EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2009)
C. Grimm and J. J. LaViola Jr. (Editors)

Modeling from Contour Drawings

Vladislav Kraevoy Alla Sheffer Michiel van de Panne †

University of British Columbia

Abstract

Occlusion contours are a natural feature to draw when tracing an object in an image or when drawing an object. We investigate the development of 3D models from multi-stroke contour drawings with the help of a 3D template model that serves as a shape prior. The template is aligned and then deformed by our method to match the drawn contours. At the heart of this process is the need to provide good correspondences between points on the contours and vertices on the model, which we pose as an optimisation problem using a hidden Markov model. An alternating correspond-and-deform process then progressively deforms the 3D template to match the image contours. We demonstrate the method on a wide range of examples.

1. Introduction

Creating 3D models from 2D input is a central problem in computer graphics. Two-dimensional input is ubiquitous and includes drawings, images, and mouse input, while 3D data is much harder to acquire. In this paper we present a new technique for using multi-stroke contour drawings to help define novel 3D geometry. We show that with the use of 3D template models and modern deformation methods, this is a tractable problem for a wide range of drawings. The knowledge for defining the new shape comes from both the drawing and the template: the drawing contains general shape, scale, and pose information that is not in the template and the template contains 3D shape information not found in the drawing, as well as shape details that may be missing in it.

Recently, several techniques have been proposed that allow users to edit 3D geometry by drawing a stroke nearby the model which then represents the desired occlusion contour for that portion of the model [KG05, NSACO05, ZNA07]. These stroke-based deformation techniques are close in spirit to our work, but are tailored for sequentially-applied, local deformations. We explore the alternative option of directly interpreting a multi-stroke contour drawing. This provides a workflow that is complementary to that of sequential stroke-based methods and mimics the sketch-then-model approach that is common practice in applications such as early stage automotive design [KS08].

Figure 1 showcases an example result produced with the proposed method. The user input consists of a number of drawn contours, traced on top of an image, that represent the desired shape of the object. Given an initial user alignment between the drawing and the template, our system then interprets the drawing by deforming the template model to fit the contours. The deformation is driven by establishing correspondences between points on the drawn contours and vertices on the model. We observe that neighboring points on a contour should generally map to nearby vertices on the model. Using this as one of the objectives in solving the correspondence problem provides a significant improvement

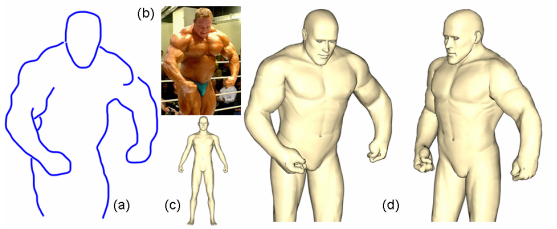


Figure 1: *A contour drawing, (a), is traced on top of an image, (b). Our algorithm corresponds and deforms the 3D template, (c), to produce the final model, (d).*

† e-mail: {vlady|sheffa|van}@cs.ubc.ca

as compared to using independently-computed correspondences. An example of this is provided in the supplemental material. Given correspondences, the model is deformed in a way that moves vertices, or rather their 2D projections, towards their corresponding contour points. The deformation is driven in an iterative correspond-and-deform fashion. This is key to the process because it minimizes the effects of less-than-ideal correspondences. In general we can initially expect to see outliers among the estimated correspondences, which are removed as the deformation proceeds. The same iterative process is also important for enabling good compromises to be made between preserving the shape of the template and matching the drawn contours.

Contributions First, we introduce the use of a hidden Markov model (HMM) [Rab89] as a representation for finding optimal correspondences between a sequence of 2D contour stroke points and a sequence of 3D template vertices. This combines local match and global continuity criteria in a principled manner. Second, we develop an iterative correspond-and-deform framework that is key to making this type of contour-based modeling work. The last contribution is that of the system itself, where we demonstrate that non-trivial contour drawings can be used as an effective geometric modeling tool.

2. Related Work

A large body of related work exists in sketch-based modeling, anchor-and-sketch driven deformation, model-based vision, and image-based modeling. For conciseness, we focus our discussion on the closest related work and representative samples of work in other areas.

The sketch-based modeling tools [IMT99, KH06, NISA07] provide some of the inspiration for our work. These techniques use underlying assumptions of smoothness in order to cope with the inherently ambiguous nature of drawings, but which also then restricts the class of objects that can be easily modeled using these methods.

Modern mesh deformation tools allow geometry to be smoothly reshaped through the manipulation of 3D handle vertices [BS08]. These tools provide an effective way to generate different poses of animal models without the use of a skeleton. However, they are less suitable for edits that require changes of shape or proportion. The *wires* framework [SF98] provides notable early examples of deforming a mesh to conform to drawn and projected space curves using correspondences based on known curve paramterizations.

Several recent papers introduce sketch-based deformation interfaces [KG05, NSACO05, KS06a, ZNA07], where individual drawn strokes can be used to deform models. The stroke provides a sequence of handle vertex targets that are then put into correspondence with vertices on the mesh. With this set of correspondences in place, mesh deformation techniques can enact the desired deformation. Kho and Garland [KG05] have the user sketch both a reference curve and a target curve and use normalized arc length to define a correspondence between them. The sketched curves serve as a type of proxy skeleton and are shown to be an effective mechanism for posing models. Nealen et al. [NSACO05] deform models using a stroke representing an occlusion contour. They also define correspondences using normalized arc length. To create the reference curve, the user defines a region of interest on the surface and then selects and trims one of its occlusion contours. Kara et al. [KS06a] use a similar approach to match strokes to a pre-segmented 3D wireframe template and further demonstrate a powerful set of tools in support of the shape design of automobiles [KDS06, KS06a, KS08] .

Most recently, Zimmermann et. al [ZNA07, ZNA08] demonstrate several further improvements, including the ability to infer a reference curve (path) on the model for a sketched stroke drawn near the undeformed model. The region of the mesh that is to be deformed is automatically estimated using the correspondence information and an effective heuristic.

We seek to move from working with individual strokes that define local deformations to processing a multi-stroke drawing that apply global deformations. The workflow changes from one of repeated 'edit and observe' to one of drawing followed by model-based drawing interpretation. The notion of a region of influence disappears and a number of other challenges are exposed. Achieving error-free correspondences between the drawn strokes and paths on the model becomes less likely given the larger scale and the lack of a user in the loop. Obtaining good 2D occlusion contours from the model for use in a 2D-to-2D matching scheme can be a significant challenge in itself [ZNA07], as is finding an unambiguous and smooth mapping back to the 3D model (see the supplemental material for an example).

Sketch-based modeling has connections with both image-based modeling and model-based vision, where *a priori* knowledge about object classes plays a significant role. Yang et al. [YSvdP05] use known 2D object templates to aid in drawing recognition, which is then coupled to hand-tailored methods for generating the corresponding 3D models. Prasad et al. [PZF07] demonstrate the modeling of smooth surfaces from image-apparent contours. Balan et al. [BSB*07] use a sophisticated database-driven model of human shape and pose in order to infer the specific human shape and pose from multi-view image silhouettes.

Lastly, hidden Markov models have been proposed for 2D shape classification using exterior silhouettes, e.g., [BM04] and others. The work of [CYES00] and [Fel05] is closest in spirit to our HMM/dynamic-programming-based correspondence determination. However, their focus is on detecting 2D-deformable shapes in images; the results do not extend in an obvious way to the mixed 2D-and-3D nature of our deformation problem.

3. Algorithm Overview

Our contour-based modeling process is comprised of a number of steps, as illustrated in Figure 2. The first step is the creation of the input drawing (§4.1). Each drawing stroke is represented by a sequence of points with associated outward pointing normals. The strokes need not include all the occlusion contours of the desired model; most of our examples have only partial contour information. Given the input contours and a 3D template, an initial alignment is established (§4.2), setting a common view direction and scale for the contours and template.

The challenge is then to use the contours to deform the template model. This is accomplished using a repeated correspond-and-deform process in a fashion reminiscent of iterative-closest-point methods (ICP). Like ICP methods, the iterative nature of this process tolerates some correspondence errors being made early on in the process, while later taking advantage of increasingly accurate correspondences. Additionally, our setting allows for compromises between two distinct goals, that of conforming to the drawn contours and that of maintaining the original shape of the template model. The supplementary materials provide an example that shows the degraded results of a deformation obtained using a non-iterative deformation, i.e., one that uses only the initial correspondences.

The task of computing correspondences requires finding a best-matching sequence of 3D model vertices for each and every drawn 2D stroke. The criteria for matching any model vertex to a stroke point include proximity and normal similarity. The normal similarity is computed in 3D, while proximity is measured in image space. Importantly, we look for the best matching sequence and not just the set of individual best matches, with the expectation that continuous input contours should generally map to a continuous vertex path on the model. This is captured using a transition cost which is minimized when the distances traveled along the drawn contour and between 3D vertices are equal. This discourages jumps in depth for the path of model vertices. The local-match costs and transition costs are used to define a Hidden Markov Model, for which the optimal solution is computed using dynamic programming.

Modern mesh deformation tools allow for effective solutions to the competing goals of meeting the vertex-and-normal constraints coming from the matching contour points, and of shape preservation. This can be viewed as a problem of matching the given contours subject to a type of shape regularization, or, alternatively, as informing a shape inference problem with prior shape knowledge. As illustrated in Figure 2(e), the final model shape does not completely conform to the input contours.

4. Pre-Processing

4.1. Input Drawing

The input to the modeling process consists of a set of open stroke lines. These can be created directly by free-hand drawing, hand-tracing contour lines over an image, or edge detection. The strokes are represented as polylines. We shall refer to the polyline vertices as *points*, thereby reserving the use of *vertices* for referring to the 3D template mesh model.

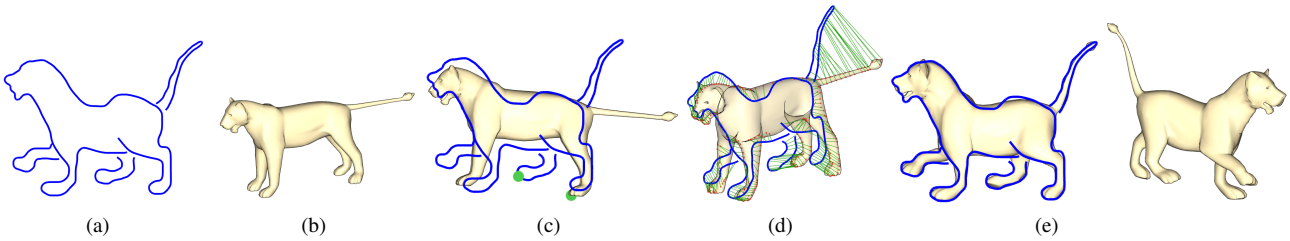
A required property of the strokes is that an outward-pointing normal can be defined for each stroke point, p_i. If the strokes form a closed or nearly closed polyline, the polyline interior is detected automatically and the *stroke normal* at each point is assigned to point outwards. If the interior is not well-defined, the inside/outside designation is provided directly by the user. Given a specified orientation, the normal n_i^p at p_i is defined as $n_i^p = ((p_i - p_{i-1}) + (p_{i+1} - p_i))^{\perp}$, i.e. the perpendicular to the tangent at p_i where p_{i-1} and p_{i+1} are the previous and next points on the stroke.

As a last step, the stroke polylines are resampled to approximately match the sampling density of the model in screen space. The latter is estimated using the mean edge length of the 3D model, scaled to screen coordinates, which we found to be a simple and effective heuristic. This will allow for a similar spacing between stroke points and their corresponding template mesh vertices, whcih is helpful for establishing the most meaningful correspondences. The examples used in our paper exhibit some natural variation in sampling density, but this has not been problematic.

4.2. Initial Alignment

Given the input drawing, the template model needs to be coarsely aligned with it. This establishes a common scale as well as a viewing direction for the template in which the template's occlusion contours best match the drawn contours. We use a semi-manual alignment where the centroid and global scale are automatically approximated, leaving the user to establish the required orientation, done with an arcball interface, and make minor refinements to scaling and translation. Minor differences in the alignment generally produced no visible difference in the final results. We use a weak-perspective camera projection for all our results.

We allow users the option of manually specifying 2D-point to 3D-vertex correspondences to facilitate interpretation in cases such as Figure 5(i), where the template and contours have very different shapes. They are also useful when an interpretation of ambiguous depth information is required (Figure 2). A good number of our examples work without any correspondences, e.g., the bodybuilder in Figure 1. For others (lion and two dogs), one or two point-to-vertex correspondences are sufficient to correctly identify matching parts as shown in Figure 5. The correspondences are used only for matching and do not affect the deformation itself.



(a) (b) (c) (d) (e)

Figure 2: *Algorithm Overview: (a) Drawn contours; (b) Template; (c) Initial contour and template alignment. The correspondence constraint (green) is added to prevent the left-right mismatch of the hind legs; (d) Initial correspondences (e) Final deformed model in two views.*

5. Determining Optimal Correspondences

We now describe the algorithm used to find optimal correspondences between drawing strokes and paths of vertices on the template model, based on a combination of orientation, proximity, and continuity considerations.

5.1. Local match criteria

We first describe the criteria that will be useful in defining a metric for the cost of matching template model vertex v to stroke point p. We consider two simple components: proximity and normal difference.

- *Proximity* is measured in the image plane as $d_P = (p^x - v^x)^2 + (p^y - v^y)^2$.
- *Normal Difference* is measured as the 3D dot product $d_N = n^p \cdot n^v$, where n^p is the normal to the stroke at p (lifted to 3D using $z = 0$) and n^v the mesh normal at v. The metric is optimal when the dot product is equal to one. By considering normal difference the matching implicitly prefers to match stroke points to occlusion contour vertices on the mesh, as in this case both normals lie in the xy-plane.

We experimented with including local surface features such as normal curvature in the image plane as a match criteria, but we found that this did not lead to a consistent improvement in the matching quality.

To enforce the general expectation that continuous contour strokes should map to continuous paths on the 3D model, we add a measure that encourages such continuity. Since each stroke is a directed one-dimensional polyline, the points on it can be ordered as $p_1, p_2, \ldots, p_n$. A simple-but-effective metric of *Continuity* is the ratio $d_C = \|v_i - v_{i-1}\|^2 / \|p_i - p_{i-1}\|^2$ where v_i and v_{i-1} are the matching vertices of p_i and p_{i-1} respectively, with distances between the mesh vertices measured in 3D and distances between stroke points measured in 2D. The optimal ratio is one, which captures the notion that traveling a given distance along the contour stroke should correspond to traveling a similar distance on the model mesh. This metric implicitly penalizes depthwise jumps on the template model.

5.2. HMM model

Given costs established by the above metrics, we wish to find correspondences that minimize a sum of these costs. Unfortunately, the costs of correspondences are coupled because of the continuity term. Our solution uses a form of dynamic programming, which implicitly considers all possible correspondence assignments without needing to explicitly enumerate them. Intuitively, this is done by breaking the problem into successive stages, where each stage is only dependent on the immediately preceeding stage.

More formally, the problem is naturally described by a Hidden Markov Model (HMM) [Rab89]. In an HMM, the input is a sequential series of observed states and the goal is to infer the corresponding sequence of hidden states that is most likely to have generated these observations. In our case, the contour stroke points are treated as observations and mesh vertices are treated as hidden states, as shown in Figure 3(c). solutions are given by a left-to-right path through the trellis, where there known costs associated with nodes and transitions. An example solution is illustrated on the trellis, and the induced correspondences are shown in Figure 3(d).

The HMM requires *emission probabilities*, i.e., the likelihood that a given hidden state will produce a given output, and *transition probabilities*, i.e., the likelihood of a transition from one hidden state to another. The proximity and normal metrics are used to compute the emission probabilities as follows:

$$P(p_j|v_i) \propto e^{-\frac{1}{2}(\frac{d_P}{\sigma_P})^2} e^{-\frac{1}{2}(\frac{d_N - 1}{\sigma_N})^2}. \quad (1)$$

The continuity metric is used to compute the transition probability:

$$P(v_i|v_{i-1}) \propto e^{-\frac{1}{2}(\frac{d_C - 1}{\sigma_C})^2}. \quad (2)$$

We use values of $\sigma_P = 0.25D$, $\sigma_N = 1$, and $\sigma_C = 0.05$ for all our examples, where D is the maximum dimension of an input image. The initial state probabilities are given by the emission probabilities for the first point, i.e., states in the first column.

The HMM problem is solved using the well-known Viterbi algorithm. The Viterbi algorithm is based on dynamic

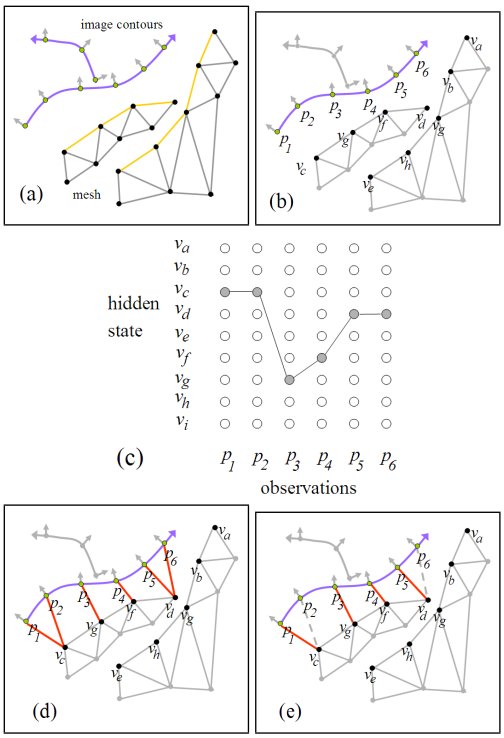


Figure 3: *Establishment of Correspondences. (a) Contours and template mesh. (b) Each contour point needs to find a best-match vertex. (c) The problem as an HMM trellis, with a solution path. (d) Best matches found by the solution path. (e) Elimination of many-to-one matches (dashed lines).*

programming and computes the optimal path through the trellis. For conciseness, we refer the interested reader to the classical introduction to the Viterbi algorithm found in [Rab89]. User-specified point-to-vertex correspondences(§4.2), if any, force the HMM solution to pass through a given point in the trellis. Lastly, the HMM solution may result in several contour points corresponding to the same mesh vertex. A post-processing pass remedies this by uniquely assigning them to the most likely contour point, determined by the emission probability as illustrated in Figure 3(e).

Considering all vertices on the mesh for each contour point would yield a problem unwieldy in size. We restrict the set of candidate vertices to those having an emission probability greater than a fixed threshold, such that a typical matching set size lies in the range of 25–100 vertices per contour stroke point. The filtering thus removes from consideration vertices that are very poor matches in terms of proximity or normal. Stroke points that are left with no matching vertices are treated as outliers and removed from consideration. We note that the HMM solution will not establish correspondences with vertices on the model that have no good matches with the drawn contours. As a result, the system preserves model features that have no matching drawing features.

6. Deformation

Given the computed correspondences, each iteration of our algorithm deforms the template by attracting the matched vertices to their contour-stroke counterparts. Since the matched vertices are expected to become occlusion contour vertices on the deformed model, it is necessary not only to pull the vertices towards their corresponding stroke points, but also to attract the normals at these vertices towards the corresponding stroke point normals. Both requirements are implemented as soft constraints in order to maintain the desired compromise between shape preservation and contour matching, and to allow for some tolerance of any remaining correspondence errors. We use mean-value encoding [KS06b] as our deformation method, although alternatives such as linear variational models could also be adapted for use with our method.

The mean-value encoding describes each vertex v_i as a function of its neighbor vertices v_j, and an estimated vertex normal $n_i(v_j)$, $v_i = F_i(v_j, n_i(v_j))$. The estimated normal $n_i(v_j)$ at v_i is computed as a function of the neighbor vertices v_j. To compute the deformed mesh the method minimizes the following functional

$$\frac{1}{2}\sum_{v_i \in V}(v_i - F_i(v_j, n_i(v_j)))^2. \qquad (3)$$

Our formulation has no hard constraints and the computed correspondences are used to specify soft positional and normal constraints. To introduce soft positional constraints we add a term $\rho((v_i^x - p_i^x)^2 + (v_i^y - p_i^y)^2)$ for each constraint to the functional in Equation 3, where (p_i^x, p_i^y) is the optimal 2D position for the vertex and ρ is the weight assigned to all soft positional constraints. To enforce soft normal constraints we add a term $\theta\|n_i^v - n_i^p\|^2$ for each constraint, where n_i^p is the normal to the contour at p_i, n_i^v is the estimated normal at v_i, and θ is the weight assigned to soft normal constraints.

To enhance the power of the human and animal templates, we use a spatially-varying deformation behavior, depending on local material stiffness, as proposed by several recent deformation frameworks [BPGK06, HZS*06]. The degree of stiffness at each vertex is defined using a weight coefficient for each entry in the sum in Equation 3. The stiffness for the lion model was derived from deformation examples [PJS06] while for other models we painted the main joints as being less stiff than the rest of the body.

6.1. Deformation Schedule

Our match-and-deform schedule exploits the improved correspondences that are achieved as the iterative process proceeds. The weights ρ and θ (§6) are initially small and are increased throughout the iterative matching and deformation process, as the accuracy of both the contour fit and the computed correspondences increases. We use values of $\rho = 0.01$

and $\theta = 0.001$ and increase them by 10% at each subsequent iteration. These values apply to all our models. The weight θ for normals is set to be much lower than ρ as changing normals has a more global impact than changing positions. The energy minimization problem is solved using Gauss-Newton optimization to obtain a deformed mesh. To speed up the process, the first few match-and-deform iterations are performed on a simplified model with about 10% of the faces.

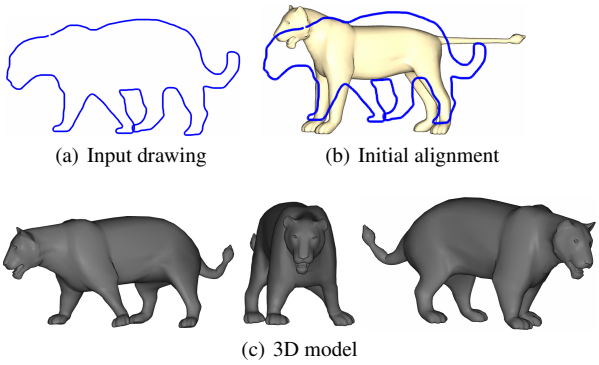
The HMM computation takes less than a second on a simplified mesh and up to two seconds on a typical full resolution template with about 20K faces. The bottleneck of the computation is the actual deformation. It typically requires one or two interior Gauss-Newton iterations to converge for each set of constraints and weights, with one iteration on a simplified template taking two to three seconds and an iteration on a full resolution one taking up to ten seconds. The method typically requires a total of ten to twenty outer iterations of matching and deformation to obtain the final result. Thus the entire process takes two to ten minutes depending on the size of the template mesh and the deformation complexity.

7. Results

We illustrate our method on a wide variety of models, which are also shown in the supplementary video. Most of the input drawings were created by tracing contours on images, while one was created using automatic edge-detection on an image (creamer cup). We test the method on both side and three-quarter views.

Figure 1 shows a muscle-bound bodybuilder recreated from a contour drawing. The model in the drawing is shown in a three-quarter view, which is challenging to interpret. The initial alignment for this model does not include any correspondence constraints. The elbows of the template model have been painted with 'soft' stiffness weights. The final model is significantly different from the template in both pose and proportions. In particular, it has large muscle-bound arms and chest. The scale-invariant deformation technique ensures that the model's arms scale appropriately in all directions and not only in the image plane defined by the contours. This property is further illustrated in the supplemental material, where we illustrate a sphere being deformed using the contour from a football. Any apparent muscle bulges on the final 3D model come from the combination of the drawn contours and the original template model. As seen in this example, contour drawing provide an effective way for creating models with different proportions and body shape.

Animals: We tested our method on several animal drawings. Contour-based modeling provides an effective way to model the shape of many animals due to the difficulties in using other techniques such as 3D scanning, which require the subject to remain still in order to obtain a quality scan. The geometric modeling of animals has commonly relied on either artists or scans of figurines. The combined contour-plus-template approach provides an effective tool to create a large variety of animal shapes and breeds from a small set of canonical templates. Our first example is that of the caricature lion shown in Figures 2. This example illustrates the ability of the technique to infer good estimates of pose and local proportions from the contour information. The panther in Figure 4 is constructed using the same template model as the lion. No correspondence constraints were used in the initial alignment. A comparison of the shapes of the final lion and panther models illustrates the significant variations in shape that can be inferred from contours. The panther is the only example we present where the 2D contour directly corresponds to the outer silhouette of the model.



(a) Input drawing (b) Initial alignment

(c) 3D model

Figure 4: *Panther*

The dogs in Figure 5 are another example demonstrating the effect of contours in changing proportions and pose. The head, tail, and torsos of the sketched hound and pug differ significantly from those of the template. Figure 5(i) demonstrates the need for the user-specified correspondences for the pug model, where the tail differs drastically from that of the template, as without them the method adopts a sub-optimal interpretation of the input strokes.

Man made object: The creamer shown in Figure 6 provides an example of a man-made object modeled with our technique. This is a challenging example in several respects. The contours are automatically derived from an edge detection algorithm and thus there are a number of contour lines that come from the surface texture rather than occlusion contours. Additionally, the template and target contours are significantly different in shape. We note that even though the algorithm nicely preserves the ribbing on the creamer, in general our method, like other local coordinate deformation techniques, does not perform best on CAD-like models with sharp corners.

Modeling occluded symmetric parts: The horse model in Figure 7 demonstrates an extension of our method which uses symmetry to better complete occluded parts. Given a contour drawing, the shape of occluded parts can be esti-

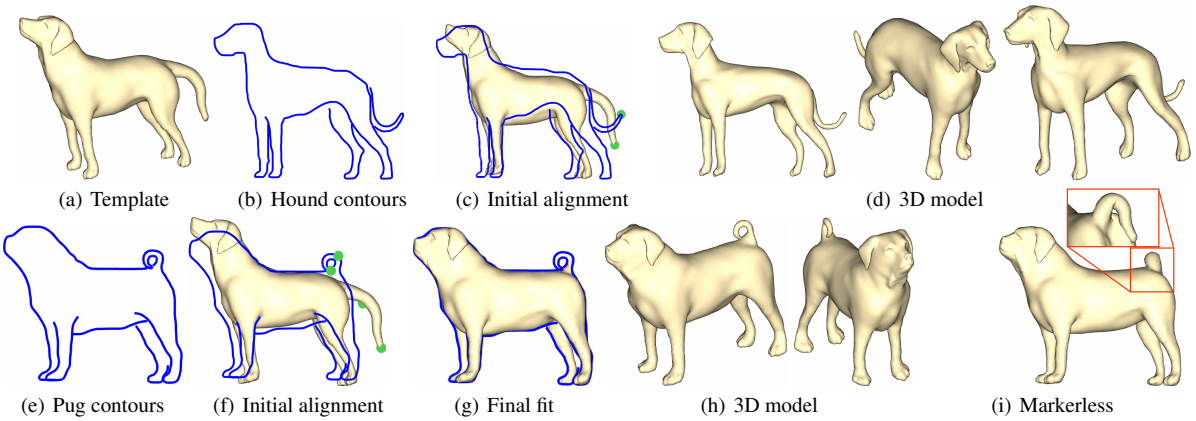


Figure 5: *Hound (d) and pug (h) modeled using the same template (a). Green dots (c,f) indicate user-specified correspondences. (i) The pug modeled with no markers on the tail.*

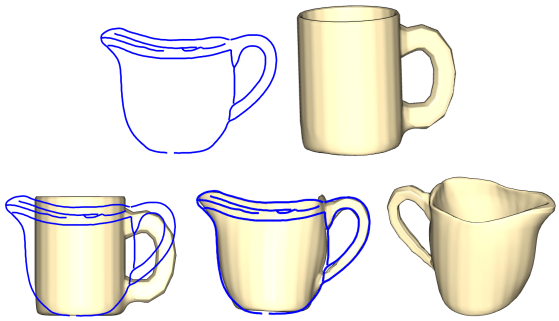


Figure 6: *Creamer: (top) drawing and template; (bottom) initial alignment (left), final fit (center) and 3D view (right).*

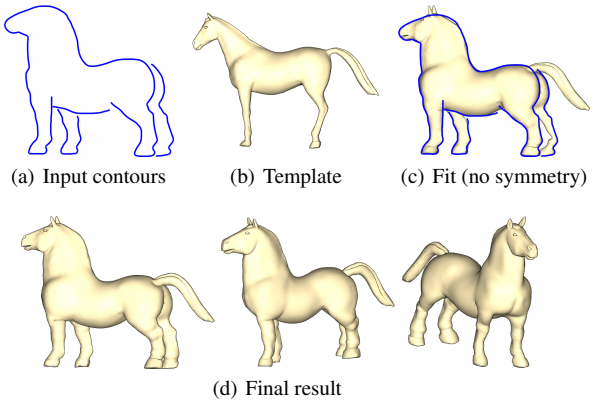


Figure 7: *Horse model illustrating use of symmetry to model occluded regions.*

mated using prior knowledge of the likely shapes. Specifically, the local body shape is likely to be left-right symmetric, even though globally the pose of the left and right halfs may differ. We take advantage of this observation, in cases where the input templates have global reflective symmetry, maintaining symmetric local shape between visible and fully- or partially-occluded parts. Due to space considerations, the details of the symmetry preservation are presented in the supplementary material.

Multiple views: In the supplemental material, we further provide an example that illustrates the use of two views to reconstruct a teddy bear model.

8. Discussion

Modeling with multi-stroke contour drawings uses a different workflow than single deformation strokes to interactively deform a 3D model. We envisage an ideal system as having access to both. The first phase of conceptual design and artwork remains firmly rooted in creating drawings, where strokes can be freely drawn without any commitment to their interpretation. In contrast, single-stroke based deformation systems are a natural fit with the sequential construction approach of current 3D modeling systems.

One of the key novelties of our method is the use of an HMM to find optimal correspondences sequences instead of working with more local point correspondences. We explicitly allow for non-silhouette vertices to participate in this match, as compared to the alternative of only using silhouette vertics. This is particularly helpful achieving good correspondences for situations such as the flat back of the horse model shown in Figure 7, where the silhouette vertices as computed from the mesh are scattered across the back of the horse rather than lying on a smooth line. While such artifacts can be cleaned up algorithmically, commiting to one or more designated silhouette paths on the 3D geometry can be tricky, particularly in areas of the model where there are joining silhouette edges. The HMM-based approach in effect avoids computing silhouette edges as an intermediate repre-

sentation for solving the correspondence problem. Instead, it directly provides an optimal solution to the sequence correspondence problem that is needed to drive the deformations.

The current system has a number of limitations. It is slower than single deformation stroke methods, although advances in parallel sparse linear system solvers and multiscale approaches are a promising avenue for achieving a sizeable speedup. A correct interpretation requires good initial alignment and sometimes benefits from extra user constraints, e.g., Figure2(c). It may be difficult to predict in advance when manual correspondences are needed. The approach requires that a good library of 3D templates be available, and it is still currently a manual process to select an appropriate template and to perform the initial registration. In some situations, a small number of user constraints may be necessary to help enforce desired correspondences between the template and drawing, e.g., Figure 5(i).

The iteractive correspond-and-deform process is ultimately a gradient-based technique and so it may converge to local minima. The method does not yet exploit clues such as T-junctions [KH06]. Thus our method can provide an interpretation that is inconsistent with such cues.

9. Conclusions

We have presented a framework for developing tailored 3D models from contour drawings and 3D templates. This represents a complementary alternative to deformation techniques based upon using single strokes. The novel HMM representation of the correspondence problem provides the reliable correspondences that are at the heart of our method. Our iterative correspond-and-deform mechanism is tolerant of early correspondence errors and helps achieve the desired compromise between matching drawn contours and shape preservation.

There are a number of exciting avenues to expore in future work. One direction is to extract more information from the drawing, specifically using sketch structure to infer relative depth and occlusion relationships from the drawing [KH06]. Developing techniques for modeling more directly from images instead of drawings is also a tantalizing prospect. Another future direction is to incorporate more information into the template. Known skeletal structure for humans and animals could be used to index into a pose-likelihood model, which would help with the process of disambiguating contours or finding sets of likely solutions.

References

[BM04] BICEGO M., MURINO V.: Investigating hidden markov models' capabilities in 2d shape classification. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 2 (2004), 281–286.

[BPGK06] BOTSCH M., PAULY M., GROSS M., KOBBELT L.: PriMo: Coupled Prisms for Intuitive Surface Modeling. *Eurographics Symposium on Geometry Processing* (2006), 11–20.

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Trans. on Visualization and Computer Graphics* (Jan/Feb 2008).

[BSB*07] BALAN A. O., SIGAL L., BLACK M. J., DAVIS J. E., HAUSSECKER H. W.: Detailed human shape and pose from images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2007).

[CYES00] COUGHLAN J., YUILLE A. L., ENGLISH C., SNOW D.: Efficient deformable template detection and localization without user initialization. *Computer Vision and Image Understanding 78*, 3 (2000), 303–319.

[Fel05] FELZENSZWALB P. F.: Representation and detection of deformable shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence 27*, 2 (2005).

[HZS*06] HUANG J., ZHANG H., SHI X., LIU X., BAO H.: Interactive mesh deformation with pseudo material effects. *Comput. Animat. Virtual Worlds 17*, 3-4 (2006), 383–392.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH* (1999), pp. 409–416.

[KDS06] KARA L. B., D'ERAMO C., SHIMADA K.: Pen-based styling design of 3d geometry using concept sketches and template models. *ACM Solid and Physical Modeling Conf.* (2006).

[KG05] KHO Y., GARLAND M.: Sketching mesh deformations. In *SI3D '05: Proceedings of the 2005 Symposium on Interactive 3D graphics and games* (2005), pp. 147–154.

[KH06] KARPENKO O. A., HUGHES J. F.: Smoothsketch: 3d free-form shapes from complex sketches. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 589–598.

[KS06a] KARA L. B., SHIMADA K.: Construction and modification of 3d geometry using a sketch-based interface. *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2006).

[KS06b] KRAEVOY V., SHEFFER A.: Mean-value geometry encoding. *Intl Journal of Shape Modeling 12*, 1 (2006).

[KS08] KARA L. B., SHIMADA K.: Supporting early styling design of automobiles using sketch-based 3d shape construction. *Computer-Aided Design & Applications 5*, 6 (2008), 867–876.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 41.

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3 (2005), 1142–1147.

[PJS06] POPA T., JULIUS D., SHEFFER A.: Material aware mesh deformations. In *Shape Modeling International* (2006).

[PZF07] PRASAD M., ZISSERMAN A., FITZGIBBON A.: Single view reconstruction of curved surfaces. *CVPR 2007* (2007).

[Rab89] RABINER L. R.: A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[SF98] SINGH K., FIUME E.: Wires: a geometric deformation technique. In *Proc. ACM SIGGRAPH* (1998), pp. 405–414.

[YSvdP05] YANG C., SHARON D., VAN DE PANNE M.: Sketch-based modeling of parameterized objects. In *Proc. EG Workshop on Sketch Based Interfaces and Modeling* (2005).

[ZNA07] ZIMMERMAN J., NEALEN A., ALEXA M.: Silsketch: Automated sketch-based editing of surface meshes. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2007).

[ZNA08] ZIMMERMANN J., NEALEN A., ALEXA M.: Sketching contours. *Computers & Graphics 32*, 5 (2008), 486–499.

Supplemental Material

Illustrations of several sub-optimal algorithm variations

The proposed method uses non-local matches and an iterative process. In Figure 1 we show the results of using only local matches with iteration, or using non-local matches but with no iteration. In either case, the results are inferior to that of the full non-local, iterative algorithm, whose final result is illustrated in Figure 2(e) of the paper.

Illustrations of several sub-optimal algorithm variations

Figure 2 illustrates what happens to the depth of an object as it is deformed to match image contours. A sphere is deformed to fit the contour of a football, which preserves it circular cross-section.

Challenges in directly computing occlusion contours

Figure 3 illustrates the noisy nature of unprocessed occlusion contours as computed from a geometric mesh, and the potentially complex ways that occlusion contours can split and join. The HMM algorithm avoids having to explicitly compute such contours for the model.

Illustration of the iterative deformation

Figure 4 shows the iterative correspond-and-deform process.

Teddy bear model reconstruction from multiple views

Multiple views can be used simultaneously or in sequence, as shown in Figure 5. To use the views simultaneously we would need to register them accurately with respect to one another. Misalignments can lead to artifacts when working simultaneously with two views, and thus we adopt a sequential strategy instead. In the given example, we deform the template using the contours from view A, then view B, and once again for view A. The image teddy differs from the template teddy in pose as well as the shape of the nose, ears, and feet. These differences are successfully recovered from the contour information with the help of a few user-specified correspondences. These are necessary mostly to correctly position occluded parts in the side view. Since the template is not symmetric, we cannot use symmetry for correct modeling of occluded parts in this case.

Details of symmetry preservation

A symmetry plane is first precomputed on the template using any of the recently developed symmetry detection methods for geometric models. When the template deforms, the global symmetry is no longer preserved. However, we expect symmetry to be locally preserved, e.g., both hands of a human have equal finger length, even if the two arms are positioned differently. Using the symmetry plane we compute a mapping for any vertex of the undeformed template mesh to its corresponding symmetric position.

Our method first deforms the model using the iterative correspond-and-deform procedure, disregarding symmetry. Users are then asked to specify the occluded regions they want repaired. Given this input, the method computes new local shape descriptors, in our case mean-value coordinates, for each vertex. First for vertices outside the specified regions it computes a new descriptor using the deformed mesh. For vertices inside these regions it computes the symmetric descriptor by first mapping the vertex and its neighboring vertices to their associated symmetric locations on the deformed model and then computing the descriptor based on those. The algorithm then reapplies the deformation using the precomputed matches with vertices in the selected regions using their symmetric descriptors and vertices everywhere else using the newly deformed ones.

Another example of using symmetry constraints

In the absence of symmetry constraints, the lion in Figure 6 the algorithm arbitrarily selects one of the front legs to match to the front-leg contour, and similarly for the back legs. This highlights the importance of consistency enforcement in the matching stage and showcasing the method's ability to preserve unmatched template geometry. With the additional enforcement of symmetry, both left and right legs are effectively matched to the singly drawn leg contours.

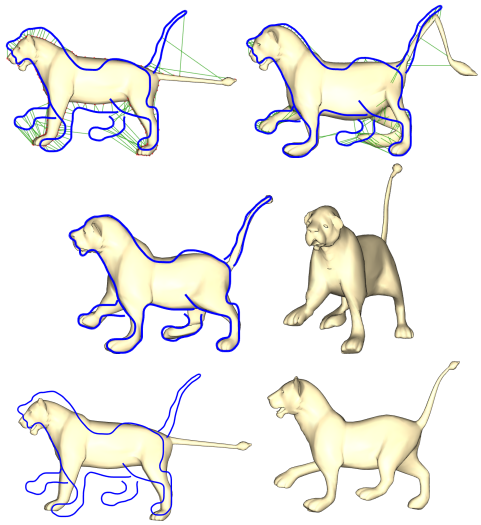
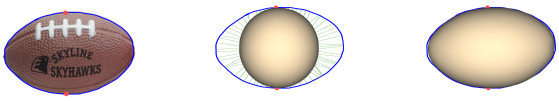


Figure 1: *Sub-optimal algorithm variations applied to the cartoon lion (Figure 2) Top: Iterative deformation using only local matches, showing the first-iteration correspondences (left) and the resulting final deformation (right). Middle: Non-iterative deformation using only initial correspondences (Figure 2(d)). Bottom: Unconstrained initial alignment produces a solution with a left-right mismatch for the hind legs.*



(a) Traced image contour (b) Initial registration and correspondences (c) Final shape

(d) Side view (e) Another view

Figure 2: *Football*

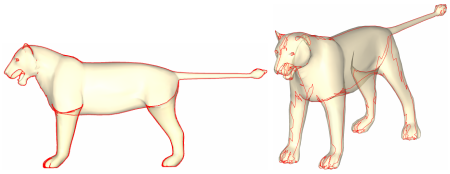


Figure 3: *Occlusion contours for the caricature lion in side view (left) rotated to visualize contour structure (right).*

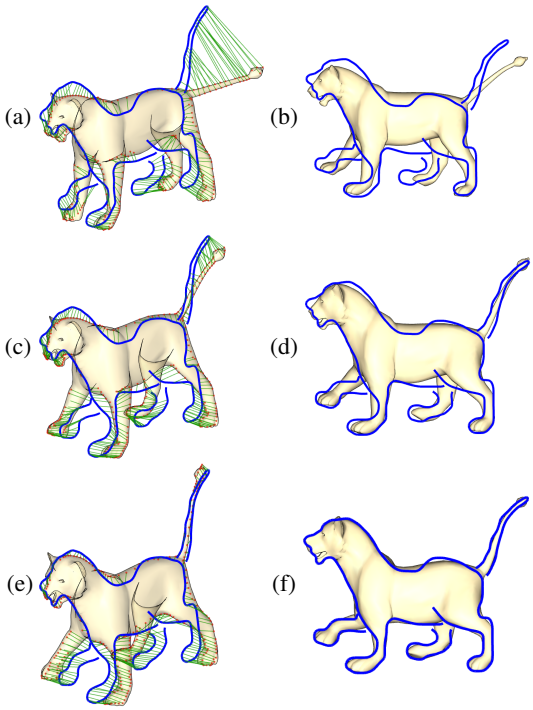


Figure 4: *The iterative correspond-and-deform process. The matches are shown in semi-transparent 3/4 view in order to showcase the impact of continuity. (a) initial correspondences; (b) deformed model after 5 iterations; (c) correspondences after 5 iterations; (d) deformed model after 10 iterations; (e) correspondences after 10 iterations; (f) final fit to contours after 15 iterations.*

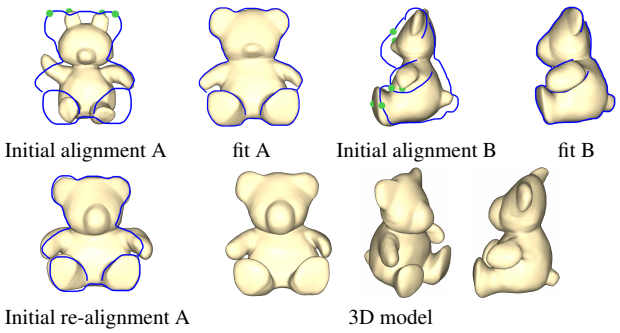
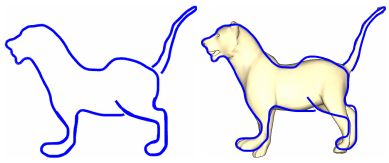
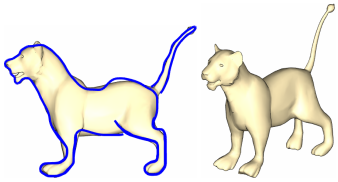


Figure 5: *Teddy bear model illustrating modeling from multi-view sketches.*



(a) Input contour (b) Fit (no symmetry)



(c) Final result

Figure 6: *Modeling a lion from a side view (two legs) contour. Symmetry is used to achieve the final result.*