Contour-based Modeling Using Deformable 3D Templates

Vladislav Kraevoy Alla Sheffer Michiel van de Panne [*]

University of British Columbia

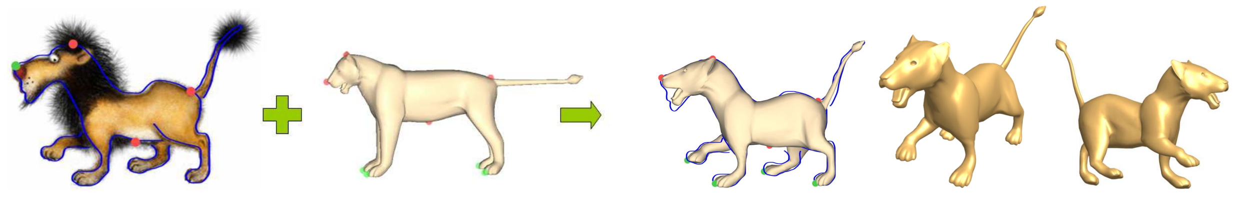


Figure 1: *Image contours are used to automatically construct a corresponding 3D model with the help of a 3D template. The image contours help inform the pose and proportions of the new shape while the template model helps inform its full 3D shape and surface detail.*

Abstract

We present a new technique for image-based modeling using as input image contours and a deformable 3D template. The technique gradually deforms the template to fit the contours. At the heart of this process is the need to provide good correspondences between points on image contours and vertices on the model. We propose the use of a hidden Markov model for efficiently computing an optimal set of correspondences. An iterative match-and-deform process then progressively deforms the 3D template to match the image contours. The technique can successfully deform the template to match contours that represent significant changes in shape. The template models can be augmented to include properties such as bending stiffness and symmetry constraints. We demonstrate the results on a variety of objects.

1 Introduction

Techniques in computer graphics and computer vision can readily benefit from each other, as demonstrated by past successes in image-based modeling and model-based vision. A parameterized 3D geometric template can provide predictive models of image features, which can then be paired with the actual observed image features. This 3D-model-feature to 2D-image-feature pairing can be used for estimating the parameters of the 3D model, including position, scale, orientation, and deformation. The template can also provide geometric detail that cannot be inferred from an image alone.

In the context of geometric modeling, our goal is to be able to create new 3D model geometry using information derived from images or drawings. Our work explores how such image-based information can be used to produce matching geometry with the help of a template. Figure 1 illustrates an example result of our system. The user input consists of tracing the image contours that define the desired proportions and pose of the lion, as well as specifying a small number of point correspondences between the template model and the image. With this input, the output geometry is automatically generated through an iterative match-and-deform process.

Making this type of contour-driven deformation work well requires solving two problems. First, robust correspondences are needed between silhouette vertices on the 3D model and points on the 2D image contours. This is a hard problem because any given silhouette vertex could plausibly match multiple image contour points, or perhaps none, and vice versa. The target contours may also represent a highly deformed version of the template object and thus the silhouette and the contours may differ significantly. Original silhouette vertices may not even be silhouette vertices of the final deformed model. Second, an appropriate deformation model needs to be developed for the 3D template geometry. The template should support a preferred shape, coupled with flexible control of the constraints that will drive the deformation process. It should also support the addition of further knowledge, such as material-like information indicating where the model may prefer to bend, as well as symmetry information.

The contributions of this paper are as follows. First, we propose a robust solution to the optimal-correspondence problem by modeling it as a hidden Markov model (HMM). Second, we develop an iterative deformation framework that interleaves finding correspondences with the application of a flexible deformation scheme tailored to our problem domain. The iterative scheme is analogous to iterative closest point (ICP) methods, with the “closest point” step and the “alignment” (deformation) step being replaced by methods of appropriate sophistication for the problem at hand. Our final contribution is the demonstration that these techniques can be used to synthesize 3D geometry from 2D images in a way that blurs the distinction between prior work on sketch-based modeling of local deformations and model-based computer vision techniques.

2 Related Work

There has been significant recent progress in developing tools for flexible geometric deformations. Many of the proposed tools are based on different frameworks for representing geometry in local coordinates [Sorkine et al. 2004; Yu et al. 2004; Lipman et al. 2005; Kraevoy and Sheffer 2006; Huang et al. 2006a]. This representation can then be used to apply deformations, blending, or other editing operations to geometric meshes. The efficiency and flexibility of such schemes can further be improved using implicit skeletal structures [Huang et al. 2006a; Yoshizawa et al. 2003], adding material

[*]e-mail: {vlady|sheffa|van}@cs.ubc.ca

properties[Popa et al. 2006; Botsch et al. 2006; Huang et al. 2006b], or using multiple level-of-detail mesh representations[Huang et al. 2006a].

While most of the tools use anchor vertex manipulation as a way to control the deformation, two recent papers introduce sketching based deformation interfaces [Kho and Garland 2005; Nealen et al. 2005]. The method of [Kho and Garland 2005] uses as input a reference stroke on the model and a target stroke, indicating the deformed shape of the reference. Nealen et al. [2005] infer the reference stroke from the target stroke, using a set of assumptions. The method computes silhouette edges or a suggestive silhouette within a user specified region and uses the silhouette as the reference stroke. User assistance is required if the silhouette needs to be trimmed or if more then one silhouette exists inside the region. Both methods assume arc-length correspondences between the points within each stroke, e.g., the half-way point of the reference stroke is chosen to correspond to the half-way point on the target stroke.

Our work is inspired by the above approaches but operates in a significantly less constrained setting where multiple image contours and model silhouettes need to be matched automatically. The proposed method computes a set of detailed point-to-vertex correspondences for points on the image contours and silhouette vertices on the model. It also supports unmatched contour points and unmatched silhouette vertices. Using this matching mechanism in an iterative correspond-and-deform process allows us to robustly handle large deformations.

Sketch-based modeling work has a strong connection with both image-based modeling and model-based vision, given that the types of contours that are useful to sketch are also often the same ones that are useful when attempting to infer geometric models from single images. The use of user drawing coupled with strong a priori knowledge about object classes has been used in [Quan et al. 2006] for modeling plants from a single photograph and in [Yang et al. 2005] for modeling airplanes, cups, and fish from sketched or traced contour lines. Earlier work [Debevec et al. 1996] supports fast architectural reconstruction from images using a priori knowledge of typical shapes and user-driven selection of key image points. Model-based vision has a long history in computer vision. A prime example is [Lowe 1991], which uses matches between the silhouette edges of the known 3D model and edges extracted from the image in order to determine the 3D pose of the model.

Deformable models have a long history and are often used for shape reconstruction from volumetric data, although they have also been used to estimate 3D reconstructions from single images. They commonly use physics-based spline formulations that are driven by image gradients and seek minimal curvature solutions [Terzopoulos and Metaxas 1991; McInerney and Terzopoulos 1996]. These models have been extended to include stronger models of prior shape, such as the PCA subspaces used in active shape models [Cootes et al. 1995]. Building an appropriate statistically-based prior shape model requires many identically-parameterized instances of the same model. In contrast, in our work the “shape prior” consists of a single 3D geometric template, optionally augmented with material parameters.

A key step of our work is the automatic computation of optimal correspondences between points on the 2D target contours and the vertices on the 3D model that form its silhouette. It is related to the problem of computing optimal matches between 2D open curves. Hidden markov models (HMMs) have been proposed as a technique for 2D shape classification using silhouette edges [Gorman et al. 1988; He and Kundu 1991; Bicego and Murino 2004]. Typically, these models assume the existence of a significant number of training examples in order to learn the HMM parameters, and are commonly demonstrated working on closed silhouette curves. Our use of an HMM model differs in a number of respects from this prior work. First, our model is developed using a single template model and likelihood models tailored to our application, instead of sets of training examples used in prior work. Second, unlike the 2D-to-2D curve matching problem, our work is solving for correspondences between 3D silhouette vertices and points on 2D open curves. The match likelihood is designed to reflect the 3D nature of our problem. Third, our HMM models produce finer-grained correspondences as a result of using a large number of silhouette samples as hidden states.

Another area of related work is the development of specific parameterized geometric models. A method of modeling horses from a set of anatomical landmarks is presented in [Simmons et al. 2002]. Other work applies data-driven approaches to modeling human geometry that has been flexibly parameterized according to both pose and proportions [Anguelov et al. 2005]. Our work differs in that our goal is to produce a model that matches given image contours, and to do so using a single 3D template model.

3 Algorithm Overview

Our modeling process can be described in terms of a number of steps, as illustrated in Figure 2. The first step is the extraction of the input contours, which are created as hand-drawn curves traced over an image or, alternatively, obtained using image processing algorithms. Each contour consists of a sequence of points with associated outward pointing normals. While the majority of the extracted curves represent desired silhouette edges, “non-silhouette” contours can also be added in order to achieve additional surface detail such as muscle ripples (see Figure 12). Contour lines are represented as open curves and it is not necessary to provide full coverage of all the silhouette edges in the model.

Given the input contours and a 3D template, an initial coarse registration needs to be established. This is accomplished using three user-selected contour-to-template (2D point to 3D vertex) correspondences. These points are marked in red in Figure 2(a) and the resulting template alignment is shown in Figure 2(c).

The next step is the most challenging, namely to deform the template model to match the contours while attempting to preserve the shape of the template. Our algorithm applies an iterative correspond-and-deform approach. At each iteration a set of optimal correspondences is established between the contours and the deforming template model. These are then used in a subsequent deformation step by attracting the model towards the contours. Figure 2(d) shows the set of correspondences used for the first iteration of deformation, while Figure 2(e) shows the final deformation after a number of correspond-and-deform iterations.

The correspondence computation matches points on the contours to vertices on the model which are both nearby and have similar normals. Additionaly, we expect to see continuity in the correspondences, i.e., consecutive points on the contours should map to nearby vertices on the model. This requirement results in the choice of optimal matches being interdependent and for which we therefore use a hidden Markov model (HMM) solution, as described in Section 6. The correspondence computation automatically discards outlier matches, skipping contour points with no close-by matches on the model.

Given the correspondences we apply a deformation mechanism which pulls the matched vertices on the model towards their cor-

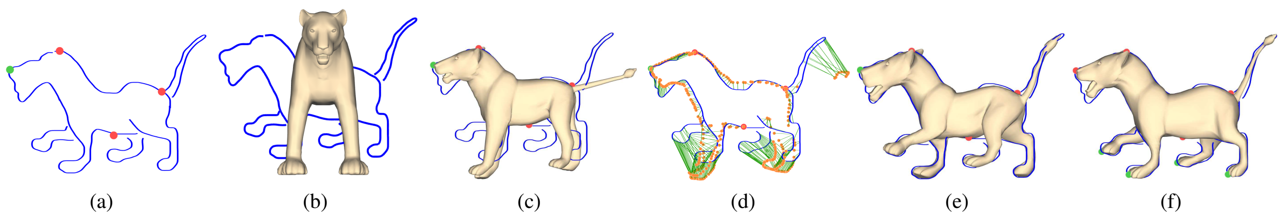


Figure 2: *Algorithm Overview. (a) Contours obtained from image; (b) Unregistered template model; (c) Initial registration of template model; (d) Initial correspondences; (e) Final deformed model; (f) Final model with extra constraints to enforce the correct mapping of the left and right legs (shown in green).*

responding points on the contours. The deformation uses soft constraints to ensure both shape preservation and contour matching. The deformation mechanism uses an extension of the mean value method [Kraevoy and Sheffer 2006] and is described in detail in Section 7.

Contours do not provide depth or occlusion information, allowing ambiguous interpretation of the described shape. The result of Figure 2(e) shows the existence of a left-right ambiguity in that the left legs of the template lion have been matched to the right legs of the image contours, and vice-versa. We introduce several mechanisms that allow the user to introduce additional information into the process for cases where the template shape provides insufficient information to resolve such ambiguities. These include reconstruction using image contours from multiple viewpoints (Figure 10), additional 2D constraints (Figure 2(f)), and directly fixing the depth of specific template mesh vertices (Figure 12).

We note that the deformable-template based framework does not create parts that do not exist in the template model, nor remove template parts that are not found in the image. This preserves parts of the template not affected by the contours, such as the head in Figure 12, while deforming the models to smoothly transition between the parts affected by the contours and those which are not.

The last step is to lift the textures from the source image and apply them to the 3D model if desired.

4 Input Contours

The input to the modeling process consists of a set of open contour lines. These can be created in several ways, including hand-tracing over an image, direct extraction from an image using an appropriate edge-detection or segmentation algorithm, or a direct sketch of desired contours. The contours are represented as polylines. In the case of image edge-detection, we convert the image to greyscale, apply an edge-detection or segmentation algorithm, threshold the resulting image, apply an erosion step until convergence in order to get single-pixel-width lines, and then convert the result into polylines. We shall refer to the polyline vertices as *points*, thereby reserving the use of *vertices* for refering to the 3D template mesh model.

A required property of the contours is that an outward-pointing normal can be defined for each contour point, p_i. If the contours form a closed or nearly closed polyline, the polyline interior is detected automatically and the *contour normal* at each point is assigned to point outwards. If the interior is not well-defined, the desired orientation is provided 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 contour specified by the orientation.

As a last step, the contour polylines are resampled to have a similar sampling density to that of the template model after initial registration as described in Section 5. This will allow a similar spacing between contour points and their corresponding template mesh vertices, which is helpful in establishing the most meaningful correspondences.

5 Initial Registration

Before determining correspondences for the first time, the template model needs to be coarsely registered with the input contours. This establishes a common scale as well as a view direction for the template in which the template's silhouette best matches the contours. We require three point p_i to vertex v_i correspondences, specified such that the points on the image are at the same depth. We therefore can arbitrarily set $p_i^z = 0$ for all of them. Given these correspondences we establish the common scale s for the model by minimizing

$$\min \sum_{(i,j)} (s\|v_i - v_j\|/\|p_i - p_j\|) - 1)^2.$$

The template is first scaled by s and then is simultaneously rotated and deformed by enforcing the correspondences as hard positional constraints, $v_i = p_i$, using the deformation framework to be described in Section 7. Users can specify additional constraints to resolve ambiguities that arise, such as the left-right reversal that occurs for the solution shown in Figure 2(e). The additional constraints can also be used to enforce exact feature correspondences. These constraints are only specified in the xy plane, i.e., $v_i^x = p_i^x, v_i^y = p_i^y$.

6 Correspondences

After initial registration, we can expect that good correspondences can be established for at least a subset of the contour points and the template vertices. At this point, one could choose to match contour points to template silhouette vertices, or vice-versa, as illustrated in Figure 3(a). We note that both contour points and template-model silhouette vertices may not always have a match. Contours may come from an edge-detection algorithm that produces spurious edges, while the template model may contain silhouettes that have no corresponding contour. We choose to look for template silhouette vertices for every contour point. This exploits the strong continuity found in the contour, namely the set of ordered points that comprise the polyline. Silhouette edges on the input mesh are not explicitly computed. Instead, it will be sufficient to identify candidate silhouette vertices, defined as vertices with normals having

$|N_z| < \varepsilon_N$. We use $\varepsilon_N = 0.2$. This will be beneficial because it allows the algorithm to consider non-silhouette vertices as candidate matches. In general, non-silhouette vertices in the initial registration may become silhouette vertices in later stages of deformation.

6.1 Match Criteria

In searching for a mesh silhouette vertex v to correspond to a given contour point p, we wish to optimize proximity and normal difference metrics.

- *Proximity* is measured as $d_P = (p^x - v^x)^2 + (p^y - v^y)^2$ and is optimal when the metric is zero. Note that proximity is measured only in the xy plane, as depth information for the contours is not available.
- *Normal Difference* is measured as the 3D dot product $d_N = n^p \cdot n^v$, where n^p is the normal to the contour 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.

Considering these two metrics alone, however, neglects the expectation that contours should map to continuous silhouettes. Since each contour is a directed one-dimensional polyline, the points on it can be ordered as $p_1, p_2, \dots, p_n$. We found that 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 the distances between the mesh vertices measured in 3D and distances between contour points measured in 2D. The optimal ratio is one, which captures the notion that traveling a given distance along the contour should correspond to traveling a similar distance on the model mesh. Depthwise jumps on the template model are also penalized with this metric.

6.2 HMM model

To combine the point-to-vertex metric with the continuity metric in a principled way, we cast the problem in terms of a hidden Markov model (HMM)[Rabiner 1989]. In this framework, contour points are treated as observations and mesh vertices are treated as hidden states, as shown in Figure 3(c). The goal is to find the most-likely left-to-right path through the trellis, i.e., the most likely sequence of vertices (hidden states) that could have produced the given contour points (observations). 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:

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

The continuity metric is used to compute the transition probability:

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

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 HMM problem is solved using the well-known Viterbi algorithm[Rabiner 1989]. User-specified point-to-vertex correspondences described in Section 5 force the HMM solution to pass through a given point in the trellis.

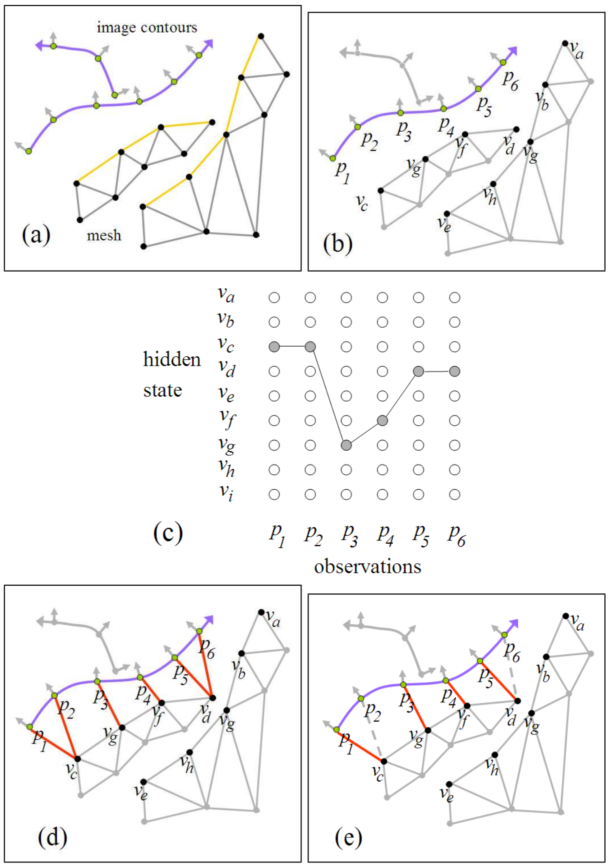


image contours
mesh
(a)
(b)
(c)
(d)
(e)
hidden state
observations
v_a v_b v_c v_d v_e v_f v_g v_h v_i
p_1 p_2 p_3 p_4 p_5 p_6

Figure 3: *Establishment of Correspondences. (a) Input contours and mesh. (b) Each contour point needs to find a best-match vertex. Vertex connectivity is ignored, but taken into account by the transition cost. (c) The problem cast as a hidden Markov model, with the solution path illustrated. (d) Best matches found by the solution path. (e) Elimination of many-to-one matches (dashed lines).*

Considering all silhouette 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. The filtering thus only removes from consideration vertices that are very poor matches in terms of either proximity or normal. Contour 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 silhouette vertices on the model that have no good matches with the contours. As a result, the system preserves model features that have no matching contour features.

Some contours may match well to multiple disjoint portions of the mesh. For example, for a dog seen in profile (Figure 8), a single well-drawn contour curve should be usable as a target contour for both of the dog's ears, the visible and the hidden. This situation can be elegantly handled using the HMM framework. The HMM trellis can be used to find second-best and nth-best solutions, although the secondary solutions we are interested in should be largely disjoint from the primary solution. We search the set of candidate vertices for the last point in the contour looking for solutions paths that are both good solutions (within some fraction of the likelihood of the best solution) as well as having a set of matching vertices that are significantly different from that of the original path. We use a maximum allowable overlap of 40%. We note that if the contour for the dog's ears is part of a much longer continuous contour then this method will fail, and thus such dual-use areas of contours should ideally be drawn separately.

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 as determined by the emission probability (Figure 3 (e)).

6.3 Non-Silhouette Contours

Contours can be added to emulate high-frequency details such as bumps or valleys present on the 2D model but missing on the template. Our algorithm encorporates those into the deformed model using a mechanism inspired by [Nealen et al. 2005]. We require the user to label these as *non-silhouette contours* and to specify an expected direction for the modified surface normals. The standard HMM mechanism is then used to establish correspondences between these contours and the model vertices. In this case the emission probability is based only on distances instead of distances and normals. Once the correspondences are established, the same deformation framework is used. Normals on the matched vertices have the contour normals as a soft constraint, which reshapes the surface to achieve the desired details (see Figure 9).

7 Deformation

Given the computed correspondences, we apply a deformation mechanism that attracts the matched vertices to their contour counterparts. To generate the required large deformations, it is necessary not only to pull the vertices towards their corresponding contour points, but to also attract the normals at these vertices towards the corresponding contour point normals. Furthermore, soft rather than hard constraints must be used when attracting the vertices and their normals towards the contours in order to maintain the tradeoff between shape preservation and contour matching, particularly as some of the matches may be inaccurate. Linear deformation methods such as [Lipman et al. 2005; Yu et al. 2004; Sorkine et al. 2004] have no obvious extensions that would support soft normal constraints. The mean-value encoding [Kraevoy and Sheffer 2006] provides a closed form formulation which can be augmented to support the required types of constraints and therefore it forms the basis of our deformation approach. We note that other non-linear formulations could also potentially be used.

The mean-value encoding [Kraevoy and Sheffer 2006] 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 using a local Laplacian mesh

$$n_i = \frac{\sum_{k=1}^{m}(v_{j_{k+1}} - l) \times (v_{j_k} - l)}{\|\sum_{k=1}^{m}(v_{j_{k+1}} - l) \times (v_{j_k} - l)\|}, \quad l = \frac{1}{m}\sum_{(i,j)\in E} v_j. \tag{3}$$

To compute the deformed mesh after several vertices are fixed at new locations, the method minimizes

$$\arg\min_{V'} G(V') = \frac{1}{2}\sum_{v_i \in V}(v_i - F_i(v_j, n_i(v_j)))^2 \tag{4}$$

where V' is the set of unconstrained vertices.

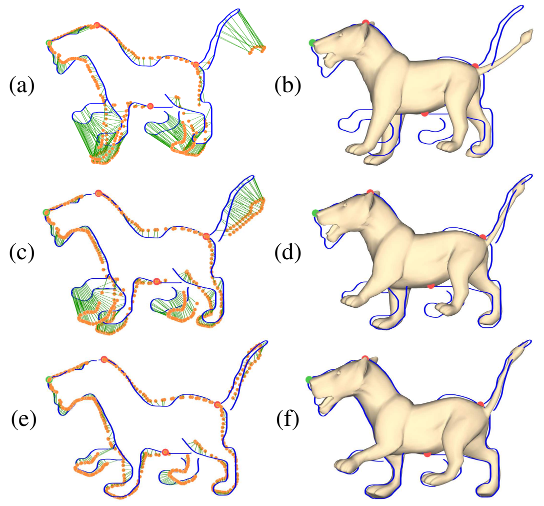


Figure 4: *The iterative correspond-and-deform process: (a) initial correspondences; (b) deformed model after 5 iterations; (c) correspondences after 5 iterations; (d) deformed model after 10 iterations; (e) correspondences after 10 iterations; (f) final fit to contours.*

7.1 Augmenting Template Properties

The mean-value encoding is purely geometric and takes no other model properties into account. To achieve more realistic results, we augment the formulation by considering material propeties and symmetry information when computing the deformed model,

Material Awareness: Most real world objects consist of nonuniform materials; as a result, during deformation the model behaviour is not uniform and depends on the local stiffness of the material. For instance, since humans and animals have stiff bones and flexible joints, pose changes lead to large deformation at the joints and smaller changes elsewhere. Several recent deformation frameworks[Popa et al. 2006; Botsch et al. 2006; Huang et al. 2006b] introduced mechanisms that account for such behaviour. We support spatially-variant deformation behavior by introducing pervertex bending stiffnesses ω_i. The sum in Equation 4 is replaced with a weighted sum

$$\arg\min_{V'} G(V') = \frac{1}{2}\sum_{v_i \in V}\omega_i(v_i - F_i(v_j, n_i(v_j)))^2. \tag{5}$$

The stiffnesses are either painted onto the model or derived from deformation examples, using a similar mechanism to [Popa et al. 2006]. The locally-variable stiffness is particularly useful for modeling changes of pose.

Model Symmetry: Many models can be expected to have symmetric local shape properties, even though their pose is not symmetric. For instance, the Hercules model (Figure 12) should exhibit general left-right symmetry of proportions despite the unsymmetric pose. The symmetry information becomes particularly important if some parts of the model are not visible in the image, or are poorly captured by the contour extraction. This is the case in Figure 12, where Hercules' right arm is hidden behind his back.

The deformation process is augmented to support symmetry in the following way. We assume that the template contains explicit symmetry information, pre-computed by any of the recent detection methods [Simari et al. 2006; Podolak et al. 2006; Mitra et al. 2006].

Given this information, for any vertex which is part of a symmetric region, we have a mapping to its symmetric position. In reflection-symmetry setups this is given by mirroring the vertex on the undeformed template mesh across the reflection plane and projecting it to the closest location on the mesh. The mapping is used after the deformation to obtain *symmetrized* local coordinates for the vertices.

Our method first deforms the model using the standard iterative procedure, disregarding symmetry. After convergence, we compute two sets of new mean-value coordinates for each vertex. First we compute the new deformed coordinates for the vertex using the deformed mesh. For each vertex in a symmetry region we then compute the symmetrized coordinates. The vertex is mapped to its associated symmetric location on the deformed model, together with its neighbouring vertices and the symmetrized mean-value encoding is computed based on the new location.

The deformation algorithm then runs with the same set of constraints as before but with the local coordinates modified as follows. For each pair of regions that should be symmetric, vertices in one region use the new deformed coordinates while vertices in the second region use the symmetrized coordinates. The coordinates are recomputed and the process is then repeated with the sides reversed. Figure 12 shows the deformation results with and without symmetry enforcement.

7.2 Additional Controls

Using the global formulation in Equation 5 it is straight-forward to incorporate the types of constraints we require into the deformation framework. In the standard formulation[Kraevoy and Sheffer 2006], hard, or exact, positional constraints on vertices are enforced by fixing the values for the vertex's coordinates and removing them from the minimization. To enforce hard constraints on only the (x,y) coordinates of the constrained vertices we simply remove only these coordinates from the minimization, instead of removing the full triplet.

To enforce 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 5, 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 computed in Equation 3, and θ is the weight assigned to soft normal constraints.

7.3 Deformation Schedule

Deformations are achieved in an iterative fashion in our framework. This exploits the improved correspondences that can be achieved as the deformation proceeds. The weights ρ and θ are initially small and are increased throughout the iterative matching and deformation process, as the accuracy of the matching increases. The default initial values we used are $\rho = 0.01$ and $\theta = 0.001$ and they are typically increased by 10% at each subsequent iteration. 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 augmented minimization problem is solved using Gauss-Newton optimization combined with a hierarchical mechanism described in [Kraevoy and Sheffer 2006] to obtain a deformed mesh. The method typically requires one or two interior Gauss-Newton iterations to converge for each set of constraints and weights, and a total of ten to twenty outer iterations of matching and deformation to obtain the final result. The entire process takes two to ten minutes depending on the size of the template mesh and the deformation complexity.

8 Texture Lifting

The original image can be used as a texture map by using the (x,y) projection of the final deformed model vertices to obtain corresponding texture coordinates. In order to avoid picking up background areas of the image where the model does not fully adhere to the target contours, vertices that lie outside or near the contour are offset inwards by a fixed amount. Alternatively, the most recent correspondence information could also be exploited to compute a more informed estimate of the final texture map coordinates. Texture map coordinates for triangles that are not visible in the image view are also computed using simple projection. If symmetry information is available, it can be used to infer texture for hidden parts from their visible counterparts. The texture seam along the silhouette is partially obscured by blurring the texture in areas near the seams. Texture synthesis could likely further improve the texture in the silhouette regions.

9 Results

We illustrate our method on a wide variety of models, including everyday objects (creamer cup, football, teddy-bear), animals (cartoon lion, panther, dog), and humans (gymnast, hercules, horse tamer, caricature head). We test the method on both profile and non-profile (hercules, horse tamer) views. The influence of the template on the final model shape is illustrated by applying the same template to different contour images (panther, cartoon lion). The ability to use contours from multiple views of the same object is illustrated on the teddy example. These examples can also be seen in the accompanying video.

The football (Figure 5) provides a simple scenario for testing the correspondence and deformation processes. The template model is a triangulated sphere and the contour is a hand-traced outline of the football. The contours do not provide depth information and thus the circular profile of the ball needs to come from the template and should ideally be preserved throughout the deformation process. The side view shown in Figure 5(d) shows that this is indeed the case. The template sphere does not have any embedded symmetry information for this example.

The creamer shown in Figure 6 shows an example that is challenging 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 silhouette contours. Additionally, the template and target contours are significantly different in shape. The creamer image is taken from a vantage point slightly above the rim and thus the inside and outside edges of the creamer upper rim are visible. In this case, the algorithm makes a reasonable choice in matching the outer silhouette line, which belongs to the far edge of the creamer. From the source image, it cannot be determined the extent to which the creamer has a circular profile. The model resulting from our deformation process has a near-circular profile near the bottom and a more elliptical profile near the top. In the absence of more information, this is a reasonable reconstruction of the estimated. There is no symmetry information embedded in the template model. 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.

Contour-based modeling provides an effective way to model the shape of animals. Our first example is that of the caricature lion shown in Figures 1, 2, and 4. 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 7 is constructed using the same template model as the lion. A comparison of the shape of the final lion model and the final panther model illustrates the significant variations in shape that can be inferred from contours. The panther's tail is distorted in the initial registration step, but is restored over multiple iterations of the deformation process. The dog in Figure 8 is another example that demonstrates the effect of contours in changing pose and proportions. The head, tail, and torso rear are significantly different.

We create a caricatured head model from a drawing of a head (Figure 9). This model presents an interesting test case for how contour-based edits from a single viewpoint can be used to achieve realistic deformation to the 3D shape. The caricature makes significant changes to the shape of the head, the shape of the nose, and the jaw line. There are no contour lines for the ears and thus the ear location, shape, and size are derived from the template model. Figure 9(f) shows the result of using non-silhouette contours to create the prominent cheekbones, nostrils, and lips that are evident in the original image. The front view of the final model and the template model shows that the caricatured 3D model has correctly preserved the 3D shape of the head while enlarging the nose width to match the enlarged nose in the original drawing.

The next example uses contours extracted from two views to reconstruct a teddy-bear model (Figure 10). Multiple views can be used simultaneously, or in sequence. We adopt a sequential strategy. 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.

Achieving correct poses and proportions is a challenging problem for image-based modeling of human figures. To demonstrate that contour based modeling is an effective tool for this task, we apply it to create models from a gymnast drawing and photographs of two different statues. Figure 11 shows the results for the female gymnast. Note that the arms are posed using incomplete contours. The correct correspondences are established on the head and result in it facing upwards as in the drawing.

Our last two examples are constructed from photographs of statues. Figure 12 shows a muscle-bound Hercules recreated from image contour information. The final model is different from the template in both pose and proportions. In particular, the Hercules statue has large muscle-bound arms and legs as reflected in the final model. The template model used for this example has relatively little surface detail, and muscle ripples are added on the abdomen and arm using non-silhouette contours. The default solution places Hercules' right arm slightly in front of the body in a statuesque pose (Figure 12(e)). Hercules' right arm is also much too small when compared to the left arm, which is large and muscular due to its guiding contour information. Adding symmetry information to the template makes the proportions of the two arms the same, as shown in Figure 12(f). Lastly, a further constraint placed on the depth value of one hand vertex drives the arm to the correct position behind the body (Figure 12(g)). Even though we do not explicitly disallow interpenetrations, this was not observed for our example models.

The horse-tamer statue (Figure 13) is a highly oblique view of a human. A more detailed template model is used in this case. The final model strikes a dramatic pose. The depth of the statue's right arm and right leg are ambiguous and thus we specify a desired depth for one vertex on the arm and one vertex on the leg.

10 Conclusions

The algorithms presented provide a powerful framework for developing tailored 3D models from images using only contour information and a flexible process for deforming 3D template models. This is a challenging problem because of the limited and ambiguous nature of the contour information. The HMM-based correspondence model provides the reliable correspondence information that is at the heart of each iteration of the deformation process. The template model itself also includes information that supports reliable model construction, including symmetry information and material stiffness properties.

There are two general directions for improving on this kind of image-based modeling work. One direction looks towards extracting more information from images. Shading or other information could be used to help inform the shape. Additional correspondences could perhaps be automatically identified. It may be possible to combine our work with recent work on object-classification in order to automatically identify the right template model.

Another direction looks are including more information into the template. For humans and animals, their known skeletal structure 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. Templates could be made to contain information about parts that could be instanced on demand. Templates for more geometric objects could contain information about the precise ways in which their shape can be expected to parameterize.

References

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: shape completion and animation of people. *Proceedings of ACM SIGGRAPH 2005 24*, 3, 408–416.

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

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

COOTES, T. F., TAYLOR, C. J., COOPER, D. H., AND GRAHAM, J. 1995. Active shape models their training and application. *Comput. Vis. Image Underst. 61*, 1, 38–59.

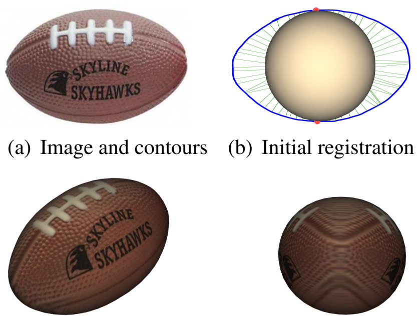
DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. 11–20.

GORMAN, J., MITCHELL, O., AND KUHL, F. 1988. Partial shape recognition using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 2, 257–266.

HE, Y., AND KUNDU, A. 1991. 2-d shape classification using hidden markov model. *IEEE Trans. Pattern Anal. Mach. Intell. 13*, 11, 1172–1184.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3, 1126–1134.

HUANG, J., ZHANG, H., SHI, X., LIU, X., AND BAO, H. 2006. Interactive mesh deformation with pseudo material effects. *Comput. Animat. Virtual Worlds 17*, 3-4, 383–392.



(a) Image and contours (b) Initial registration

(c) 3D model (d) Side view

Figure 5: *Football*

Kho, Y., and Garland, M. 2005. Sketching mesh deformations. In *SI3D '05: Proceedings of the 2005 Symposium on Interactive 3D graphics and games*, 147–154.

Kraevoy, V., and Sheffer, A. 2006. Mean-value geometry encoding. *International Journal of Shape Modeling (to appear)*.

Lipman, Y., Sorkine, O., Levin, D., and Cohen-Or, D. 2005. Linear rotation-invariant coordinates for meshes. *Proceedings of ACM SIGGRAPH 2005 24*, 3, 479–487.

Lowe, D. G. 1991. Fitting parameterized three-dimensional models to images. *IEEE Trans. Pattern Anal. Mach. Intell. 13*, 5, 441–450.

McInerney, T., and Terzopoulos, D. 1996. Deformable models in medical images analysis: a survey. *Medical Image Analysis 1*, 2, 91–108.

Mitra, N. J., Guibas, L. J., and Pauly, M. 2006. Partial and approximate symmetry detection for 3d geometry. In *ACM Transaction on Graphics, Proc. of SIGGRAPH '06*, 560–568.

Nealen, A., Sorkine, O., Alexa, M., and Cohen-Or, D. 2005. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph. 24*, 3, 1142–1147.

Podolak, J., Shilane, P., Golovinskiy, A., Rusinkiewicz, S., and Funkhouser, T. 2006. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (TOG) 25*, 3, 549–559.

Popa, T., Julius, D., and Sheffer, A. 2006. Material aware mesh deformations. In *Shape Modeling International*.

Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., and Kang, S. B. 2006. Image-based plant modeling. *ACM Trans. Graph. 25*, 3, 599–604.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE 77*, 2, 257–286.

Simari, P., Kalogerakis, E., and Singh, K. 2006. Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Symposium on Geometric Processing*.

Simmons, M., Wilhelms, J., and Van Gelder, A. 2002. Model-based reconstruction for creature animation. *Proc. Symp. on Computer Animation 2002*, 139–146.

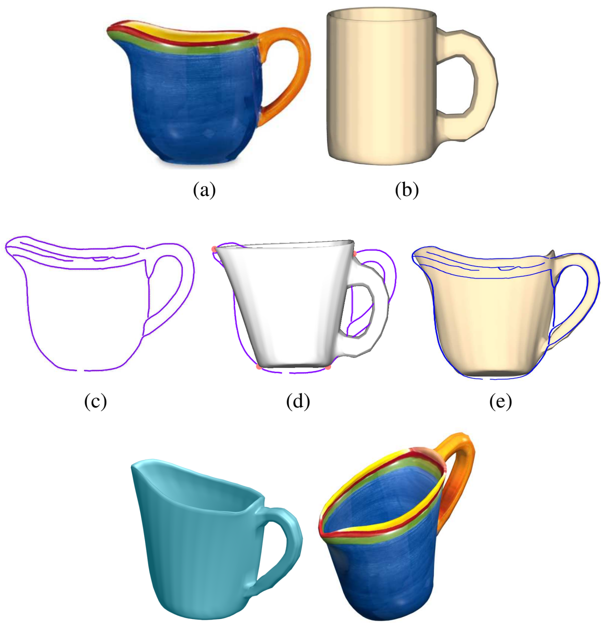
Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. 2004. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 175–184.

Terzopoulos, D., and Metaxas, D. 1991. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE PAMI 13*, 7, 703–714.

Yang, C., Sharon, D., and van de Panne, M. 2005. Sketch-based modeling of parameterized objects. In *Proceedings of Eurographics Workshop on Sketch Based Interfaces and Modeling*.

Yoshizawa, S., Belyaev, A. G., and Seidel, H. P. 2003. Free-form skeleton-driven mesh deformations. In *Proc. 8th ACM Symp. on Solid Modeling and Applications*, 247–253.

Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., and Shum, H. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. on Graphics 23*, 3, 644–651.

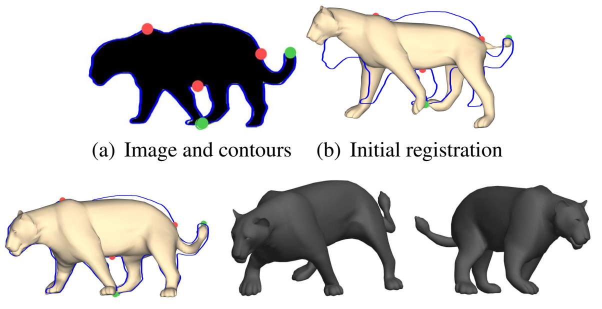


(a) (b)

(c) (d) (e)

(f) 3D model

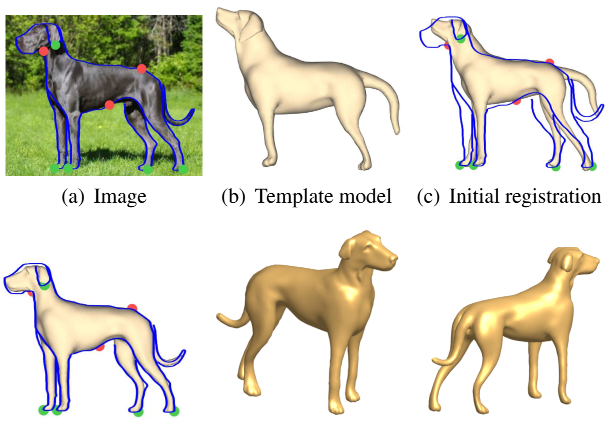
Figure 6: *Creamer: (a) Image; (b) Template model; (c) Contours from edge detection; (d) Initial registration; (e) Final fit; (f) 3D model.*



(a) Image and contours (b) Initial registration

(c) Final fit (d) 3D model

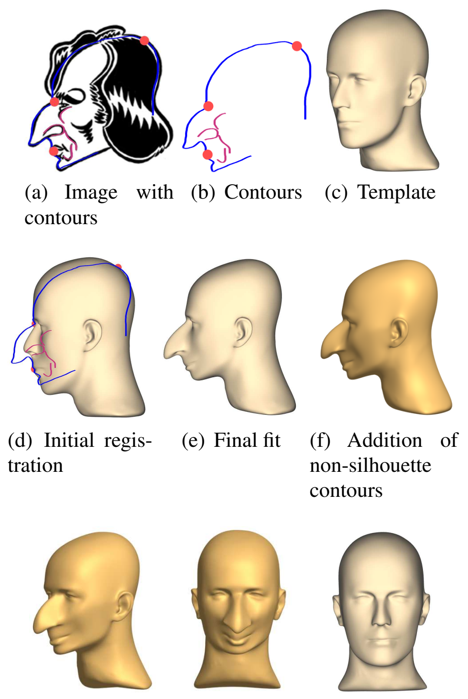
Figure 7: *Panther*



(a) Image (b) Template model (c) Initial registration

(d) Final fit (e) 3D model

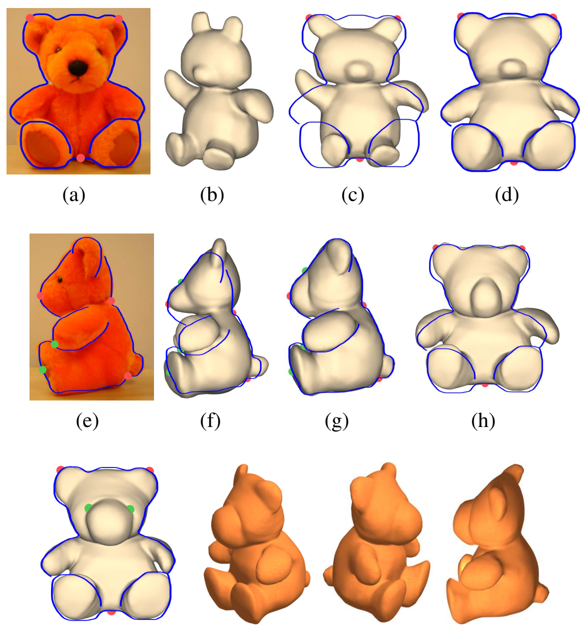
Figure 8: *Dog*


(a) Image with contours (b) Contours (c) Template

(d) Initial registration (e) Final fit (f) Addition of non-silhouette contours

(g) Another view (h) Front view (i) Template front view

Figure 9: *Caricature head*


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

(e) (f) (g) (h)

(i) (j)

Figure 10: *Teddy bear: (a) View A; (b) Template model; (c) Initial registration to view A; (d) Final fit to view A; (e) View B; (f) Iniital registration to view B; (g) Final fit to view B; (h) Initial re-registration to view A; (i) Final fit to view A; (j) 3D model.*

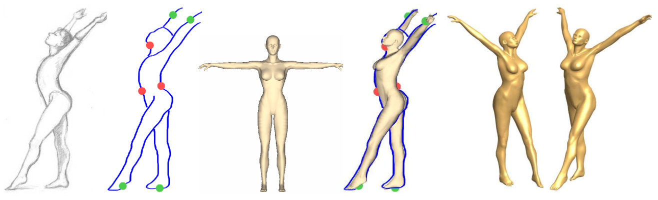
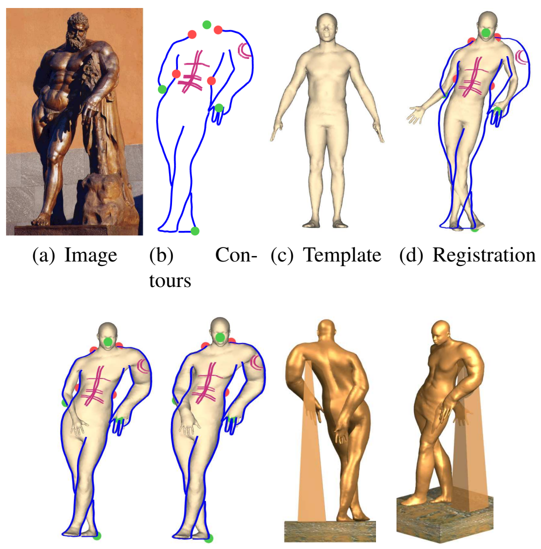
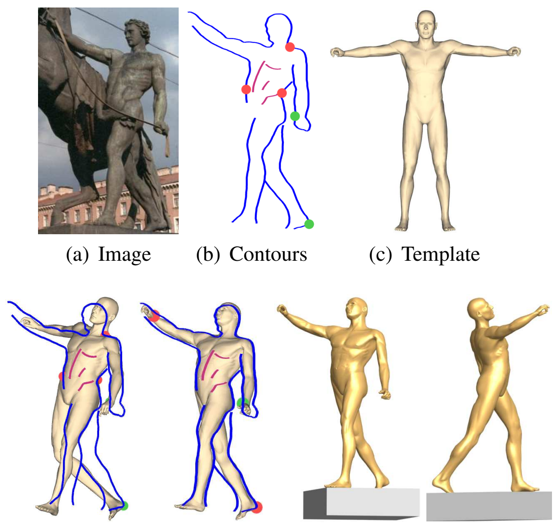

(a) Image (b) Contours (c) Template (d) Final fit (e) 3D model

Figure 11: *Gymnast*


(a) Image (b) Contours (c) Template (d) Registration

(e) Fit, no symmetry (f) Fit with symmetry (g) 3D model, added hand constraints

Figure 12: *Hercules*


(a) Image (b) Contours (c) Template

(d) Initial registration (e) Final fit (f) 3D model

Figure 13: *Horse tamer*