. These could be easily adapted to 2D, and if a database of Geon shapes were used this type of distance matching could provide accurate extraction for sets of shapes with varied internal data. A degree of flexibility is implied by the distance metric, however the difficulty developing such a system is defining the initial Geon shapes for large input range.

As outlined in Section 2.2, perceptual categorisation [11] [81] is one of the dominant visual recognition theories and can be loosely summarised by the idea that small primitives, combined with their spatial relationship, create

a full image. Much as image properties such as line stroking and the colour palette have a direct influence on the perceived style of an image, Perceptual Categorisation suggests that the size and position of small primitives within an image can also be considered as image properties, and therefore have a direct influence over the visual style of the image.

When creating and using Geons as a style metric, the properties are split into a number of categories. There are the direct metrics such as the size and shape of a Geon, but also the indirect metrics as suggested by perceptual categorisation which are the size, shape, rotation and position of the Geon in relation to other parts of the image. This insight forms a fundamental part of the Geon system outlined in this chapter.

‘The Development of Features in Object Concepts’ [148] looks at the relative importance of each geon within an object. It forms the hypothesis that different object categories have different orders of geon importance, and that geons differ in importance and form between people.

This is taken one step further by Ullman et al. [169] who also look at Geons in their paper “Visual features of intermediate complexity and their use in classification”. They find that visual features of intermediate complexity (IC) are of best use in image classification. A training set of cars and faces was used to test an image recognition algorithm. Geons of different sizes and complexity were used in the recognition process, which was based upon standard recognition techniques. It was found that IC features were optimal for the task. Ullman et al. continue on to state that the advantage of IC features has a biological grounding and that they tend to emerge naturally in this type of task.

Looking at this research in the context of visual style and specifically the task of Geon representation, it suggests that keeping Geons at an intermediate complexity should produce the highest quality results in both analysis and remapping. The exact definition of IC features is outlined in the paper and can be used as a reference when creating these Geons.

Although not explicitly stated in the paper, Chris Hecker [77] implemented a system using IC features in 3D as a method to retarget animations to unknown and complex character morphologies. The paper found that categorising structural components (‘hand’, ‘mouth’, etc) allowed for much

better generalisation of the animation, something that standard animation re-mapping did not do well. This provides evidence that IC features perform better than low level features and suggests that there may be a way to use structural components to develop a better generalisation of visual style.

In the context of this thesis, a *Geon* is defined to be *a unit of intermediate complexity within the image that is obvious to an observer*. This is represented mathematically by a single primitive that encompasses the Geon. This is usually a rectangle in the form of an oriented bounding box.

The idea of image decomposition is not new, and similar types of multi-component models are commonly used for object recognition and classification. However no existing studies could be found using context-based decomposition systems to perform visual style categorisation or transformation. The closest systems implemented basic geon classification with the aim of simulating or representing the cognitive model of how the human perception system works. While the full field of cognitive psychology and recognition models are outside the scope of this thesis, a literature review by Palmeri & Gauthier [141] gives a good overview of the most widely recognised models.

These cognitive models are useful when considering papers such as ‘Abstraction of 2D Shapes in Terms of Parts’ [122]. This research describes shape abstraction, “using a new synthesis of holistic features” which creates simplified shapes that retain important features. The holistic features can be explained by the cognitive models which show the importance of corner-invariant detail scaling in relation to the perception of style. The ability of their algorithm to perform context-sensitive part division is important when considering the need for context when creating metrics for manipulation of style.

Hummel & Stankiewicz implemented a program called Metricat [85] that looked at Geons in 3D models, classifying them purely on their relative locations. This was enough to classify different objects, however because the program was designed to model the brain’s object classification approach it also modelled the debilitating factors the human brain introduces into classification. Implementing a similar system without the artificial drawbacks would be more appropriate for visual style manipulation. Despite the limitations, the results Metricat produces prove that categorisation via the use

of shape abstraction is both possible and effective.

In their paper *Shape Distributions*, Osada et al. [138] implement a visual search system for 3D models. While this deals only with classification of objects, it works well and presents a number of interesting measurements and metrics that could be taken from 3D models and potentially 3D line drawings. In the context of style measurement, these could be very interesting bases for metrics and descriptors that could be extracted from images and recorded along with other Geon information such as position.

Bartlett Mel [120] implements an object recognition system and uses it to explore the idea that feature-point matching algorithms closely represent the neurobiological recognition process. The accuracy reflects similar algorithms, however when evaluating degradation the system coped especially well with piece-wise scrambled images, . One conclusion was that “object recognition in shape space must in some cases permit an object whose global shape has been grossly perturbed to be matched to itself, such as the various tangled forms of a telephone cord [...]”. Later discussion of local and global feature matching emphasises again that Geons, and specifically subdivision of an object into Geons, is key to understanding an object at all feature levels.

5.2.2 Image Subdivision

A large amount of the research in this chapter relies on access to a database of Geons extracted from images. Initial feasibility studies were done by hand, however an automated system requires a method of subdividing characters into Geons irrespective of pose, and identifying characters within images.

One system that comes very close to achieving the aim of extracting Geons from characters is the research outlined in ‘Learning Primitive Shapes in Cartoon Designs’ by Islam et al. [89]. Simple geometric shapes are used to represent parts of cartoon images within a predefined layout (in their results they use a humanoid shape). Principal Component Analysis is used to identify similar geons in other occurrences of the same character, and a Support Vector Machine classifier is used to find whole examples of the character within a larger database. The classification success rate was 89% and while Islam et al. struggled differentiating objects with similar styles,

this is a positive when viewed in the context of the proposed system and supports the idea that style classification can be performed using only basic metrics. Unfortunately this research is of little use as an automated geon extraction system as every image was required to be annotated by character and concept artists before use.

Li, X. et al. [107] present a comprehensive method for decomposition of complex shapes into Geons, thereby developing a structural representation of the context. This is ideal for the proposed algorithm, and the method performed well on 3D meshes. However the internal structure is built from an analysis of the object surface, a feature that is exceptionally difficult to extract or even approximate from 2D images. A similar 3D extraction system is used by Theobalt, C. et al. [166] for object replacement and space filling. The difficulty of applying these methods to 2D images, combined with the target focus upon a limited domain means that a specialised subdivision algorithm is easier to implement and likely to function better in the 2D case.

One method for shape extraction would be to use a prototype Geon and a recognition algorithm to find occurrences of this in other images. Most widely used image recognition algorithms use a variation of SIFT [112] or SURF [19] followed by a feature reduction and matching step often based on RANSAC [61]. These algorithms find occurrences of a prototype image in a larger target image, coping well with variances in shading, perspective and scale. However, cartoon content often does not produce good feature points, and data within the geons is not geometrically or linearly transformed. An exception to this is research by Hu & Nagai [83] who adapt feature points for text line data, however the template matching is designed for over-fit data and works poorly finding matches in the sparse feature cloud often generated by cartoon imagery.

Two solutions could potentially overcome this problem. A system based on exemplar based recognition would be less susceptible to non-linear transforms, and has been demonstrated to work on large datasets [65] and characteristically similar datasets [92]. However, the heavy training requirement of such a system defeats the purpose of implementing one in this context. Graph matching [167] [51] also copes well with non-linear transforms, however often relies on SIFT-based methods for featurepoint extraction and therefore

struggles with cartoon images. Even in the case of global features [183] graph matching fails for cartoon images. It has however been used to some success on a pixel-to-pixel basis [101].

One completely different approach is to use a vectorized version of the outline with a neural net classifier. Xiong et al [186] found that this worked better for shapes with outline noise, a key problem area for cartoon shape recognition. However this type of result with a neural network was not easily reproducible, and constant outline deviations were found to cause the recognition rate to drop sharply. Alternatively, if a single point could be found with certainty within each Geon, a distance-based region growing algorithm [113] could be used to extract the shapes.

One of the best solutions for image matching line drawings is to use a shape context descriptor [21]. This has high accuracy when matching line drawings and monochrome images, however, requires that target images have little background interference. The requirement is for a robust image extraction algorithm that can find multiple different occurrences in complex target images using only one source image. This is made easier because the data is restricted to line drawn cartoons, and the approximate location of subsequent occurrences is known due to the Geon structure. Because of these very specific requirements, a custom search algorithm is implemented.

Another approach to Geon extraction could be found through shape simplification. Mi, X et al. [122] propose a method for shape abstraction, whereby a complex outline is simplified so as to preserve only the important details. To do this they create perceptual parts organised by holistic features based on the shape, features, boundaries and global organisation. These heuristics could be used to rank or classify Geons within the proposed system.

Overall, a number of researchers explore the idea of using Geons for image understanding and recognition. The aim of this chapter is to apply this understanding in the field of visual style, to recognise and modify images based upon the artist. A number of technical challenges exist, in regards to creating an object database and subdividing characters into geons. However a large body of prior research exists and at least one of these techniques should be adaptable with only minor modifications to this new application.

5.3 Evaluating Geons

In Section 5.2 research papers were discussed that introduced the idea of Geons, and used these in object recognition and other systems. However, it appears that no existing research uses Geons in the context of visual style. This chapter proposes that Geons can be used to identify, categorise, and modify the visual style of an image. This can potentially be in concert with existing research that identifies image properties such as line and colour. Before developing an in-depth system based upon this premise, initial exploratory research must be carried out to assess the feasibility of using Geons in this area.

To best evaluate the relationship between Geons and visual style, external factors are reduced by constraining the input as much as possible. To this end the input images are all simple greyscale line drawings that show full-body characters from either the front or the side. This research focuses initially on humanoid characters because they are common in drawn media, providing a large body of comparable and easily subdividable objects to draw upon for input images. Additionally, characters provide a reasonable number of easily identifiable Geons that have a direct correspondence between images, artists, and styles. The characters have similar technical characteristics - line style, colour, overall size, and yet because they are drawn by different artists they are perceptually distinct. These constraints reduce external factors that could influence the measured correlation between Geons and style. Figure 5.2 shows examples of the characters used for input.

For the initial study, three datasets are chosen. While the intricacies of visual style and the utility of Geons can be explored later, these simple datasets allow basic tests to be carried out that will show whether Geons are a viable method for measuring style. Because artists with distinct visual styles are used, the categories 'artist' and 'visual style' are used interchangeably within this section.

Based on the research of Ullman et al. [169], a Geon is defined to be *a unit of intermediate complexity within the image that is obvious to an observer*. In the context of human characters, this corresponds roughly to the arms, legs, body, and head. In the example images, the arms are often too small or



Figure 5.2: Examples of the input characters. Row (a) shows characters from Calvin and Hobbes by Bill Watterson, row (b) shows Peanuts by Charles Schulz, and row (c) characters from PhD Comic by Jorge Cham.

poorly defined to be used as a Geon, and therefore characters are split into the head, upper torso, and lower torso. The splits are performed at the neck and waist.

Figure 5.3 shows an image being manually decomposed into Geons. Geons are aligned with the perceived axis, and subdivided so that the centre of the borders of adjacent oriented bounding boxes meet exactly at each split line. At least 4 unique characters were used for each artist, with a minimum of 4 instances of each character to give 16 datapoints and 48 Geons per artist. Metrics are associated with each Geon, and the differences and similarities are explored to determine if it is statistically possible to determine a difference between the three chosen artists. The metrics used in this section are height, width, angle, and position. Figure 5.4

Figure 5.5 is a scatterplot that shows the size of head geons from characters by Jorge Cham and Charles Schulz. The datapoints form two distinct

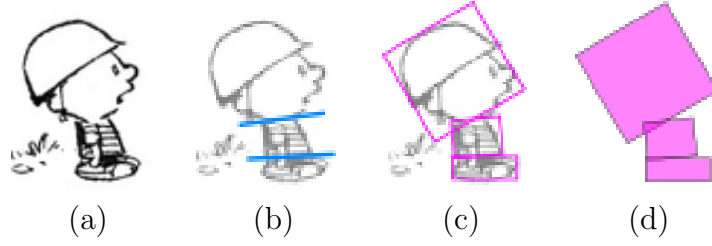


Figure 5.3: An input character (a) with perceived splits at the neck and waistline (b) is decomposed manually into Geons (c & d).

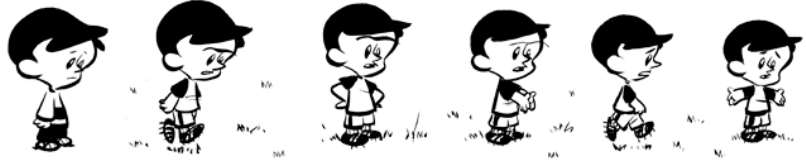


Figure 5.4: Examples of the instances of a single input character. This row shows 6 instances of the character 'Sherman', extracted from panels from the comic strip Misery Lover Sherman by Chris Eliopoulos.

groups and shows that very simple Geon metrics can be used to identify artists. Due to the different data sources the two groups have unequal sizes and variances, and so Welch's T-Test is used to analyse the point distribution along the principal axis $(0.89, -0.46)$, giving a T-Value of 9.66. Performing a two-tailed test gives a P-Value of less than 0.00001, which confirms that the difference between the two artists is statistically significant.

However, this clear division does not always occur. Plotting the size of head geons between Charles Schulz and Bill Watterson gives seemingly uncorrelated data, as can be seen in Figure 5.6. Performing the same T-Test between the datasets gives $p = 0.77$, which shows the difference is not statistically significant. From an intuitive point of view this makes sense, because the two artists draw characters in a similar style and it can be seen that the size of the characters and their subcomponents are similar. However, humans can still perceive a clear difference in the two cartoons, and so with the correct metrics it should be possible to find a distinction.

Height and width are first order metrics, along with rotation and position.

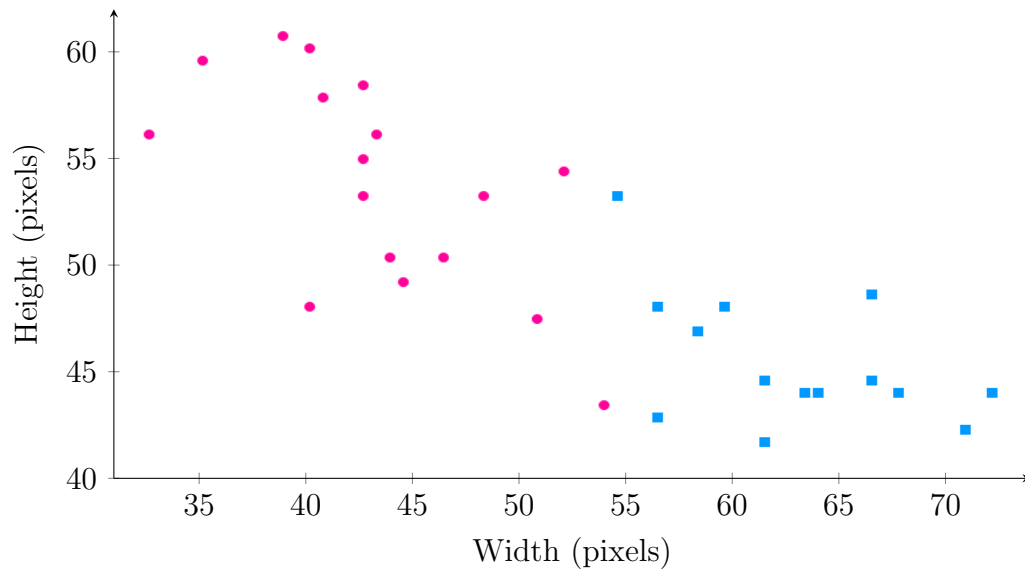


Figure 5.5: Scatterplot showing the width and height of head Geons in images drawn by Jorge Cham (pink) and Charles Schulz (blue). This simple metric is enough to clearly distinguish the two artists.

Second order metrics can be plotted using the differences between the first order metrics. The relative position between two neighbouring geons is a second order metric, and is often more useful than the first order absolute position. Figure 5.7 shows a complex pairing of metrics that were found to easily differentiate the two artists. Each point on the plot represents a second order metric, for example the relative sizes of the head and the body, or the body and the feet. The horizontal axis shows the distance between the neighbouring Geons, and the vertical axis shows the difference between the diameters of the two Geons. Points closer to the left show Geons that are similar in size to their neighbours, and points closer to the bottom show a larger overlap or a stronger locational correlation between the Geons. Points are linked to show Geons within the same object. Each character is therefore represented by a line, connecting the head-body Geon with the body-feet Geon.

The most obvious feature in this plot is the distinct grouping in the Geon data from Bill Watterson. The two groups represent the two different types of characters drawn by Watterson — adults and children. These have

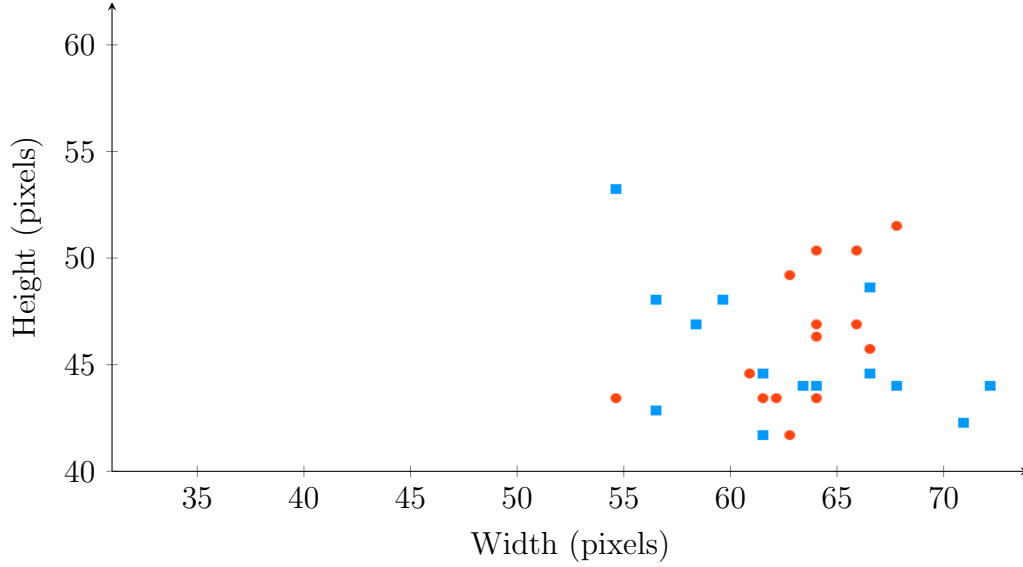


Figure 5.6: Scatterplot showing the width and height of Geons from images drawn by Bill Watterson (orange) and Charles Schulz (blue).

distinctly different spacing between Geons, but as can be seen by the angles of the connecting lines, the shape progression from head to feet is the same, and with the exception of two outliers, it remains completely different from Schulz.

Analysing a third order metric shows a clear distinction between the two datasets. The slope of the connecting line between the matching Geon pairs is measured and shown in Table 5.8. Welch's T-test is again used to compare the two distributions. The T-value is 6.27 with a P-Value less than 0.00001, indicating a significant result. This shows that there is a separation between the two datasets and clearly demonstrates that it is possible to identify differences between artists even in cases where the Geon metrics are very similar.

This exploration shows that using Geon metrics to recognise style is a viable approach. One of the biggest challenges in automating this system is determining which metrics should be used for the classification, as there are a large number of different combinations to select from and only a small number of these show significant differences between artists and datasets. The significant metrics change between artists, and it was found that higher

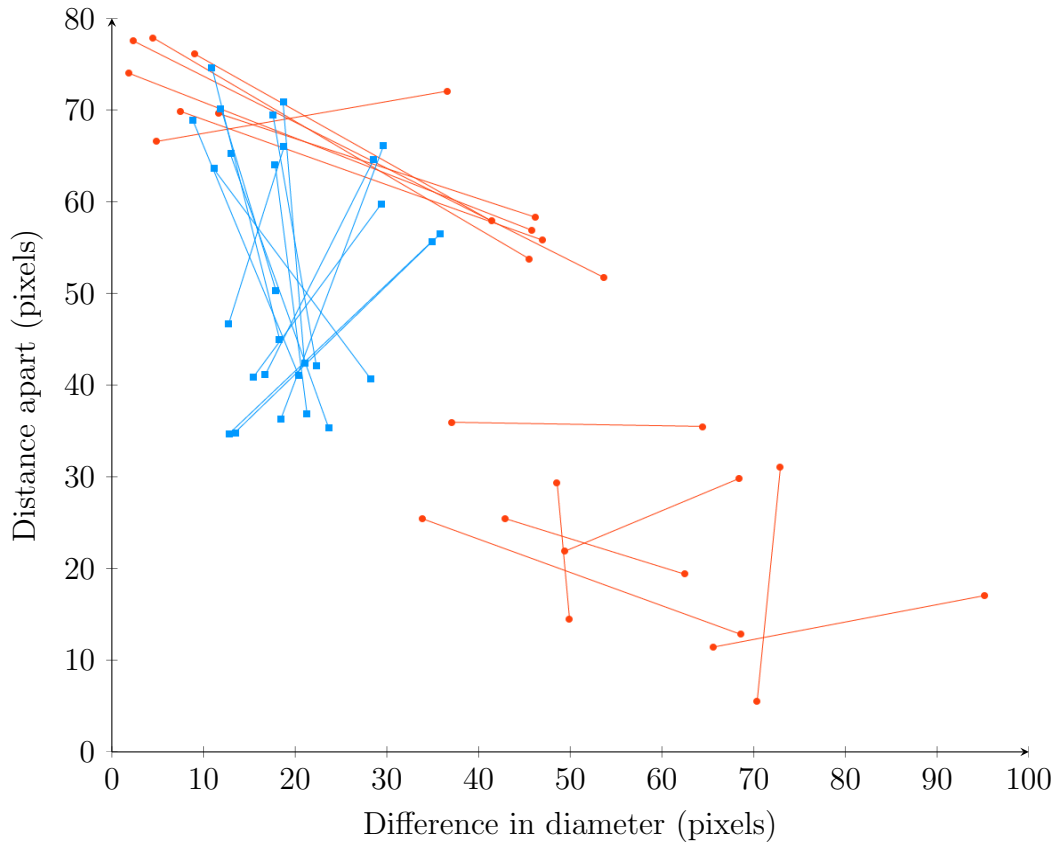


Figure 5.7: Linked scatterplot showing second order metrics. Points represent Geon pairs, with the horizontal axis showing the difference in diameter and the vertical axis showing the distance between the Geons in the pair. Lines connect Geon pairs within the same object. Orange shows Geons drawn from images by Bill Watterson, and blue from Charles Schulz.

order metrics were important in cases where datasets were similar. This finding helps to explain why previous work often had limited results, as only one value was commonly used. For example, MetriCat looked only at absolute Geon position, which suffices for object recognition, but doesn't vary as much due to artistic style as second order relative position does.

Equally important to finding distinctive metrics is knowing when metrics have little correlation to style. An example is the size of the legs or feet Geon. Because the input characters were a mix of front and side orientations, the Geon width could be based upon either the width or depth of the feet,

	μ	σ
Bill Watterson	1.17	0.55
Charles Schulz	-0.08	0.45

Figure 5.8: Mean and standard deviation for the line slope in Figure 5.7. Measurements are in radians. Welch’s T-Test for separability gives a significant result of $p < 0.00001$.

and may vary between instances of the same character more than between artists. A similar problem is differentiating between character-specific and style-specific metrics. Using overall character proportions provides a clear distinction between Charles Schulz and Jorge Cham because the artists draw children and adults respectively. As can be seen in Figure 5.7, Bill Watterson draws characters in both classes, and any attempt to use this metric will correctly classify by type but incorrectly classify by artist. Section 5.8 outlines a method of determining whether metrics such as these should be discounted from any categorization or style transformation being performed.

Overall, this section validates the idea that Geons are related to artistic style. In Section 5.7, a software database system is outlined that allows the creation of a metric probability mapping. This enables automatic style recognition and modification based upon Characteristic Proportions.

5.4 *Algorithm Overview*

The core algorithm analyses a database of images and detects consistent metrics across the dataset. It uses these consistent metrics to measure the properties of an image and determine which properties are characteristic to the artists style. This enables style-based operations such as classifying images by artist, or changing the artistic style of an image. Manually building this database and finding the characteristic properties takes a considerable manual effort, and so much of the work in this chapter focuses upon building a database from unknown source images and finding this data automatically.

From an initial image of a character or object, a structural description is automatically created to outline the significant components. A combination

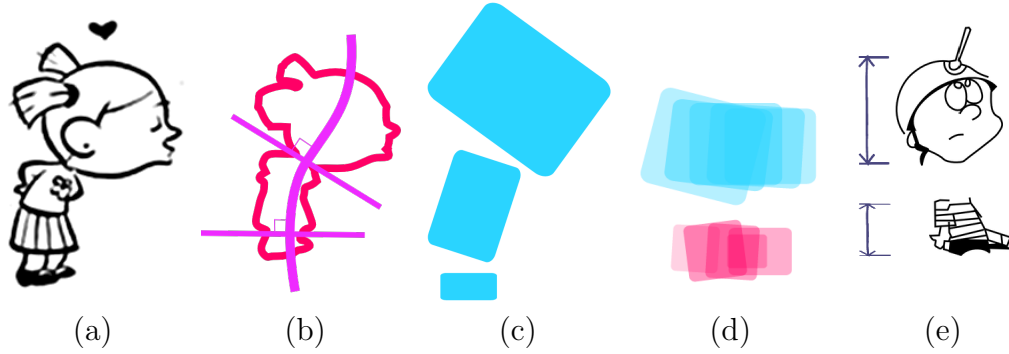


Figure 5.9: The process of modifying an image using Characteristic Proportions is fairly involved, but can be broken down into a number of main areas. The initial image (a) is decomposed (b) into a structural description (c). This is repeated for a number of images and characters by a single artist to build up a description database (d). Images by different artists can then be transformed (e) to better match the range of Characteristic Proportions from the original artist.

of the bounding hull, image shape and complexity are used to subdivide the object into the basic geometric units (Geons) that are used to measure characteristic proportions. This initial 'prototype' information is used to extract similar objects from a large number of source images. After the new objects are extracted and corresponding Geons fitted, the characteristic proportions are entered into the database. Figure 5.9 shows a brief overview of these steps.

These proportions are measured across a range of images from the same artist, and the distribution of measurements is used for categorisation. Each property is categorised depending upon whether it is part of the artists' characteristic style or unique to the specific input image and therefore irrelevant when performing the style modification. The style modification itself takes an image by the source artist and modifies the proportions to those characteristic to the target artist. Attention is needed to ensure the image content is preserved correctly when the properties are modified.

This process transforms an image into the same style as the target artist. The full algorithm is described in the following sections, and the results displayed in section 5.9.

5.5 *Prototype Subdivision*

To extract the target proportions automatically from a large set of images, a basic prototype is supplied to demonstrate the object. This prototype is selected by the artist or user, and is simply an example of the character without any background. This needs to be divided into unique shapes that represent graphically significant components of the object. These are later used to extract the object from the background in target images, and then measure their rotation and scale.

The object prototype is important because it heavily influences the extraction and measurement of source images that are used to compile the artist database.

A naive algorithm has been developed to do the separation. It relies on the exaggeration present in most cartoon images and the texture pattern if the image is textured. If the image is not textured, then the internal lines are used. Based on a sample of common images, an assumption is made that large variations in width or internal complexity indicate the boundaries between different parts of the object. A common case is a cartoon character, where the ideal focus is upon the proportions of head, body and legs. These form three geons of intermediate complexity.

First, a centreline is drawn through the longest axis of the image. If the overall shape of the main character or object bends, the centreline is also bent to account for these. If the object forks, then the most significant branch is chosen based upon its width and length. The complexity of a raster image at a point on the centreline is measured as the sum of the absolute value of derivatives along an orthogonal cross-section, while the complexity of a vector image is the number of line crossings along the cross-section. A bounding hull is also drawn around the image, using a single step point contraction as outlined in Section 6.4.3. The object outline is not guaranteed to be closed, because it has been drawn by an artist and the style may rely on gaps or spaces. To ensure the border contraction doesn't encroach too far into the object, a limit is imposed. This trades technical accuracy for a correct visual border. Figure 5.10(a) shows how this bound is used to calculate the width in respect to the centreline. Contraction is used in place of a convex bounding

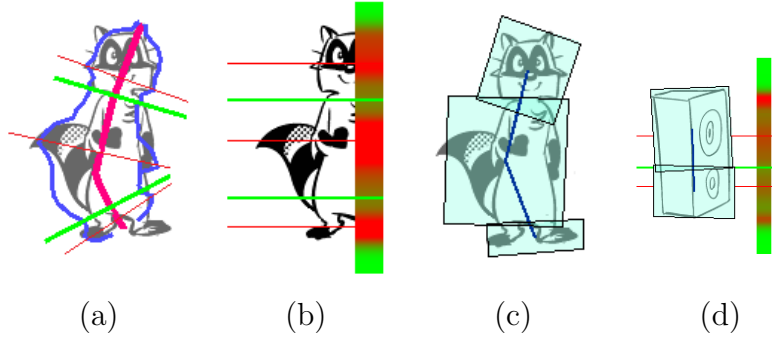


Figure 5.10: A cartoon character and object automatically subdivided into appropriate Geons. The pink centreline and blue bounding hull line (a) are used to find appropriate subdivisions (green) based on width. Note the bounding hull is centered and therefore does not visually match the image borders. The image complexity (b) is shown untransformed, and green lines mark approximate locations of low complexity. The character geons (c) and object geons (d) are created based on these subdivision lines.

hull because the convex sections of the outline are integral to the subdivision process.

$$f(t) = \frac{\frac{w(t)}{w_{max}} + c(t)}{2}$$

split when $f'(t) = 0$ and $f''(t) > 0$ (5.1)

where t is the distance along the centreline from 0 to 1, $w(t)$ is the outline width orthogonal to the centreline at a point, and c is the image complexity in the range $[0..1]$.

After these measurements are established, an evenly weighted combination of smallest widths and lowest complexity are used to find potential segmentations for the object, as per Equation 5.1. The final number of sections is determined by the size of the image and the strength and placement of the potential cuts. Geons are created aligned to the centreline, using the height of the section and the maximum width of the section. Figure 5.10 shows the result of this subdivision process.

The width of the image is measured at the adjacent boundaries of each Geon. The adjacent boundaries are considered to be those where the connectivity graph crosses, where changing the scale of the Geons could cause discontinuities in the image. These measurements are later used to ensure correct scaling along the seams when modifying a target image.

In addition to boundaries, internal measurements are recorded using an arbitrary size grid aligned to the Geon. It was found that a subdivision of 6 gridlines produced the best visual results for the style transform. At each intersection, the complexity of the image is recorded, and along each gridline the endpoints are recorded where they cross the bounding hull. These measurements are later used to influence the contents of a Geon when scaling.

This method works without intervention for a range of common and simple cartoon objects, however does not correctly subdivide complex or branching images. In this case the object can be subdivided and separate prototypes generated from the pieces. This is considered a viable solution, because complex objects are of limited use in this system. If the topology is too complicated, it reduces the chance of finding a corresponding object in the database of the other artist. If an object cannot be subdivided, then geons can be placed manually as a last resort.

A more detailed body segment extraction method is proposed in Section 6.4.3 and the results are often higher quality than those in this section. However the methods are largely incompatible due to the way they are used. Research in *skeletonization for model reconstruction* uses a single well-defined image source that has been drawn intentionally by an artist to fit a specific pose. Building a database for Geon representation requires a large number of different images, and so they cannot by definition conform to a single pose. Additionally, analysis using characteristic proportions is designed to be run on existing datasets, and the artist cannot have been expected to draw characters in a given way. Using prototype extraction trades accuracy for robustness while still providing acceptable data for Characteristic Proportion research.

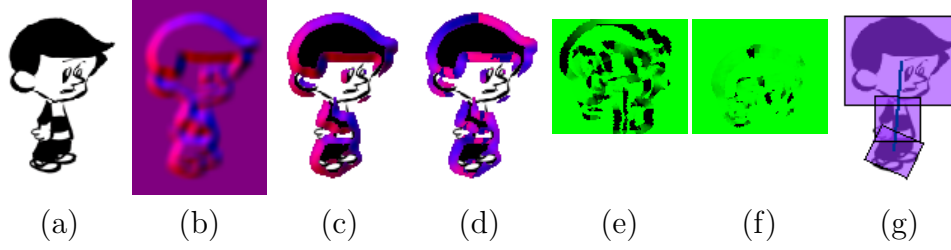


Figure 5.11: Geon Extraction is performed using an algorithm designed for this specific case. The derivatives of the source (not shown) and target (a) images are combined with the flat colours (b), and vectors represent the line directions (c). A loose correlation is performed (d) at every pixel, where gradients and colours are matched within a radius. Correlation is performed across a range of rotations and scales, and the lowest difference (e) is considered to be a match. Subsequent Geons are matched within smaller areas (f) determined by the prototype. The final output (g) contains the size, rotation and position of each Geon in the target image.

5.6 Image Extraction

To build a database with a large number of entries, instances of the target object or character need to be measured within a large input dataset. Occurrences of a character are often found in context, and so need to be extracted from the background before they can be used. This is especially challenging, because the character in the target image is usually in a different pose to the character provided as a prototype. These differences in pose cannot be ignored because they provide the data required to perform the style transfer. Each Geon therefore needs to be extracted individually.

This extraction is performed using a brute-force algorithm that uses line directions and cartoon fill to match two images at a point. Processing is performed on most complex Geon first to reduce the number of false matches. In the rare case where no matches are found, the process is repeated with a less complex Geon. This metric is determined using the same algorithm as outlined in Section 5.5.

Unlike photos, the images being matched have strong lines and large areas of flat colour. A special featuremap is generated for each image that equally emphasises these features. First a signed vector field $v(x, y)$ is gen-

erated from the image intensity channel to allow easier access to the line data, and then thresholded to remove insignificant slopes or noise. Specifically, data is removed in areas of low intensity change, specifically where the length of the vector is below a given threshold. If $\|v(x, y)\| < 0.5$, the signed vector is replaced with the flat colour from the image. If the image instead has a pattern or noise, this is removed by averaging the colour. Prototype images may also specify transparent areas that will not be used during correlation. This is important when matching a prototype image with a blank background with a target image that has an illustrated background. Making the prototype background transparent ensures it does not falsely reject a correct foreground match because the backgrounds do not match.

The edge direction is found by taking the gradient of the derivative in both the X and Y direction, and normalising the resulting vector. A naive search is then performed by measuring the loose correlation of the source and target for every pixel in the target image, with the source image rotated at increments of 1° and scaled by 1 pixel up to a range of 10%.

The loose correlation performed is similar to standard image correlation, except that overlapping pixels are also checked against nearby pixels to allow for differences in the drawings. Equation 5.3 shows the correlation between source and target for a single pixel. This loose correlation solves the situation where lines are mis-aligned by only a few pixels, yet do not match and hence distort the correlation result. Bearing in mind the previously described sensitivity threshold of $\|v(x, y)\| < 0.5$, Equation 5.2 shows how the contribution of each pixel is based upon the vector direction if applicable, or based upon the flat colour if there is no vector. The best radius for the sample dataset was determined to be 2% of the image size, but this needed to be increased for excessively noisy or different images. The closeness of a match at a specific transformation is taken to be the sum of all pixel correlations, and the transformation with the lowest energy is the best match.

$$f(x, x', y, y') = (1 - |s_v(x, y) \cdot t_v(x + x', y + y')|) + \quad (5.2)$$

$$|s_c(x, y) - t_c(x + x', y + y')| + r$$

$$match_{pixel}(x, y) = \min \left[f(x, x', y, y') \right]_{\substack{x+r \\ y+r \\ -r < x' < r \\ -r < y' < r}} \quad (5.3)$$

where s is the source image and t is the target image, with v and c representing the derivative vector channel and pixel colour channel respectively. r is the sampling radius.

After the transformation of the first Geon is found, subsequent correlations do not need to be run across the entire image. The Geon structure in the prototype is used to determine the search areas for subsequent Geons, with the correlation being run in a radius of $2r$ around the location indicated.

This method worked sufficiently well for the sample dataset, requiring no training and needing calibration only in outlying cases. In these cases, it was sufficient for the user to manually indicate the approximate area for each of the geons. However, the naive search across all pixels in the image is an $O(n^3)$ algorithm and optimisation of this algorithm is left for future work. Additionally, this matching algorithm does not work on photographic images or images with high texture content that is liable to be miscategorised as lines. Given the specific nature of the dataset, these are acceptable tradeoffs.

This type of image matching would ideally be performed using vectorized data such as that produced in Chapter 4. This format has lower noise and better detail for low resolution images. However, a vector approach is not feasible due to the lack of robust vector matching algorithms, and the requirement for matching different poses. Overall, the delivered Geon data is the primary focus and in this respect the exact extraction procedure is not critical to the use of Characteristic Proportions in artistic style manipulation.

5.7 Database Analysis

This algorithm provides the data needed to perform a proportion transfer. Within the database are a set of objects, with their subcomponents (Geons) in different poses. Some Geon properties are important when representing an artists's style, while others are either irrelevant or apply only to a specific drawing of an object. By analysing the database for consistent configurations across the dataset, and discarding the properties that are drawing-specific, the Characteristic Proportions for a specific artist can be measured.

Figure 5.12 shows a database of extracted Geons for a particular artist. The Geon scale and position is measured relative to the prototype image, removing the scale bias that would otherwise be caused by different sized target images.

Further bias can be removed by running the analysis process on a variety of objects of the same type, from the same artist. This works for simply objects in a scene, however running the process on characters often results in clear categorisations between different types of character within an artists' portfolio. Groupings such as 'Adult' and 'Child' or 'Good' and 'Bad' emerge. The differences in these groupings are more pronounced than the differences between artists. However the differences between, for example, the 'Child' groups from different artists is still significant enough to perform an effective style transfer.

Several measurements are recorded for each Geon and compared to the other Geons in the same class, in all other instances of the character. The size, angle, and distance to adjacent Geon(s) are compared. In addition, the differences in angle and size between adjacent Geons within a character are measured. For each property, the database records the average (μ), the standard deviation (σ), and the total range in each direction (r). Exceptional scenes in the source image, or errors in the Geon matching process can adversely affect the range, so outliers are discarded for this property.

To successfully transfer the characteristics between datasets, it is necessary to decide which properties are part of the characteristic style and which are specific to an individual character or image. If a property has a small variance and is similar across a wide range of images from the same artist,

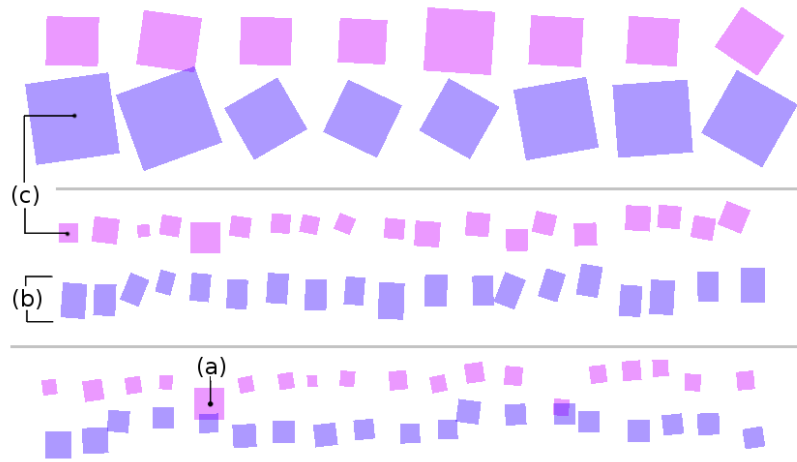


Figure 5.12: This is a visualisation of the database showing Geons from two characters in ‘Misery Loves Sherman’. The rectangles show the size and rotation for each Geon identified in the target images. Each colour denotes a character, blue representing Sherman and magenta representing Fran; and each row represents a specific class of Geon — the head, body, and legs respectively. Three categories of Geon are clearly visible. The size of (a) is unique to that Geon, and needs to be recognised as an outlier; The height of the body Geons (b) is consistently different for each character, and so is not a characteristic property for the artist; Lastly the ratio between the size of the Geon classes marked by (c) is consistent across both characters, making this ratio characteristic to that artist. All three types of Geon need to be recognised and categorised during the analysis stage.

then it is likely to be part of their specific style. If a property has a large variance then it is probably dependent on the image or situation and therefore not of use as a style metric. This is difficult to measure, because most properties have different scales and there is no easy method to determine what a ‘normal’ variance is for each of them. Therefore, comparison is performed on a property from each class of Geons.

The properties to be compared are outlined in Figure 5.13. It can be seen that the rotation of the head Geon has a very wide range, as does the width of the leg Geon. This is because the character was looking in different directions in the source images, and because the legs were drawn differently across a range of comics. The categorisation of these properties

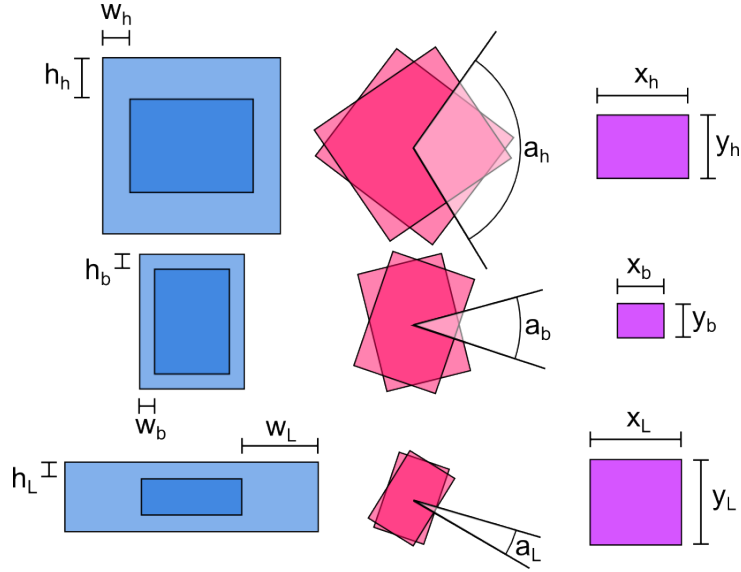


Figure 5.13: This figure shows the possible range of values for a single set of character Geons. Twelve instances of the character Lex from ‘Not Just Nicky’ were analysed. The left column shows the sizes of the Geons. The outer box shows the maximum size, and the inner box shows the minimum size. The middle column shows the range of rotations, and the right column shows the allowable range for each Geons position.

depends upon the target artist, but with a large range it is unlikely they are characteristic properties. In contrast, the rotation of the legs and the location of the body have very little deviation in their measurements and are likely to be characteristic properties. The properties are compared between artists to make that classification.

If a property shows a significant difference ($\mu > 0.5(\sigma_1 + \sigma_2)$) between two artists, then the property with the low deviation is counted as part of the style and the property with the high deviation is considered to be image specific. If there is no significant difference between artists, then properties are compared within a single artist. Each property is compared to the equivalent in the other classes of Geon. For example, the property measuring rotation of the head is compared to those measuring rotation of the body and the legs. The same test and categorisation is applied.

If there is still no significant difference found, then one of two situations

has occurred. Either neither artist has that specific type of property in their characteristic style and the property is a byproduct of the shapes and situation within the images—this is common for rotation, where the orientation of Geons depends upon what’s happening in the image. Or alternatively, both artists have this property as a strong component of their style—this is common for the ratio of sizes between adjoining Geons. There is no ideal method for determining which of these options is correct, but the following formula gives reasonable results.

$$\text{when } \begin{cases} 2\sigma_m < r_m & , \text{ the property is characteristic} \\ > r_m & , \text{ the property is image specific} \end{cases} \quad (5.4)$$

where σ_m is the standard deviation for a given property, and r_m is the corresponding range.

Equation 5.4 relies upon the fact that the set of values is not an exact standard distribution. By measuring the ratio of the standard deviation to the range, it is possible to guess the type of distribution, and hence the type of the property. If σ is small compared to r , the distribution is likely to be clustered around the mean with only a few outliers, and the property is therefore likely to be part of the artists’ style. If σ is large compared to r , the distribution is likely to have a wide spread. This means the property is unique to each image. The distribution scaling value 2 can be modified to change the weighting of characteristic to image specific properties, however this weighting is not critical because the hierarchy of transformations outlined in Section 5.8 ensures that the most significant characteristic properties are transformed first.

This categorisation method is also run on the weighted grid extracted during the prototype stage (Figure 5.14). The length of each gridline is placed in the database and categorised as an artist or image property. If neighbouring gridlines are categorised differently, the transformation process can introduce discontinuities. Therefore an additional step is performed where each line is given an influence value (initialised to 0 and 1 for image and artist properties respectively) and then weighted exponentially across the Geon from the border so that neighbours do not have large differences. When performing



Figure 5.14: In addition to scaling Geons, their contents are influenced using a weighted grid. The grid is extracted from the prototype image, including weightings based on complexity (a). If an image is scaled using the artist-significant gridlines, discontinuities are obvious (b). Weighting these values across the Geon produces a better result (c).

the transfer, this prevents large discontinuities.

Once each metric has been categorised as an object-specific metric or an artistic style-specific metric, the proportion transfer can be performed.

5.8 Proportion Transfer

The final stage to complete a style transfer using these Characteristic Proportions is to apply the properties of the stylistic features from the source image set to the corresponding features in the target image. Each geon in the source image is transformed to fit the characteristic proportions of the target style. Properties that measure the differences between geons depend on the properties of the geons themselves, and hence it is possible to end up in a situation where there are two possible alternatives to modify a property. In this case, the alternatives may not be compatible and it may not be possible to transfer all properties correctly. Therefore, an order of precedence is established whereby the metrics with the most impact will be processed first. Four passes are run across the metrics, and properties within each pass are solved beginning with the geon that has the lowest standard deviation.

1. In the first pass, properties that are marked as characteristic for both the source image and the target image are processed first, and in order starting from the property with the largest difference. This is given pri-

ority because removing the influence of the original artist and replacing it with the target artist creates the biggest impact.

2. Properties marked as characteristic for only the target image are processed second. If possible, the characteristic will be modified to take on the target image characteristics. However, this is only done if it does not interfere with the transformations performed in the first pass.
3. A special case arises where properties are marked as characteristic only for the source image. The style transfer is more effective if all original characteristic properties are removed, however there is no target characteristic value. Instead of completely transforming the geon, properties are adjusted by a random amount in the direction of the target mean.
4. Remaining properties aren't marked as characteristic for either artist, and are likely to be important to the image itself. These are left unmodified.

When transferring proportions, it is important to introduce a random element to correctly reproduce the natural distribution of property values that arises from hand-drawn images. Even if the same artist draws the same image several times, the different versions will never be exactly identical. Therefore if transformed images are used in several locations, or if several different images are transformed, they should not have exactly the same proportions. To better reproduce the target characteristics, random values are generated for the target characteristic that fit with the measured average, distribution, and range. By following the precedence rules outlined above, it should be possible to ensure that the random element is only introduced into characteristic properties. This is important, because image properties (such as the direction a character is looking, or perhaps an outstretched hand pointing at an object) are likely to be integral to the context and should not be disrupted.

The result of introducing these random values and performing the transformation is that discontinuities appear at the borders of neighbouring Geons. Image (a) in Figure 5.15 demonstrates a case where Geons have been scaled

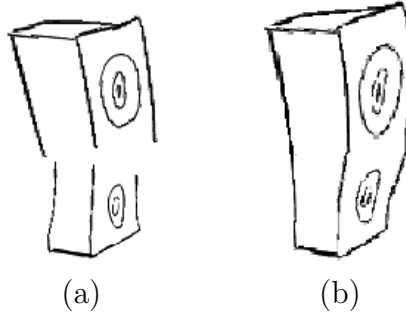


Figure 5.15: When scaling, discontinuities are introduced at the borders of neighbouring Geons. Scaling the contents based upon the ratio of the images at the seam ensures a better fit. The contents are scaled based on their complexity as shown in Figure 5.14.

disproportionately and the discontinuities are highly visible. Both the image prototype and the image extraction steps record the actual width of the image at the borders of Geons. The connectivity information from the prototype generation is used to determine adjacent borders on neighbouring Geons, which are scaled to preserve line continuity as shown in Figure 5.15 (b).

After applying border scaling, the contents of the Geon are also scaled so as to preserve the image content while still allowing for Geon matching at the seams. Figure 5.14 shows this process.

Another common transformation for geons is an unequal scaling of the dimensions. This causes graphical artifacts because it modifies line curvature and significant points such as eyes or hands become stretched unrealistically. Stretching linearly often disrupts the perspective or distorts geometric objects. Note that this is not affected nor solved by the grid scaling, which runs independently of the dimensions of the Geon. There are two common problem cases that are both handled with the same algorithm. Row (a) in Figure 5.16 shows the case where geons are reduced disproportionately in one direction, resulting in flattened shapes. Row (b) shows the extension of geons, often resulting in stretched shapes such as the flower on the chest and the waist band.



Figure 5.16: Scaling lines linearly produces suboptimal results. Row (a) shows downsizing for a single geon and row (b) shows upscaling for a geon in context. The original image is presented in column 1, and column 2 shows a linear scaling. This produces artifacts such as flattening of the feet and a rotation of the waistband. Column 3 shows the results of scaling when curves are preserved and these artifacts no longer occur.

To solve this issue, images should be scaled only at places with minimal texture or features. The complexity along the image is recalculated based upon the scaled, border-aligned, content-shifted Geon, in the direction of the unequal scaling. The sections of least complexity cannot be selected, because these are at the edges of the Geon where the initial prototype splitting occurred, so instead a cross-section is selected as close to the middle as possible. This is then stretched or removed entirely depending upon the direction of scaling. Directly removing a section of the image introduces the same discontinuity problem as with Geon borders, and so the same weighted scaling is used to ensure continuity.

5.9 Results

This algorithm was designed to extract an individual artist’s style and apply it to other drawings. The process focuses on the proportions of an image, and in this respect it successfully transfers the characteristics from one artist to another. The process is entirely automated, an outcome that is important in contributing to the original goal of reducing artist workload. The effect of

style translation using characteristic proportions can be seen clearly when the modified characters are viewed in context. In Figure 5.1, characters are shown from 'Sandra and Woo' and 'Misery Loves Sherman', as originally drawn by Puri Andini and Chris Eliopoulos respectively. Andini's characters have their characteristic properties modified to those of Eliopoulos' characters, and are then placed into a cartoon panel alongside the original. Due to the proportion transfer, the characters do not appear out of place and the result of style transfer allows Andini's characters to appear correctly in the same scene as Eliopoulos' characters.

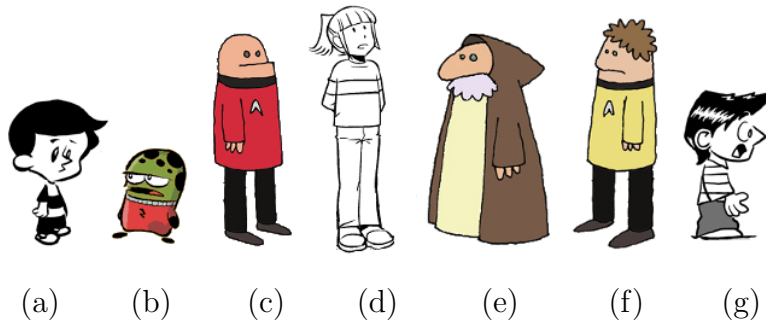


Figure 5.17: Source images used for the transfers in Figure 5.18. Images are from 'Misery Loves Sherman' (a & b), 'YodaBlog' (c, e & f), 'Sandra and Woo' (d), and 'Not Just Nicky' (g).

The process has been run on a number of line drawn datasets with success. Three datasets are used in the associated figures, and where available, at least ten instances of each object are used to build the database. Figure 5.17 shows several different characters drawn by four different artists. A target style, generated from one of the other artists, was applied to each character and the results are shown in Figure 5.18. The introduction mentioned the difficulty of ensuring stylistic consistency across a large range of work. Although this algorithm would struggle to perceive the difference between two artists who strove to draw in the same style, it produces usable results in most other cases.

Due to the random influence in the proportion transfer, the results are never identical when running the transfer algorithm multiple times. Fig-



Figure 5.18: Several examples of the style transfer method. The left column is an example of the target style and the other columns have characters converted to that style.

ure 5.19 shows five different transformations using Sandra from 'Sandra and Woo' as the input image and 'Misery Loves Sherman' as the target style.



Figure 5.19: Introducing a random element to the transform correctly reproduces the scatter of values in the target style. The algorithm is run on one source character five times to generate a range of results. It can be seen that each generated character conforms to the target style, yet has its own unique pose.

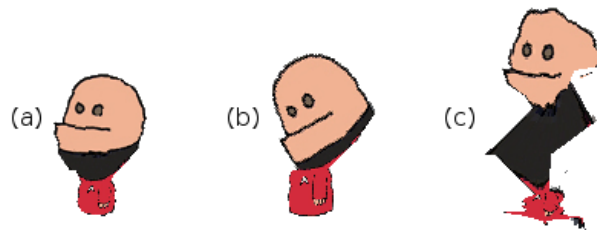


Figure 5.20: A character from 'Yodablog' is transferred to the style of 'Misery Loves Sherman'. Transfer without a target database (a) produces acceptable results. Without the source database (b), the angle of the head is incorrectly transferred, resulting in stretching. Without any database (c), the internal gridlines are directly adjusted to approximate the shape of sherman - giving incorrect results.

The methods outlined in this chapter produce reasonable results for datasets and images that fit the specific requirements outlined at each step. However often data is not available that fits these requirements, and running the process provides mixed results. One common scenario is a lack of source or target material, forcing the process to be run with only one side (or neither side) of the database populated. In the case where no database of source images

exists, the single source image has all properties marked as artist-specific so that only characteristic properties are replaced on the target image. Figure 5.20 shows images produced when one or both databases are lacking. In the case where no database of target images exists, the process can be less effective but still produce usable results. Finally, running the style translation process is possible with no databases at all, however the results can range from passable to unusable.

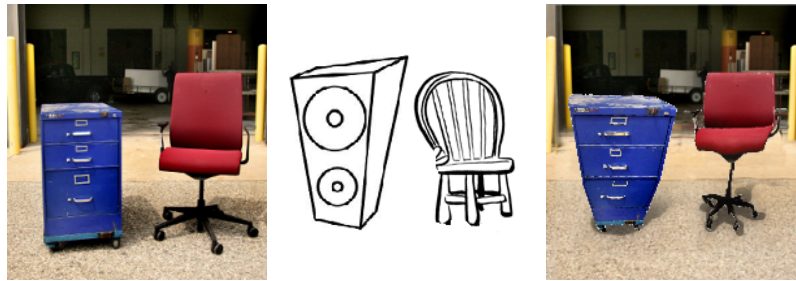


Figure 5.21: An example of transferring a cartoon style to a photographic image.

It is also possible to perform style translation on photos and images that are not line-drawn, although some of the processing stages must be skipped or performed manually. Figure 5.21 shows a situation where the results appear correct. In addition to modifying images, the geon database can be used to identify the artist for existing images. By measuring how closely the new figures conformed to the characteristic properties of both artists a confidence value could be built and the character's artist identified. Figure 5.22 shows how this geon recognition produces correct results even for very similar datasets.

Characteristic properties can be used to classify unknown images by testing against Geon databases from individual artists, and finding the artist with the closest metrics. In Figure 5.22, three additional figures were chosen from the existing artists and correctly recognised.

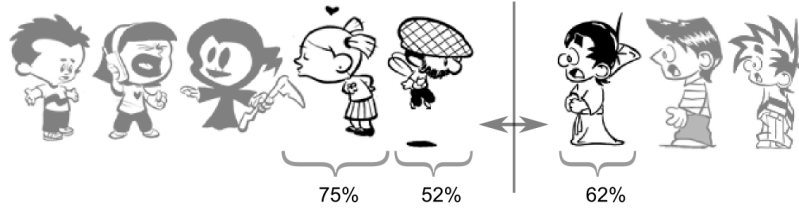


Figure 5.22: Three previously unused figures were compared with the analysed characteristic properties for each artist. The confidence of each categorisation is shown below the figure. The characters in grey on either side were used to build the database of geons against which the matching was carried out.

5.10 Conclusion

The success of the algorithm outlined in this chapter is an encouraging step into the field of artistic style. The proportion transfer runs without requiring human judgement and produces acceptable results for a range of images.

Human perception has yet to be rivalled by computers, in large part due to the difficulty of contextual analysis. By focusing on characteristic proportions, this algorithm fills the gap between the functional context-free style analysis of Hertzmann et al. [79], and the non-functioning attempts at fully contextual systems. The spatially linked geons extracted from the image gives enough information to avoid the distortion problems encountered by Freeman et al. [64]. Measuring geons allows spatial context to be taken into account without needing to classify the object itself.

While the results of style transfer using Characteristic Proportions seem reasonable to an observer, a better indication would be to perform some type of verification. Research such as the Structural Vectorization in Chapter 4 lends itself well to mathematical verification, however judging the *correctness* of artistic style is difficult. A participant based study could be performed where viewers categorise or rate sets of modified and unmodified images, or attempt to identify computer generated objects from an original set. This type of test could verify a style transfer, but still would not measure degrees of success objectively. Because any objective classification of artistic style

relies so heavily on perception and cognitive science, it would be a significant project in itself. A more rigorous verification of this chapter is therefore left to future researchers or other fields.

The algorithm produces reasonable and predictable results when using line drawn cartoons. If both cartoons use similar types of line art, transformed images match the style of the target image accurately. However, two arbitrarily chosen artists are unlikely to have the same type of line art, and may draw in colour. In these cases, further processing is needed to fully complete the style transfer. Chapter 3 begins to address this issue and can be used in conjunction with characteristic proportion transfer to provide better overall style transfer. More advanced line style integration is however left to future work.

Chapter VI

Model Reconstruction

6.1 Introduction

The aim of this thesis is to reduce the workload of content creators by developing techniques based upon or enabled by visual style metrics. In this chapter, the boundary between 2D and 3D work is explored, and methods proposed to ease the artist’s burden when moving content between these mediums. An automatic model creation method is outlined that allows easy integration of concept and user generated artwork into a game. Specifically, the research deals with low to mid resolution content that is useful for rapid prototyping or applications such as mobile games.

In particular, the work in this section focuses on removing the need for user guidance during 3D reconstruction and producing automatically a production-ready model that includes required standard properties such as UV texture coordinates and bone influence values. Texture coordinates are an integral part of displaying the model in a way that accurately reflects the original concept art, and the bone influence values allow animation of generated models. The work in this section explores the use of automatic generation for characters, including humans, robots, creatures, and other animate entities. Characters such as these are central to the majority of animated and interactive digital media, and provide a difficult and important use case while at the same time limiting and focusing the scope of the project within a manageable subset. By focusing on this character subset we can also make a number of assumptions about the content, and produce a higher quality output than a general solution can achieve.

The main contributions of this chapter are two conversion methods. One is a rescale method that deforms an artist-defined base mesh to better reflect

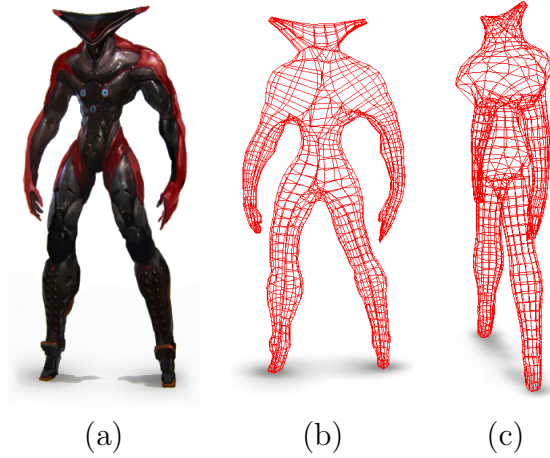


Figure 6.1: Front (b) and side (c) views show the 3D mesh automatically generated by our algorithm using a single piece of concept character art (a).

the target 2D concept artwork, and is appropriate for content retouching and style consistency evaluation between 2D and 3D. The other method is a shell-based meshing algorithm that converts 2D outlines to 3D meshes using a heuristic that balances the skeletal relevancy of the mesh against reduced visual artefacts. By extracting a skeleton structure, approximating the 3D orientation and analysing line curvature properties, appropriate centrepoints can be found around which to create cross-sectional slices and build the final triangle mesh as seen in Figure 6.1. Additionally, this section outlines a low complexity automatic skeletonisation algorithm for raster images, with an optional user-controlled complexity parameter.

The work outlined in this section aims to cover a number of important goals. The foremost is the use of single-view input; this is important because it allows a much wider range of input based upon concept artwork. It means that artists do not have to change their workflow, nor produce additional work outlining alternate views, and helps in general to ensure the technique remains widely applicable. The second major goal is to produce 3D models entirely automatically. Removing complex intervention from the generation process allows for faster prototyping, and content creators avoid learning another specialist system. Removing the need for artistic guidance also allows for easier integration of user generated content. Other goals are to ensure

the generation process runs on a mid-level system in reasonable time, and that the end result is a game-ready or render-ready model. Depending upon the technique used a number of constraints are imposed on the input format and content, these are outlined in Sections 6.3 and 6.4 respectively.

The following sections outline the existing research in this area, and introduce two new approaches to model generation. An evaluation is performed, comparing the two approaches and comparing automatic generation with artist created models. The results show that a single 2D input image can be used to generate a visually correct rigged 3D low-polygon model suitable for use in realtime applications.

6.2 Previous Work

6.2.1 Overview

Single-View Modeling has long been a difficult research problem and has received a lot of attention. Three dimensional forms that appear obvious to an experienced human observer are difficult to process without contextual knowledge, and the breadth of existing research showcases many different approaches to the problem of extracting an implied 3D structure.

Existing research can be organised into three clear groups, each with it's own approach to the problem.

The most common approach is the creation of a program, workbench, or framework that allows for artist-assisted model creation from 2D sketches. This can range from systems that interpret special types of input [7] and modelling operations [189] to systems that act on the appearance of simple lines as they are drawn [130]. These systems can use data from a single dimension [111], but more commonly allow multiple dimensions to be used, either from a single view [7] or from multiple views [161].

Another popular approach is the reconstruction of geometric shapes such as those found in technical drawings. These are predominantly single-view solutions that use inferred reference lines [59] or lines internal to the sketch [123] to calculate the depth of the sketched lines. The method of surface generation differs significantly from the freeform systems mentioned in the previous paragraph and this CAD-style approach often requires input shapes with

clearly defined planes and edges.

The third approach to 3D mesh creation from 2D input is to rescale and reshape a generic base mesh to reflect the input image. This can be done using solid silhouettes [175] or sketched lines [187]. While rescaled mesh results often reflect the input better than constructed results due to their origin from a template, this advantage degrades quickly when the skeletal topology of the source image and template mesh do not match.

Cook & Agah [43] carried out a survey of artist-assisted and sketch-based 3D modelling techniques, and categorised a wide range of papers based upon approaches to the problem. Olsen et al. [136] also performed a comprehensive survey of all significant sketch based modelling advances up until 2009. They quantify Sketch Based Interfaces and Modelling (SBIM) as the field of research focusing on interfaces that attempt to simplify or automate the process of creating a 3D model from a 2D sketch. Both papers survey and evaluate a number of artist-driven interfaces, CAD-based systems, and several automatic methods. The common goal with these systems is to allow freeform artist input and provide responsive 3D mesh generation based upon the input.

An interesting overview is provided by the SBIM taxonomy on page 16 of *Sketch Based Modeling : A Survey*, where all of the methods evaluated in the paper are listed and grouped according to their functionality and their aims. Olsen et al. find many of the same categories that are outlined in this section, including CAD-based systems (labelled ‘Engineering’) and artist-driven systems (‘freeform’). They list a category called ‘Multi-view’, however this is never expanded upon and the papers labelled as such do not match with the definition used in this section. The taxonomy also shows that the majority of papers surveyed allow for additional editing techniques after, or during, the creation of the mesh. Many of the systems also support advanced interfaces. Given the aim of our research is to create an entirely automatic system, this suggests that the primary focus should be in replacing those steps in existing systems that still require artistic input.

Our research stands apart from the majority of these techniques, due to the use of a well defined input but a lack of time sequence data, multiple views, and artist interaction. A select few papers attempt to create systems

with similar goals or restrictions, and these core papers provide the best groundwork for developing our algorithm.

6.2.2 Core Techniques

One paper that looks at mesh creation from a single view is ‘Image-assisted modeling from sketches’ by Olsen & Samavati [134]. A user defined outline, including annotations for holes and object features, is used together with the discrete distance transform to create inflated 3D meshes. The planar meshing method used to create the surface supports holes, and these are used to good effect in the examples. However, the uniform distance transform creates slightly flattened surfaces in the case of fully-expanded objects such as spheres and cylinders. This problem was also discovered and addressed by Nealen et al. in their paper *FibreMesh* [130].

The ability to define multiple shapes within the image is an important contribution to the character of the final models. Depth-mirrored details such as wheels, fins, legs and wings are all shown. ‘Image-assisted modeling’ shows a method that is close to the aims of our system, and while we aim to remove user interaction entirely, the techniques shown in Image Assisted Modeling show that it is possible to create high quality models from a single view. Texture distortion on seams is something we can improve on.

Single-View Sketch Based Modeling [7] by Andre & Saito is a recent paper that uses lofting to generate 3D meshes based upon outlines sketched from a single viewport. Although targeted towards an artist-assisted system, their algorithm uses only outlines without needing image content. The outlines drawn in a special format by an artist who is acquainted with their system. The outlines provide cross-sectional data for components within a model, and are used to generate a curved centreline along which appropriately sized slices are extruded. Andre & Saito aim for a single-view approach where the user is not expected or required to use multiple views. Due to the large influence of small amounts of noise in the sketched lines, the user must sometimes supply additional reference lines to constrain the generated shape. The overall method works on a variety of shapes, including but not limited to characters; however the models are visibly low quality and inaccurate, due

in large part to the sketch input.

The single-view input constraint makes this paper highly relevant to our research. In addition, Andre & Saito use a method of lofted sections (also seen in literature as ‘extruded sections’, ‘slices’, or ‘planes’) to construct their meshes. A similar extrusion method is used in the paper ‘Reconstructing Polyhedral Swept Volumes from a Single-View Sketch’ [159], however Andre & Saito use multiple slices which allows for greater control over the contours of the silhouette. This is important as it allows a better representation to be created from the limited input data. The paper also tries to differentiate between CAD style drawings with straight lines and freeform organic drawing. A cubic corner extraction method is used to identify sharp corners, and these are treated differently when creating the lofted sections.

Another recent paper uses a similar mesh generation method to the one we propose. Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces by Tai, Zhang & Fong [161] presents a two step method for creating 3D models based on one or more sketched profiles. The sketched profiles may be at different orientations, and for each profile the morphological skeleton is extracted using constrained Delaunay triangulation. Medial discs are generated for the skeleton, and used as the basis of a convolution surface that is weighted and merged with the other profiles. Their enhanced convolution model allows for custom cross-sections in the generated surface, and an implicit isosurface is created to render the object. Additional sculpting tools are provided to for the artist to create extra details that were not generated from the profiles. The system is primarily designed to work with multiple profiles from multiple angles and even though each profile is generated from a single view, the algorithm as a whole cannot be classified as strictly single-view.

Typical convolution surfaces often end up looking too smooth and don’t represent hard edges or creases very well. Tai et al. avoid this problem by providing boolean operations that modify the mesh after the implicit surface is generated. This works acceptably for internal details but is difficult to use to refine the main body. In addition, this type of surface generation does not produce appropriate mesh topology for animation or texture mapping.

Despite these drawbacks with the surface generation, this mesh generation

method uses only outlines as an input and yet manages to generate appropriate 3D models for a wide variety of inputs with minimal user intervention. The skeleton filling method used in the paper allows for unconstrained input types, something that our research also aims to achieve. One of the biggest benefits of skeleton based generation is that it allows custom cross-sections for the created surface, something that is not possible within the framework used by Andre & Saito. This degree of control over the final model is especially important when seen in the context of our single input image goal, as it allows us to make the most of limited input data.

These core papers provide a reasonable insight into the current state of the art and a give a clear technical overview of the techniques upon which our own research is based. The main direction of our research is to use a skeleton based mesh generation with lofted segments. This avoids a number of the significant down-sides that would arise if employing any of these core methods to create game-ready models from single view inputs. Using lofted sections allows us to generate a topology appropriate for realtime animation and texturing, and generating sections based on an internal skeleton allows for a wider range of topology and most importantly, custom cross-sections.

6.2.3 Artist Assisted Systems

In addition to the core techniques it is also worth looking at the methods used across a wider field, as this can help highlight areas in the 3D creation process that often cause issues, as well as algorithms that are commonly used to solve technical problems. One of the biggest problems facing our research is the use of hand-drawn user data, because lines may not align or meet correctly and the image is often not drawn with correct perspective distortion. This is often the case in artist-assisted systems, and almost all of those surveyed solved the problem in different ways.

The interactive modelling system proposed by Lui & Huang [111] is one of the few systems that accepts a generic single non-line based image for input. The system mixes human interaction with automatic algorithms to convert a single photograph into a 3D model for use in a virtual reality system. The rationale behind requiring user input is to avoid the use of complex

algorithms in computer vision and computational geometry.

Colour clustering is used to extract the initial outline shape from the photo. A type of skeleton is extracted by constructing a minimum spanning tree from feature points extracted using the KLT tracker. This produces a type of topology that is neither a morphological nor topological skeleton, and that segments the image in such a way that Delaunay triangulation can be used to fill the areas between the spanning tree branches. These triangles form a 2D mesh. Human interaction is required to turn the 2D mesh into a 3D mesh. The user is provided with tools to lift the vertices away from the centre plane. If vertices are moved so far as to create distorted triangles a subdivision process is applied to constrain triangles to a given maximum size. The 3D mesh is either mirrored on both sides or filled with a flat back-face. The created meshes, whilst approximating the input image, have highly distorted triangulation with obvious nexuses.

The use of photographic input means that the initial segmentation and recognition stages are closer in scope to our concept artwork than the majority of systems that use vector or line-based input. In addition, Liu & Huang address the question of base model rescaling versus generation from scratch, with their system avoiding the use of base models entirely. After evaluating model-based methods in Section 6.6, this was the same conclusion we came to and supports our use of generation over adaptation methods. However, their use of a minimum spanning tree and KLT feature points seems an overly complex approach to triangulation.

One of the stated goals of the paper is a focus on showcasing a complete system using basic algorithms such as colour clustering, as opposed to researching tangential areas such as complex image segmentation. Creating simple modules like this makes the system both extendable and improvable as new algorithms are developed. However the use of basic techniques introduces downsides, such as a failure to deal with shapes containing holes, and limited facilities for complex texture mapping.

Overall the system aims to achieve similar goals to the work in this section of the thesis, but sits on the side of manual intervention as opposed to an entirely automated system. If the missing steps are able to be automated, the research by Lui & Huang could potentially provide a simple groundwork

to start from.

A second example of photo-based input is also provided by J Ventura et al. [173] in an artist-driven system for creating 3D pop-up photos. Their interface lets artists steer the subdivision of an image into layers that represent different depths. While the examples given are of single-view characters such as a baby boy and an elephant, their aim is only to modify the camera orientation within a given window. The extracted 3D information is too sparse to allow a full 3D model to be built.

Chen et al. [37] also outline a similar system, where edge detection and user guidance are combined to allow 3D primitives to be drawn over the top of photographs. The photograph is then texture wrapped onto the model, which can be edited in various ways by the artist.

‘Sketch-based subdivision models’, by Nasri & Karim [128] presents a method for intuitively creating subdivision control cages and surfaces through the use of a sketching interface. Using sketched lines for the mesh profile, a low resolution reverse Catmull-Clark subdivision is used to create control polygons for curve segments. The combined control polygon is simplified by merging internal polygons and then extruded to create a 3D control mesh. After the subdivision surface has been created, the system allows for artist control over the shape handles for further modification of the model such as the addition of sharp features.

Nasri & Karim claim that “the industry standard for the creation and animation of 3D models for the last decade or so has been subdivision surfaces”. While this is the case for production-quality rendering, in the context of video games quad and triangle-based meshes still dominate due to the hardware requirements for realtime rendering. In the examples, it can be clearly seen that the subdivision models follow the same style - rounded edges with flat front and back surfaces. While this works for the examples shown (a hand, and a fan), it would produce strange looking characters in most instances.

Teddy is another fully developed sketch-based interface for 3D design, developed by Igarashi, Matsuoka & Tanaka [87]. It is similar in some ways to Single View Sketch Based Modeling [7], except that it allows drawing from multiple view angles. The interface is designed explicitly for concept 3D model construction, and a range of typical geometry creation tools are

provided, such as extrusion, clipping, and a body fill. The user draws freeform strokes on the screen from any camera angle and the system constructs 3D polygonal surfaces to match. The paper focuses primarily on the range of input operations needed to support a basic sketch interface, and then briefly touches upon the mesh construction algorithm.

The mesh is constructed in stages following each command from the user. The typical 'fill' command used to create geometry is performed by triangulating the initial 2D polygon drawn by the user, and then extracting the chordal axis. Triangles are reoriented so as to form fans at endpoints and junctions, and the spine trimmed to remove all edge-touching branches. Extrusion, clipping and wrapping are performed by constraining typical mesh operations to the user's input spines.

The mesh construction step recalculates the mesh topology in real-time as the user creates and destroys parts of the model. Even with complex operations this results in a well distributed topology with a minimal number of small or unbalanced polygons, suitable for animation even though the topology does not conform to standard 'good practice'. This can be seen in *LifeSketch* [188], where the Teddy algorithm was successfully combined with a skeletal animation system to create a platform for a user study.

The input to Teddy is in the form of point-defined spines from the user's gestures and 2D painted strokes. The system relies on this time and view-annotated input data to perform the construction operations, and while the isosurface generation is a more viable approach than convolution surfaces [161], the underlying model generation would be unlikely to work with single view images, or with painted concepts. This time-sequence approach to data construction is common in a number of other systems, such as SKETCH [189], which focuses on input via gestural commands and builds up scenes using collections of user-input primitives. Unfortunately no clear examples of how well this works are provided, and the focus is primarily upon the user input and not the mesh construction or display.

The paper 'Improved Skeleton Extraction and Surface Generation for Sketch-based Modeling' [105] presents research based on Teddy, performing both skeleton extraction and surface lofting. In this paper, lofting is performed edge-to-edge. This produces acceptable results, however does not

allow for the generation of asymmetric features that a edge-to-centreline system does.

FiberMesh [130] is another program and 3D modelling technique that creates implicit surfaces from a collection of 3D curves. These 3D control curves are user generated and can be created using two different methods. Control curves can be created from scratch in the 2D plane and filled to create a 3D model. Drawing upon the model will create a control curve stuck to the surface, than can then be manipulated. Additional tools within the FiberMesh framework give control over the curves with functions such as smoothing, deformation, and erasing parts of the control line. Two types of curve (open and closed) are available, and can be used to create creases in the model.

One new technique proposed in this paper is the use of nonlinear function optimisation to create properly rounded meshes. Previous work covering similar systems typically used a least-squares solver that concentrated curved surfaces near control lines and ended up with an uneven model. The nonlinear solution in the paper is significantly more complicated but produces better results, meaning that this may be a better approach in our case with a single input image, where it's not possible to use data from additional views to improve results.

The idea of using open and closed type curves is pervasive amongst control curve modelling systems. What this paper does differently is allow for a smooth transition between creases and soft curves that aren't possible in systems such as Andre & Saito's Single View Modeling [7]. The issue of surface corner transition is solved in 'ILoveSketch' by Bae et al. [14], but much like FiberMesh, this system is based primarily on user input and has no facility for automatic creation. A similar system is also proposed by Igarashi et al. [86] in the paper 'Smooth Meshes for Sketch-based Freeform Modeling'. This is designed as an artist driven system using implicit surfaces. There is no simple control over the cross-section, and it requires an artists interaction to perform edge and joint operations.

In many ways FiberMesh is similar to Teddy and SKETCH. All three are multi-view modelling programs, and in the case of FiberMesh a number of the operations wouldn't be possible in a single-view implementation. However,

unlike both Teddy and SKETCH, FiberMesh is not time dependant and the collection of control curves can be created and stored in any order with the same 3D result.

The final artist-driven modelling system worth analysing is Just DrawIt: a 3D sketching system, by Grimm & Joshi [71]. This is another multi-view sketch system that lets users sketch curves in 3D based upon 2D user input from a number of different directions. What differentiates Just DrawIt are the protocols that Grimm & Joshi develop to process uncertain user input. Uncertainty can be due to artists' sketching styles and the difficulty of drawing long lines accurately in one stroke. A system of 'overdraw' allows for gradual improvements to an underlying mesh, and user-selected tools allow the artist to switch between two modes, an 'improve' mode and a 'create' mode, each with different ways of interacting with the line-based surface model. The 3D mesh is created by processing the sketched curves to create patch-based implicit RBF Hermite surfaces. Implicit RBF Hermite surfaces have been used in sketch-based construction previously [31] but their use is complicated because of the need to generate implied properties such as surface normals at the curves and snapped join points for curves that cross near each other. In some cases, user guidance is required to generate these correctly.

Although Just DrawIt is a multi-view system it shares the same target goals as us to create a 3D mesh based on 2D sketches. Grimm & Joshi state at the beginning of their paper that "[...] in practice, this is nearly impossible because people do not create consistent drawings.". Most other systems ignore this issue or deal with the most obvious problems by smoothing the input, however the detailed stroke uncertainty analysis outlined in the Just DrawIt paper provides numerous useful techniques for correcting the worst consistency problems. Section 4.1 in 'Just DrawIt' outlines a technique to allow for stroke scratching within the framework of a single line, and section 4.2 addresses 3D joint merging from a single viewpoint.

Using these techniques would require our single-view concept art to be vectorised and the depth estimated for various lines, and unfortunately even then 'Just DrawIt' requires too much user intervention for the system to be used entirely automatically.

In general, artist-assisted systems tend to be multi-view and rely on an artist performing set operations on the geometry or creating input data in a specialised format. In general this type of approach is unfeasible for pre-drawn single-view concept artwork, but a number of smaller algorithms outlined in these papers offer innovative approaches to solving problems of artist consistency, surface generation, and automation.

6.2.4 *CAD-based Systems*

A number of papers approach the problem of model generation from an entirely different direction. CAD-based systems attempt to create solid objects from technical or architectural drawings, often by calculating depth as opposed to the common use of implied depth in artist-driven systems. The most important aspect common to most CAD-based construction systems is that little or no user interaction is required, and input imagery is often only single-view.

One of the most successful attempts at closing the gap between 3D generation of solid and organic surfaces is ‘3D Sketch-Based Model Reconstruction and Rendering’ by Mitani, J et al. [123]. Their paper proposes a system that takes a user-drawn CAD-style image in 2D and attempts to find the best-fit bounding boxes, including warping of the surface to better fit non-linear shapes. While the results show impressive accuracy, this comes at the cost of a limited input range. In section 4.1 they outline a number of constraints that must be fulfilled for the system to work, including having mirror symmetry and specified planar faces. The edge graph used in the paper can be easily extracted due to the sketch-based input method, whereby the image construction is captured during the drawing phase on a graphics tablet.

The corner and edge detection allow a base 3d model to be created with planar faces. Curve fitting and reconstruction is then used to compare the drawn line between two corners with the reconstructed edge. The curved line is projected into 3d space using the reconstructed bounds and the camera properties, and is then used to modify the curvature of the faces it bounds. This two-step process of rough cage creation followed by detail adjustments results in models that appear correct at both macro and micro scale.

Few other papers in this group attempt to model curved surfaces. Indeed, the technique outlined in Reconstructing Polyhedral Swept Volumes from a Single-View Sketch [159] requires the single-view input sketch to contain well-defined corners. The input image is also assumed to contain one profile view, and show the tail end of an extruded solid. Orthogonal and non-orthogonal corners are recognised in the image, and used to find the plane of the profile and the orthogonal direction of the extrusion. These can then be used to create the 3D model. The system relies on the input image being a relatively correct geometric sketch.

One of the primary constraints and advantages of our system is that it uses a single input view. Suh [159] outlines a method of determining orientation given only a single view, and while this only applies to CAD-style mechanical objects, any input image with potential right-angled corners could be analysed and use this method to determine the initial orientation. Three-quarter views are common among concept artwork, and being able to un-distort the initial image could result in higher quality models.

Another system that is similar in technique to those outlined above is the Smart Sketch System for 3D Reconstruction Based Modeling by Naya et al. [129]. The paper outlines a freehand CAD system that automatically straightens freehand input lines and aligns them in viewspace with a predefined projection. A second input step is then performed with the surfaces being sketched on top of the reference lines. The program automatically fills these in to create the model. Although the view angle is unconstrained, examples show that the system works from a single-view input sketch. This two-step mesh generation algorithm appears to afford better control over how the surface is created, as guidelines and the reference frame can be set up first and then adjusted to fix alignment and perspective problems before any geometry is created. However, to implement this using concept art would also require a 2-step input, perhaps in the form of a sketched wireframe followed by the painted concept image.

The majority of 3D reconstruction techniques that fall within this category have trouble reconciling the implied accuracy of technical drawings with the low accuracy of hand-drawn sketches. Fen et al. [59] use vanishing points, detected by tracing any straight lines in the image, to determine the focal

length and perspective of a drawing. This is then used to identify corners and estimate depth data to recover the 3D object. This worked well for accurate drawings, however for inaccurate drawings such as sketches the algorithm had greater trouble calculating the vanishing points and focal length.

In opposition to artist-assisted sketching interfaces, CAD-based generation systems rarely rely on time-series data, nor multi-view data input. Some of these techniques could prove applicable when generating character models, such as in the case of robotic or highly stylised drawings. However this situation is not as common as organic, arbitrary shaped characters and the principles behind polygonal mesh synthesis from sketches is fundamentally different from rounded mesh synthesis. In addition, the concept artwork used as a template does not often have a rigid reference frame and may not be strictly orthogonal, perspective, or isometric. CAD-based single view reconstruction relies upon accurate adherence to this outside frame of reference.

6.2.5 Rescaled Base Meshes

A third approach to 3D mesh creation from 2D input is to rescale and reshape a generic base mesh to reflect the input image. There are several advantages to be gained by using this type of method, such as the ability to store one set of artist-generated information that can be re-used by non-artists or even automatically. Rescaled mesh results are often better quality than artist-driven constructed results due to their origin from a template, however this quality comes at the cost of flexibility with rescaling systems often having much stricter input requirements than freeform design systems.

One in-depth and successful example of mesh rescaling is by Wang et al. in their paper Virtual Human Modeling from Photographs for Garment Industry [175]. Wang et al. focus on accurately matching a character mesh to a supplied image for use in retail applications such as virtual clothes fitting. With this target, the focus of the paper is upon creating a virtual model as closely aligned to the input as possible, within the limited input framework of adult human body types. The system uses two orthogonal photographs of a person, and rescales an existing 3D mesh to reflect the dimensions and shapes of the persons silhouette.

The process has several steps and is conceptually straight-forward, although Wang et al. favour complex algorithmic solutions that afford a slight edge in quality over conventional approaches. The model creation process begins by using the Chan-Vese algorithm to obtain the outline of the human via active contour matching. This outline is subdivided into feature points, demarcating body sections at individual points such as the neck, armpits and hands. These feature points are matched between views, and then each segment of the 3D model is morphed in the view plane to conform to the outline.

The morphing step introduces several new ideas that increase the closeness of the match. In particular, global and local deformation metrics are introduced, and used to stop overlapping parts being deformed incorrectly. If two sections of the 3D model overlap in one of the orthographic views, the existing overall (global) deformation of each section is used to decide which of them the local deformation is most consistent with. This is an elegant way to solve the single-view input overlap problem and appears to work in the most frequent occurrences. While the mesh results are considerably better quality due to their origin from a template mesh, this advantage degrades quickly when the source character does not match the skeleton topology of the template. In addition, different models are required to represent body shapes that differ in ways not represented by the outline, for example the difference between the male and female body type.

Although the techniques presented by Wang et al. [175] have a highly constrained input range, a large number of the algorithms can be expanded to deal with more complex and non-human characters. The detection of feature points in the silhouette allows for accurate matching of a 3D mesh to a 2D image, and additionally provides enough contextual information to perform different operations on different parts of a model. This is equally applicable to all types of single-view model generation methods, and turned out to be one of the key steps in our research. Wang et al. find these feature points based on predefined settings appropriate to adult human outlines, however a more generic selection method could be used to allow for a wider range of input shapes.

When viewing 3D models from any single direction, there will inevitably

be overlapping and occluded sections that need to be differentiated. The idea of using global and local deformation metrics to differentiate between layers is a novel and potentially widely applicable idea. Wang et al. make good use of a standard falloff term, based on the distance from the skin to the bone, to allow more intuitive manipulation of lines that occlude or intersect in the view plane despite depth differences. While it may be difficult to implement for autogenerated meshes, some type of template or database system could potentially be used in these situations to identify overlapping edges.

A second paper that is similar in goal to Wang et al. but differs significantly in technique is *Sketching-out Virtual Humans, From 2D Storyboarding to Immediate 3D Character Animation* by Mao, Qin & Wright [114]. This paper shows pose generation for pre-existing but unrigged skeletons based upon sketched input. It also shows morphing of an existing body mesh to fit a target profile sketch. Three individual steps are used to perform the morphing; first a rigid step transforms the proportions of whole body segments; then the mesh is scaled within each segment to match the outlines; finally, a physical fat distribution step is performed to approximate real body distribution.

The two areas of interest in the paper, mesh posing and shape modification, are similar in scope to the aims of our research. However, the paper lacks technical detail and the results are underwhelming. The pose step lacks the ability to deal with detail such as feet and hands, and no intersection testing is performed, allowing for self intersecting limbs. While the mesh morphing does create models that approximate the input shape, fine details are lost and numerous inconsistencies are visible. Furthermore, although the characters have obviously different proportions that are related in some way to the target shape, they all share the same style and in this respect would fail to achieve our goal of replicating the underlying style in the 3D model.

Focusing more clearly on modifying the shape of every part of a model, *Sketch-based Modeling of Parameterized Objects* by Yang et al. [187] matches 2D sketches with 2D template outlines and then uses deformation rules to change the sizes of subcomponents. The matching process is based on curve feature vectors, and accuracy was improved by using the location as a contextual indicator. The resulting 3D object is constructed from a base model

using measurements extracted from the 2D sketch. The examples in the paper are applied to cups, aeroplanes, and fish, and while the algorithm appears to work well within the object classes it requires an intensive setup process for each class. Given that our research is restricted to character models, a template system such as this could be a viable option. The results by Yang et al. show that their system works well even when objects vary greatly in shape and size within each type class.

In a similar vein to Image-Assisted Modeling, Structured Annotations by Gingold, Igarashi & Zorin shows a system for creating 3D models based upon annotated 2D sketches. The major difference between these two papers is that structured annotations uses 3D primitives, such as spheres and cylinders, drawn directly onto a 2D image. by matching the silhouettes and centrelines, the shapes can be used as constructive geometry to build complicated objects.

One of the advantages of this system is that the geometry depth can be better calculated by looking at the intersections and placements of the primitives in 2D. So long as the image is not orthographic, the view offset means that placement of mirrored geometry or annotations can be analysed to find the depth offset and the correct point at which to attach the primitives. The largest issue is that final models still look like a collection of simple primitives used with constructive solid geometry. The overall system is reminiscent of the creature editor used in EA Games' *Spore* [76], and the final models show the same characteristics.

When dealing with single-view input there are limited datasets to perform processing on. The silhouette of the input shape is one of the most easily accessible and easily understandable datasets, and ideally suited to mesh rescaling. Our approach builds upon this existing research by using the silhouette as the primary input source, and a number of the ideas from Virtual Human Modeling by Wang et al. proved useful in solving the major problems. Although their implementation tended toward complex solutions for minor problems, the underlying ideas and techniques often gave new ways of looking at the issues.

6.2.6 *Alternative Approaches*

In addition to the three standard approaches to mesh generation outlined above, there are numerous papers that touch upon the problem of mesh generation as a by-product to their goal, and papers that outline techniques applicable to mesh creation that may provide alternative approaches to the problem. The most relevant of these papers are outlined here.

Tsai & Lu [168] outline a technique for realtime mesh splicing to create new characters in their paper ‘A Rapid Mesh Fusion Method to Create 3D Virtual Characters in Games’. Given two or more different meshes, parts are separated from each mesh, for example a head from a bird and a body from a horse, and spliced together to create a new animal with the head of a bird and the body of a horse. Because the parts are unlikely to initially match perfectly at the join, the bulk of the paper is dedicated to creating a matching seam using a form of local deformation and mesh section generation.

Although Tsai & Lu assert that their method creates new characters, the mesh fusion requires both existing models and produces a limited result set of derivative characters. Given a large enough database, the technique could be used to effectively mix and match pieces to conform to a target model, however the size of the required piece database precludes this from being a viable option. Mesh morphing is however an important tool to make meshes conform to outside specifications or different forms, and the technique used in this paper is unique in that it’s driven by cross-sectional slices. This type of cross-sectional slice is a key component of mesh lofting used by most of the papers in Section 6.2.3. In addition, Section F (Meta-Opening Generation) in their paper shows generation of a new mesh using implied data from the cross-section and the target body piece.

An important paper from a technical standpoint is ‘SmoothSketch, 3D Free-Form Shapes from Complex Sketches’ by Karpenko & Hughes [96]. Smoothsketch finds hidden edges for organic geometry and reconstructs the mesh based upon an inflation algorithm that uses a combination of user set parameters and the width of the drawn 2D shape. The hidden edges are found by identifying T-joints in the line topology, and reconstructing the hidden contour based on the curvature properties of the visible one.

With only the outline for data, the best metric for finding depth is to use the width of the object because this is often related to depth. The problem with this is that the resulting model will always have a cylindrical cross section, something clearly visible in the generated objects of Karpenko & Hughes and not ideal for character modelling. However, the ability to find hidden edges and understand line intersection structure would help our mesh generation greatly. As in the case of Just DrawIt [71], the largest barrier to using these methods is extracting the line data to process. This is difficult to extract and in some cases nonexistent in our painted input images.

One technique we found to be directly applicable to our research was the method of measuring and extending line curvature outlined in SmoothSketch. This is used in our research as the metric for line curvature in both the skeleton and generated meshes, and produced better results than any of the other metrics trialled.

In their paper 'Automatic 2.5d cartoon modelling' [4], An et al. replace 2D models with 3D substitutes. By implementing a Non-Photorealistic Rendering (NPR) technique, they successfully addressed the developer's concerns about stylistic consistency while still allowing for a 3D content creation pipeline [5]. This paper relies upon pre-existing 3D or video content and as such a system such as the one outlined in this chapter would be invaluable in further speeding up this process. At the other end of the spectrum, Van et al. [172] look at creating game models by extracting data from multiple views based upon user segmentation. This paper uses video input, but the segmentation technique could be applicable to multiple-view sketched models.

When rescaling and adjusting 3D meshes, care must be taken to avoid obvious artefacts and unrealistic distortions. In the paper 'Building Efficient, Accurate Character Skins from Examples', Mohr & Gleicher [125] use a set of examples to correct deformations in a model. A similar method could be used to smooth or eliminate artefacts caused by inconsistent scaling, or by re-positioning of the base model due to a difference in the input and model poses. However, a more general exemplar system would be required because in the case of character generation the possible mesh configurations aren't known in advance.

When generating a character model based upon custom input, there are two ways to deal with differing input poses. One is to ensure that the input strictly adheres to a given pose, such as the default animation ‘T-pose’; while the other is to allow custom poses and then provide a method which can detect the pose and apply a reverse transformation to bring the character back to the default. In their paper *Three-dimensional Proxies for Hand-drawn Characters* [91] Jain et al. show a system that maps 3D proxies to hand-drawn images and animations. This allows 2D animations to interact with 3D effects in a scene, and vice versa. The system revolves around their method of pose matching, where hand annotated stick figures provide the driving data for setting a 3D model pose. A set of 3D datapoints from a motion capture is used to generate the depth data. In addition, the original drawing must be segmented by an artist so that the generated model has the correct limb sizes and proportions.

In addition to providing an insight into pose manipulation, this research provides a good example of an application that would benefit from our model generation research. Our automatically generated mesh could be substituted for the proxy and remove the need for user input when designating limb sizes, thereby reducing the user input for the software by a third.

Overall, a large body of research exists in the area of automatic model and sketch-based generation. This section collated and evaluated this research, showing the strengths and weaknesses of different approaches and highlighting the techniques that are best suited for additional automation. The rest of this chapter focuses on two primary approaches to completely automatic model generation. Section 6.3 shows a base model re-scaling system, while Section 6.4 outlines a method for generating lofted sections based upon an extracted skeleton.

6.3 Mesh Rescaling

Mesh rescaling is an approach to 3D mesh creation that uses the outline of an input image to reshape a generic base mesh. In their research paper for the clothing industry, Wang et al. [175] found that this produced meshes that corresponded well with the input shape. However, existing methods

often require multiple input images to function, and do not cope well with complex body shapes. This section outlines a rescaling method that uses a single input image and allows for custom base meshes so as to function on more than just humanoid characters.

There are several advantages to be gained from using a rescaling method, with the most important being the ability to store one set of artist-generated information and re-use this during the creation process which can then be run by non-artists, or even automatically. Information such as the depth and cross-sectional data for humanoid shapes is difficult to generate automatically, and mesh topology is arguably the single most important aspect [3] [182] when creating meshes for animation. An artist-generated model will have a much higher chance of deforming correctly, even after rescaling is performed. In addition, multiple complexity levels can be predefined to suit a realtime graphics engine and rescaled as a set.

6.3.1 Implementation

While the results shown by Wang et al. are high quality, the rescale method is designed for realistic human proportions and unlikely to work across a wide range of humanoid shapes. Therefore we take a multi-level rescaling approach as mentioned by Mao et al. in *Sketching-out Virtual Humans*.

A pre-defined 3D model is subdivided into significant sections and used as a base to perform rescaling operations on. A correspondence is created between each segment of the model and an automatically extracted segment of the 2D character image which was provided as input. A global rescaling and repositioning step is performed to align the base model with the input image and ensure the limbs have the correct proportions and the model fits the overall form. The vertices within each individual section are then rescaled independently of the other sections to ensure that the form of the segment closely matches the target. This is the local rescale step.

Figure 6.2 shows the base model that is used throughout the rescaling. The mesh is created by an artist and segments automatically created based upon the underlying armature by assigning vertices a group based on the highest bone weight. A complexity option is exposed to the user that re-

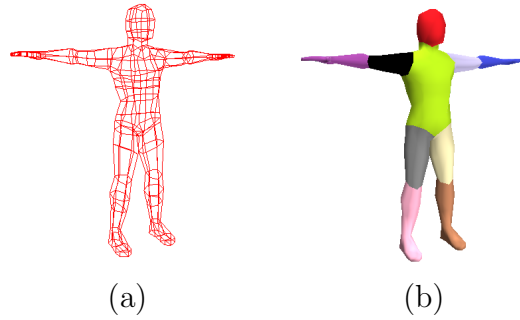


Figure 6.2: A base model (a) is used to provide the starting topology when rescaling. Sections corresponding approximately to limbs in the 2D image are identified in the base model (b).

stricts groups to bones with a skeleton depth under a given threshold. If no underlying bone structure is supplied with the base model, it is possible to use a 3D skeletonisation algorithm to subdivide the mesh. Examples of this are shown in papers by Pan et al. [142], Willcocks & Li [180], and Katz & Tal [98].

6.3.1.1 Geon Extraction

To perform rescaling on the subdivided 3D model, it is also necessary to subdivide the 2D image into matching sections. A definition of subobject composition and scale was outlined in Chapter 5 in the form of Geons, and these are used as the input to the rescaling algorithm. An alternative method to subdivision is to use Voronoi subdivision [27]. This results in cleaner segments, but lacks the geometric information such as rotation and scale that can be found using bone extension and Geons. Note that in this Chapter, the input images used are typical of video game concept artwork. This specific style presents a single front or three-quarters view of a hand painted character, usually in a T-pose or similar idle state.

Initial rescaling tests were performed manually, with the user subdividing the 2D image into Geons. The basic Geon extraction algorithm outlined in Section 5.6 was also trialed for a number of tests, and while this allowed for an entirely automatic rescaling procedure the lack of Geon detail made it

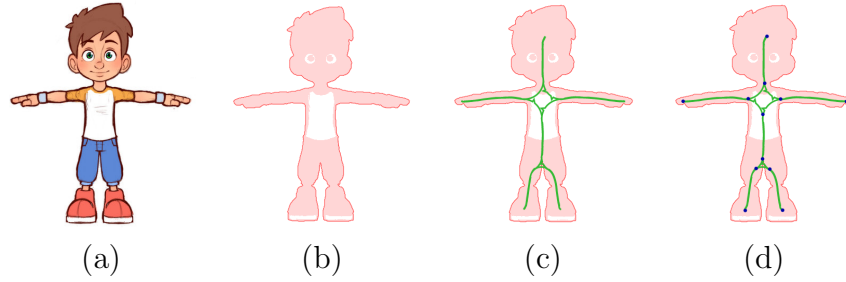


Figure 6.3: Soft skeleton joints can be used to reliably segment a 2D image based upon the outline. Using the skeletonisation method in Section 6.4.3 the input image (a) is reduced to an outline (b) and the soft skeleton (c) generated using encroachment techniques. Joints and line ends can be used to identify interesting skeleton features (d).

ineffective for use in the rescaling procedure.

The skeletonisation algorithm outlined in Section 6.4.3 was developed to enable model generation using lofted segments, but it can also be used as a subdivision method for 2D images based upon the outline. This works especially well because we are only dealing with character images constrained to a neutral pose (either the T-pose or inverted-V pose, both of which are standard poses for character model sheets within the visual effects industry. The subdivision process is started by generating the soft skeleton and interest points as shown in Figure 6.3.

After skeletonisation, the interest points are used to divide the body into major sections according to a template. Simple extrapolations are used to create and extract the Geons. The system uses a pre-programmed human skeleton template to identify the function of each bone within the extracted skeleton topology. The template is based on the assumed pose, where for example the uppermost vertical bone represents the head, whilst the outermost horizontal bones are the arms.

Figure 6.4 shows the main steps in the Geon extraction process. The skeleton interest points shown in Figure 6.3 are used to construct the main skeleton bones in two steps; all points are connected with all adjacent neighbours; then all bones that do not fit the pre-programmed template are removed. This results in Figure 6.4(a). The blue bones will be used to create

rectangles than encompass the entire limb, and therefore need to extend the entire length of each segment. All bone extremities are therefore extended to meet the body outline.

Because the bones were extracted based in part upon line curvature, they do not capture joint information for knees or elbows in the case where arms and legs are fully extended. Because the required input is a T- or V-pose, this problem is almost certain to occur. Rescaling the 3D model requires knowledge of these joints, and so their position is estimated based on the pre-programmed skeleton template. If bones identified as arms or legs are only one segment long, or are multiply connected at one end (indicating a direct joint to the backbone) then they are subdivided as shown in Figure 6.4(c).

The Geons are constructed using the line length and the average outline width along the line, calculated using Equation 6.1. This produces the result shown in Figure 6.4(e), where each section of the image is overlaid by a rectangle representing the rotation, width and length. However, the major bones may not be topologically connected due to the removal of bones that did not fit the template, and therefore some areas are not covered by any bone. Using a predefined order of importance (body \rightarrow head \rightarrow limbs), Geons are scaled along their length until the associated bone tips touch another geon or reach the outline of the character. The width is then recalculated. Figure 6.4(g) and (h) show how this final scaling step ensures complete coverage of the character.

$$width = \frac{1}{|s_a - s_b|} \int_{s_a}^{s_b} \|o_1(t) - o_2(t)\| dt \quad (6.1)$$

where s_a and s_b are the endpoints defining a skeleton bone and $o_1(t)$ and $o_2(t)$ are the outline curves on either side of the bone.

6.3.1.2 Global Rescale and Reposition

The global adjustment step tries to align the major parts of the body with the given target image. The Geons, labelled according to the pre-defined template, are linked with their corresponding segment in the 3D model. Equation 6.2 is then used to reposition and rescale the 3D segments so as to match

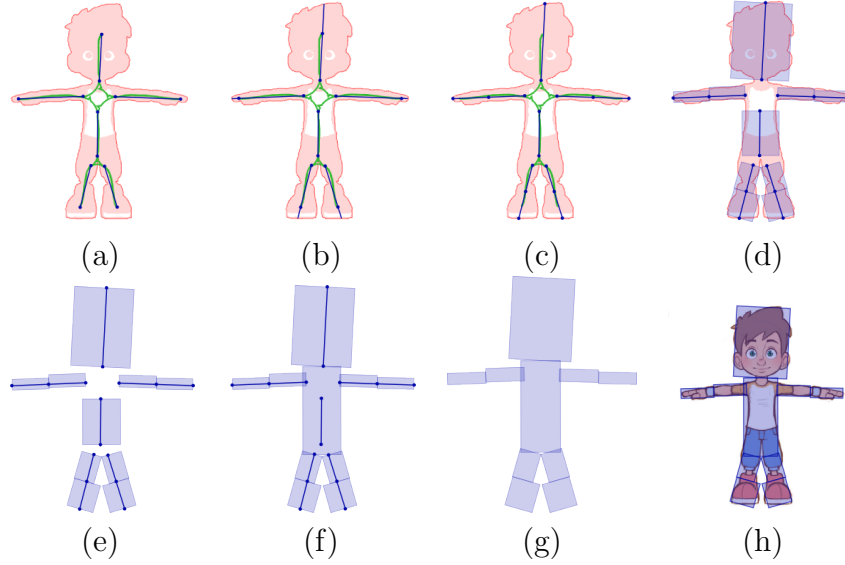


Figure 6.4: The soft skeleton calculated in Figure 6.3 is divided into bones (a) that are extended (b) and subdivided (c) so that a set of Geons (d) can be easily created. Minor scale adjustments (e & f) result in a set of Geons (g) that represent the sections of the input image (h).

the form of the corresponding 2D Geons.

Because the Geons are 2D, the depth must be inferred, and this is done simply by linking the depth with the width, thereby ensuring that the cross-section of each segment is scaled without distortion. The length is scaled independently.

The transform matrix TG in Equation 6.2 is calculated by finding the difference between the target properties and the base properties in geon coordinates. This is used to transform the vertices in the base model to create a new model where the relationship between the components better reflects the source image. If the scaling between two adjacent sections is too severe then visible discontinuities can appear and in some cases the scaled segments can overlap. While this looks bad at the global stage, local rescaling fixes the majority of these alignment issues.

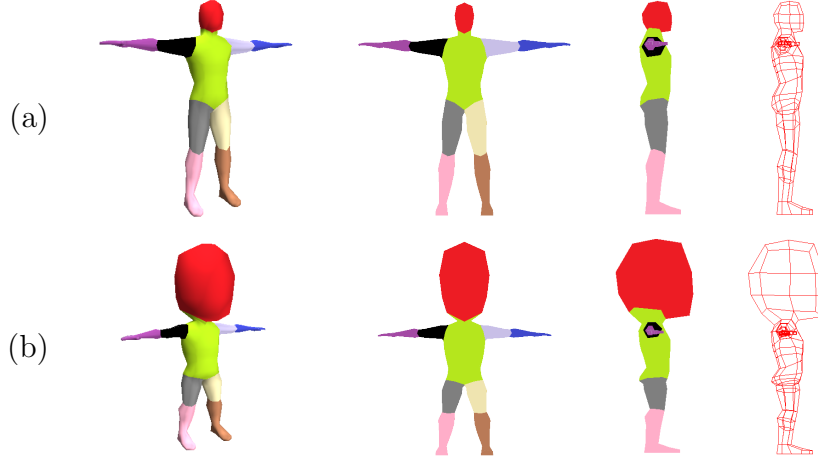


Figure 6.5: Example of global scaling on a character. Each section in the base model (a) is individually stretched (b) so as to line up with the target image. This ensures that the segments are in the correct place and are the correct size for local scaling. The target image is shown in Figure 6.3(a).

$$\begin{aligned}
 TG &= \begin{bmatrix} \frac{G_w}{T_w} & 0 & 0 & 0 \\ 0 & \frac{G_h}{T_h} & 0 & 0 \\ 0 & 0 & \frac{G_w}{T_w} & 0 \\ T_x - G_x & T_y - G_y & 0 & 1 \end{bmatrix} \\
 v'(x) &= v(x) * B^{-1} * TG * B;
 \end{aligned} \tag{6.2}$$

where v is a set of 3D vertices attached to bone matrix B and the corresponding target geon G and template geon T . w and h represent the geon width and height.

6.3.1.3 Local Rescaling

As seen in Figure 6.5, the global rescaling step performs the bulk of the work, and the resulting model represents the general proportions of the input image. However the model still does not reflect the input target image, in part because the shape of each segment does not match the shape of the outline. This is especially evident in the arms in Figure 6.5(b) which are still

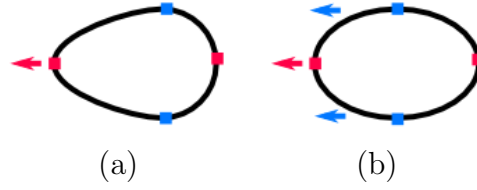


Figure 6.6: A contrived example of mass skew due to border vertex repositioning. Image (a) shows the model cross-section if the border vertices (red) are moved or scaled without adjusting the other parts of the cross-section (blue). Image (b) shows how correcting for skew can result in a more even cross-section.

thin compared to the reference image.

A local rescaling step is therefore proposed to modify the internals of each segment. This is similar to the process outlined by Wang et al. but modified so as to use edgeloops and to run with only one reference image. In their paper, Wang et al. adjust border vertices individually in a 2-Dimensional plane so as to better fit the target outline. A problem occurs when the outlines have a parallel offset from the outline, because one side of the model will be pulled out and the other pushed in, but the section mass will remain centred. Figure 6.6 shows how this results in a skewed cross section and unrealistic weight distribution. In practice this occurs rarely when using real humans as input and was not a problem for the examples given in *Virtual Human Modeling*, however one of the aims of this chapter is to deal with more distorted and exaggerated body types as input and therefore this problem occurs more frequently.

Our method solves this issue by using edgeloops to scale in depth as well as the 2D front plane, thereby adjusting the mass distribution along with the edges. Border vertices are scaled such that the profile of the base model fits the profile of the input image. Modifying a border vertex causes the associated edge loop to be scaled according to Equation 6.3, which ensures that the cross section is scaled equally. As with global scaling, this assumes that the cross-section of the loop selected from the base mesh is not significantly different in shape to the implied cross-section of the final model. This assumption holds in the majority of cases, and the result is a model that more



Figure 6.7: Edge loop extraction demonstrated on the right arm (a) of the base model. The resulting edge loops (b) provide an easy method to modify the local size without distorting the cross-section.

closely matches the shape of the input image.

$$E'(i) = E(i) * (v' - v) \quad (6.3)$$

where $i = 0 \rightarrow n$, n is the number of vertices in the edgeloop E , and v is the border vertex with new position v' .

Orthogonal edge loops are selected based upon a standard mesh loop selection algorithm that recursively selects vertices in a given direction until a loop is formed or the edge of the mesh is reached. The implementation used in this section was developed by the Blender Foundation [26]. Whilst there is a possibility for edge-loops to be user-defined and an override capability is provided to allow for this, dynamically generating the rings allows for an easier interface with a single artist-driven base mesh selection. In addition, automatic edge loop extraction reduces human interaction and allows for a more streamlined re-scaling process. Figure 6.7 shows the edge loop selection process performed on a part of the 3D base mesh.

6.3.2 Rescaling Results

The results of local and global re-scaling are generally representative of the original input image, and the algorithm works with a high success rate on images that fit the criteria outlined in Section 6.3.1.1. After a once-off setup of a base mesh and a 2D skeleton topology, the process is entirely automated. This outcome is important in respect to the original goal of reducing creation

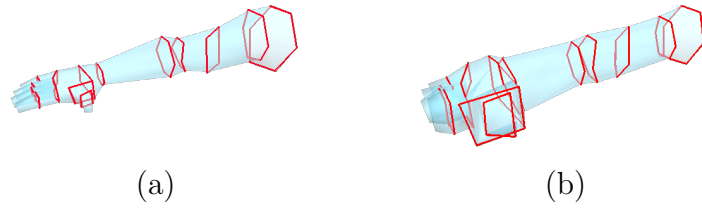


Figure 6.8: Example of local scaling performed on an arm (a) by adjusting the size of the edge loops (b).

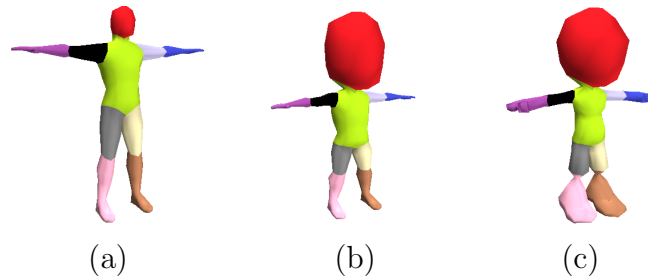


Figure 6.9: The three stages of mesh re-scaling, illustrating the base mesh (a), global rescaling of the segments (b) and the local adjustment of vertices (c).

time and artist workload.

Figure 6.9 shows how the two re-scaling steps each contribute a different type of modification to the base mesh, and how the final model reflects the input image. If UV co-ordinates are calculated using a front-projection, the original input can be applied as texture to the rescaled model. Figure 6.10 shows an example of the best-case scenario where the 3D model is a fair representation of the artists concept image.

Unfortunately, rescaling models has a number of limits that show even when rescaling relatively simple target images. In Figure 6.11 the concept image has sharp changes in width where the arms and legs turn into hands and feet. Whilst the rescaling copes well with the image in general and the resulting model represents the input image fairly, the sharp edges of the original image are lost. This is due to the even distribution and small number of edgeloops in the base model. Potential solutions to this include redistribut-

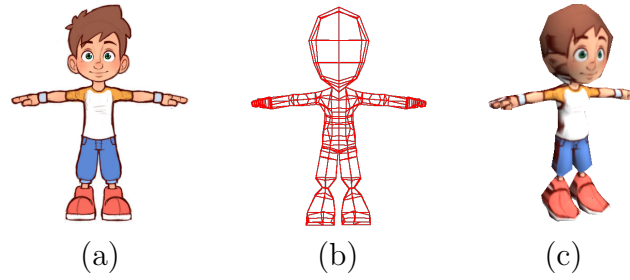


Figure 6.10: The rescaled mesh (b) closely reflects the input image (a). A front-projection can then be used to apply the original image as a texture (c).

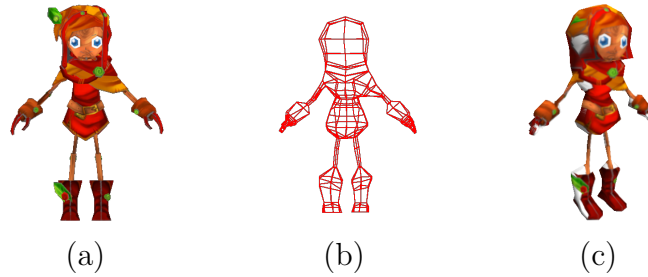


Figure 6.11: Issues arise when rescaling the base model to match sharp changes of edge or details that require more than the number of vertices available in the base mesh. Note the loss of sharp transitions between the concept image (a) and the rescaled wireframe (b). This also causes texture re-mapping issues with the final model (c).

ing the edgeloops along the bone to areas with sharp changes in width; or inserting new edgeloops into the mesh using a subdivision method. In practice, however, these solutions were found not to work as well as expected due to vertex overlap and the creation of cavities when closely-packed rings with different orientations were scaled by the same amount.

In general, the results show that rescaling is a viable method for creating 3D models based upon a single pice of concept art.

6.3.3 Rescaling Discussion

Despite some minor problems, the results outlined in Section 6.3.2 show that rescaling produces 3D models that closely represent the 2D input. Every technique has advantages and disadvantages, and in the case of single-view model creation this rescaling method can be compared to the major alternative - mesh generation.

The major advantage of mesh rescaling over generation is the ability to retain important internal details that can be difficult or impossible to infer from a single image. These can include non-occluded body features such as the nose, breasts, or equipment that does not impact the outline and therefore requires extra processing to distinguish. Equally, occluded details such as fingers in the side-view of a hand or front-view of toes in a foot can be preserved and correctly represented in the created model by including them in the base mesh. Mesh generation requires significantly more contextual knowledge to be able to infer and generate this type of detail. Another important advantage is the ability to provide artist-guided cross-sections, which are important for musculature and body sections such as the head which are not perfectly cylindrical. This type of irregular cross-section can however also be inferred when generating meshes, based upon knowledge of the underlying skeleton and the type of bodypart being generated.

The biggest limitation with this mesh rescaling method is the highly constrained input format. Whilst 2D character model-sheets are not uncommon, character concepts for prototyping are typically less formal and often not in a T-pose. A potential solution is to use the extracted skeleton to generate a reverse pose transform and modify the image so as to be in the correct pose. However the distortion is often too extreme for the segmentation step to function. The input constraints also include the reliance upon a base model. This means that only character limb configurations that have been thought of and modelled can be used as input. There is currently also no facility for automatically matching an input to a base model, and therefore this introduces an extra manual step in the process. Other problems arise from the topology imposed by the base model. It is possible that parts of the model do not have enough edge loops to represent the mesh, and the

resulting model lacks detail. On the other hand, if the model has too many adjacent edgeloops then rescaling can cause cavities or mesh overlap.

Mesh rescaling is ideally suited for small tweaks and fixing inconsistencies between concept artwork and 3D models, because in general the results are accurate but the method is constrained to situations where the input data falls within a very limited range. The mesh generation method outlined in the following section copes with a wider range of input and as a practical tool has proven to have a more streamlined workflow with a better practical application.

6.4 Model Generation from Concept Artwork

A second common approach to sketch based modelling is constructing a mesh from scratch based upon various 2-Dimensional cues. A system based upon mesh generation has several advantages over model re-scaling, including encompassing a wider range of input topologies and allowing for greater flexibility over the final shape of the mesh. The rest of this chapter focuses on a novel shell-based meshing algorithm that allows for the ability to change the cross-sectional profile of the model based upon the type of character and even the type of limb in the model. The implementation also demonstrates an end-to-end system with no user interaction, optimised to produce the best results across a range of input. Optional user interaction can be used to create more advanced effects/refinement for higher quality models.

6.4.1 Algorithm Overview

The core algorithm analyses a single piece of concept artwork and approximates a skeleton based on the outline. This is then used to construct a rigged triangle mesh appropriate for use in realtime graphics applications. Manually creating 3D models from 2D concept art often requires time and technical skill, and much of the work in the following sections focuses upon automatically performing the steps in this pipeline to allow faster prototyping and better integration of user generated content.

To generate a 3D mesh the algorithm requires one input image with the three assumptions that: the background is not heavily textured; the character

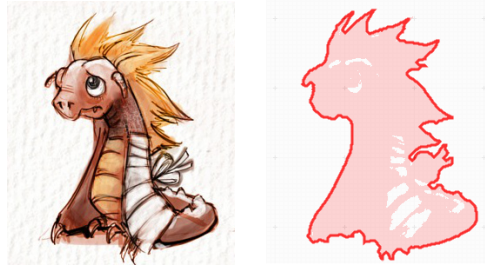


Figure 6.12: Character outline generation using image palletisation and vectorization. The generated polygonal outline is shown in red.

is oriented in general toward the front; and there is minimal self-occlusion or touching between the character's limbs.

The process begins by extracting the concept image from any subtle background shading by reducing the colour palette the image and selecting the largest connected area as the background. Everything else is considered to be the character, and a polygonal outline is generated using the Potrace vectorization algorithm [150]. This outline can be seen in Figure 6.12 and is used for both skeleton extraction and mesh creation.

A skeletonisation process based upon the work of Willcocks & Li [180] is applied to the image, balancing two core iterative operators to extract a skeleton with the appropriate complexity and positioning. The process is modified by adding an extra term based upon the image content that aligns the generated skeleton better with respect to the shapes inside the image.

The 3D mesh is created by generating arcs (also known as shells) from the outline in toward the centre while the ends diverge in the depth plane. The proposed method changes the cross-sectional profile based upon the characteristics of the outline. The positioning and size of these shells is critical to creating an appropriate mesh, and the cross-sectional shape that is generated based on line curvature has a distinct influence on the feel of the final mesh. Bone rigging and skinning for animation is performed on the mesh, and the original concept image is modified and used as a texture.

This process creates a 3D triangle mesh representation of the original 2D concept image. The full algorithm is described in the following 3 sections, and the results are presented in section 6.5.

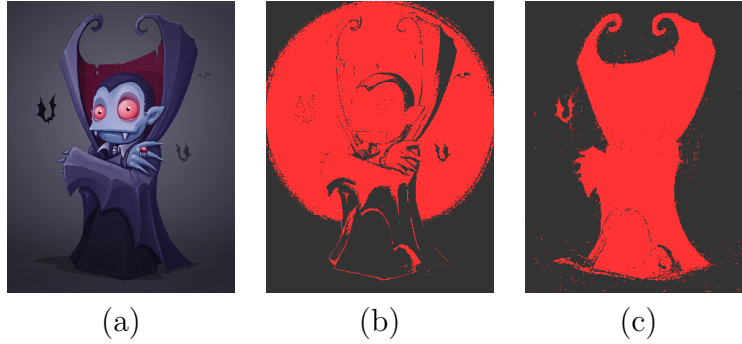


Figure 6.13: Figure showing the difference between region extraction in different colour spaces. The original image (a) is very difficult to segment due to the graded background and the similar colours throughout the image. Using a threshold to separate the background is impossible using RGB channels as even the best case (b) barely matches the outline. Using the CIE LAB colourspace (c) gives a clear distinction.

6.4.2 Concept Image Preparation

The first step when generating a model is to extract the character outline and estimate the underlying skeleton. As outlined in Section 6.4, the input for automatic model construction is assumed to be a piece of concept artwork on a plain background. The character or object can be in unknown pose or rotation within the constraints given in Section 6.4.1, although images with large amounts of hidden or obscured detail will result in less accurate models. Because the background is not heavily textured, a simple thresholding process can be used to extract the character outline.

Because the input images are hand-drawn art and therefore interpretation is based primarily on colour and intensity perception, using a perception based colour space could produce subdivision results close to what an artist would achieve. One option is proposed by Chong et al. [39], however in this section all image processing is performed in the CIE LAB colourspace as it is more commonly used and is available in most image processing libraries. Using CIE LAB still gives better results over the same operations in RGB, HSV or HSL. Figure 6.13 shows the difference between the best cases when performing a thresholding operation in RGB and CIE colourspaces.

This simple approach works for backgrounds with solid colours, gradients

and some shadows. It fails in cases with textured or complicated backgrounds. This could potentially be supported by implementing a more advanced background extraction algorithm. This is outside the scope of this research, but a comprehensive review of background segmentation techniques is covered in *Image Segmentation Evaluation* [190] as well as the older *A Review on Image Segmentation Techniques* [140].

Background thresholding extracts an accurate outline for the character, however in a number of cases the character must also be segmented internally. A robust and simple approach is to use k-means clustering or the medium cut (boxcut) algorithm. These identify trends in the point-cloud data generated from the image colours. K-means clustering can be difficult to initialise. While other clustering techniques have been adapted specifically for colour-image quantization, such as the centre-cut algorithm [93], PCA, Agglomerative Hierarchical Clustering and the density based DBSCAN [162], many of these do not translate as well into CIE colour space. The initialisation issue was addressed in a paper adapting k-means clustering for colour quantization [36], and this is the implementation used in this chapter.

The extraction threshold for the background colour is selected by quantizing the image into two colours, and then selecting the colour with the least amount of variation within the quantized x-means box.

6.4.3 Skeletonisation

Generating a 3D mesh from scratch relies on access to a skeleton that represents the underlying body structure as well as possible. Many skeletonisation algorithms exist, and producing the best result for arbitrarily shaped character images requires selecting the best approach.

Medial transforms are perhaps the most well established method for 2D skeletonisation, having been proposed in 1967 [27] and tweaked in various different ways up until the present [16] [126] to solve problems such as the influence of surface noise on branching. Pixel based methods are also common, with an overview being provided by Lam et al. [104]. Skeletons can also be extracted by joining bi-tangent circles or maximal discs within a shape [10], or through the use of Delaunay triangulation to create a chordial

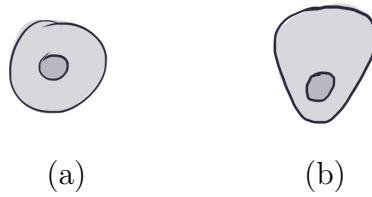


Figure 6.14: Two sketched cross-sections of a leg or arm bone. A bone placement in the geometric center (a) of a limb does not reflect reality. By generating a skeleton that is offset according to muscle mass, a more realistic placement (b) can be achieved.

axis [147]. It is also possible to perform single-step skeletonisation using a vector outline. Mayya & Rajan [119] propose a technique called Voroni skeletons that extract a morphological skeleton using Voroni cells and a pruning algorithm. A large number of vectorization algorithms are also outlined further in Section 4.2.

The difference between the Skeletonisation outlined in this section and the Structural Vectorization outlined in Chapter 4 is the type of content they try to represent. Both algorithms attempt to represent the image data in a format that is easy to parse, however Structural Vectorization extracts the centreline from collections of strokes in low resolution images whereas the Skeletonisation needed for this section runs using a single image outline and does not produce a morphological topology.

This distinction is important, because a morphological topology always gives a skeleton in the geometric centre of an outline. When using a skeleton for bones or skin generation, this is unlikely to be representative. Instead, the bones should be generated using outline and texture cues so as to best estimate overlying muscles, clothes, or other features. Without context, only a rough estimate can be achieved, but analysis using the methods outlined in Section 6.6 show that this is still better than using a morphological skeleton. Figure 6.14 shows a visualisation of the placement differences.

As well as 2D vectorization and skeletonisation methods, another class of algorithms also exists. 3D curved skeletons can be created from 3D models, often through a process of mesh contraction or decimation. The advantage

of using a 3D skeleton created via a mesh contraction algorithm is that it is easier to effectively reverse the process and create a model by building outwards from the skeleton. Although the skeleton may not be centred as in the case of traditional morphological skeletons, extra information can be gained from the density of contracted points and the relative mis-alignment of the skeleton within the mesh. Adapting a 3D contraction algorithm to run on a 2D border retains many of these advantages.

The evaluation paper *Curve-Skeleton Properties, Applications and Algorithms* [47] compares the results of basic implementations from each category of skeletonisation algorithm. The evaluation properties include metrics such as whether the skeleton is ‘centred’, ‘robust’ or ‘hierarchical’, and the paper looks at how important each of these is in terms of algorithm design and how often these are core ideals in existing papers. Four major classes of algorithm are then chosen and implemented in their basic form. This allows the authors to compare the results across a dataset and they attempt to determine the algorithm class that produces the best results in all categories.

Cornea et al. form the conclusion that “From these results, it is clear that the potential field method yields the cleanest and smoothest curve-skeleton at the initial stage”. However when comparing the results from best-of-class algorithms topological thinning appears to produce the most useable skeleton. The authors allow for implementation error, stating “We cannot claim that each of our implementations is fully representative for an entire class of algorithms, as many improvements to the resulting curve-skeletons can surely be made”. Whilst the best skeletonisation algorithm is arguable, the research by Cornea et al. at least allows two major approaches to be ruled out as viable options. As shown in their paper, both Distance Field and Geometric methods produced lacklustre results across the entire range of input models, with false branching, high complexity, and outright incorrect skeleton placement.

Skeleton Extraction by Mesh Contraction [12] is an example of a recent curve-skeleton extraction method, producing skeletons that would be ideal for model generation. By iterating two steps in repetition the mesh is contracted and connected until it forms a single line bent skeleton. The major contribution of the paper appears to be adjustments to recentre the skeleton

after the initial extraction.

The two contraction steps are a smoothing operation and a connection algorithm. The smooth operation is performed by applying a Laplacian that moves the points along their approximate normals. While there are minor problems such as the skeleton’s susceptibility to mesh density, the major issue is bone misalignment during contraction. Because the contraction step distance is based upon mesh thickness at the vertex, it is possible that the skeleton is offcentre, and may even be created outside of the mesh. An embedding refinement step is therefore introduced that merges vertices in the curved skeleton based upon their centeredness.

Feature-Varying Skeletonization [180] proposes a mesh skeletonisation algorithm that runs automatically but allows the user a single complexity control that modifies the detail level of the skeleton. By alternating between two refinement stages and modifying the weight of each, they successfully decompose triangle meshes into logical skeletons. The mesh contraction method is similar to that proposed by Au et al. [12] but due to the weighting of the steps doesn’t need to do embedding refinement.

The two refinement algorithms are smoothing and merging. The smoothing step contracts the bounding hull toward its spatial centre removing local noise at the cost of increased density and reduced fidelity. The merging step reduces this spatial density and consolidates important geometric features, however is susceptible to local noise. By using the correct amount of each operator in turn, a locally noisy surface topology can be contracted to its visual centrelines and a node topology created.

By treating a silhouette as a linked mesh structure, a 3D curved skeleton creation algorithm can be run in 2D and produces a topology structure ideal for the generation of a 3D model.

6.4.3.1 *Soft Skeleton*

The first requirement of a character skeletonisation algorithm is that it produces a visually correct topology that imitates the structure implied by the outline and not just the geometric centre. To achieve this we adapt the 3D smoothing and merging steps from Willcocks & Li’s paper [180] to run on

2D polygonal shapes, and then add a third term that allows image contents to influence the skeleton generation.

The process starts by creating a polygonal bounding hull of points $P = \{p_0 \dots p_i \dots p_n\}$ based on the image silhouette. The smoothing step contracts this bounding hull toward its spatial centre removing local noise at the cost of increased density and reduced fidelity. It operates iteratively by placing a point at the average of its untransformed neighbours:

$$p'_i = \frac{p_{i-1} + p_{i+1}}{2} \quad (6.4)$$

where p is a set of consecutive points with n items and i is a value from 0 to n .

The increasing point density after a smoothing step causes convergence issues, as the smoothing step is dependant on point density while the encroachment step is not. This causes the two operators to become unbalanced, unless the density is conserved by introducing a simplification term into the smoothing step:

$$\begin{cases} p'_i = \frac{p_{i-1} + p_{i+1}}{2} & \text{if } \|p_{i-1} - p_{i+1}\| < \omega \\ p_i & \text{otherwise} \end{cases} \quad (6.5)$$

where ω is a threshold distance calculated as:

$$\omega = \frac{1}{n} \sum_{i=0}^n \|p_i - p_{((i+1) \bmod n)}\| \quad (6.6)$$

where p is a set of consecutive points containing n items.

While Willcocks & Li use an iterative merging operator, we perform a single merge pass at the end to create a rigid skeleton. This is necessary because our mesh creation process relies on extracting the two-sided curving centreline data before merging but after the full contraction is complete. To give the same effect, the iterative merge operator is replaced by an encroach operator that moves all points 'inwards' along their local normal:

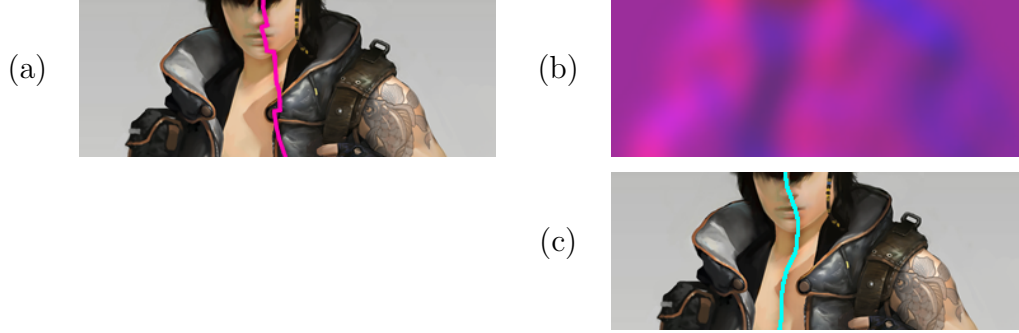


Figure 6.15: Without weighting (a), the extra width of the shoulder and clothes causes discontinuities in the contracted polygon. Weighting the encroach step by the gradient of the low frequency component of the image (b) causes the skeleton structure to be influenced not just by the outline but also by the contents of the image (c). Image (b) shows $\Delta G_{x,y}$ encoded in the red (vertical) and blue (horizontal) channels.

$$p'' = p' + (p'_{i-1} - p'_{i+1})^\perp G(p') \quad (6.7)$$

where p is a point in P , the outline polygon, and G is a frequency term as explained below. If the p'' lies outside of P , the result for that point is discarded and the original position used.

This iterative encroach step reduces spatial density and consolidates important geometric features, however it is susceptible to local noise and as a method of skeletonisation is not ideal because there is no influence from the internal lines and information in the image. Therefore an additional term G is calculated using Equation 6.8 and added to the encroach operator, using local image complexity in addition to the image outline to create the skeleton more correctly. Figure 6.15 shows the difference in skeleton placement when local complexity is taken into consideration.

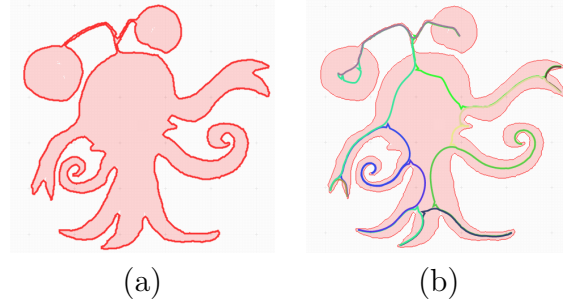


Figure 6.16: Alternating between the smooth and encroach steps condenses the outline (a) to a two-sided soft skeleton structure. Internal lines (b) show the position of P halfway through the contraction process. The coloured lines show the division of bone segments for surface curvature measurement in Section 6.4.4.

$$\begin{aligned}
 k(x, y) &= e^{-\left(\frac{x^2+y^2}{2r^2}\right)} \\
 G(x, y) &= 1 - \frac{1}{(2r)^2} \sum_{\substack{-r < x' < r \\ -r < y' < r}} k(x', y') \frac{I(x + x', y + y') - I(x, y)}{I_{max} - I_{min}}
 \end{aligned} \tag{6.8}$$

where I is the image intensity and r is the radius of the Gaussian kernel k . This is chosen to be $1/100^{th}$ of the image diagonal.

6.4.3.2 Rigid Skeleton

The soft skeleton is used for mesh generation, however realtime animation and character rigging requires the use of solid bones. These are created by finding splits, merges, end points and inflection points in the soft skeleton and connecting them according to the topology. The polygon formed by the soft skeleton is traversed in order and the backing edges, created when p'' lies outside of P during the encroach phase, are removed to ensure a bone is not created on each side of the two-sided soft skeleton. Figure 6.17 shows the results of both skeletonisation stages.

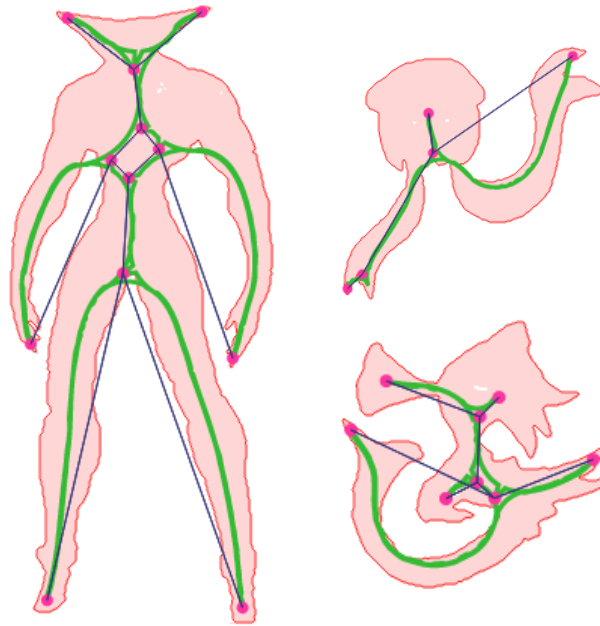


Figure 6.17: Examples of the finished skeletonisation process, showing data derived from the red outline. The bent skeleton is drawn in green, while the nodes are linked in blue

One of the largest underlying problems with this technique is that creating an accurate skeleton from a single view requires contextual awareness or image understanding based upon experience. This results in certain cases that are unlikely to be processed properly by the algorithm.

Even when including adjustments based on image content, such as Equation 6.8, large breaks or features in the image content may not be reflected correctly in the skeleton because the silhouette has the largest influence on the final result. An example is an arm that sits flush with the side of a body and is not represented by the outline, and therefore does not contract in a way that would create the new branches required to represent the arm correctly.

A similar issue arises when generating joints in the skeleton. Unbent limbs in the source image make it difficult to identify joints such as knees or elbows, and these sections are often recognised as only one bone. This can be solved by looking at basic contextual information. Long, non-branching

limbs near the edges of a skeleton are likely to represent arms or legs and are split so as to contain a single middle joint. A second check can be performed once animations are transferred to the object, where joints that do not bend significantly across any of the animations can be considered spurious and removed.

Images conforming to our initial requirements seldom contained these problems and usually produced accurate skeleton data without any user interaction. With this information extracted correctly, all of the required data exists to generate the mesh.

6.4.3.3 *Principal Axis*

A number of operations that are used to find properties of the mesh require a median or best-fit axis for the skeleton. This orients the character in space and gives geometric meanings to metrics such as *width* or *spread*. Because characters may be either vertical (for example with humanoids) or horizontal (many four legged animals), an orientation independent technique such as Principal Component Analysis (PCA) is required. The assumption is made that the longest axis of the input character is the primary bone in the skeleton (i.e. "the spine"). This assumption held true for all images within the input dataset.

In this implementation, PCA is run on a larger dataset comprising all the outline points after the final encroach step and after selective thinning is performed. To improve the accuracy, all points that lie on an external limb (classified as any limb where any endpoint has only one neighbour) are removed from the dataset. The remaining data best represents the spine of the character and means the best-fit axis is still correct even in cases where the arms or legs are wider than the model is tall, or where external appendages such as wings or antennae would otherwise influence the fit.

6.4.4 *Mesh Construction*

In this section a 3D synthesis algorithm is outlined based partially on swept volumes (also known as swept hulls or swept shells) as outlined by several papers in Section 6.2. Swept shells construction is performed by taking the

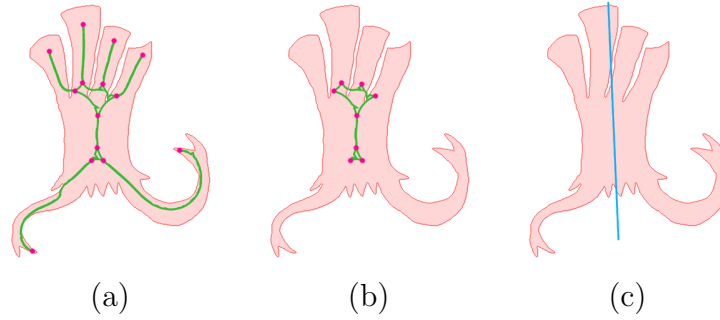


Figure 6.18: Finding the principle axis. The skeleton (a) is stripped back to the spine segments (b) and principal component analysis performed to find the best fitting line (c).

surface when a cross-section is moved along a line in 3D space. For organic mesh reconstruction, solutions using swept shells generally meet with success despite being susceptible to noise and relying on the input having a clear primary axis. These issues are alleviated due to the skeletonisation algorithms' smoothing step that reduces noise, and the primary axis that is pre-generated in Section 6.4.3.3.

6.4.4.1 Establishing Orientation

Concept artwork is often drawn off-centred, such as in three-quarter view, and therefore it is not possible to assume a front-facing orthographic view and so the initial orientation needs to be established to correctly build the mesh. Suh, Y outlines a method for determining initial orientation in his paper *Reconstructing Polyhedral Swept Volumes from a Single-View Sketch* [159], however the algorithm was designed for technical drawings and concept artwork often does not have sufficient right-angled corners.

Instead the orientation is found using side-equality measures based upon the centreline and the ratio of the extent of the armature. If the skeleton topology is found to be symmetrical using symmetry-axis decomposition [144], we make the assumption that the character it represents is also symmetrical. This can be verified by comparing the ratio of extent of each pair of matching limbs in relation to the character's principal axis as gener-

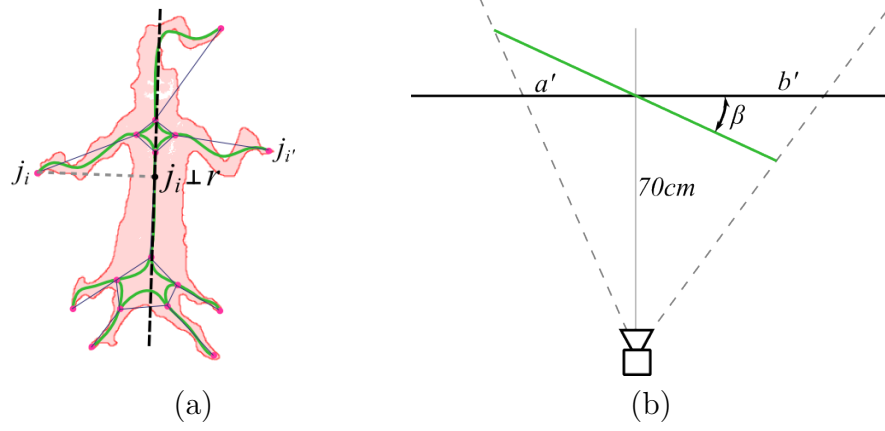


Figure 6.19: If a model topology is symmetrical along an axis, it may be possible to find the orientation the image was drawn at. Image (a) shows the skeleton layout with matching limbs, while image (b) shows how this ratio is used to calculate the orientation of the character depicted in the input image. Image (b) is a top down view where the green line represents the model plane and the black line represents the screen.

ated in Section 6.4.3.3. If the character is posed, this ratio will be different for each pair. If Equation 6.9 holds true for the skeleton, the average extents are calculated using Equation 6.10 and re-orientation is performed. Due to the hand-drawn nature of the input, variation between the limbs should be expected and therefore the ratio of extents does not have to be perfectly equal for re-orientation to occur. Figure 6.19 shows the layout of the implied camera setup.

$$\frac{j_i - (j_i \perp r)}{j_{i'} - (j_{i'} \perp r)} \approx \frac{j_{i+...n} - (j_{i+...n} \perp r)}{j_{i'+...n} - (j_{i'+...n} \perp r)} \approx \dots \quad (6.9)$$

$$a = \frac{1}{n} \sum_{v=0}^{n-1} j_{(i+v)}, \quad b = \frac{1}{n} \sum_{v=0}^{n-1} j_{(i'+v)} \quad (6.10)$$

where j is a set of joints in the skeleton, including end points, and i and i' represent a pair of topologically symmetrical points in regards to the centreline r , where r is calculated as the best-fit line for all j . a and b are calculated to be the average extents for each side of the skeleton.

For a front-facing image the orientation can be estimated by projecting the horizontal offsets back to an implied camera with respect to the centreline. The distance to the camera is taken to be 70cm, which is the recommended viewing distance from a monitor and a value assumed to influence the average perspective for digitally drawn concept artwork [9]. The screen's internal DPI is used to convert the pixel based bone lengths into centimetres, which can then be used to determine the orientation of the drawing using Equation 6.11.

$$\begin{aligned}\theta_1 &= \tan^{-1} \left(\frac{a'}{70} \right), & \theta_2 &= \tan^{-1} \left(\frac{b'}{70} \right) \\ \beta &= \tan^{-1} \left(\frac{\sin(\theta_1)\sin(\theta_1 + \theta_2)}{\sin(\theta_2) - \sin(\theta_1)\cos(\theta_1 + \theta_2)} \right)\end{aligned}\quad (6.11)$$

where a' and b' are the average skeleton extents generated using Equation 6.10 and converted to centimetres, and β is the calculated angle of the model around the Y axis.

Determining the orientation with this method fails where a topologically symmetrical character has artificially stunted limbs or if the artist did not draw in perspective, however the failure case is a zero orientation ($\alpha = 0$) and has no negative impact on the surface generation stage. Error is also introduced by drawing and sketching inconsistencies, although averaging results for all symmetrical node pairs reduces the impact of this.

6.4.4.2 Lofted Surface Generation

The mesh is generated based upon the silhouette of the figure and the previously generated soft skeleton. As mentioned in Section 6.3, mesh topology plays an important role in defining the shapes of an object, and creating believable deformations during animation. In sections without joints, lofted segments create edgeloops based on quads in a similar way to an artist. However in areas of detail the edgeloops often end up distorted.

The initial surface is created using a modified 'lathe' procedure, where limbs and body components are constructed by creating appropriately sized

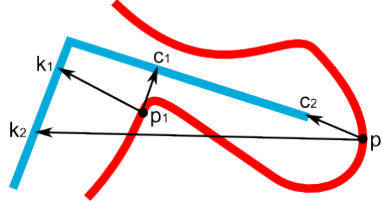


Figure 6.20: This diagram shows the best and worst case placing for the arc endpoints k and c given certain points $p_{1..2}$ on the outline (red). p_1 shows a good result for k_1 , whereas c_1 would create overlapping arcs perpendicular to the outline normal. p_2 creates the opposite result where c_2 is a useable local centre but k_2 would create arcs that lay outside of the outline.

rings around the skeleton. To begin creating a mesh, arcs are created from the boundary in to their local centre. Equation 6.12 defines two possible local centres, each with different advantages.

$$\begin{aligned} c_i &= \text{get closest point on } S \text{ to } p_i \\ k_i &= \text{intersection of } n_i \text{ and } S \end{aligned} \quad (6.12)$$

where S is the soft skeleton calculated in Section 6.4.3, n is a set of normal vectors calculated from the edges of the outline polygon p .

As can be seen in Figure 6.20, the difference between the arcs created by these two methods can be quite significant. This figure shows a simulated side view of a model, with the red line representing the outline, and the blue line representing the skeleton. The 3D arcs are created by placing a 3D disc at the centrepoint (k or c), with radius r equal to the length of the vector. The disc is oriented so as to minimise the dot product between the disc normal and the bone direction, while still ensuring that the circumference passes through p .

k acts better as a local centre because it creates a more evenly distributed mesh at corners and looks better visually. Due to the use of a projected normal small variances in the outline polygon can cause large discontinuities in k and introduce visual artifacts. The higher the noise, the less influence

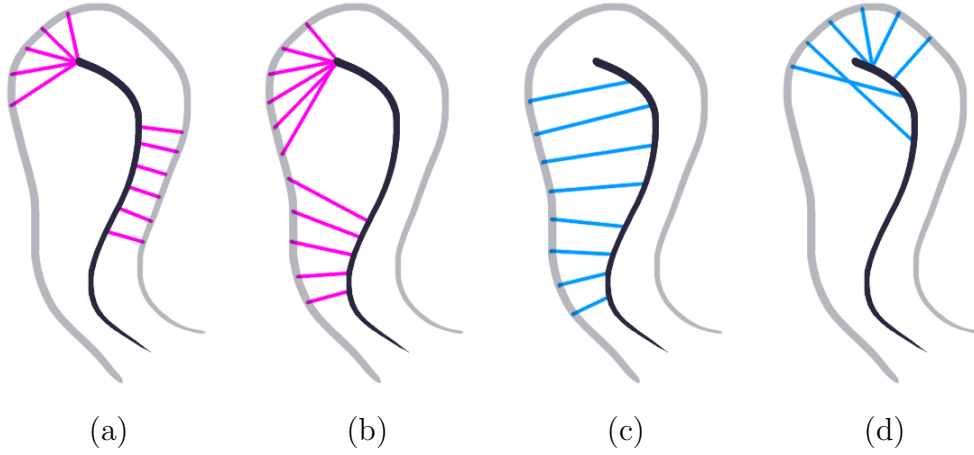


Figure 6.21: This diagram expands on the best and worse cases for centre-point placement, showing a series of arcs for each of the different approaches in Figure 6.20. Images (a) and (b) show the best and worst cases for the shortest distance respectively. Notice the large gap in (b). Images (c) and (d) show the best and worst cases for using the bone normal. Notice the overlapping arcs in (d).

k should have over the final solution. In contrast, c will always provide a point that can be used but introduce banding artifacts by causing groups of consecutive points to have the same local centre. Figure 6.21 shows how the placement of

The accuracy of k also decreases with distance and in some edge cases the normal can be almost parallel to the skeleton, creating intersection points a considerable distance away from the original point. To offset this, an additional term is created that has no influence on the balance of the terms when the distances are similar, but favours c exponentially when the distance between k_i and p_i increases.

$$\omega_d = \frac{\|c_i - p_i\|}{2\|k_i - p_i\|} \quad (6.13)$$

where p_i is the point on the outline.

A similar inaccuracy occurs in c when it is offset from the outline polygon normal and causes the created slices to be visually incorrect. Equation 6.14

sets up a scaling term for this.

$$\omega_\alpha = |c_i - p_i| \cdot n_i \quad (6.14)$$

where p_i is the point on the outline and the result ω_α is clamped in the range $[0 : 1]$.

The final property that effects the choice of centrepoint is the noisiness of the curve, which is calculated using Equation 6.15. The more noise there is in the line, the less accurate properties based upon the line normal or tangent will be.

$$e = n_i - \left| \sum_{k=i-10}^{i+10} n_k \right| \quad (6.15)$$

where n is a set of normal vectors calculated from the edges of the outline polygon.

Weighting our choices of centre position, the final slice centrepoint is calculated using Equation 6.16.

$$p' = ce(1 - \omega_d)\omega_\alpha + k(1 - e)\omega_d(1 - \omega_\alpha) \quad (6.16)$$

6.4.4.3 Cross Section

After calculating the best centrepoint around which to construct the slices, the mesh itself is created by extruding a cross-section along the skeleton. Generating a realistic cross-section often requires information about the type of object being created, and using a perfectly round, centred mesh is not always appropriate. The type of character and character material is often implied by the properties of the silhouette. In "Single-View Sketch Based Modelling", Andre & Saito differentiate between straight lines and freeform rounded shapes. A cubic corner extraction method identifies sharp corners, and these are treated differently when creating the lofted sections. The method in this section aims for a similar differentiation between straight and curved lines, however it modifies the shape of the slice as it is lofted and

not across the entire section. This allows for sections that differ in depth and shape over the length of the extrusion line. This is an important distinction because it allows for smooth joins where multiple lofted sections meet and where different sized sides join.

Shape discovery is performed on a local level by analysing the separated line segments shown in Figures 6.16 and 6.17. One of the best profile indicators is the line curvature and the number of sharp corners in subsegments of the image. Equation 6.17 generates points that define the cross-sectional profile to loft along the skeleton. The terms in the last line create a round surface, which is blended with the square surface in the second line according to the line sharpness:

$$\begin{aligned}
 r &= ||p' - p|| \\
 v(x, z)_{(\alpha=-\pi \rightarrow \pi)} &= r \left(\lfloor \sqrt{2} \cos(\alpha) \rfloor, \lfloor \sqrt{2} \sin(\alpha) \rfloor \right) s \\
 &+ r \left(\cos(\alpha), \sin(\alpha) \right) (1 - s)
 \end{aligned} \tag{6.17}$$

where p and p' are corresponding points on the centreline and outline respectively, and s is a sharpness factor between 0 and 1, calculated in Equation 6.21.

The general approach is that sharp or rough lines indicate that the concept art is non-biological or at least polygonal in nature and the generated mesh should contain fewer round faces, whereas curved or smooth lines suggest that the generated mesh should look more organic. Figure 6.23 shows how this works in practice.

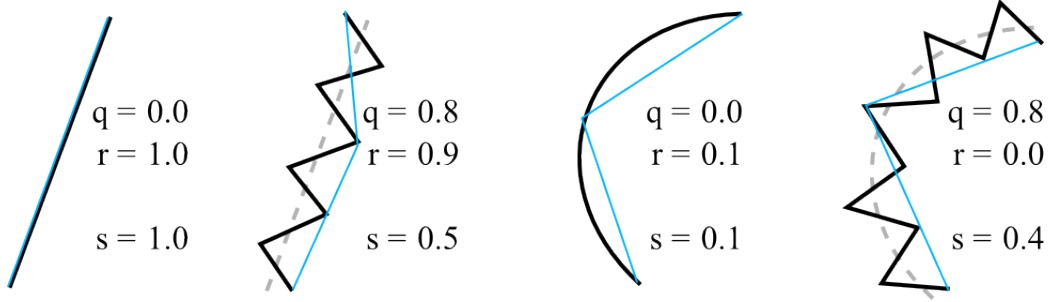


Figure 6.22: This diagram shows the q , r and s values calculated in Equations 6.19 6.20 and 6.21 respectively. The blue line shows the calculation for r and the dashed grey line a curve with $c_{x=0 \rightarrow n} = c_\mu$. A low s value is taken to indicate soft or organic objects, while a high s value is likely to be generated by mechanical objects with square edges. This is reflected in the type of mesh cross-section used to create the model.

$$c_t = \sum_{x=0}^{n-2} 1 - ||v_{x+1} - v_x| \cdot |v_{x+2} - v_{x+1}||$$

$$c_\mu = \frac{c_t}{n-2} \quad (6.18)$$

$$q = \frac{1}{c_t} \sum_{x=0}^{n-2} |c_\mu - |v_{x+1} - v_x| \cdot |v_{x+2} - v_{x+1}|| \quad (6.19)$$

$$r = 1 - |v_{n/2} - v_0| \cdot |v_n - v_{n/2}| \quad (6.20)$$

$$s = \left| r - \frac{q}{2} \right| \quad (6.21)$$

where c_t and c_μ are the total and average curve calculated using v , the set of n vertices from line segment L . The roughness q , segment bend r and sharpness s are all clamped in the range $[0 : 1]$.

Equation 6.18 calculates the average bend of a line segment, which is used by Equation 6.19 to calculate how smooth the line segment is by looking at how far each subsegment differs from the average. A value of 0 indicates a smooth line while 1 is line that does not conform to the average curvature. Equation 6.20 approximates the angle that the line curves through, a value of

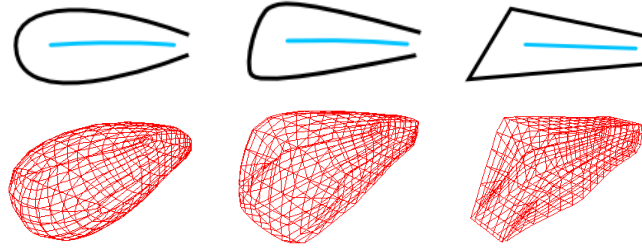


Figure 6.23: Examples of mesh generation showing cross-sectional adjustment based upon line curvature and with respect to sharp corners.

0 means the line does not change direction, while 1 indicates a right angle or higher. These values are combined in Equation 6.21 which gives the overall 'sharpness' of a line segment. A line segment that is rough, or straight will generate higher values, while a smooth curve will result in lower values. Figure 6.22 shows the sharpness value for a number of scenarios.

The mesh itself is created by generating points in the depth plane according to Equation 6.17. The terms in the first line create a round surface, which is blended with the square surface in the second line according to the line sharpness calculated in Equation 6.21. These cross-sections are then offset by the skeleton centrepoint p and joined in order along the skeleton to create the final model.

Due to the centrepoint weighting applied in Equation 6.16, the mesh slices are unlikely to join seamlessly across centrelines and one concave polygonal hole will be present on each side. These are characterised by long thin segments and numerous branches, and can be filled using any of the standard polygon fill methods such as monotone polygon decomposition [117] or Las Vegas triangulation [41]. In our implementation Ear Clipping [57] is used, and can be optimised significantly by removing the ear search stage. An ear is considered to be a polygon with two sides lying along the edges of the generated arcs, and the side spanning the gap between the generated meshes. We don't need to search for ears because after the mesh generation the arc that projects farthest from each end points of the skeleton will always contain at least one point from a polygonal ear. This can therefore be used to start the clipping process.

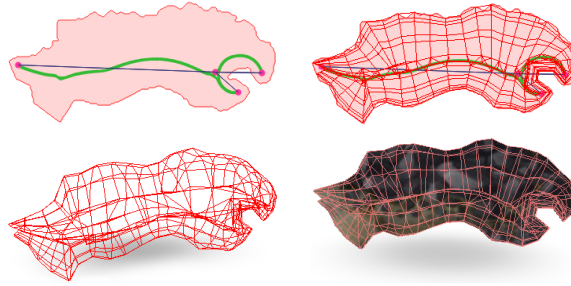


Figure 6.24: This image shows the generated arcs and shells for a basic skeleton extracted from the image outline. The weighted centrepoint creates segments that are not overlapping and merge correctly at corners and ends.

When compared to the worst case performance in Figure 6.20, shells created with the weighted centrepoint appear more natural and are free of artefacts. Figure 6.24 shows a typical wireframe generated from skeleton and outline data.

Even with adjusted centrepoints and consideration to line curvature, there remain situations where the mesh generation will be less than ideal. When a mesh is generated near corners that have an acute angle caused by straight or convex lines, the local midpoints may be distributed a long way apart. This causes the polygonal fill to create a large flat 'ear' [57] that will not deform correctly with skeletal animation. A similar situation occurs at the join between thin outlines and larger objects they are attached to, where neighbouring arcs differ greatly in size. Both of these issues could be solved by inserting extra arcs where needed. However placing these arcs is a non-trivial task, and because visible artefacts are rare it is left unsolved.

6.4.5 *Mesh Refinement*

There are two further approaches used to refine the surface topology and correct any inaccuracies in the depth that arise from lack of information in the outline. Because the mesh is only generated using the outline, analysis of the image internals can provide extra depth cues. These techniques only apply to specific situations and struggle to produce consistently good results across a wide range of images. With input limited to only a single view, further

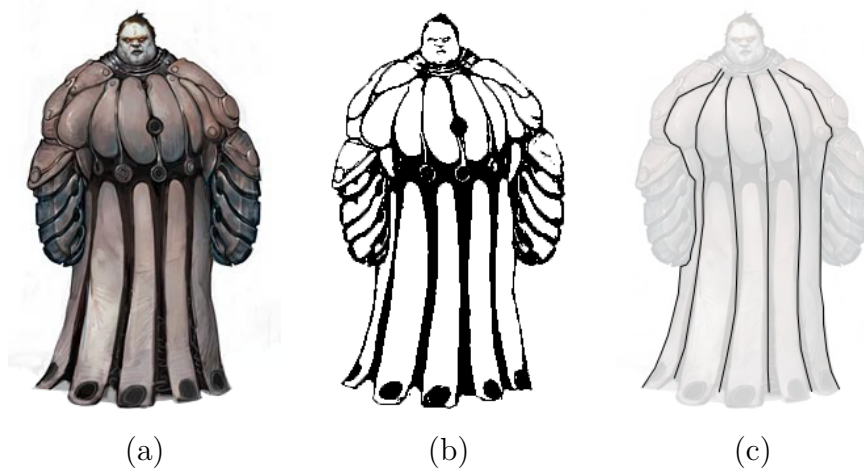


Figure 6.25: Contour extraction from the source image (a) using thresholding (b) from Section 6.4.2 and vectorization techniques from Chapter 4. Lines not aligned to the primary axis were discarded (c) and several intersections manually aligned.

refinements will always have diminishing returns. However, in isolated cases they can make a significant difference.

6.4.5.1 *Depth cues from Contours*

Refinement of mesh structures can be carried out through topological analysis. By re-projecting significant contours within an image into camera space, their depth can be found and compared with the generated 3D mesh. This method was implemented, however not used in the final system because it was too difficult to reliably extract the contours. While in some cases depth cues from contours could be used to enhance the model, the evaluation methods in Section 6.6 found that on average it made the model quality worse.

Figure 6.25 shows how thresholding is performed on the input image before the lines are extracted. Contiguous contours are selected by discarding lines where the angle does not match the image's primary axis, according to Equation 6.22. This means that all selected contours will have a total angle of $< 90^\circ$. The axis is found using the method in Section 6.4.3.3.

$$\text{significant if} \quad \left| \widehat{(v_{n-1} - v_0)} \cdot \hat{c} \right| > \omega \quad (6.22)$$

where v is the set of n vertices in a given contour, c is the primary axis, and ω is a threshold value chosen to be 0.9 through evaluation of different options in Section 6.6.

The most reliable results for depth extraction were achieved using a simple algorithm. The average distance between contours is recorded and used to find depth via a direct linear mapping. No allowance is made for camera perspective correction or a correct cylindrical projection. The camera setup uses the same assumptions as Section 6.4.4.1. Equation 6.23 shows how this is calculated from the contour line data:

$$\begin{aligned} w_\mu &= \frac{1}{(n-1)m} \sum_{\substack{0 \leq u \leq n-1 \\ 0 \leq t \leq m}} \|c_u(t) - c_{u+1}(t)\| \\ w(u, t) &= c_{\lfloor u-0.5 \rfloor}(t) - c_{\lceil u+0.5 \rceil}(t) \\ \text{depth}(u, t) &= d \frac{w_\mu - w(u, t)}{w_\mu} \end{aligned} \quad (6.23)$$

where (u, t) are mesh co-ordinates from 0 to n and 0 to m respectively. $c_{u=0 \rightarrow n}(t = 0 \rightarrow m)$ are n contour lines made up of m vertices each, and d is the virtual camera distance. w_u is the average width between contours, and $w(u, v)$ is the width between adjacent contours at a point.

In general the form is correct but has less resolution than the model created using the outline. The local accuracy is better in cases where external features that show up in the outline are irrelevant to the content. However, this is a very limited technique because connected contour line data is essential. Most images don't have appropriate contours, and while many have distorted textures that produce high quality contours, the rate of 'false' contours from incorrectly textured or irrelevant patterns is quite high.

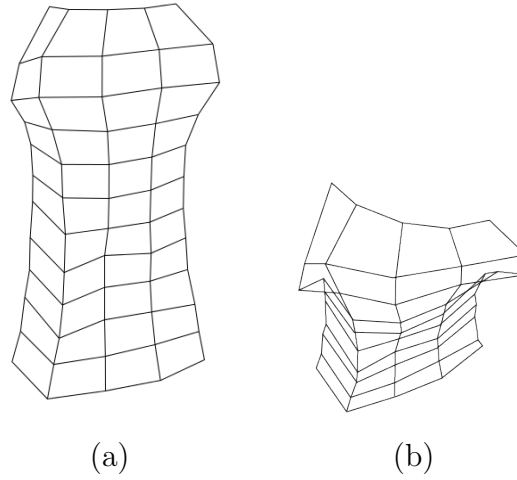


Figure 6.26: Front (a) and top (b) views showing a mesh generated with depth information taken only from the contours.

6.4.5.2 *Depth Compression for Cloth*

While the proposed mesh generation methods create representative meshes for solid objects, some unique image details such as large thin cloth areas can cause issues with mesh generation. Details that are often incorrectly represented by the model creation process include wings or feet with thin membranes, cloth such as sails, or metal sheets. This is due to the lack of contextual or material information that would be needed to identify these areas, and therefore this type of object is incorrectly given depth when instead it should remain flat. Figure 6.27 shows how deep cross-sections produce an artificially inflated representation of a flat object.

Some types of cloth are still represented correctly, such as those that are wrapped around other objects and therefore still have a 3D form. Although no mesh data is generated for the underlying object the cloth still deforms relatively intuitively during animation because the vertices are weighted and linked to multiple bones. This deformation would also work in the case of flat objects if the mesh was correctly generated.

To implement this, a depth scaling step is added to the mesh generation process to override the default segment size for designated parts of an object.

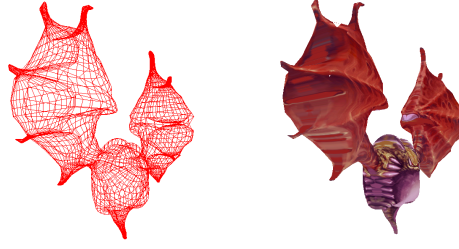


Figure 6.27: Textured and wireframe view showing a bat model created based on a single image input. Without material or context cues, the meshing algorithm has no way to know that the wings should be flat and therefore creates round cross-sections.

Equation 6.24 shows this:

$$v'_i(x, y, z) = (\overrightarrow{1, 1, 1 - s}) v_i \quad (6.24)$$

where $v_{i=0 \rightarrow n}$ is a set of all mesh vertices and s is the depth scaling term between 0 and 1 where 0 leaves the model untouched and 1 generates a flat plane.

The difficulty inherent in this problem is defining the scaling term s for all sections of the model. Identifying parts of a model based on context typically requires a large database [164] or alternatively a smaller training dataset [68]. With only a single input image, image metrics provide the best fallback and therefore the hue and texture are used to identify additional material properties. Due to the wide range of input images accepted by the system, no one metric was able to produce consistent results for automatic detection, and therefore if this step is used as part of the refinement process then a user selection parameter must be used. This parameter allows the operator to select an optional hue and texture to be identified as cloth. The automation of this step is left as future work.

Selecting the cloth areas based on texture falls within the domain of texture classification. One common method for texture classification is to use wavelet transforms [108] [149] [106]. Liapis et al. use a database to match against, and so this method is only effective for images containing prede-

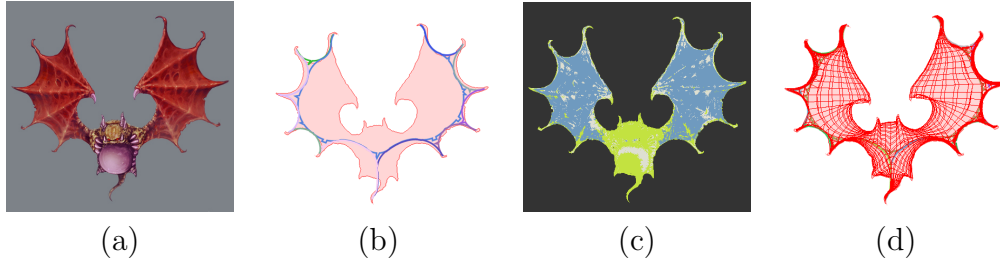


Figure 6.28: This image shows the standard mesh construction steps of taking a concept image (a) and extracting the skeleton (b). User directed hue is enough to segment the image and isolate the cloth (c). Whilst the mesh generation (d) looks identical from the front, depth values are overridden based on the cloth map to create flat wings.

finer textures. One way to avoid this limitation is to use an unconstrained clustering algorithm, most commonly in concert with Gabor filters [90] [155]. Bhiwani et al. [22] compare the effectiveness of a number of different classification methods when applied to texture features, finding that Canberra and Bray Curtis distances produce the best results.

Even when using current texture classification methods the colour segmentation still may not correctly align to picture. An example can be seen in Figure 6.28(c) where areas of texture are not continuous and ill defined near the edges. The image is however logically segmented, so to reduce the error around the edges depth variation is performed on whole skeleton sections. A section is considered to be cloth if more than half of the skeleton points are within the cloth texture.

This occasionally introduces a disconnect problem where adjacent and back-to-back skeleton segments (Section 6.4.3.1) have different depths, creating surface discontinuities. To solve this issue each set of paired skeleton vertices are given the same depth value, taken to be the maximum depth of the two. The cloth allocation and skeleton depth matching are performed in Equation 6.25. The image I used in this operation is the untransformed concept artwork.

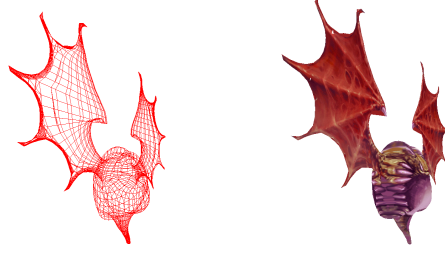


Figure 6.29: Textured and wireframe images showing a bat model created using material cues. The wings have been tagged as cloth material, and therefore are flattened into a defined plane to better represent the intent of the input image. Notice the improvement in this image over the wings created in Figure 6.27.

$$\begin{aligned}
 S(p), S'(p') &= \text{median} \left[2 \frac{|I_h(p) - C_h|}{\max(I_h)} T(p) \right]_{p=0 \rightarrow S_n, S'_n} \\
 s &= \max(S(p), S'(p))
 \end{aligned} \tag{6.25}$$

where p is a vertex on the bent skeleton segment S with matching side p' on segment S' with n skeleton vertices. I_h is the image hue channel, and C_h is the cloth hue. T is a map corresponding to the cloth texture calculated using wavelet transforms [108] where 0 is no match and 1 is a perfect match.

As can be seen in Figure 6.29, cloth-based flattening changes the character and accuracy of the model significantly. There is a marked difference between the inflated model in Figure 6.27 and this version. Because of segment-based scaling and the percentage-based texture type detection, the cloth demarcation for the original model does not need to be overly accurate. A typical real-world example of this is the more complex goblin model in Figure 6.30, where the light grey cloth is detected in the head of the staff.

While this simple method of cloth flattening works to a certain degree, there are still a number of issues that can arise. In several cases, the bones and structural components underlying the cloth are also flattened. Ideally these would be separated and individual 3D meshes generated. The cloth

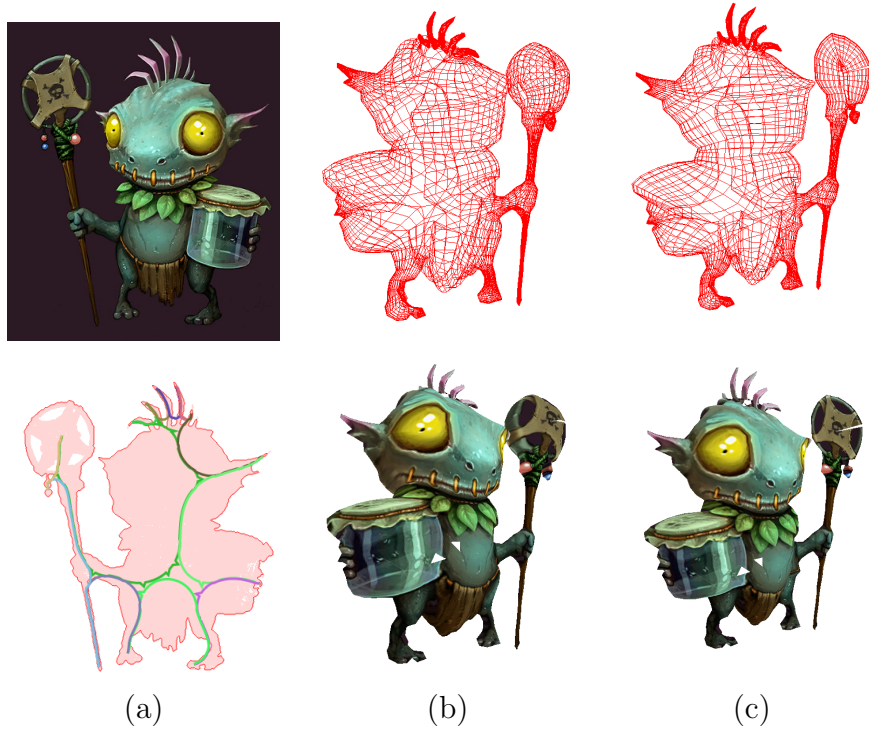


Figure 6.30: In most situations the effect of forced depth compression for materials is more subtle. Without refinement, the concept image (a) gives depth to the cloth on the tip of the baton (b). When hue separation is used to identify the cloth from the tip of the baton, the segment depth sharing outlined in Equation 6.25 ensures the entire tip is correctly flattened (c).

could then be draped over or spread between the structural components. Implementing this would increase the complexity of the algorithm significantly for only minor gains.

The use of user specified input also breaks with the original aim of automatic model generation. Automatic cloth texture selection is a difficult operation that relies on complex image understanding, and is outside the scope of this section. Due to the need for manual intervention, this cloth flattening method was therefore not used on the models when performing evaluations or generating models in Sections 6.5 and 6.6.

6.4.5.3 Rigging and Animation

In addition to mesh generation, practical use of a 3D model requires an armature and texture coordinates. Numerous studies have focused upon generating skeletons based upon existing meshes, and several of these methods [160] [143] [144] [180] extract structures similar to the curved 2D skeletons from Section 6.4.3. Other automatic skeletonisation methods focus specifically on animation [17] and deformation [110] [98] of the mesh, and even include skeletonisation within the framework of sketch-based interfaces [28].

The paper *A novel template-based automatic rigging algorithm for articulated-character animation* [144] classifies models based upon their skeletal topology. This is done by extracting any 3D skeleton from the model “using a suitable skeletonisation algorithm”, and then measuring additional topological properties. This includes the symmetry axis, the number of significant joint nodes, and the lengths of leaf chains attached to the central nodes. The symmetry axis is found by analysing the topology and identifying nodes that have equal leaf chains and segments with similar directions, and is used in this section when attempting to re-map animation between non-matching skeletons. The general classification approach of using skeleton junctions and topology to classify animal type - such as fish, birds, and humans - is not robust when considered in the framework of concept artwork. The skeleton topology produced in this chapter may be missing segments, or contain additional segments representing weapons.

An example of a system that attempts to unify modelling and rigging is *RigMesh* by Borosan et al. They maintain that the separation of modelling and rigging is difficult for artists, and therefore create a system that rigs models and stores their pose as they are created. A base skeleton is automatically generated after each change in the mesh, using a modified version of the Douglas-Peucker algorithm on the chordial axis. The modifications take cylindrical diameter into account in addition to the standard salient points. Mesh weighting to bones is performed locally using the pinocchio algorithm [17]. A number of interesting techniques are used in this paper, most notably the chordial axis algorithm for skeleton extraction. The chordial axis is a medial skeleton extraction algorithm that can be run on vector

outlines instead of raster images.

The system in this section makes a clear distinction between mesh generation and rigging, but hides this from the end user as a ‘black-box’ style system. Because there is no ability for the artist to go back and edit the mesh we do not have to regenerate the skeleton at any stage, instead the entire system is presented as a single step where basic controls allow influence over the entire process. This allows us to process the mesh in such a way as to make skeleton construction easier.

Having generated the underlying mesh itself, a number of key properties can be accessed and used to augment existing algorithms. The soft 2D skeleton is projected into 3D by aligning bone and orientations as closely as possible to an existing animated skeleton, after which the mesh is attached to the bone structure. Equation 6.26 shows this simple alignment step.

$$T_b = T_{parentofb} * A(b')_{rotation} * A(b')_{scale} \quad (6.26)$$

where T_b is a 3D transform matrix for the bone b in the skeleton, and A is the target armature with corresponding bones b' .

$$\begin{aligned} p_i &= clamp(v \perp B_i) \\ d_i &= ||v - p_i|| \\ \alpha_i &= |v - p_i| \cdot n \\ s_i &= \frac{d_i}{\sum_{n=0}^3 d_n} \\ influence_i &= s_i \alpha_i \end{aligned} \quad (6.27)$$

where $B_{i=[0..3]}$ is a list of the closest 4 bones to the mesh vertex v with normal n . Bones are represented in point-vector form and a limit of four is used as this is the maximum supported number in many 3D engines. p_i is the vertex



Figure 6.31: An attack animation is applied to a generated model of a monster. The original positions and rotations are correctly transferred and scaled to the appropriate lengths of the generated armature. Vertex mapping ensures the mesh follows the bone structure, including axial rotations such as the twist of the spine.

v projected onto B_i and clamped between the two ends.

Unbent limbs in the source image make it difficult to identify joints such as knees or elbows. This can be mitigated by looking at basic contextual information. The spine is found using Symmetry-Axis Decomposition outlined by Pantuwong et al. [144]. Long, non-branching, mirrored limbs that are leaf nodes to the spine are assumed to represent arms or legs and are split to contain a middle joint. To allow for situations where this assumption does not hold, a second check can be performed once animations are transferred to the object. Joints that do not bend significantly across any of the animations can be considered spurious and removed. Section 6.1 shows a similar operation performed in 2D.

The mesh is then attached to the bone structure. Two common attachment methods are vertex groups [6] and bone envelopes [62]. While the underlying structure uses vertex groups so as to be easily used in realtime applications, these groups are essentially calculated using bone envelopes based upon the generated armature. The influence of a bone over a vertex is based on the distance to the bone and the surface normal at the vertex, and is calculated in Equation 6.27.

Simple tests have shown that the projected 3D armature and the vertex weighting generated by Equation 6.27 can be used in a range of motions without significant artifacts. Motion retargeting [69] [77] can therefore be used to map existing animations to the generated skeleton. Figure 6.31 shows an example of this.



Figure 6.32: Texture interpolation and inaccuracies in model generation can mean that textures do not fully cover the entire model surface and background colours (a) can be visible. Border seam expansion is used to reduce these artefacts (b). A bright background colour was used for demonstration purposes.

6.4.5.4 Texturing

Texturing a model with only one source image poses a number of problems, many of which are beyond the scope of visual style research. Problems such as texture seam artifacts are easy to solve, whilst others such as texture estimation for occluded projection would require existing algorithms to be adapted for this use. The common issue is one of texture synthesis, as the model has been generated from a single view and texture information for all occluded faces is missing. Existing texture synthesis research can be leveraged to create the missing textures, and fortunately a wealth of texture synthesis research exists.

Numerous papers deal with general pattern synthesis based upon small sections of known texture [42] [78] [139] [177] [178] [132], or image background synthesis for photo expansion or foreground removal [174]. There is however little research into texture synthesis for occluded projection or character-specific texturing. Pattern synthesis techniques are difficult to apply to model textures because they require a prototype image or generation formula. Edge-fill and replacement algorithms are also difficult to implement because they are likely to copy unique details from the texture that would make the cloning process obvious. Solving these issues is outside the scope of this thesis, and a simple re-mapping approach is used as a viable tradeoff.

Projected texture mapping is used to apply the single front view to generated model vertices. This causes both sides of the model to have the same texture which often results in obvious artefacts. Aside from this issue, the

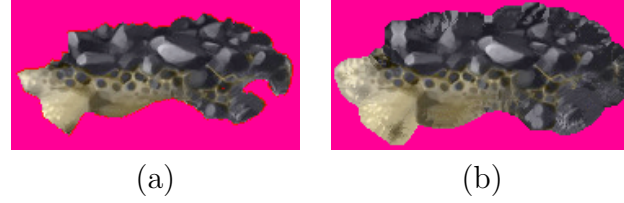


Figure 6.33: Border seam expansion is used to reduce artefacts when mapping a texture (a) to the model. The red border indicates the area that is visible on the model but due to antialiasing and the influence of the background is the wrong colour. Expanding the border by 10 pixels (b) means that every mapped pixel on the model is correct.

most common problem is texture seam tearing. Figure 6.32 shows how the background colour can show through due to texture interpolation and inaccuracies in the generated model.

$$T'(p) = \begin{cases} \|T(p) - o(p)\| < d & 2o(p) - T(p) \\ > d & T(p) \end{cases} \quad (6.28)$$

where T is the texture, o is the closest point on the outline to p representing a set of all pixels outside the outline. d is a predefined distance threshold set to 8 pixels, which is the largest distance for texture interpolation on modern graphics hardware.

Figure 6.33 shows how texture border expansion can be used to cover these seams. Because the texture borders need only be expanded to cover a small number of pixels, repeating or cloned image content is not an issue. Panel (b) shows how the contents of the image are reflected about the outline extracted in Section 6.4.2, providing texture continuity and ensuring texture matching at seams.

While viewing the model from the back will still make the nature of projected texturing obvious, the generated detail from border expansion gives the model better quality when seen side-on.

6.5 Results

The algorithm outlined in Sections 6.4 to 6.5 was designed to generate a low-resolution application-ready 3D mesh from a single-view 2D concept artwork. The process focuses on front view character artwork and in this respect it successfully generates useable 3D models that represent the underlying artwork. The process is entirely automated, an outcome that is important in contributing to the original goal of reducing creation time and artist workload.

The process has been run with success on a number of different datasets from different artists. Figure 6.34 shows a typical piece of character concept art and the resulting generated mesh. This is an example of the best-case mesh generation, because the concept art fits all the required criteria and does not contain any unusual shapes that could cause artefacts. Figure 6.37 shows a difficult case where the input image contains two overlapping characters. Although they are not segmented correctly, the 3D result is still a cohesive and useable mesh.

If cloth is wrapped around other objects and still has form, the resulting model is usually correct. Figure 6.36 shows some examples of this. Although no mesh data is generated for the implied legs, the cloth still deforms intuitively during a walk cycle because the vertices are weighted and linked to multiple leg bones.

Figure 6.35 shows how the slices used in mesh generation produce the correct depth and form. Perspective views show how the shoulders and chest bulge outwards, while the feet remain small and are correctly texture mapped. The aim of mesh generation is to create a model as close to the source image is possible, and given the minimal visible differences between the concept art and the generated models, these results can be considered a success.

6.6 Evaluation

Although the results in Section 6.5 appear correct at a glance, it is difficult to evaluate whether the generated mesh is an accurate 3D representation of

the 2D image. This is a traditionally subjective judgement and is difficult to measure. However, in the case of computer game concept artwork it is often possible to find matching 2D and 3D artwork, and these can be used as a benchmark in an evaluation. Using the same 2D image as a source, a model created by an artist can be compared to a generated model.

The evaluation outlined in this section uses a surface difference metric to compare the generated results with a number of artist created 3D models. In select cases where no original concept art was available, the input image was generated from the artist-created model. A number of approaches have been suggested for comparison of 3D models [40] [127] [163] and these share many of the same ideas. The evaluation in this section uses two measurements; the hausdorff distance, outlined in *Interactive Hausdorff Distance Computation for General Polygonal Models* [163]; and the averaged surface error metric outlined by Aspert, N et al. in *Metro - Measuring Error on Simplified Surfaces* [40]. A set of 15 concept artworks and associated models were used for the surface difference comparisons.

In addition to evaluating the final model, this comparison was used during development to refine the output. Developing an end-to-end system without user interaction necessitates a number of tradeoffs, and the selection of so-called ‘magic numbers’ (such as thresholds and bias) needs to be done so as to produce the best results across a wide range of input. Changes to these numbers, as well as changes to the algorithm itself can often be difficult to evaluate. The average surface difference across the model set was a valuable way to determine how changes to the generation effected changes in the output. Parameters were iterated across a range and the results for 15 different model sets compared to determine the value that produced the best result across the largest number of test cases.

The Hausdorff distance computation requires that the two models are aligned as well as possible in 3D space. Models created using reprojection (i.e. models where the 2D image has been created by rendering the 3D model) are aligned by default. However, models where the 2D image is a piece of concept artwork require alignment. For the results in this section, the models were manually rotated to use the same world co-ordinate system,

then automatically scaled and translated so that their computed AABBs were the same.

When calculating these surface metrics, the values are calculated from vertices in both models to avoid biasing due to differences in mesh complexity. Each distance is scaled by the sum of the adjacent triangles areas to avoid bias from areas of high point density. The normalised hausdorff distance is a value between 0.0 and 1.0, representing the average distance in world space units between the surfaces of two models of unit size. 0.0 represents a perfect correlation and is the case if the two models are identical. An uniformly distributed random pointcloud would return a value of 0.5 and is used as the benchmark for uncorrelated data. Numbers less than this can be returned in cases where the surfaces are ordered in such a way that the average distance between them is more than half of the unit length. For similar models, the normalised hausdorff distance is typically between 0.0 and 0.1.

A number of interesting results arise, and these can be roughly categorised based upon the surface metrics. Models with a low hausdorff distance and low error show that the generated model matches closely the ‘ideal mesh’ represented by the artist’s model. Models with a high hausdorff distance and a high surface error show a global offset or inconsistency. This is easier to fix than local issues, and can in cases be as simple as rescaling or performing an offset. Typical causes of this are skeleton alignment issues within the outline and depth differences between the generated and target mesh, for example if the target mesh is on a lean or generally offset along the z-axis.

The most interesting results to analyse are models that have a low hausdorff distance and a high error. This usually indicates local errors, and although the results are less than ideal they showcase the parts of images that aren’t generated correctly. Common examples of this include depth issues where protruding feet and hands are not generated; 2D concept artwork where the outline is not representative of depth; and the lack of generation of hidden details, such as legs under dresses.

Method	μ	θ	$\mu\%$	$\theta\%$	min	max
Rescaling	0.0115	0.0112	98.85	1.13	2.637×10^{-6}	0.0868
Generation	0.0139	0.0135	98.61	1.36	0.089×10^{-6}	0.0726

Table 6.1: The model surface metrics. μ is the normalised hausdorff distance, while θ is the surface error metric. For ease of interpretation these are also shown as a percentage according to the formula $\mu\% = 100(1 - \mu)$, where 100% represents a perfect match. The *min* and *max* columns shown the minimum and maximum surface errors.

6.6.1 Method Comparison

Sections 6.3 and 6.4 outline two different methods for creating 3D models from a single image. Whilst the benefits of each have been looked at qualitatively in Sections 6.3.2 and 6.5, a surface difference evaluation allows us to quantitatively measure the difference between the two methods and compare them to an artist-modelled benchmark. To accurately measure the difference between generated models and the benchmark, a 2D initial image is generated by re-projecting the 3D model back into 2D. This ensures that the model represents the ideal target mesh and can be used as a valid benchmark.

Figure 6.38 shows the reprojected ninja character in the correct T-pose with the associated 3D models. This character is an ideal model for generation, with a known topology and a simple outline. The original model is then compared with the generated models. The surface difference metrics are shown in Table 6.1. Both generation methods produce good results, although the rescaled mesh has a slightly better correlation and a moderately better standard deviation. The difference in standard deviation is important here, as it shows the rescaled mesh is more consistent and is likely to better reflect the original model, whilst the slightly higher standard deviation of the generated mesh indicates that there are likely to be certain features or parts that are more incorrect.

A visualisation of the surface correlation is shown in Figure 6.39. This highlights the differences between the two generation methods and provides more context for the calculated metrics. The generated model has two areas of significant divergence - around the neck and chest muscles, and at the feet.

Method	μ	θ	$\mu\%$	$\theta\%$	min	max
Rescaling	0.0345	0.0324	96.54	3.24	2.787×10^{-6}	0.204
Generation	0.0202	0.0191	97.98	1.91	0.090×10^{-6}	0.153

Table 6.2: The model surface metrics for the squid model in Figure 6.40. These are labelled as per the outline in Table 6.1.

The difference in the feet is easy to understand, and can be seen clearly when comparing the meshes in Figure 6.38. Because the feet are viewed from the front and no additional data is used when generating the mesh, the resulting mesh incorrectly reflects the depth of the feet and has only a cylinder in their place. The surface error around the chest area is due to the depth in the musculature in the original model that is not replicated in either the generated model or the rescaled model.

To test the generation methods in a more extreme case, the squid-type monster in Figure 6.40 is used. Whereas the ninja model in Figure 6.38 showed an ideal generation case, this model contains multiple appendages, an irregular shaped body, and irregular surface data. Although the character is unconventional, it is not unreasonable to expect this type of input when considering the target applications of video game prototyping and the entertainment industry. Geons could not be calculated automatically for this model, so the layout was specified manually before running the rescale procedure.

Table 6.2 shows the surface correlation metrics, and Figure 6.41 shows the distribution of depth differences across the model. While understanding depth information for the body requires a better contextual understanding of the creature, it could be possible to generate depth for the antennae and tentacles using predictive organic modelling such as that proposed by Ijiri et al. [88]. This, however, is left for future work.

6.6.2 Breadth of Input

Table 6.3 shows the generated evaluation data for 12 of the models. The remaining 3 were used for rescaling and had significant artefacts when generating meshes. It is possible to gain a good insight as to how accurate the

Model	μ	θ	$\mu\%$	$\theta\%$	min	max
Oriental	0.0298	0.0283	97.01	2.84	2.149×10^{-6}	0.132
Woman	0.0146	0.0142	98.54	1.42	12.89×10^{-6}	0.070
Gaston	0.0315	0.0299	96.84	2.99	0.787×10^{-6}	0.143
Hanks	0.0144	0.0140	98.55	1.41	0.035×10^{-6}	0.085
Humpty	0.0335	0.0317	96.65	3.17	5.460×10^{-6}	0.167
Jabinello	0.0471	0.0431	95.29	4.31	2.200×10^{-6}	0.255
Lin	0.0221	0.0212	97.79	2.12	1.672×10^{-6}	0.116
Lin's Mum	0.0370	0.0344	96.3	3.45	2.351×10^{-6}	0.160
Ninja	0.0139	0.0135	98.61	1.36	0.895×10^{-6}	0.072
Slug	0.1407	0.1159	85.92	11.6	26.66×10^{-6}	0.322
Undead	0.0180	0.0175	98.19	1.75	1.068×10^{-6}	0.087
Squid	0.0202	0.0191	97.98	1.91	0.901×10^{-6}	0.153
Average	0.0352	0.0319	96.44	2.382	4.755×10^{-6}	0.147

Table 6.3: The evaluation data for 12 models. Figure 6.42 shows two examples of the type of model used.

generated models are by comparing the measurements to the best-case and worst-case scenarios. The best case is obviously a perfect match at 100%, and in cases where the model is distinctly humanoid the generated model approaches this level of accuracy. Examples of this are the *Woman*, *Hanks*, *Ninja* and *Undead* models. On the opposite end of the scale, two random point-clouds generate a correlation of 50% (each point is, on average, half of the model size away from it's matching pair). None of the models come close to this and so the failure case demonstrated in Section 6.5 is used as a benchmark for a badly generated model with $\mu = 0.2$ and $\theta = 0.2$.

The problems with some generated models are obvious, and indeed models that score badly in the evaluation generally push the limits of what input can be accepted, or outright break the initial requirements. However it can be less obvious what issues arise with the average model.

Figure 6.42 shows the source and generated models for two low-poly game-ready models typical of the characters in Table 6.3. The models, *Lin* and *Lin's Mum* scored ($\mu = 0.022, \theta = 0.021$) and ($\mu = 0.037, \theta = 0.034$) respectively in the evaluation, making these both around average for the current method. The heatmaps can be used to identify the characteristics of the

model that were hardest for the generation algorithm to understand and build a mesh around. In the case of *Lin's Mum* the crossed hands stand out in front of the body and are not generated. This is a very difficult case that includes several occluding layers. The opposite issue occurs with the *Lin* model where the long hair obscures the outline of the head and neck, causing the algorithm to generate a larger mesh than required.

6.6.3 Discussion

This evaluation confirms that generated and rescaled models reflect the input concept artwork, often with less than 5% surface difference from the target image and in the best case within 1.5%. The evaluation also identifies the biggest graphical issues and areas where the generation method fails consistently.

As outlined in Section 4.3, there are still many problems to be solved even with a limited input dataset. Models with few details and big shapes often have a high conformance, while internal and complex details are not generated. Rescaling can produce better results in these cases, but fails on a wider range of input. Generated models work better across a wider range of body shapes, but fail on occluded details. Additional processing steps such as cloth flattening can be performed, but this requires artist or user intervention.

The paper *Interactive 3D Modeling Using Only One Image* [111] also looked at the differences between generation and rescaling. Although only informal tests were carried out, Liu et al. came to similar conclusions about the advantages and limitations of each system, concluding that generation was the better method for unconstrained input.

The largest issue with both automatic generation and re-scaling is recognising small details and details that are unrelated to the profile. Increasing the accuracy of these details further is difficult, and greatly increases the complexity of the generation algorithm for diminishing returns.

6.7 Conclusion

This Chapter outlined two automatic model creation methods that create production-ready models from single pieces of concept artwork. Section 6.3 proposed a re-scaling algorithm using a premade base mesh, while Sections 6.4 to 6.5 proposed a reconstruction algorithm that generates models from an extracted skeleton. Both methods are suitable for use in rapid prototyping and content generation pipelines with limited format input data. Model generation runs without requiring human judgement and produces acceptable results for a range of character types and shapes. With the generation of bone structure and vertex weighting data, models are appropriate for use in low-resolution realtime applications such as video games.

This is a step forward in the area of automatic content generation, and has the potential to save large amounts of time when prototyping and developing low-poly products.

A mesh surface difference algorithm was implemented in Section 6.6 and was used to evaluate the results of the algorithm against artist created content. Model generation using lofted segments showed good results across a wide range of input, with a reasonable match against the artist generated versions of each model.

As outlined in Section 6.5, there are still many problems to be solved even with our limited input dataset. Creating a mesh from a single image is a complex topic that relies in no small part upon human perception. The addition of an extra dimension inherently requires the creation of extra data with an eye to artistic style and character design. These algorithms fill the gap between context-free general meshing algorithms and contextually driven manual creation of character meshes by an artist. It is a small first step into an area of research that has great potential benefit to the video games industry.

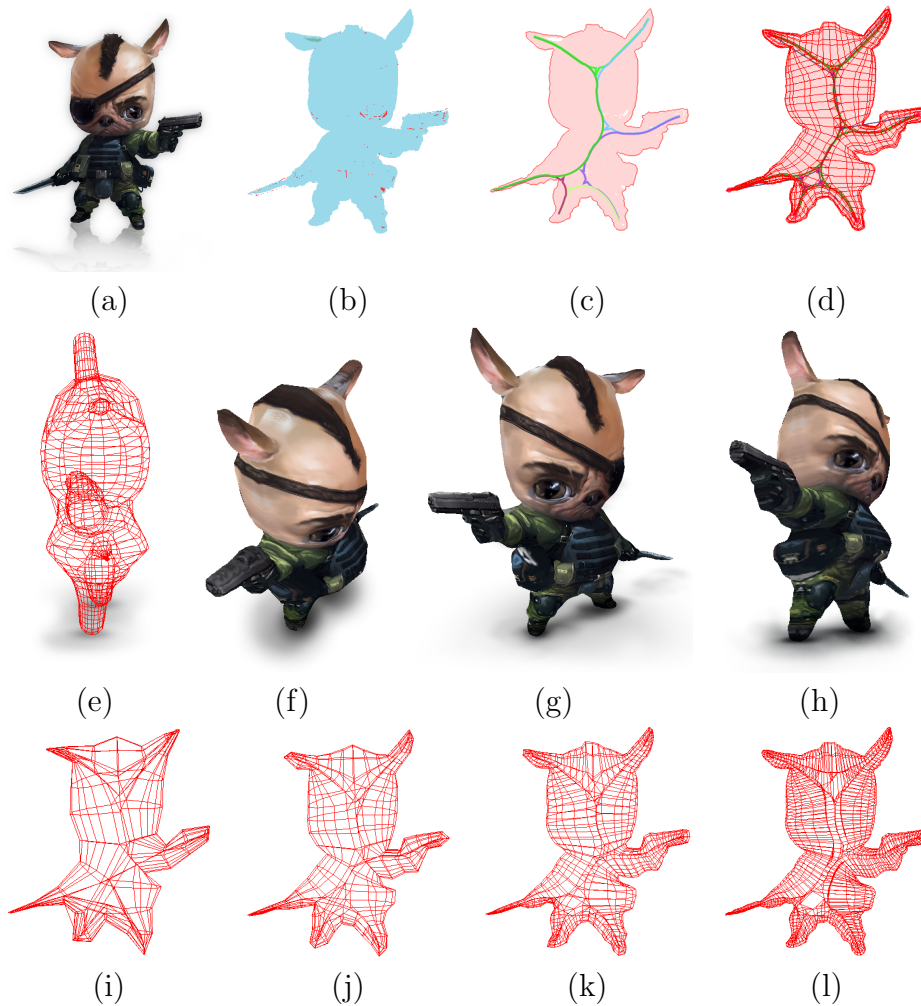


Figure 6.34: An example of mesh generation from a typical concept image. The 2D concept (a) is separated from the background (b) and a bent skeleton (c) is generated and used to create polygonal shells (d). The profile view (e) shows clearly how the shells create a 3D representation of the input image. The final model (f - h) is mapped with textures based on the initial concept artwork and contains bone and vertex weighting information appropriate for animation. Details such as the hair and ears are correctly transformed into 3D, and mechanical objects such as the gun and knife have sharp edges. The complexity of the model is determined by the number of sections generated by the algorithm, and can in turn be used to generate different LOD (Level of Detail) meshes for realtime applications (i - l).

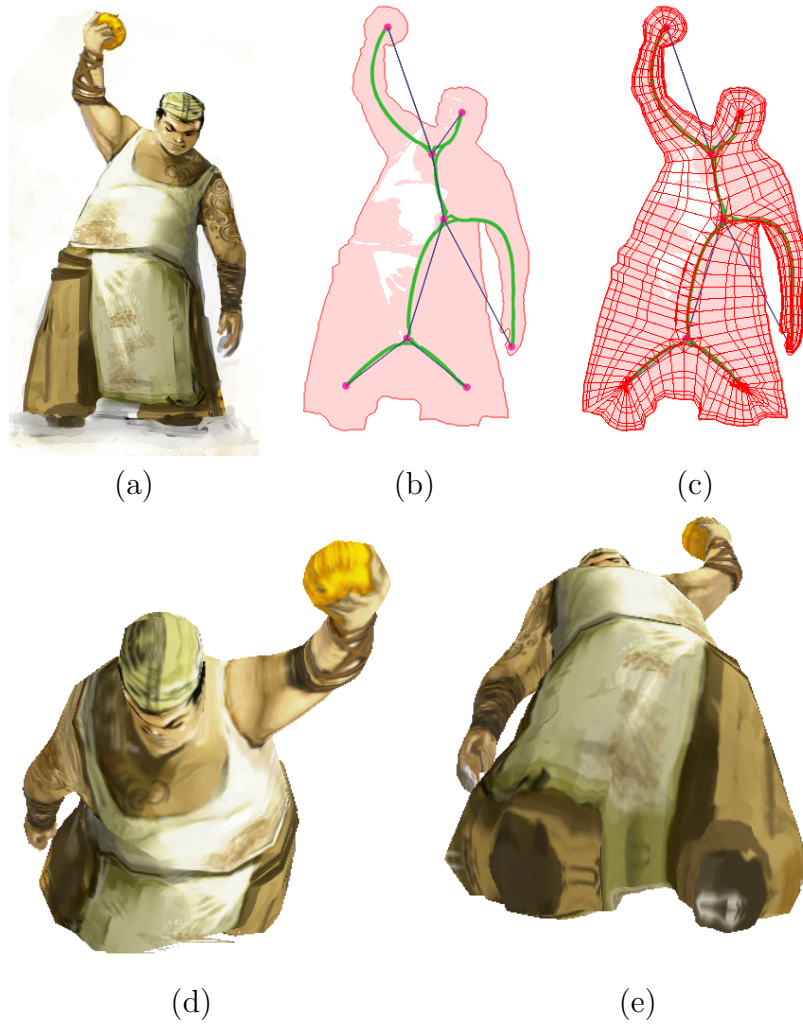


Figure 6.35: Several steps illustrating mesh generation from concept artwork. The concept artwork (a) is analysed and a bent skeleton (b) used to produce swept shells (c) that form the final model. Perspective views from the top (d) and bottom (e) show that the correct body shape is produced in 3D.

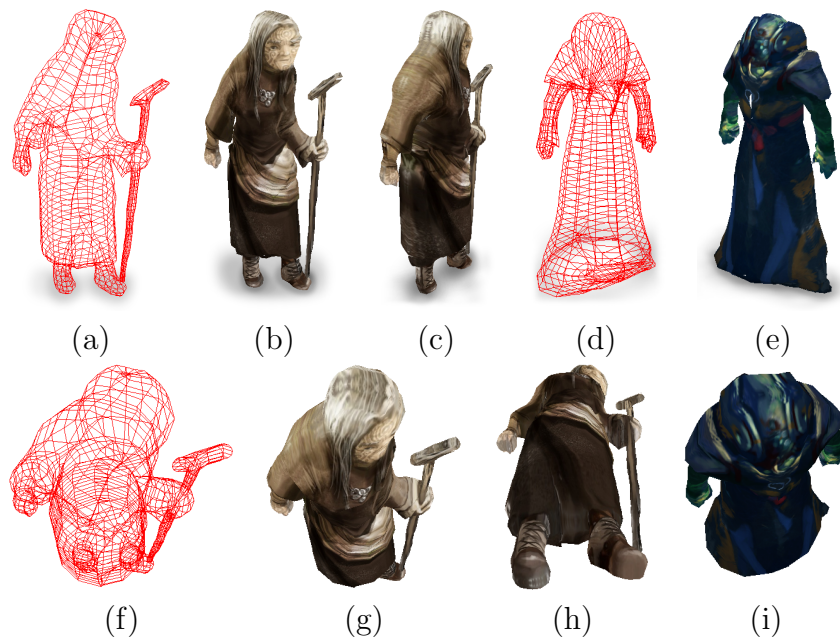


Figure 6.36: Mesh generation for fabric components produces acceptable results (a, b & f) if the fabric is not a thin sheet and instead is given form by the obscured (b & c) or implied (e) underlying objects. The top (g & i) and bottom (h) views show how the depth is preserved.

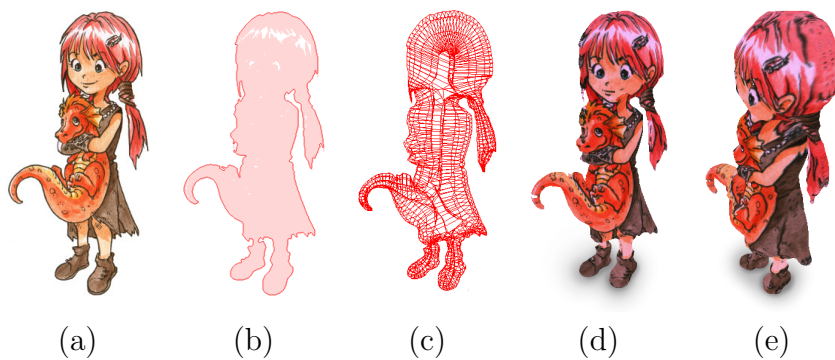


Figure 6.37: Comparing the front (d) and side (e) views of a model shows how the original concept art (a) can be stripped of dark borders and used to map the sides of the generated model. This figure also shows multiple characters in the input image. While the resulting mesh (c) is still a good 3D representation, the lack of segmentation (b) means that animation and repositioning is unlikely to work correctly.

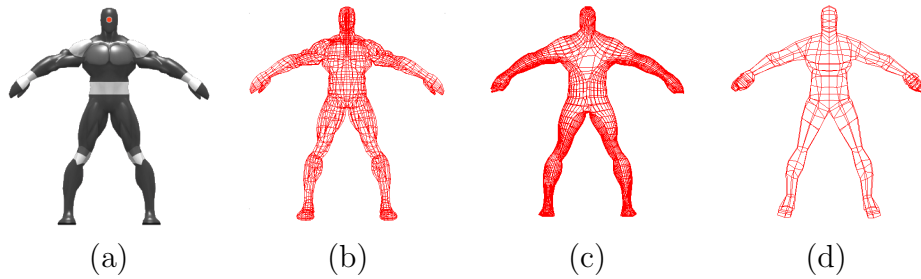


Figure 6.38: The reprojected source image (a) and three associated models for evaluation. Image (b) shows the original 3D model whilst (c) is generated from scratch using the method in Section 6.4.4 and (d) is rescaled from a generic base mesh in Section 6.3.

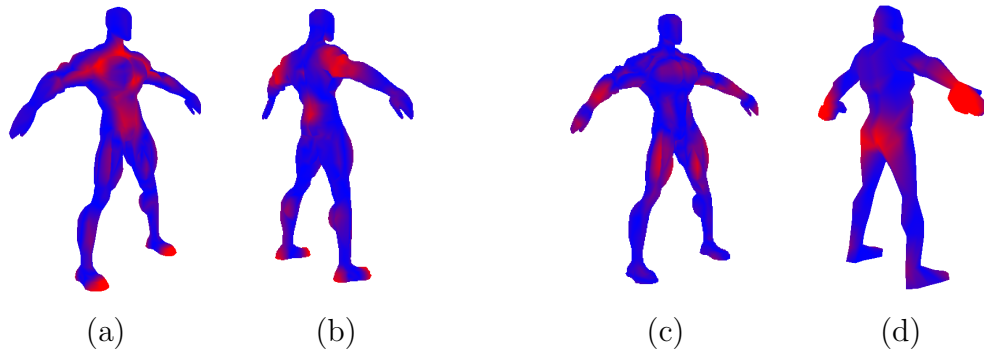


Figure 6.39: Evaluation of different generation techniques using a surface error metric. Blue shows areas of high correlation, while red shows areas of low correlation. Images (a) and (b) show the front and back respectively of the original model compared to the generated model. Note especially the large error around the feet and the neck muscles, both of which are difficult to generate. In comparison, these parts have high correlation in the rescaled model (c) and (d), however it can be seen in image (d) that the rescaled hands have low correlation due to the incorrect depth scaling.

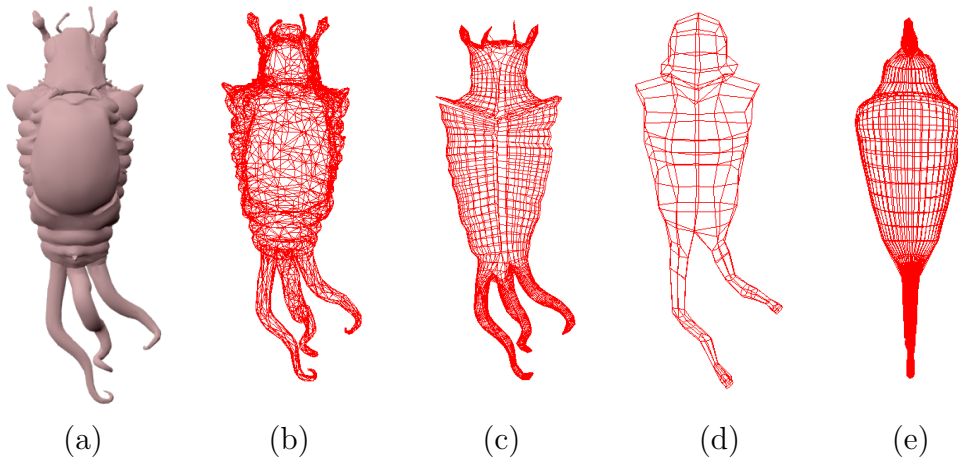


Figure 6.40: The reprojected source image (a) and three associated models for evaluation. Image (b) shows the original 3D model whilst (c) is generated from scratch using the method in Section 6.4.4 and (d) is rescaled from a generic base mesh in Section 6.3. Image (e) shows the worst-case scenario, which is the side view of the generated model. The extent of the missing details can be seen in Figure 6.41.

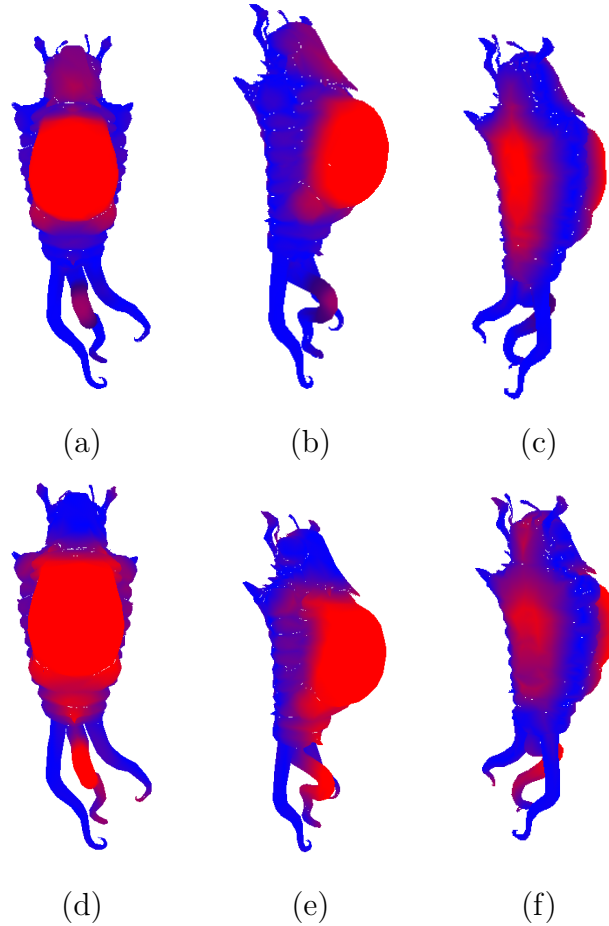


Figure 6.41: Evaluation of different generation techniques on the squid model. Blue shows areas of high correlation, while red shows areas of low correlation. Images (a), (b) and (c) show the front, side and back respectively of the original model compared to the generated model. The distribution of error is very similar to the rescaled model (d - f), although subtle differences can be seen and in general the rescaled model has a lower correlation. This is especially visible when comparing the two front views (a) and (d). Comparing the back views (c) and (f) also shows the complete lack of a centre tentacle in the rescaled model. The rescaled model appears to have a better correlation with the head, and while it is difficult to compare the antennae the tips are blue in the generated model and red in the rescaled model which indicates a lack of correlation as they move away from the body.

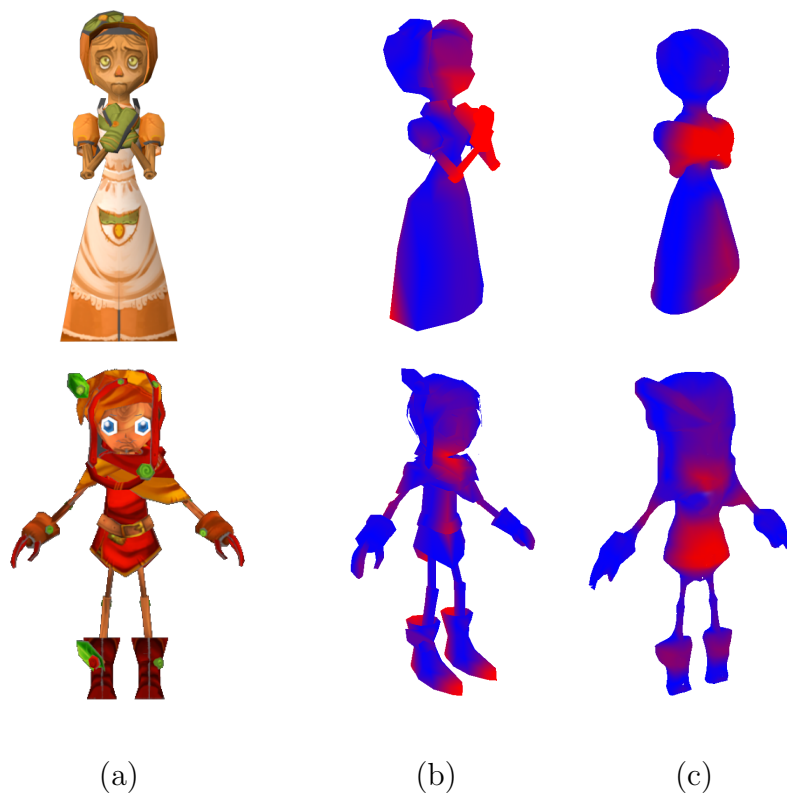


Figure 6.42: Typical examples of the models and generated images used to create the results in Table 6.3. The top row shows the model *Lin*, and the bottom row *Lin's Mum*. Column (a) is the re-projected source image, with rows (b) and (c) showing the surface difference metric projected into the artist generated model and the automatically generated model respectively.

Chapter VII

Conclusion

7.1 *Limitations and Future Work*

Style is difficult to quantify, and even within the limited scope of this thesis numerous assumptions were made to keep algorithmic complexity down and ensure viable results could be generated. Even so there are areas where the algorithms fail, and the techniques outlined in each chapter rarely work outside of the imposed constraints. This section outlines some of the largest assumptions and limitations, and outlines areas where future work would have the largest impact in improving the results.

The initial results show enough promise to open up the area of visual style understanding to further research. Future work could encompass better understanding of image contents and fix outlying cases in the existing methods, or it could focus upon expanding the work in this thesis to cover more properties, styles, and a wider range of input data.

When studying both Line Style and Characteristic Proportions, it was assumed that the measured metrics encompassed the most significant contribution to the overall visual style. While this is true for the datasets used, there are examples of monochrome cartoon images where the artists' style is best described by other metrics, such as the type of shading used, the position of the image within a page, or the contents of any associated text.

Additionally, greyscale and coloured images are likely to have significantly more style related properties than the measured line and Geon information. No metrics were used to differentiate types of line stroking or borders between different colours in images. Properties such as the colour palette, the lighting setup or even the original medium are candidates for representing

stylistic data. If these unmeasured properties hold more information than the measured line and Geon properties, then the style identification and manipulation methods outlined in Chapters 3 and 5 will not produce meaningful results.

To expand the style manipulation methods further and ensure they are robust when dealing with a wider range of inputs, future work should concentrate on developing a wider range of more robust metrics, such as the colour, shading, or lighting properties outlined above. Additionally, more accurate transformation could potentially be developed using techniques that better understand image content. Rescaling and reshaping lines often caused overlaps or created gaps, and similar artifacts would occur with additional metrics. Developing new methods that performed these transformations with fewer artifacts would improve the style transformation.

Methods based around Characteristic Proportions only operate at a level of intermediate complexity. This assumes that input images have a roughly consistent complexity and that intermediate level details are the most significant when measuring style properties. Prior research supports the assumption that IC-features offer the best image metrics, this does not exclude the fact that other detail or complexity levels may offer additional important details.

The existing database comparison methods could be extended to operate at more detail levels. Ideally each Geon could contain a number of children that are themselves IC-features in relationship to the parent. An example where this is important are the facial features within the head Geon of a character. The difficulty however lies in extracting and labelling the parts, because within cartoon and hand-drawn imagery these vary significantly in appearance. Future work expanding upon Characteristic Proportions would need to develop faster and more robust sub-image extraction techniques.

One of the largest visual artifacts in both the generated and rescaled 3D models is the lack of a back-facing or side-facing texture. The ability to generate a model from only one image introduces the limitation that only one texture is available for UV-mapping and therefore a significant amount of occluded texture information is lacking. Minor problems around texture seams are solved through texture expansion, however there still remain a

number of issues. The most obvious artifact is the repetition of faces or uniquely identifying details on the back of a model. A smaller problem is that encircling details such as belts or clothing seams often do not align correctly where the texture wraps from the front to the back.

There is significant room for improvement through further research in the field of texture synthesis. Using the Geon and complexity information could provide contextual cues to identify details in the texture that should be unique to the front face. Texture generation could replace these details for the back faces, or a more complex type of UV-mapping could be developed to replace the linear view projection currently used. Any improvement of the occluded textures would make a large difference to the appearance of the final model.

A second major limitation in the model generation process arises when inferring depth only from the front view and without any context. This results in areas including the feet and the face that are flattened and do not have the 3D form that would be expected.

There are two approaches future research could take to address this issue. A hybrid system could be developed using the best of rescaling and generation. A 2D image could potentially be decomposed into Geons with associated depth information extracted from a predefined database. At the current Geon level this would cope with larger details such as feet, and further detail could be implemented progressively at smaller levels.

A second approach would be implementing additional depth cues similar to the contour extraction already in use. The single-view concept artwork used in many of the examples had a shadow component that could give additional details about the depth. Self-shadowing and ambient occlusion cues could also be used in cases where the artwork is more realistic. There appears to be little existing work in this area.

Overall there are a number of limitations with the content manipulation and creation methods outlined in this thesis. It is anticipated that future work could address these and improve the results without adding additional user or input requirements. However, the difficulty of extracting information from such constrained input means that diminishing returns are seen for each new technique implemented. This thesis outlines algorithms that

demonstrate a tradeoff between accuracy and complexity.

7.2 Conclusion

This thesis explores the areas of visual and artistic style in the context of computer science, using these traditionally intangible artistic properties to enhance existing content manipulation algorithms and develop new content creation methods. Line style research showed that appearance differed between artists, and this property could be both measured and adjusted. The proportions of image subsections and their relationships also differed between artists, leading to an understanding of Characteristic Proportions that allowed the isolation of artistically significant properties. This enabled a deeper level of style transfer and categorisation.

A vectorization technique was developed specifically for line based cartoon content, leading to a line scaling algorithm that preserves image form. Chapter 6 then outlined two different approaches to 3D content creation using 2D image properties. Both methods are suitable for use in rapid prototyping and content generation pipelines with limited format input data, and are designed to run without requiring human judgement.

What we perceive as style is a hugely complex artifact of the human visual system combined with lifelong environmental experience. While the results in this paper begin to merge standard computer graphics methods with a more implicit understanding of object appearance, this is only currently possible with a constrained input range and numerous underlying assumptions. The results are a tradeoff between algorithmic complexity, breadth of input options, and human oversight.

Chapter 3 explored the relationship between line properties and artists. Prior research was successfully reproduced, and further statistics developed and evaluated. Moreover, these distributions were found to be representative of the artistic style and not of the content of the image, meaning that individual artists could be distinguished by the line properties alone. Testing showed that the style of an image could be changed by modifying only the line properties. Chapter 4 outlined a method to more easily extract the un-

derlying structure of vectors within an image. While there is no single correct solution for the vectorization of strokes in a cartoon image, this algorithm provides data formatted expressly toward style based algorithms.

Chapter 5 takes the idea of style manipulation further by exploring the Characteristic Proportions of images. A new method is outlined that automatically extracts an individual artist's style, which can then be applied to other drawings. The ability to classify properties as artist specific or image specific allows for a much more accurate understanding of the image style than was previously possible. A range of tests was carried out across a large body of artwork, confirming that style translation and classification is possible using these properties.

Line and content style properties are also carried into Chapter 6, where single images of character concept artwork are used to automatically create 3D models. Two automatic model creation methods were explored for use in rapid prototyping and content generation pipelines. Section 6.3 proposed a re-scaling algorithm using a pre-made base mesh, while Sections 6.4 to 6.5 discussed a reconstruction algorithm that generates models from an extracted skeleton. This type of model generation runs without requiring human judgement and produced good results for a range of different characters.

A mesh surface difference algorithm was then used to evaluate the results of the algorithms against artist created content. Both methods reflect the input concept artwork with only a small difference between the target model and the automatically generated results. The evaluation was also used to identify problem areas and fine-tune parameters in the algorithm.

Overall, this research provides style manipulation abilities that are missing from modern digital art creation pipelines, with potential applications reaching beyond the scope of video games and electronic entertainment. This thesis expands on the complicated and relatively unexplored area of artistic style, and hopefully opens up further opportunities in stylistic manipulation. It is a small first step into an unexplored area of research with a great potential benefit.

7.3 Attribution

This research would not have been possible without the support of the artists who generously let me use their work and who provided high-resolution sketches and source files. Thank you.

7.3.1 Line Art

7.3.1.1 With Permission:

JORGE CHAM . Piled Higher and Deeper. *Figure 5.2.*
www.phdcomics.com

CROW. Not Just Nicky. *Figures 5.17, 5.18, 5.22.* www.nickyitis.com

CHRIS ELIOPOULOS. Misery Loves Sherman. *Figures 4.1, 4.6, 4.7, 4.8, 4.13, 4.14, 5.1, 5.11, 5.14, 5.16, 5.17, 5.18, 5.22.*
www.miserylovessherman.com

OLIVER KNÖRZER & POWREE. Sandra and Woo. *Figures 4.11, 5.1, 5.10, 5.17, 5.18, 5.19.* www.sandraandwoo.com

THIERRY VIVIEN. YodaBlog. *Figures 3.1, 3.7, 4.16, 4.17, 5.17, 5.18, 5.20.* www.yodablog.net, www.thierryvivien.com

7.3.1.2 Fair Use:

DANIELLE CORSETTO. Girls With Slingshots. *Figures 3.1, 3.7, 3.8.*
www.girlswithslingshots.com

CHARLES M SCHULZ. Peanuts. *Figure 5.2.*

BILL WATTERSON. Calvin and Hobbes. *Figures 5.2, 5.3.*

7.3.2 Concept Artwork

7.3.2.1 With Permission:

ASHLEIGH BARRETT. www.altaeyyr.deviantart.com *Kaolin*

ANDREA BIANCO. www.portfolio.andreabianco.eu *Consecrated, Creature Concept 2, Creature Concept 5, Drone Class 1, Ghost, Mecha Concept Sketch, Night Hiss, Pilot Suit, Starship Concept Sketch*

MERLIN CHENG (NANYANG POLYTECHNIC). merlinkun@yahoo.com *Monster Concept Art (Figures 6.16, 6.17, 6.18)*

NICHOLAS CLOISTER. www.rpgcreatures.com *Jarrazen, Dungeon Level 1 Creature, Neverenn, Agi Septhoran*

LAURA DUGGAN. shamazindustries@hotmail.com *Robot Concept Art*

ALEXEY GRISHIN. www.sephiroth-art.deviantart.com *Vampire (Figure 6.13), Swamp Imp (Figure 6.30), Red Bat (Figures 6.27, 6.28, 6.29)*

DAVID HAKOBIAN. dhconnect@yahoo.com *Creature Designs*

GARRET ARNEY-JOHNSON. www.garretaj.com *Scrapper Mechanic, Artillery, Ninja Bot, Rat Rod, SR81 Pilot, Tank Unit, The Empirors War Throne*

EVGENIY KHE. www.evgeniykhe.deviantart.com *Heavy Robot, Very Heavy Robot, Scetch, '3'*

KONSTANTIN MAJSTRENKO. skate_boarding@mail.ru *Sketch Cyborg (Figure 6.1), Cute Girl (Figure 6.15), Solid Dog (Figure 6.34), Ann, Concept 2, Concept 3, Cyber Bear, Cyber Soldier, Contest Metro, Goblins Concept, ID, Orc 2, QWERTY, Robot 1, Speed 77*

CHRISTOPHER ONCIU. www.cxartist.deviantart.com *Monster Mania (Figures 6.24, 6.31, 6.32, 6.33), Blue Bagonfo, Creature Character Design Sheet, Drake Monster, Earth Golem, Land Dragon, Rihorix*

JIANRAN PAN. www.ichitakaseto.deviantart.com *Walker Concept, Alien Crawler Bot Concept*

PROG WANG. www.progv.deviantart.com *4 Feet Mech, Anti Infantry Robot, Assault Mecha, Cute Face, Daily Mech Paintings, Daily Practice, Dinosaur Mech, Enter the Mech, Hummer, Just a Mech, Main Battle Robot, Sketchs, Sniper Type, Still a Mech, Tactical Guitar, Weapon Design*

TIEREIGHT. www.tiereight.deviantart.com *Man Reference, Guardian - 237, Monster Concepts Batch 4, Trojan Leader, UEC President, UEC Admiral*

ANDY / HALCYONBRUSH. *Halo Cryptum Merse Creature*

7.3.2.2 *Licensed under Creative Commons:*

BLENDER FOUNDATION. Sintel Production Files, CC-BY. *Ishtarrians Other (Figures 6.12, 6.35, 6.36), Dragon Concept Art (Figure 6.37), Full Sintel Concept, Sintel Style.* www.sintel.org

7.3.2.3 *Reproduced under Fair Use:*

SANGJUN LEE. . *Figure 6.25.* <http://www.sangjunart.com/>

LUIGI LUCARELLI. *Leo's Forest. Figures 6.3, 6.4, 6.10.*

7.3.3 *3D Models*

7.3.3.1 *With Permission:*

ANNIE MENESES & GAVIN MCDOWELL. www.royalsharkart.com *Lin the Squire (Figures 6.11, 6.42), Lin's Mum (Figure 6.42)*

NICK KUIJTEN. www.pyroxene.deviantart.com *Mech Armor Suit, Akemi Homura*

7.3.3.2 *Royalty Free:*

TUCHO FERNANDEZ CALO. *Archer.* . www.artbytucho.com

DAVE GIBBONS. Slug. . www.theqiwoman.com

MICKAEL LEBIHAN. Gaston Lagaffe. .

TRUONG. Laevar. *Figures 6.40, 6.41.*

Cyborg Ninja. *Figures 6.38, 6.39*

Uncredited. *Psycho-Lops, Raccoon, Chinaman, Tsucora Quori, Demon, Humpty, Amy, Boy, Old Man, Sephiroth, Squall, Jabbinello*

7.3.3.3 Creative Commons:

CLINT BELLANGER. Human Base Model BY-CC-SA. *Figures 6.2, 6.5, 6.7, 6.8, 6.9.* www.clintbellanger.net

GORD GOODWIN. Tom Hanks Model BY-CC-SA. . www.gord-goodwin.blogspot.com

TIZIANA LONI. Undead BY-CC. . www.unbruco.it

References

- [1] S.o.e. player studio. <https://player-studio.soe.com/>, 2013.
- [2] Spore creepy & cute parts pack. <http://www.ea.com/uk/spore-creepy-n-cute>, 2013.
- [3] Topology. <http://web.archive.org/web/20131031035644/http://livyatan3d.wordpress.com/2012/11/10/topology/>, 2013.
- [4] AN, F., CAI, X., AND SOWMYA, A. Automatic 2.5 d cartoon modelling. In *Image and Vision Computing New Zealand* (2011), vol. 20, pp. 149–154.
- [5] AN, F., CAI, X., AND SOWMYA, A. Perceptual evaluation of automatic 2.5d cartoon modelling. In *Knowledge Management and Acquisition for Intelligent Systems*, D. Richards and B. Kang, Eds., vol. 7457 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 28–42.
- [6] ANDERSON, E. F. *Real-time character animation for computer games*. PhD thesis, The National Centre for Computer Animation, 2001.
- [7] ANDRE, A., SAITO, S., AND NAKAJIMA, M. Single-view sketch based surface modeling. *IEICE Transactions* (2009), 1304–1311.
- [8] ANH, N. T. L., KIM, Y.-C., AND LEE, G.-S. Morphological gradient applied to new active contour model for color image segmentation. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication* (2012), ACM, p. 21.
- [9] ANKRUM, D. R. Viewing distance at computer workstations. *Workplace Ergonomics* 2, 5 (1996), 10–13.

- [10] ARCELLI, C., AND DI BAJA, G. S. Euclidean skeleton via centre-of-maximal-disc extraction. *Image and Vision Computing* 11, 3 (1993), 163 – 173.
- [11] ASHBY, F. G. *Multidimensional models of perception and cognition*. Psychology Press, 1992.
- [12] AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. Skeleton extraction by mesh contraction. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 44.
- [13] AVIDAN, S., AND SHAMIR, A. Seam carving for content-aware image resizing. *ACM Trans. Graph.* 26, 3 (July 2007).
- [14] BAE, S.-H., BALAKRISHNAN, R., AND SINGH, K. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (2008), ACM, pp. 151–160.
- [15] BAI, X., LATECKI, L., AND LIU, W.-Y. Skeleton pruning by contour partitioning with discrete curve evolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29, 3 (march 2007), 449–462.
- [16] BAI, X., LATECKI, L. J., AND LIU, W.-Y. Skeleton pruning by contour partitioning with discrete curve evolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29, 3 (2007), 449–462.
- [17] BARAN, I., AND POPOVIĆ, J. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.* 26, 3 (July 2007).
- [18] BATTIATO, S., GALLO, G., AND MESSINA, G. Svg rendering of real images using data dependent triangulation. In *Proceedings of the 20th spring conference on Computer graphics* (2004), ACM, pp. 185–192.

- [19] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.
- [20] BECK, J. From the ashes of mythos: The art of torchlight. http://www.gamasutra.com/view/feature/132512/from_the_ashes_of_mythos_the_art_.php, sep 2009.
- [21] BELONGIE, S., MALIK, J., AND PUZICHA, J. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24, 4 (apr 2002), 509–522.
- [22] BHIWANI, R., KHAN, M., AND AGARWAL, S. Texture based pattern classification. *Int J Comput Appl* (2010), 0975–8887.
- [23] BIEDERMAN, I. Recognition-by-components: a theory of human image understanding. *Psychological review* 94, 2 (1987), 115.
- [24] BINFORD, T. O. Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control (Miami, FL)* (1971).
- [25] BLAKE, A., AND ISARD, M. *Active shape models*. Springer, 1998.
- [26] BLENDER FOUNDATION. Mesh loop select. `editmesh_mods.c` in Blender 2.58a, Nov 2012. Line 1854.
- [27] BLUM, H. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form* (1967), 362–380.
- [28] BOROSÁN, P., JIN, M., DECARLO, D., GINGOLD, Y., AND NEALEN, A. Rigmesh: automatic rigging for part-based shape modeling and deformation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 198.
- [29] BOXER, S. From a germ of an idea to the spore of a franchise. <http://www.theguardian.com/technology/2006/aug/03/pcgames.games>, aug 2006.

- [30] BRAND, M., AND HERTZMANN, A. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192.
- [31] BRAZIL, E. V., MACEDO, I., SOUSA, M. C., DE FIGUEIREDO, L. H., AND VELHO, L. Sketching variational hermite-rbf implicits. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium* (Aire-la-Ville, Switzerland, Switzerland, 2010), SBIM '10, Eurographics Association, pp. 1–8.
- [32] BRUBAKER, J. How to color a comic : Part 1. <http://www.remindblog.com/2009/11/23/how-to-color-a-comic-part-1/>, nov 2009.
- [33] BUCHANAN, P., DOGGETT, M., AND MUKUNDAN, R. Structural vectorization of raster images. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand* (2012), ACM, pp. 319–324.
- [34] BUCHANAN, P., DOGGETT, M., AND MUKUNDAN, R. Transferring characteristic proportions to modify the artistic style of cartoons. In *Proceedings of the 30th Computer Graphics International Conference* (jun 2012).
- [35] BUCHANAN, P., MUKUNDAN, R., AND DOGGETT, M. Automatic single-view character model reconstruction. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2013), SBIM '13, ACM, pp. 5–14.
- [36] CELEBI, M. E. Effective initialization of k-means for color quantization. In *Image Processing (ICIP), 2009 16th IEEE International Conference on* (2009), IEEE, pp. 1649–1652.
- [37] CHEN, T., ZHU, Z., SHAMIR, A., HU, S.-M., AND COHEN-OR, S.

- D. 3-sweep: extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 195.
- [38] CHENG, M.-M. Curve structure extraction for cartoon images. In *Proceedings of The 5th Joint Conference on Harmonious Human Machine Environment* (2009), pp. 13–25.
- [39] CHONG, H. Y., GORTLER, S. J., AND ZICKLER, T. A perception-based color space for illumination-invariant image processing. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 61.
- [40] CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum* 17, 2 (1998), 167–174.
- [41] CLARKSON, K., TARJAN, R., AND WYK, C. A fast las vegas algorithm for triangulating a simple polygon. *Discrete & Computational Geometry* 4 (1989), 423–432.
- [42] COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. Wang tiles for image and texture generation. *ACM Trans. Graph.* 22, 3 (July 2003), 287–294.
- [43] COOK, M. T., AND AGAH, A. A survey of sketch-based 3-d modeling techniques. *Interact. Comput.* 21, 3 (July 2009), 201–211.
- [44] COOTES, T. F., EDWARDS, G. J., AND TAYLOR, C. J. Active appearance models. In *Computer Vision ECCV98*. Springer, 1998, pp. 484–498.
- [45] COOTES, T. F., AND TAYLOR, C. J. Active shape modelssmart snakes. In *BMVC92*. Springer, 1992, pp. 266–275.
- [46] COOTES, T. F., TAYLOR, C. J., COOPER, D. H., AND GRAHAM, J. Active shape models-their training and application. *Computer vision and image understanding* 61, 1 (1995), 38–59.

- [47] CORNEA, N. D., SILVER, D., AND MIN, P. Curve-skeleton properties, applications, and algorithms. *Visualization and Computer Graphics, IEEE Transactions on* 13, 3 (2007), 530–548.
- [48] COVELIERS, W. Postmortem : American McGee’s Grimm. Gamasutra Feature Article. http://www.gamasutra.com/view/feature/3910/postmortem_american_mcgees_grimm.php, jan 2009.
- [49] CROWLE, R. The making of little big planet. In *Computer Arts Magazine* (jan 2010), vol. 132, Computer Arts UK.
- [50] CYGANEK, B., AND SIEBERT, J. P. *An introduction to 3D computer vision techniques and algorithms*. John Wiley & Sons, 2011.
- [51] DASIGI, P., AND C.V.JAWAHAR. Efficient graph-based image matching for recognition and retrieval. In *Proceedings of National Conference on Computer Vision, Pattern recognition, Image Processing and Graphics (NCVPRIPG-2008)* (jan 2008).
- [52] DE BEECK, H. O., WAGEMANS, J., AND VOGELS, R. The effect of category learning on the representation of shape: dimensions can be biased but not differentiated. *Journal of Experimental Psychology: General* 132, 4 (2003), 491.
- [53] DECARLO, D., AND FINKELSTEIN, A. Line Drawings from 3D Models. SIGGRAPH, Course 7, 2005.
- [54] DECARLO, D., AND STONE, M. Visual explanations. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (2010), ACM, pp. 173–178.
- [55] DORI, D., AND LIU, W. Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* 21 (March 1999), 202–215.

- [56] DOUGHERTY, E. *An introduction to morphological image processing*. Tutorial texts in optical engineering. SPIE Optical Engineering Press, 1992.
- [57] EBERLY, D. Triangulation by ear clipping. *Magic Software, Inc* (2002).
- [58] ELLIMAN, D. A really useful vectorization algorithm. In *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances* (2000), GREC '99, pp. 19–27.
- [59] FANG, F., AND LEE, Y. 3d reconstruction of polyhedral objects from single perspective projections using cubic corner. *3D Research 3* (2012), 1–8.
- [60] FISCHER, S., LIENHART, R., AND EFFELSBERG, W. Automatic recognition of film genres. In *Proceedings of the third ACM international conference on Multimedia* (1995), ACM, pp. 295–304.
- [61] FISCHLER, M. A., AND BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June 1981), 381–395.
- [62] FOUNDATION, B. Armature envelopes. <http://www.blender.org/development/release-logs/blender-240/armature-envelopes/>, December 2005.
- [63] FRANCKE, M., AND LUNDEEN, R. How valve connects art direction to gameplay. In *Microsoft GameFest* (jul 2008).
- [64] FREEMAN, W. T., TENENBAUM, J. B., AND PASZTOR, E. C. Learning style translation for the lines of a drawing. *ACM Trans. Graph.* 22 (January 2003).
- [65] FROME, A. L. *Learning Local Distance Functions for Exemplar-Based Object Recognition*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2007.

- [66] FUNCH, B. S. *The psychology of art appreciation*. Museum Tusculanum Press, 1997.
- [67] FUNKHOUSER, T., AND KAZHDAN, M. Shape-based retrieval and analysis of 3d models. In *ACM SIGGRAPH 2004 Course Notes* (2004), ACM, p. 16.
- [68] GALLEGUILLOS, C., AND BELONGIE, S. Context based object categorization: A critical survey. *Computer Vision and Image Understanding* 114, 6 (2010), 712 – 722.
- [69] GLEICHER, M. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 33–42.
- [70] GOLDSTONE, R. L. Unitization during category learning. *Journal of Experimental Psychology: Human Perception and Performance* 26, 1 (2000), 86.
- [71] GRIMM, C., AND JOSHI, P. Just drawit: a 3d sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (Aire-la-Ville, Switzerland, Switzerland, 2012), SBIM '12, Eurographics Association, pp. 121–130.
- [72] GROCHOW, K., MARTIN, S., HERTZMANN, A., AND POPOVIĆ, Z. Style-based inverse kinematics. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 522–531.
- [73] GUPTA, A. Visual-style variation as a narrative device in animated productions. Master's thesis, RMIT University, Melbourne, Australia, 2004.
- [74] HAGTVEDT, H., HAGTVEDT, R., AND PATRICK, V. M. The perception and evaluation of visual art. *Empirical Studies of the Arts* 26, 2 (2008), 197–218.

- [75] HAYASHI, T., ONAI, R., AND ABE, K. Vector image segmentation for content-based vector image retrieval. In *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on* (2007), IEEE, pp. 695–700.
- [76] HECKER, C. Structure vs. style. Game Developers Conference, Presentation. San Francisco, California., 2008.
- [77] HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. Real-time motion retargeting to highly varied user-created morphologies. In *ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 27:1–27:11.
- [78] HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 327–340.
- [79] HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. M. Curve analogies. In *Proceedings of the 13th Eurographics workshop on Rendering* (2002), pp. 233–246.
- [80] HILLIARD, K. Game freak talks pokmons first numbered sequel. <http://www.gameinformer.com/b/features/archive/2012/09/17/gamefreak-talks-pok-233-mon-s-first-numbered-sequel.aspx>, sep 2012.
- [81] HOFFMAN, D. D., AND SINGH, M. Saliency of visual parts. *Cognition* 63, 1 (1997).
- [82] HOULE, G. F., BLINOVA, K., AND SHRIDHAR, M. Handwriting stroke extraction using a new xytc transform. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on* (2001), IEEE, pp. 91–95.

- [83] HU, Y., AND NAGAO, T. Matching of characters in scene images by using local shape feature vectors. In *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on* (2003), IEEE, pp. 207–212.
- [84] HUANG, M., YANG, M., LIU, F., AND WU, E.-H. Stroke extraction in cartoon images using edge-enhanced isotropic nonlinear filter. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry* (2010), VRCAI '10, pp. 33–38.
- [85] HUMMEL, J., AND STANKIEWICZ, B. Two roles for attention in shape perception: A structural description model of visual scrutiny. *Visual Cognition* 5, 1-2 (1998), 49–79.
- [86] IGARASHI, T., AND HUGHES, J. F. Smooth meshes for sketch-based freeform modeling. In *ACM SIGGRAPH 2007 courses* (2007), ACM, p. 22.
- [87] IGARASHI, T., MATSUOKA, S., AND TANAKA, H. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2009), SIGGRAPH '99, ACM Press / Addison-Wesley Publishing Co.
- [88] IJIRI, T., OWADA, S., OKABE, M., AND IGARASHI, T. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 720–726.
- [89] ISLAM, M. T., NAHIDUZZAMAN, K. M., PENG, W. Y., AND ASHRAF, G. Learning primitive shapes in cartoon designs. *Trans. MLDM* 4, 1 (2011), 17–29.
- [90] JAIN, A. K., AND FARROKHNI, F. Unsupervised texture segmentation using gabor filters. *Pattern recognition* 24, 12 (1991), 1167–1186.

- [91] JAIN, E., SHEIKH, Y., MAHLER, M., AND HODGINS, J. Three-dimensional proxies for hand-drawn characters. *ACM Trans. Graph.* 31, 1 (Feb. 2012), 8:1–8:16.
- [92] JEONG, J., HWANG, C., AND JEON, B. An efficient method of image identification by combining image features. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication* (2009), ICUIMC '09, pp. 607–611.
- [93] JOY, G., AND XIANG, Z. Center-cut for color-image quantization. *The Visual Computer* 10 (1993), 62–66.
- [94] JUPP, J., AND GERO, J. S. Visual style: Qualitative and context-dependent categorization. *AIE EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* 20, 03 (2006), 247–266.
- [95] KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics* 31, 4 (2012), 55.
- [96] KARPENKO, O. A., AND HUGHES, J. F. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics* 25, 3 (July 2006), 589–598.
- [97] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active contour models. *International journal of computer vision* 1, 4 (1988), 321–331.
- [98] KATZ, S., AND TAL, A. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 954–961.
- [99] KEGL, B., AND KRZYZAK, A. Piecewise linear skeletonization using principal curves. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on* (2000), vol. 3, pp. 131–134.

- [100] KÉGL, B., AND KRZYŻAK, A. Piecewise linear skeletonization using principal curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24, 1 (2002), 59–74.
- [101] KEYSERS, D., DESELAERS, T., AND NEY, H. Pixel-to-pixel matching for image recognition using hungarian graph matching. In *In DAGM 2004, Pattern Recognition, 26th DAGM Symposium* (2004), pp. 154–162.
- [102] KOPF, J., AND LISCHINSKI, D. Depixelizing pixel art. In *ACM SIGGRAPH 2011 papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 99:1–99:8.
- [103] LAKSHMI, J. K., AND PUNITHAVALLI, M. A survey on skeletons in digital image processing. In *Digital Image Processing, 2009 International Conference on* (2009), IEEE, pp. 260–269.
- [104] LAM, L., LEE, S.-W., AND SUEN, C. Y. Thinning methodologies—a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence* 14, 9 (1992), 869–885.
- [105] LEVET, F., AND GRANIER, X. Improved skeleton extraction and surface generation for sketch-based modeling. In *Proceedings of Graphics Interface 2007* (2007), ACM, pp. 27–33.
- [106] LI, S., WANG, Y., AND LOHMANN, B. Texture classification using discrete multiwavelet transform. *ADVANCES IN MODELLING AND ANALYSIS-B-* 46, 1/2 (2003), 1a–14a.
- [107] LI, X., WOON, T. W., TAN, T. S., AND HUANG, Z. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), I3D '01, pp. 35–42.
- [108] LIAPIS, S., ALVERTOS, N., AND TZIRITAS, G. Maximum likelihood texture classification and bayesian texture segmentation using discrete wavelet frames. In *Digital Signal Processing Proceedings, 1997*.

- DSP 97., 1997 13th International Conference on* (1997), vol. 2, IEEE, pp. 1107–1110.
- [109] LIN, H., CHEN, W., AND WANG, G. Curve reconstruction based on an interval b-spline curve. *The Visual Computer* 21, 6 (2005), 418–427.
 - [110] LIU, P.-C., WU, F.-C., MA, W.-C., LIANG, R.-H., AND OUHYOUNG, M. Automatic animation skeleton using repulsive force field. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on* (oct. 2003), pp. 409 – 413.
 - [111] LIU, S., AND HUANG, Z. Interactive 3d modeling using only one image. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2000), VRST '00, ACM, pp. 49–54.
 - [112] LOWE, D. G. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2* (1999), ICCV '99.
 - [113] MANCAS, M., GOSSELIN, B., AND MACQ, B. Segmentation using a region-growing thresholding. In *Electronic Imaging 2005* (2005), International Society for Optics and Photonics, pp. 388–398.
 - [114] MAO, C., QIN, S. F., AND WRIGHT, D. K. Sketching-out virtual humans: from 2d storyboarding to immediate 3d character animation. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology* (New York, NY, USA, 2006), ACE '06, ACM.
 - [115] MARAGOS, P., AND SCHAFER, R. Morphological skeleton representation and coding of binary images. *Acoustics, Speech and Signal Processing, IEEE Transactions on* 34, 5 (1986), 1228–1244.
 - [116] MARCHESE, S. Inside a video game developer studio. <http://news.bbc.co.uk/2/hi/technology/6472467.stm>, mar 2007.

- [117] MARK DE BERG, MARC VAN KREVELD, M. O., AND SCHWARZKOPF, O. *Computational geometry (2nd revised ed.) : Algorithms and Applications*. Springer-Verlag, Berlin New York, 2000, ch. Chapter 3: Polygon Triangulation.
- [118] MAYYA, N., AND RAJAN, V. Voronoi diagrams of polygons: A framework for shape representation. *Journal of Mathematical Imaging and Vision* 6 (1996), 355–378.
- [119] MAYYA, N., AND RAJAN, V. Voronoi diagrams of polygons: A framework for shape representation. *Journal of Mathematical Imaging and Vision* 6, 4 (1996), 355–378.
- [120] MEL, B. W. Seemore: combining color, shape, and texture histogramming in a neurally inspired approach to visual object recognition. *Neural computation* 9, 4 (1997), 777–804.
- [121] MESTETSKII, L. Fat curves and representation of planar figures. *Computers & Graphics* 24, 1 (2000), 9–21.
- [122] MI, X., DECARLO, D., AND STONE, M. Abstraction of 2d shapes in terms of parts. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (2009), NPAR '09, pp. 15–24.
- [123] MITANI, J., SUZUKI, H., AND KIMURA, F. 3d sketch: Sketch-based model reconstruction and rendering. In *Workshop on Geometric Modeling'00* (2000), pp. 85–98.
- [124] MITCHELL, J. Connecting visuals to gameplay at valve. In *Montreal International Game Summit* (nov 2008).
- [125] MOHR, A., AND GLEICHER, M. Building efficient, accurate character skins from examples. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 562–568.

- [126] MONTERO, A. S., AND LANG, J. Skeleton pruning by contour approximation and the integer medial axis transform. *Computers & Graphics* 36, 5 (2012), 477 – 487. *Shape Modeling International (SMI) Conference 2012*.
- [127] N, A., SANTA-CRUZ, D., AND EBRAHIMI, T. Mesh: measuring errors between surfaces using the hausdorff distance. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on* (2002), vol. 1, pp. 705–708.
- [128] NASRI, A., KARAM, W. B., AND SAMAVATI, F. Sketch-based subdivision models. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2009), ACM, pp. 53–60.
- [129] NAYA, F., CONESA, J., CONTERO, M., COMPANY, P., AND JORGE, J. Smart sketch system for 3d reconstruction based modeling. In *Lecture Notes in Computer Science* (2003), pp. 58–68.
- [130] NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. Fiber-mesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.* 26, 3 (July 2007).
- [131] NGUYEN, V., MARTINELLI, A., TOMATIS, N., AND SIEGWART, R. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on* (2005), IEEE, pp. 1929–1934.
- [132] NGUYEN, H. M., W. B. C. D. P., AND LUTTEROTH, C. Poisson blended exemplar-based texture completion. In *Thirty-Seventh Australasian Computer Science Conference (ACSC 2014)* (Auckland, New Zealand, 2014), B. Thomas and D. Parry, Eds., vol. 147 of *CRPIT*, ACS, pp. 99–104.
- [133] NORIS, G., HORNUNG, A., SUMNER, R. W., SIMMONS, M., AND

- GROSS, M. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG)* 32, 1 (2013), 4.
- [134] OLSEN, L., AND SAMAVATI, F. F. Image-assisted modeling from sketches. In *Proceedings of Graphics Interface 2010* (2010), Canadian Information Processing Society, pp. 225–232.
- [135] OLSEN, L., AND SAMAVATI, F. F. Stroke extraction and classification for mesh inflation. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium* (2010), Eurographics Association, pp. 9–16.
- [136] OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85–103.
- [137] ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. Diffusion curves: a vector representation for smooth-shaded images. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 92.
- [138] OSADA, R., FUNKHOUSER, T., CHAZELLE, B., AND DOBKIN, D. Shape distributions. *ACM Transactions on Graphics (TOG)* 21, 4 (2002), 807–832.
- [139] PAGET, R., AND LONGSTAFF, D. Texture synthesis via a noncausal nonparametric multiscale Markov random field. *IEEE Transactions on Image Processing* 1998 7, 6 (1998).
- [140] PAL, N. R., AND PAL, S. K. A review on image segmentation techniques. *Pattern Recognition* 26, 9 (1993), 1277 – 1294.
- [141] PALMERI, T. J., AND GAUTHIER, I. Visual object understanding. *Nature Reviews Neuroscience* 5, 4 (April 2004).

- [142] PAN, J., YANG, X., XIE, X., WILLIS, P., AND ZHANG, J. J. Automatic rigging for animation characters with 3d silhouette. *Computer Animation and Virtual Worlds* 20, 2-3 (2009), 121–131.
- [143] PANTUWONG, N., AND SUGIMOTO, M. Skeleton-growing: a vector-field-based 3d curve-skeleton extraction algorithm. In *ACM SIGGRAPH ASIA 2010 Sketches* (New York, NY, USA, 2010), SA '10, ACM, pp. 6:1–6:2.
- [144] PANTUWONG, N., AND SUGIMOTO, M. A novel template-based automatic rigging algorithm for articulated-character animation. *Computer Animation and Virtual Worlds* 23, 2 (2012), 125–141.
- [145] PARKER, D., AND DEREGOWSKI, J. B. *Perception and artistic style*. Elsevier, 1991.
- [146] PATTON, P. Why second life fails. <http://pennycow.blogspot.com/2013/02/why-second-life-fails.html>, feb 2013.
- [147] PRASAD, L. Morphological analysis of shapes. *CNLS newsletter* 139, 1 (1997), 1997–07.
- [148] SCHYNS, P., GOLDSTONE, R., AND THIBAUT, J. The development of features in object concepts. *Behavioral and Brain Sciences* 21, 1 (1998), 1–17.
- [149] SEBE, N., AND LEW, M. S. Wavelet based texture classification. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on* (2000), vol. 3, IEEE, pp. 947–950.
- [150] SELINGER, P. Potrace: a polygon-based tracing algorithm. 2003.
- [151] SHAMIR, L., MACURA, T., ORLOV, N. D., ECKLEY, M., AND GOLDBERG, I. G. Impressionism, expressionism, surrealism : Automated recognition of painters and schools of art. *ACM Trans. Appl. Percept* (2010).

- [152] SHAPIRO, A., CAO, Y., AND FALOUTSOS, P. Style components. In *Proceedings of Graphics Interface 2006* (2006), Canadian Information Processing Society, pp. 33–39.
- [153] SHEFFIELD, B. The art of the form: Hyung-tae kim speaks. http://www.gamasutra.com/view/feature/4310/the_art_of_the_form_hyungtae_kim_.php, mar 2010.
- [154] SHUGRINA, M., BETKE, M., AND COLLOMOSSE, J. Empathic painting: interactive stylization through observed emotional state. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2006), ACM, pp. 87–96.
- [155] SIVALINGAMAIAH, M., AND REDDY, B. V. Texture segmentation using multichannel gabor filtering. *IOSR Journal of Electronics and Communication Engineering (IOSRJECE)* 2 (2012), 22–26.
- [156] STEGER, C. Subpixel-precise extraction of lines and edges. *International Archives of Photogrammetry and Remote Sensing* 33, 3 (2000), 141–156.
- [157] STEPIN, M. hq4x. <http://web.archive.org/web/20070717064839/www.hiend3d.com/hq4x.html>, 2003.
- [158] SUH, J. W., AND KIM, J. H. Stroke extraction from gray-scale character image. *Proceedings of 5th IWFHR* (1996), 593–598.
- [159] SUH, Y. S. Reconstructing polyhedral swept volumes from a single-view sketch. In *IRI’06* (2006), pp. 585–588.
- [160] TAGLIASACCHI, A., ZHANG, H., AND COHEN-OR, D. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph.* 28, 3 (July 2009), 71:1–71:9.
- [161] TAI, C.-L., ZHANG, H., AND FONG, J. C.-K. Prototype modeling from sketched silhouettes based on convolution surfaces. *Comput. Graph. Forum* (2004), 71–84.

- [162] TAN, P.-N., ET AL. Cluster analysis : Basic concepts and algorithms. In *Introduction to data mining* (2007), Pearson Education India, pp. 488–566.
- [163] TANG, M., LEE, M., AND KIM, Y. J. Interactive hausdorff distance computation for general polygonal models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009)* 28, 3 (2009).
- [164] TANGELDER, J., AND VELTKAMP, R. A survey of content based 3d shape retrieval methods. *Multimedia Tools and Applications* 39, 3 (2008), 441–471.
- [165] TENENBAUM, J. B., AND FREEMAN, W. T. Separating style and content with bilinear models. *NEURAL COMPUTATION* (2000), 1247–1283.
- [166] THEOBALT, C., RÖSSL, C., DE AGUIAR, E., AND SEIDEL, H.-P. Animation collage. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), SCA '07, pp. 271–280.
- [167] TORRESANI, L., KOLMOGOROV, V., AND ROTHER, C. Feature correspondence via graph matching: Models and global optimization. In *Proceedings of the 10th European Conference on Computer Vision: Part II* (Berlin, Heidelberg, 2008), ECCV '08, Springer-Verlag, pp. 596–609.
- [168] TSAI, M., AND LU, T. A rapid mesh fusion method to create 3d virtual characters in games. In *Computer Sciences and Convergence Information Technology, 2009. ICCIT '09. Fourth International Conference on* (2009), pp. 393–398.
- [169] ULLMAN, S., VIDAL-NAQUET, M., AND SALI, E. Visual features of intermediate complexity and their use in classification. *Nature neuroscience* 5, 7 (jul 2002).

- [170] UNUMA, M., ANJYO, K., AND TAKEUCHI, R. Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 91–96.
- [171] VALVE CORPORATION. Dota 2 charater art guide. <http://media.steampowered.com/apps/dota2/workshop/Dota2CharacterArtGuide.pdf>, 2012.
- [172] VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., WARD, B., AND TORR, P. H. Videotrace: rapid interactive scene modelling from video. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 86.
- [173] VENTURA, J., DIVERDI, S., AND HÖLLERER, T. A sketch-based interface for photo pop-up. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling* (2009), ACM, pp. 21–28.
- [174] VIVEK KWATRA, ARNO SCHDL, I. E. G. T., AND BOBICK, A. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003 22*, 3 (July 2003), 277–286.
- [175] WANG, C. C. L., WANG, Y., CHANG, T. K. K., AND YUEN, M. M. F. Virtual human modeling from photographs for garment industry. *Computer-Aided Design 35*, 6 (2003), 577–589.
- [176] WAYNER, P. Identification of artistic styles using a local statistical metric. In *Proceedings of IEEE Artificial Intelligence Applications* (1991), pp. 110–113.
- [177] WEI, L.-Y. Deterministic texture analysis and synthesis using tree structure vector quantization. In *Proceedings of the XII Brazilian Symposium on Computer Graphics and Image Processing* (Washington,

- DC, USA, 1999), SIBGRAPI '99, IEEE Computer Society, pp. 207–214.
- [178] WEI, L.-Y., AND LEVOY, M. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 479–488.
 - [179] WELSH, O. Spore first impressions. <http://www.eurogamer.net/articles/spore-first-impressions>, feb 2008.
 - [180] WILLCOCKS, C. G., AND LI, F. W. B. Feature-varying skeletonization - intuitive control over the target feature size and output skeleton topology. *The Visual Computer* 28, 6-8 (2012), 775–785.
 - [181] WILLIAMS, S. G. D., AND GREEN, R. Mr painting robot.
 - [182] WILLIAMSON, J. hq4x. <http://web.archive.org/web/20131031040128/http://cgcookie.com/blender/2012/11/28/why-topology-matters-modeling/>, 2013.
 - [183] WISKOTT, L., FELLOUS, J.-M., KRUGER, N., AND VON DER MALS-BURG, C. Face recognition by elastic bunch graph matching. In *Image Processing, 1997. Proceedings., International Conference on* (1997), vol. 1, pp. 129–132.
 - [184] WU, K., AND LEVINE, M. D. Recovering parametric geons from multiview range data. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on* (1994), IEEE, pp. 159–166.
 - [185] XIA, T., LIAO, B., AND YU, Y. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 115:1–115:10.

- [186] XIONG, S.-S., AND ZHOU, Z.-Y. A complex nonlinear exponential autoregressive model approach to shape recognition using neural networks. *Instrumentation and Measurement, IEEE Transactions on* 49, 6 (2000), 1298–1304.
- [187] YANG, C., SHARON, D., AND VAN DE PANNE, M. Sketch-based modeling of parameterized objects. In *ACM SIGGRAPH 2005 Sketches* (New York, NY, USA, 2005), SIGGRAPH '05, ACM.
- [188] YANG, R., AND WÜNSCHE, B. C. Life-sketch: a framework for sketch-based modelling and animation of 3d objects. In *Proceedings of the Eleventh Australasian Conference on User Interface-Volume 106* (2010), Australian Computer Society, Inc., pp. 61–70.
- [189] ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. Sketch: an interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 163–170.
- [190] ZHANG, H., FRITTS, J. E., AND GOLDMAN, S. A. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding* 110, 2 (2008), 260 – 280.
- [191] ZHANG, L., HUANG, H., AND FU, H. Excol: An extract-and-complete layering approach to cartoon animation reusing. *IEEE Transactions on Visualization and Computer Graphics* 18, 7 (July 2012), 1156–1169.
- [192] ZHANG, S.-H., CHEN, T., ZHANG, Y.-F., HU, S.-M., AND MARTIN, R. R. Vectorizing Cartoon Animations. *IEEE Transactions on Visualization and Computer Graphics* (2009).
- [193] ZHEN, J. S., BLAGOJEVIC, R., AND PLIMMER, B. Automated labeling of ink stroke data. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (2012), Eurographics Association, pp. 67–75.