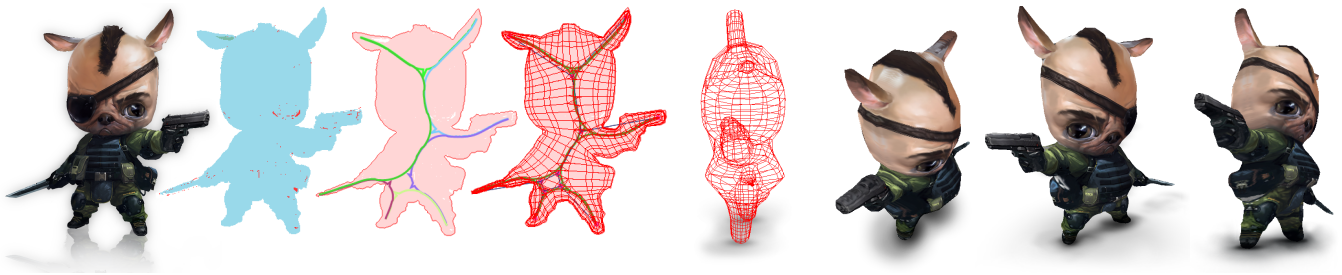
Automatic Single-View Character Model Reconstruction

Philip Buchanan[*] R. Mukundan[†] and Michael Doggett[‡]



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 1: *This paper presents a method for automatically constructing 3D meshes from a single piece of concept artwork. This figure shows how a typical 2D concept image (a) is separated from the background (b) and how 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.*

Abstract

In this paper we present a new method for automatically constructing 3D meshes from a single input image. With the increasing content demands of modern digital entertainment and the expectation of involvement from users, automatic artist-free systems are an important step in allowing user generated content and rapid game prototyping. Our system proposes a novel heuristic for the creation of a 3D mesh from a single piece of non-occluding 2D concept art. By extracting a skeleton structure, approximating the 3D orientation and analysing line curvature properties, appropriate centrepoints can be found around which to create the cross-sectional slices used to build a final triangle mesh. Our results show that a single 2D input image can be used to generate a rigged 3D low-polygon model suitable for use in realtime applications.

CR Categories: I.2.10 [Vision and Scene Understanding]: Modeling and recovery of physical attributes— [I.3.7]: Three-Dimensional Graphics and Realism

Keywords: mesh generation, model reconstruction, object geometry, character modelling, skeletonization

[*]philip.buchanan@pg.canterbury.ac.nz
[†]mukundan@canterbury.ac.nz
[‡]mike@cs.lth.se

1 Introduction

Content creation in modern entertainment is one of the most time consuming components of a project. Our research aims to reduce the workload of content creators by providing tools that allow easy integration of concept and user generated artwork into a game. Our research deals specifically with low to mid resolution 3D content that is useful for rapid prototyping or applications such as mobile games.

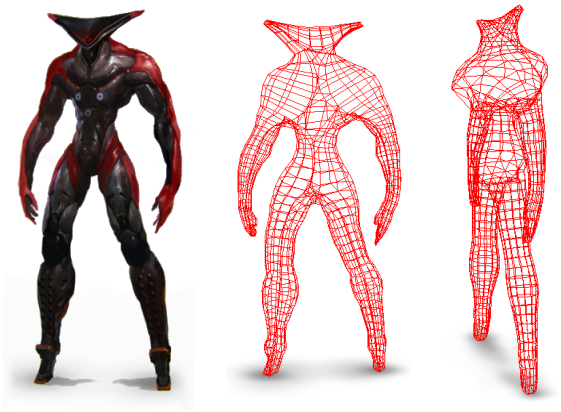
The work in this paper focuses on removing the need for user guidance during 3D reconstruction and producing a production-ready model that includes required standard properties such as UV texture coordinates and bone influence values. The main contribution of this paper is an end-to-end system with no user interaction, optimised to produce the best results across a range of input. This paper also demonstrates a 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. Additionally, this paper contains a low complexity automatic skeletonization algorithm for raster images, with an optional user-controlled complexity parameter.

The algorithm converts 2D outlines to 3D meshes using a heuristic that balances the skeletal relevancy of the mesh against reduced visual artefacts, using line style metrics to influence the 3D style of the generated mesh. 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 2. To ensure this technique remains widely applicable, only a single input image is required.

Our results show that a single 2D input image can be used to generate a rigged 3D low-polygon model suitable for use in realtime applications.

2 Previous Work

Single-View Modelling 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



(a) (b) (c)

Figure 2: *Front (b) and side (c) views show the 3D mesh automatically generated by our algorithm using a single piece of concept character art (a).*

process without contextual knowledge, and the breadth of existing research showcases many different approaches to the problem of extracting an implied 3D structure.

Two survey papers [Cook and Agah 2009] [Olsen et al. 2009] give a comprehensive look into the state of the art. Papers are classified by the type of system, the techniques used, and the tools available to artists. Practical solutions often aim to augment an artist by giving them more control over depth while sketching [Grimm and Joshi 2012] or the ability to create other 3D properties from a 2D viewport [Yotam Gingold and Zorin 2009]. Other avenues of research match templated patterns [Yang 2006] or use photos [Fang and Lee 2012] [Liu and Huang 2000] as a starting point. A large number of papers cover reconstruction of geometric shapes such as those found in technical drawings, requiring shapes with clearly defined planes and polygonal shapes [Suh 2006] [Fang and Lee 2012] [Naya et al. 2003]. Mitani et al. [2000] attempt to enlarge the potential application of this by warping constructed surfaces to better fit non-linear shapes, however use of this technique is limited in practice and does not suit the concept art style of our input imagery.

A different approach is shown in the prototying modelling research by Tai, C. et al. [2004]. Their method uses both the silhouette and a skeleton to produce an analytical convolution surface with a circular cross section. The silhouette and skeleton data are also commonly used by single-view image-based reconstruction techniques such as Interactive 3D Modelling Using Only One Image by Liu S. & Huang Z [2000] who produce a surface via Delaunay triangle subdivision. Wang et al. [2003] use the silhouette to accurately modify an existing character mesh to match a supplied image for use in retail applications such as virtual clothes fitting. While the mesh results are considerably better quality than constructed results 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.

'Smoothsketch' [2006] is a recent and effective technique that estimates 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. Full 3D character reconstruction is demonstrated by Mao et al. [2006] with the aim of speeding up a development workflow, prioritising user speed over quality.

The majority of similar work is based in artist-driven systems or interfaces, often based upon sketched lines. Single-View Sketch Based Modelling [2009] is a recent paper that uses a similar method to the one we propose, albeit to solve a different problem. Although targeted towards a user-based system, their algorithm uses only outlines without needing image content. Similarly, Olsen & Samavati [2010] propose an outline inflation method to good result, showing that it is possible to create high quality models from a single view. Their method also allows artist control over the distance function, enabling a custom cross-sectional profile. Sketch-based construction methods such as *Teddy* [1999] and *FibreMesh* [2007] often allow input from multiple views, although Andre & Saito aim for a single-view approach where the user is not expected or required to use multiple views during model construction. Each of these papers take a different approach to surface generation, with *FibreMesh* using implicit surfaces, *Teddy* using skeleton based realtime triangulation, Olsen & Samavati use a distance function, a paper by Nasri, Karam & Samavati [Nasri et al. 2009] uses subdivision surfaces, while Andre & Saito extrude appropriately sized slices along a curved centreline.

The common feature to these sketch-based interfaces and most of the aforementioned 3D model creation algorithms is their use of artistic or user guidance during the creation process. This guidance ranges from limited manipulation of variables through to specifically formatted input such as stroke data records, something that is unavailable for our input images. In this paper we propose a synthesis algorithm that runs automatically without intervention. This is achieved through additional steps and by optimising parameters to produce the best results across a range of input

For organic mesh reconstruction, solutions using swept shells generally met with success despite being susceptible to noise and relying on the input having a clear primary axis. Swept shells allow the most influence over the cross section of the model and this approach was chosen for use in this paper as it best supports our aim of forming the surface appropriately based upon stylistic traits of the input image. Extending the capabilities shown by Oslen & Samavati, our system allows for different cross-sections over the whole model.

Our system takes a single view input image and outputs a rigged and textured mesh. By focusing on characters we can also make a number of assumptions about the content and produce a higher quality output than a general solution can achieve.

3 Algorithm Overview

Our 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 this paper 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 our algorithm requires one input image with the three assumptions that: the background is not heavily textured; the character 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 palletising 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 [2003]. This outline can be seen in Figure 3 and is used for both skeleton extraction and mesh creation.

Figure 3: *Character outline generation using image palletisation and vectorization. The generated polygonal outline is shown in red.*

A skeletonization process based upon the work of Willcocks & Li [2012] is applied to the image, balancing two core iterative operators to extract a skeleton with the appropriate complexity and positioning. We modify the process 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. Our 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. Bone rigging and skinning for animation is performed on the mesh, and the original concept image 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 displayed in section 8.

4 Skeletonization

Our method for creating a 3D mesh relies on access to a skeleton that represents the underlying image structure as well as possible. Many skeletonization 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 skeletonization, having been proposed in 1967 [Blum 1967] and tweaked in various different ways up until the present [Montero and Lang 2012]. Skeletons can also be extracted by joining bi-tangent circles or maximal discs within a shape [Arcelli and di Baja 1993], or through the use of Delaunay triangulation to create a chordial axis [Prasad 1997].

The method used in this paper is a point contraction algorithm adapted from *Feature-varying Skeletonization* [Willcocks and Li 2012] and using elements from *Skeleton Extraction by Mesh Contraction* [Au et al. 2008]. Although designed for 3D meshes, this type of iterative and modularised approach allows easy adaptation of the algorithm through the addition of extra terms. This is important because overlapping components within an image don't create an outline, necessitating the addition of extra influence values that use a gradient field to better align the skeleton with the actual image contents.

4.1 Soft Skeleton

The first requirement of a character skeletonization 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 to run on 2D polygonal shapes, and then add a third term that allows image contents to influence the skeleton generation.

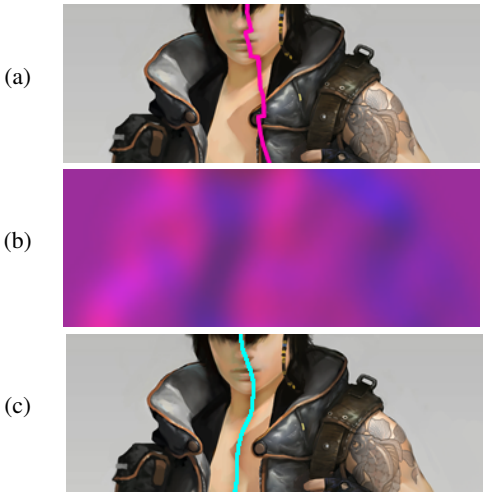


Figure 4: *Without weighting (a), the extra width of the shoulder and clothes causes discontinuities in the contracted polygon. Weighting the encroachment step by the gradient of the low frequency component of the image (b) causes the skeleton structure to be influenced not just by the outline but also by the contents of the image (c). Image (b) shows $\Delta G_{x,y}$ encoded in the red (vertical) and blue (horizontal) channels.*

The process starts by creating a polygonal bounding hull of points $P = \{p_0 ... p_i ... p_n\}$ based on the image silhouette. A smoothing step contracts this bounding hull toward its spatial centre removing local noise at the cost of increased density and reduced fidelity. The standard operator placed each point at the average of its untransformed neighbours, however 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. Therefore we introduce a simplification term into the smoothing step to create the following density-conserving operator:

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

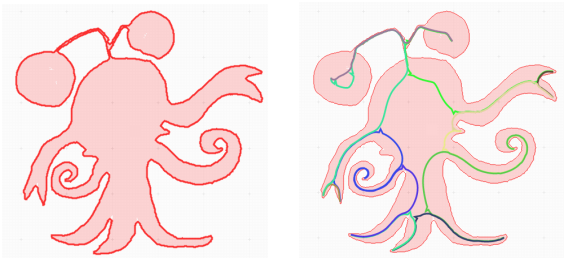
Where ω is a threshould distance calculated as:

$$\omega = \frac{\sum_{i=0}^{n} p_i - p_{(i+1) mod n}}{n} \tag{2}$$

and 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 centerline data before merging but after the full contraction is complete. To give the same effect, the iterative merge operator is replaced by an encroachment operator that moves all points 'inwards' along their local normal:

$$p'' = p' + \perp \left(p'_{i-1} - p'_{i+1}\right) G(p') \tag{3}$$



(a) (b)

Figure 5: *Alternating between the smooth and encroachment 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 5.*

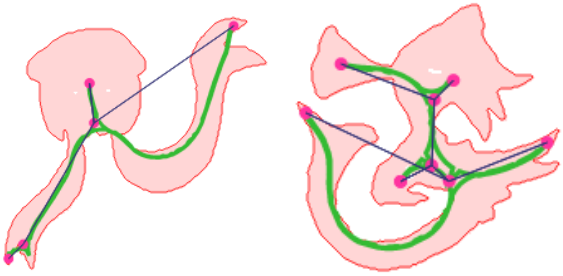


Figure 6: *Examples of the finished skeletonization process, showing data derived from the red outline. The skeleton is drawn in green, while the nodes are linked in blue*

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 encroachment step reduces spatial density and consolidates important geometric features, however it is susceptible to local noise. The term G is calculated using Equation 4 so that local image complexity is used in addition to the image outline to ameliorate noise and create the skeleton more correctly. Figure 4 shows the difference in skeleton placement when local complexity is taken into consideration. While not applicable to all input images, the addition of this step improved results on average by 1.23% when compared to the basic outline using the evaluation procedure in Section 7.

$$k(x) = e^{-\frac{x^2}{(2r/3)^2}}$$
$$G(x,y) = \sum_{\substack{-r<x'<r \\ -r<y'<r}} k(x')k(y')\left[I(x+x', y+y') - I(x,y)\right] \quad (4)$$

where r is the radius of the kernel, in our tests chosen to be one 100^{th} of the image diagonal.

Figure 6 shows the results of both skeletonization stages.

5 Mesh Construction

5.1 Establishing Orientation

Concept artwork is often drawn off-centered, such as in three-quarter view, and therefore we cannot assume a front-facing orthographic view and need to establish the initial orientation to correctly build the mesh. This can be done based upon the centerline and the ratio of the extent of the armature. If the skeleton topology is found to be symmetrical using symmetry-axis decomposition [Pantuwong and Sugimoto 2012], 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. If the character is posed, this ratio will be different for each pair. If Equation 5 holds true for the skeleton, the average extents are calculated using Equation 6 and re-orientation is performed.

$$\frac{j_i - (j_i)_{||r}}{j_{i'} - (j_{i'})_{||r}} = \frac{j_{i+\ldots n} - (j_{i+\ldots n})_{||r}}{j_{i'+\ldots n} - (j_{i'+\ldots n})_{||r}} = \ldots \quad (5)$$

$$a = \frac{1}{n}\sum_{v=0}^{n-1} j_{(i+v)}\,, \quad b = \frac{1}{n}\sum_{v=0}^{n-1} j_{(i'+v)} \quad (6)$$

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 centerline r, where r is calculated as the best-fit line for all j. $j_{||r}$ represents a projection of j onto r, while a and b are calculated to be the average extents for each side of the skeleton.

For a front-facing image we can estimate the orientation by projecting the horizontal offsets back to an implied camera with respect to the centerline. We select the distance to the camera to be 70cm, which is the recommended viewing distance from a monitor and a value we assume to influence the average perspective for digitally drawn concept artwork. 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 7.

$$\begin{aligned} \theta_1 &= tan^{-1}\left(\frac{a'}{70}\right), \quad \theta_2 = tan^{-1}\left(\frac{b'}{70}\right) \\ \beta &= tan^{-1}\left(\frac{sin(\theta_1)sin(\theta_1+\theta_2)}{sin(\theta_2) - sin(\theta_1)cos(\theta_1+\theta_2)}\right) \end{aligned} \quad (7)$$

where a' and b' are the average skeleton extents generated using Equation 6 and converted to centimetres, and β is the calculated angle of the model.

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.

5.2 Surface Generation

Meshes are created using a modified 'lathe' procedure, where limbs and body components are constructed by creating appropriately sized rings around the skeleton. To begin creating a mesh, arcs are created from the boundary in to their local centre. Equation 8 defines two possible local centres, each with different advantages.

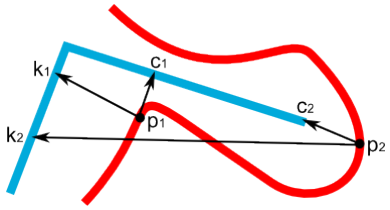


Figure 7: *This diagram shows the best and worse case placing for the arc endpoints k and c given certain points $p_{1..2}$ on the outline (red). p_1 shows a good result for k_1, whereas c_1 would create overlapping arcs perpendicular to the outline normal. p_2 creates the opposite result where c_2 is a useable local centre but k_2 would create arcs that lay outside of the outline.*

$$\begin{aligned} c_i &= \text{get closest point on } S \text{ to } p_i \\ k_i &= \text{intersection of } n_i \text{ and } S \end{aligned} \tag{8}$$

where S is the bent skeleton calculated in Section 4, n is a set of normal vectors calculated from the edges of the outline polygon p.

As can be seen in Figure 7, the difference between the arcs created by these two methods can be quite significant.

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 artefacts. The higher the noise, the less influence k should have over the final solution. In contrast, c will always provide a point that can be used but introduce banding artefacts by causing groups of consecutive points to have the same local centre.

The accuracy of k also decreases with distance and in some edge cases the normal can be almost tangential 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 increases.

$$\omega_d = \frac{||c_i - p_i||}{2\,||k_i - p_i||} \tag{9}$$

where p_i is the point on the outline.

A similar inaccuracy occurs in c when it is offset and causes the created slices to diverge too far from the outline normal. Equation 10 sets up a scaling term for this.

$$\omega_\alpha = \widehat{c_i - p_i} \cdot n_i \tag{10}$$

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 affects the choice of centrepoint is the noisiness of the curve, which is calculated using Equation 11. The more noise there is in the line, the less accurate properties based upon the line normal or tangent will be.

$$e = n_i - \sum_{k=i-10}^{i+10} n_k \tag{11}$$

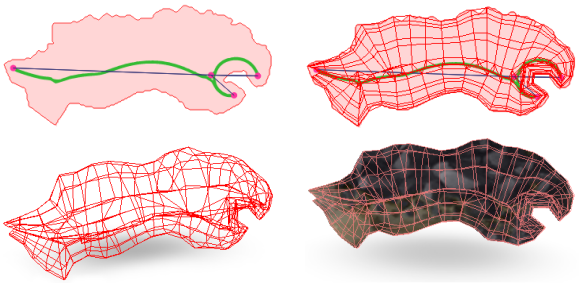


Figure 8: *This image shows the generated arcs and shells for a basic skeleton extracted from the image outline. The weighted centrepoint creates segments that are not overlapping and merge correctly at corners and ends.*

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 12.

$$p' = ce(1-\omega_d)\omega_\alpha + k(1-e)\omega_d(1-\omega_\alpha) \tag{12}$$

After calculating the best centrepoint around which to construct the slices, the mesh itself is created by extruding a cross-section along the skeleton. Creating a perfectly round, centred mesh is not always appropriate, so we need to look at the type of character and character material implied by the properties of the silhouette. This can be done on a local level by analysing the separated line segments as shown in Figure 5. One of the best profile indicators is the line curvature and the number of sharp corners in parts of the image. Equation 13 generates points that define the cross-sectional profile to loft along the skeleton. 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:

$$\begin{aligned} v_{i=-\pi\to\pi} &= \left|\left|p' - p\right|\right| (cos(i) + sin(i))s \\ &+ (1 - \frac{|i|}{\pi})(1-s) \end{aligned} \tag{13}$$

where s is between 0 and 1, and calculated thus:

$$c_t = \sum_{w=0}^{n-1} |v_w \cdot v_{w+1}|\,, \quad \rho = \frac{d}{40}$$

$$c_\mu(w) = \sum_{x=w-\rho}^{w+\rho-1} |v_x \cdot v_{x+1}| \frac{n}{c_t} \tag{14}$$

$$s = \sum_{w=0}^{n-1} \frac{|v_w \cdot v_{w+1} - c_\mu(w)|}{c_t} \tag{15}$$

where c_t and c_μ are the total and average curve calculated using v, the set of n normal vectors from line segment L. ρ is the point density calculated using the largest bound of the outline polygon d, and s is the segment sharpness which is clamped in the range $[0:1]$.

Equation 14 calculates the average bend of a line segment. . Because the curvature is calculated using an average, the size of the

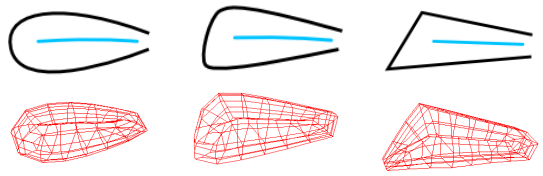


Figure 9: *Examples of mesh generation showing cross-sectional adjustment based upon line curvature and with respect to sharp corners.*

input image affects the result and the density term ρ is used to adjust for this. The best value for this term was found using the iterative evaluation in Section 7.

A straight line segment will always generate a value of 0, while a value of 1 signifies that the line is equal to the average curvature in the outline. Values higher than one are clamped. Equation 15 calculates the sharpness of a line segment. Sharp 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 lines suggest that the generated mesh should be more organic. Figure 9 shows how this works in practice.

Due to the centrepoint weighting applied in Equation 12, 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 [Mark de Berg and Schwarzkopf 2000] or Las Vegas triangulation [Clarkson et al. 1989]. In our implementation Ear Clipping [Eberly 1998] is used, and can be optimised significantly by removing the ear search stage. Sections adjacent to end points from the skeleton will always be polygonal ears and can be used to start clipping.

When compared to the worst case performance in Figure 7, shells created with the weighted centrepoint appear more natural and are free of artefacts. Figure 8 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' [Eberly 1998] 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 due to the fact visible artefacts are rare it is left unsolved.

6 Rigging and Texturing

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 [Tagliasacchi et al. 2009] [Pantuwong and Sugimoto 2010] [Pantuwong and Sugimoto 2012] [Willcocks and Li 2012] extract structures similar to our curved 2D skeletons. Other automatic skeletonization methods focus specifically on animation [Baran and Popović 2007] and deformation [Liu et al. 2003] [Katz and Tal 2003] of the mesh, and even include skeletonization within the framework of sketch-based interfaces [Borosan et al. 2012].

Any of these methods could be used to create a new 3D skeleton based upon the generated mesh, however we already have a 2D skeleton generated during the mesh creation step and this can be used to create the rigged mesh. Additional joints are created by finding splits, merges, and inflection points in the soft skeleton and connecting them according to the topology.

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. Long, non-branching, mirrored limbs that are leaf nodes to the spine (found through symmetry-axis decomposition in Section 5) 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.

The connecting 2D bones are then projected into 3D using the orientation from Equation 7 and the mesh attached to the bone structure. Two common attachment methods are vertex groups [Anderson 2001] and bone envelopes [Foundation 2005]. While our 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 16.

$$\begin{aligned} p_i &= clamp(v \perp B_i) \\ d_i &= ||v - c_i|| \\ \alpha_i &= |c_i - v|\,\dot{n} \\ \\ s_i &= \frac{d_i}{\sum_{n=0}^{4} d_n} \\ influence_i &= s_i \alpha_i \end{aligned} \tag{16}$$

where $B_{[0..3]}$ is a list of the closest 4 bones to the mesh vertex v with normal n. Four bones are used as this is the influence limit in many 3D engines.

Simple tests have shown that the projected 3D armature and the vertex weighting generated by Equation 16 can be used in a range of motions without significant artefacts. Motion retargeting [Gleicher 1998] [Hecker et al. 2008] can therefore be used to map existing animations to our generated skeleton. Figure 10 shows an example of this.

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 our algorithm. Even including adjustments made based on image content by Equation 4, large breaks or features in the image content may not be reflected correctly in the skeleton because the silhouette has the largest bearing 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.

Images conforming to our initial requirements rarely contained these problems and produced useable skeleton data. Once this information has been extracted correctly, all of the required data exists to generate the mesh.

Figure 10: *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.*

(a) (b)

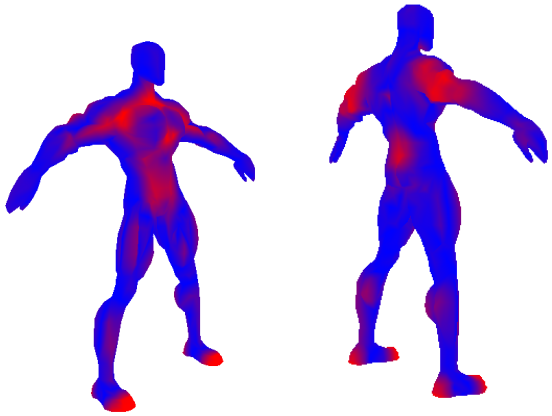
Figure 11: *Border seam expansion is used to reduce artefacts when mapping a texture (a) to the model. The red border indicates the area that is visible on the model but due to antialiasing and the influence of the background is the wrong colour. Expanding the border by 10 pixels (b) means that every mapped pixel on the model is correct.*

Texturing a model with only one source image poses numerous problems, many of which are beyond the scope of this paper. Numerous papers deal with general pattern synthesis based upon small sections of known texture [Hertzmann et al. 2001] [Cohen et al. 2003] [Paget and Longstaff 1998], or image background synthesis for photo expansion or foreground removal [Vivek Kwatra and Bobick 2003]. There is however little research into texture synthesis for occluded projection or character-specific texturing. In lieu of a better texture, we therefore use the original source image and perform a texture border expansion as seen in Figure 11. This removes the seam mapping artifacts that are usually caused by sketch lines or background colour creep from antialiasing. The generated detail gives the model better quality when seen side-on.

7 Parameter Evaluation

Developing an end-to-end system without user interaction necessitates a number of tradeoffs and the selection of 'magic numbers' such as thresholds and bias needs to be done so as to produce the best results across as wide a range of input as possible. Changes to these numbers, as well as changes to the algorithm itself can often be difficult to evaluate, and therefore a numerical evaluation was used to determine the best system setup.

The evaluation uses a surface difference metric to compare the generated results with a number of 3D models created by an artist based only upon the input image (in select cases, the input image was generated from the artist-created model). A number of approaches have been suggested for comparison of 3D models [Cignoni et al. 1998] [N et al. 2002] [Tang et al. 2009] and these share many of the same ideas. Our evaluation compares both the hausdorff distance and the averaged surface error metric outlined by Aspert, N et al. and visible in Figure 12. Parameters such as those in Equation 15 are iterated across a range and the generated results compared for 15 model sets to determine the value that produces the best result across the largest number of test cases.



(a) (b)

Figure 12: *Evaluation of a generated model against a version created by a 3D-artist using a surface error metric. Blue shows areas of high correlation, while red shows areas of low correlation compared to the generated model. In this model the hausdorff distance is 8.95% of the largest bounding dimension, while the surface error is 1.36% with σ = 3.2%*

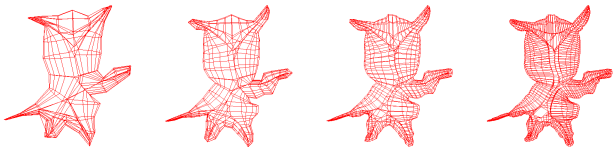


Figure 13: *The complexity of the model is determined by the number of sections generated by the algorithm, and can in turn be used to generate different LOD (Level of Detail) meshes for realtime applications.*

8 Results

Our algorithm was designed to generate a low-resolution application-ready 3D mesh from a single-view 2D concept artwork. The processes 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. Our implementation is written in javascript and the results in this section took approximately 20 seconds to generate from each concept image on a modern laptop (Intel Core i7 2.13GHz; 4GB RAM).

The process has been run with success on a number of different datasets from different artists. Figure 1 shows a typical piece of

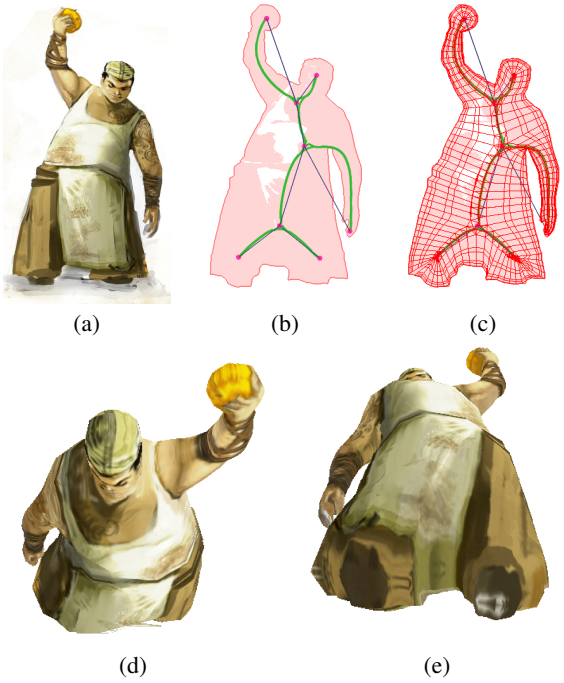


Figure 14: *Several steps illustrating mesh generation from concept artwork. The concept artwork (a) is analysed and a bent skeleton (b) used to produce swept shells (c) that form the final model. Perspective views from the top (d) and bottom (e) show that the correct body shape is produced in 3D.*

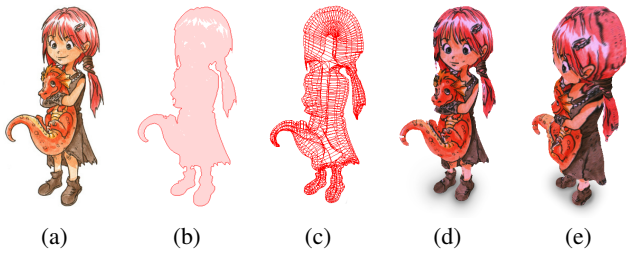


Figure 15: *Comparing the front (d) and side (e) views of a model show how the original concept art (a) can be stripped of dark borders and used to map the sides of the generated model. This figure also shows multiple characters in the input image. While the resulting mesh (c) is still a good 3D representation, the lack of segmentation (b) means that animation and repositioning is unlikely to work correctly.*

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 13 shows how the complexity of the model can be adjusted to suit the needs of a realtime game engine. Figure 15 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.

Some unique image details cause issues with mesh generation, such as large thin cloth areas. 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. Our process does not analyse image textures and cannot infer context, and therefore these areas are given depth when they should remain flat. However,

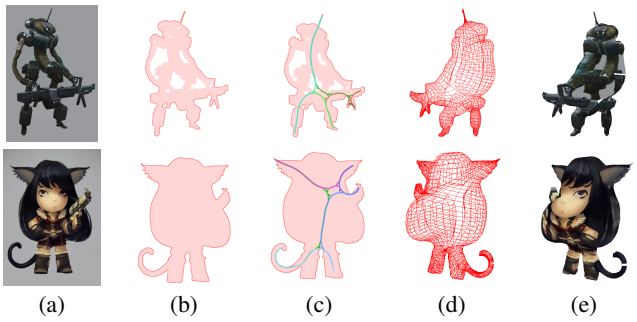


Figure 16: *Two failure cases demonstrating concept images that produce suboptimal results. Row 1 shows the results of an image containing self-occlusion and image holes. The outline (b) is only created around the outermost limbs, and therefore neither the skeleton (c) nor the generated mesh (d & e) correctly reflect the narrow body. Row 2 shows a case where the outline is technically correct (b), but is not related to the depth of the image (a) and therefore the generated mesh (d & e) is nonsensical.*

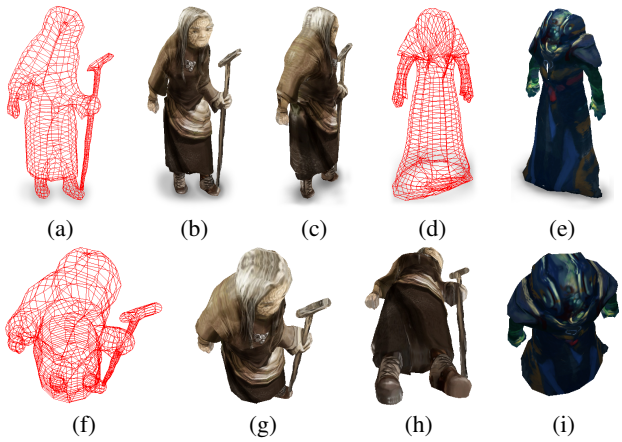


Figure 17: *Mesh generation for fabric components produces acceptable results (a, b & f) if the fabric is not a thin sheet and instead is given form by the obscured (b & c) or implied (e) underlying objects. The top (g & i) and bottom (h) views show how the depth is preserved.*

if cloth is wrapped around other objects and still has form, the resulting model is usually correct. Figure 17 shows some examples of this. Although no mesh data is generated for the implied legs, the cloth still deforms relatively intuitively during a walk cycle because the vertices are weighted and linked to multiple leg bones.

More general problems with the input images can result in failed model generation. One common issue is incorrect background segmentation, where areas of shadow or texture cause incorrect outline identification and thus the resulting process runs on invalid data. While this usually occurs with images that don't conform to the initial assumptions in Section 3, cases such as images with horizon lines or gradient backgrounds can also cause problems. Another common issue is during the mesh generation step when the outline does not correctly reflect the width or shape of the character. Two examples of this are shown in Figure 16 where the depth of the final mesh doesn't match with expectations.

Figure 14 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 our generated models, we consider these results to be a success.

9 Conclusion

This paper proposes an automatic single-view model reconstruction algorithm that is suitable for use in rapid prototyping or content generation pipelines with limited format input data. The reconstruction runs without requiring human judgement and produces acceptable results for a range of character types and shapes. With the generation of bone structure and vertexing weighting data, models are appropriate for use in low-resolution realtime applications such as video games.

Although the results appear "correct" at a glance, it is difficult to evaluate whether the generated mesh is an accurate 3D representation of the 2D image. Although this is a subjective judgement, a survey based evaluation of success could allow a comparison between machine and artist generated results. Alternatively, the surface difference metric used in Section 7 to evaluate parameter and algorithm selection could be expanded to provide comparison between multiple mesh construction methods. As outlined in Section 8, there are still many problems to be solved even with our limited input dataset, and an evaluation would be the next step in identifying the biggest graphical issues.

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. Our algorithm fills the gap between context-free general meshing algorithms and contextually driven manual creation of character meshes by an artist. It is an small first step into an area of research that has great potential benefit to the video games industry.

Acknowledgements

To the New Zealand Government Foundation for Research Science and Technology, Stickmen Studios, ELLIIT and Intel Visual Computing Institute, Saarbruecken, Germany for funding. Copyright of original images belong to the respective authors and are reproduced here with permission. Some images are creative commons.

Christopher Onciu, cxartist.deviantart.com — fig. 8, 10 & 11
Konstantin Maj, zoonoid.deviantart.com — fig. 1, 2, 4, 13, 16 & 17(e, i)
Merlin Cheng, Nanyang Polytechnic — fig. 5 & 6
Blender Foundation, www.sintel.org — fig. 3, 14, 15 & 17(a-d, f-h)

References

ANDERSON, E. F., 2001. Real-time character animation for computer games.

ANDRE, A., SAITO, S., AND NAKAJIMA, M. 2009. Single-view sketch based surface modeling. *IEICE Transactions*, 1304–1311.

ARCELLI, C., AND DI BAJA, G. S. 1993. Euclidean skeleton via centre-of-maximal-disc extraction. *Image and Vision Computing 11*, 3, 163 – 173.

AU, O. K.-C., TAI, C.-L., CHU, H.-K., COHEN-OR, D., AND LEE, T.-Y. 2008. Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27*, 3 (Aug.), 44:1–44:10.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3 (July).

BLUM, H. 1967. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*, 362–380.

BOROSAN, P., JIN, M., DECARLO, D., GINGOLD, Y., AND NEALEN, A. 2012. RigMesh: Automatic rigging for part-based shape modeling and deformation. *ACM Transactions on Graphics (TOG) 31*, 6 (Nov.), 198:1–198:9.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2, 167–174.

CLARKSON, K., TARJAN, R., AND WYK, C. 1989. A fast las vegas algorithm for triangulating a simple polygon. *Discrete & Computational Geometry 4*, 423–432.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3 (July), 287–294.

COOK, M. T., AND AGAH, A. 2009. A survey of sketch-based 3-d modeling techniques. *Interact. Comput. 21*, 3 (July), 201–211.

EBERLY, D. 1998. Triangulation by ear clipping.

FANG, F., AND LEE, Y. 2012. 3d reconstruction of polyhedral objects from single perspective projections using cubic corner. *3D Research 3*, 1–8.

FOUNDATION, B., 2005. Armature envelopes. `http://www.blender.org/development/release-logs/blender-240/armature-envelopes/`, December.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 33–42.

GRIMM, C., AND JOSHI, P. 2012. Just drawit: a 3d sketching system. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SBIM '12, 121–130.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. In *ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, SIGGRAPH '08, 27:1–27:11.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 327–340.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. 409–416.

KARPENKO, O. A., AND HUGHES, J. F. 2006. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics 25*, 3 (July), 589–598.

KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, SIGGRAPH '03, 954–961.

LIU, S., AND HUANG, Z. 2000. Interactive 3d modeling using only one image. In *Proceedings of the ACM symposium on Virtual*

reality software and technology, ACM, New York, NY, USA, VRST '00, 49–54.

LIU, P.-C., WU, F.-C., MA, W.-C., LIANG, R.-H., AND OUHYOUNG, M. 2003. Automatic animation skeleton using repulsive force field. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, 409 – 413.

MAO, C., QIN, S. F., AND WRIGHT, D. K. 2006. 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*, ACM, New York, NY, USA, ACE '06.

MARK DE BERG, MARC VAN KREVELD, M. O., AND SCHWARZKOPF, O. 2000. *Computational geometry (2nd revised ed.) : Algorithms and Applications*. Springer-Verlag, Berlin New York, ch. Chapter 3: Polygon Triangulation.

MITANI, J., SUZUKI, H., AND KIMURA, F. 2000. 3d sketch: Sketch-based model reconstruction and rendering. In *Workshop on Geometric Modeling'00*, 85–98.

MONTERO, A. S., AND LANG, J. 2012. Skeleton pruning by contour approximation and the integer medial axis transform. *Computers & Graphics 36*, 5, 477 – 487. ¡ce:title¿Shape Modeling International (SMI) Conference 2012¡/ce:title¿.

N, A., SANTA-CRUZ, D., AND EBRAHIMI, T. 2002. Mesh: measuring errors between surfaces using the hausdorff distance. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, vol. 1, 705–708 vol.1.

NASRI, A., KARAM, W. B., AND SAMAVATI, F. 2009. Sketch-based subdivision models. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM'09)*, ACM, New York, NY, USA, 53–60.

NAYA, F., CONESA, J., CONTERO, M., COMPANY, P., AND JORGE, J. 2003. Smart sketch system for 3d reconstruction based modeling. In *Lecture Notes in Computer Science*, 58–68.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3 (July).

OLSEN, L., AND SAMAVATI, F. F. 2010. Image-assisted modeling from sketches. In *Proceedings of Graphics Interface 2010*, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, GI '10, 225–232.

OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. 2009. Sketch-based modeling: A survey. *Computers & Graphics 33*, 1 (Feb), 85–103.

PAGET, R., AND LONGSTAFF, D., 1998. Texture synthesis via a noncausal nonparametric multiscale Markov random field.

PANTUWONG, N., AND SUGIMOTO, M. 2010. Skeleton-growing: a vector-field-based 3d curve-skeleton extraction algorithm. In *ACM SIGGRAPH ASIA 2010 Sketches*, ACM, New York, NY, USA, SA '10, 6:1–6:2.

PANTUWONG, N., AND SUGIMOTO, M. 2012. A novel template-based automatic rigging algorithm. *Comput. Animat. Virtual Worlds 23*, 2 (Mar.), 125–141.

PRASAD, L. 1997. Morphological analysis of shapes. *CNLS newsletter 139*, 1, 1997–07.

SELINGER, P. 2003. Potrace: a polygon-based tracing algorithm.

SUH, Y. S. 2006. Reconstructing polyhedral swept volumes from a single-view sketch. In *IRI'06*, 585–588.

TAGLIASACCHI, A., ZHANG, H., AND COHEN-OR, D. 2009. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph. 28*, 3 (July), 71:1–71:9.

TAI, C.-L., ZHANG, H., AND FONG, J. C.-K. 2004. Prototype modeling from sketched silhouettes based on convolution surfaces. *Comput. Graph. Forum*, 71–84.

TANG, M., LEE, M., AND KIM, Y. J. 2009. Interactive hausdorff distance computation for general polygonal models. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009) 28*, 3, to appear.

VIVEK KWATRA, ARNO SCHDL, I. E. G. T., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003 22*, 3 (July), 277–286.

WANG, C. C. L., WANG, Y., CHANG, T. K. K., AND YUEN, M. M. F. 2003. Virtual human modeling from photographs for garment industry. *Computer-Aided Design 35*, 6, 577–589.

WILLCOCKS, C. G., AND LI, F. W. B. 2012. Feature-varying skeletonization - intuitive control over the target feature size and output skeleton topology. *The Visual Computer 28*, 6-8, 775–785.

YANG, C., 2006. Sketch-based modeling of parameterized objects.

YOTAM GINGOLD, T. I., AND ZORIN, D. 2009. Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics (TOG) 28*, 5, 148.