Morphable Surface Models *

Christian R. Shelton
Massachusetts Institute of Technology

February 23, 2000

Abstract. We describe a novel automatic technique for finding a dense correspondence between a pair of n-dimensional surfaces with arbitrary topologies. This method employs a different formulation than previous correspondence algorithms (such as optical flow) and includes images as a special case. We use this correspondence algorithm to build Morphable Surface Models (an extension of Morphable Models) from examples. We present a method for matching the model to new surfaces and demonstrate their use for analysis, synthesis, and clustering.

1. Introduction

The goal of this paper is to describe a general method for learning models of surface classes without user intervention. The technique works on surfaces of any dimension embedded in a Euclidean space and is not specific to any particular modality. The Morphable Surface Models of this paper are a generalization of Morphable Models (described below).

The key problem in building such models is finding correspondences between surfaces. We define a correspondence to be a relation from points on one object to their matching points on the other object. In images such relations are often called flow fields and have found a wide variety of uses. Here we extend the notion from images to surfaces in general[1] and employ the correspondence algorithm to build Morphable Surface Models.

The remainder of this introduction describes previous work in Morphable Models and surface matching. Section 2 describes the correspondence algorithm by first describing the minimization problem, then the algorithm for performing the minimization, and finally the role of surface simplification. In section 3, we describe how to use the

* This paper describes research done within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences and at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. This research is sponsored by grants from the National Science Foundation, ONR and Darpa. Additional support is provided by Eastman Kodak Company, DaimlerChrysler, Siemens, ATR, AT&T, Compaq, Honda R&D Co., Ltd., Merrill-Lynch, NTT and Central Research Institute of Electric Power Industry.

[1] A gray scale image can be described as a surface or height-field. Color images can similarly be described (now in 5-dimensional space instead of 3-dimensional space).

© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

correspondence algorithm to build models. To do so, we introduce a method for matching the models to surfaces, an inner product on the space of correspondences, and how to apply bootstrapping. Section 4 describes experimental results obtained by building models of various types of surfaces and gives an illustration of the steps of the correspondence algorithm. Lastly, section 5 gives some conclusions and possible extensions.

1.1. Morphable Models

It is known that linear models in the image space are generally poor models for classes of images. Such a model describes members of the class as alpha-blendings of the examples. This in general produces poor results. Unless the images align pixel-for-pixel, the alpha-blending of two images will not produce a third image in the same class of images: instead the synthesized image will be two "ghosts" of the combined images.

The difficulty lies in the fact that every pixel of the new image is a linear combination of the pixels at the same location in the other images. These other pixels may not be related to each other and thus taking a linear combination of them will produce nothing of worth. However, if we can match "corresponding" pixels across the set of example images, then linear combinations of these corresponding pixels will work better as a model (Beymer and Poggio, 1996).

For this paper we define a correspondence to be a relation between two objects that maps each of the points on one object to its "corresponding" point on the other object (where we appeal to intuition for the definition of corresponding). For images, a correspondence is simply a flow field from one image to another. Morphable Models are constructed from a set of example images by finding flow fields from an arbitrary base example image to all of the other example images. As a generative model, the output is the base image warped by a linear combination of the flow fields. The parameters of the model are the weights of the linear combination. The pixel values aligned by the correspondences may also be combined linearly to produce an image that is not only a combination of the "shapes" of the examples (the flow fields) but also the "texture" of the examples (the pixel values).

In this paper, we consider a more general morphable model. Instead of dealing only with images, we allow the objects to be surfaces of arbitrary dimension embedded in Euclidean space. Images are a particular example of such a surface (a gray-scale image is a two-dimensional manifold in three-dimensional space: one dimension for each image axis and one dimension for intensity). Thus we will build a general

correspondence technique for any pair of surfaces (replacing the flow field algorithm for images) and, from that, a general technique for building morphable models of arbitrary surfaces. At the end of this paper, we will show some simple examples of the uses of Morphable Surface Models.

1.2. Related Work

Morphable Models have been applied mainly to images. Jones and Poggio (1998) has a good description of their image method and related image-based techniques. We would refer to their introduction as the best description of the Morphable Model literature. Briefly, the result that new views of a 3D object can be synthesized linearly from three sample views (Ullman and Basri, 1991; Shashua, 1992) spurred the creation of models of image classes as linear combinations of examples (Poggio and Vetter, 1992; Vetter and Poggio, 1995; Beymer et al., 1993; Beymer and Poggio, 1996).

Since Jones and Poggio (1998), Vetter et al. (1997) developed a bootstrapping technique (used and redescribed in this paper) for better automatic construction of Morphable Models. Recently, Blanz and Vetter (1999) extended Morphable Models to 3D shapes by describing the shape as its projection onto a 2D surface (which can then be unrolled and treated as an image). From there they were able to use Morphable Models to reconstruct 3D shape from a single view. Kang and Jones also have used such a projection technique to constrain reconstruction.

Active Appearance Models (Cootes et al., 1998) also try to match images but require user specified correspondences. Deformable Intensity Surfaces (Nastar et al., 1996) treat images as surfaces and use an automatic matching technique not dissimilar to the one described here. However, the topology of the surface is essentially limited to that of a grid. Active Contour Models (or Snakes) (Kass et al., 1988; McInerney and Terzopoulos, 1995) are also similar to the work in this paper. They operate on surfaces of arbitrary dimensions and topologies. However, they are used to match a surface to gradients in volumetric data instead of to other surfaces.

In this paper we have used the term "n-dimensional surface" to mean a surface with n orthogonal tangent vectors at every point. In Active Contour Models this phrase is taken to imply that the surface lies in an n-dimensional space (and the surface itself has a dimensionality less than n). We have used the phrase "dimensionality of the embedded space" to denote such a quantity.

The difficult part of building a morphable model and many other modeling techniques is finding correspondences. We believe this is the

first paper to propose an algorithm for automatically finding surface correspondences between surfaces with arbitrary topologies directly. The energy minimization method here is based on that described in (Shelton, 1998), has certain similarities to elastic networks (Durbin and Willshaw, 1987), and has energy terms related to the surface reconstruction work of Hoppe and others (Hoppe et al., 1992; Hoppe et al., 1993).

2. Matching Surfaces

To build a Morphable Surface Model, we must first be able to find the correspondences between the example surfaces. This will serve as the foundation for model construction algorithm. We define an energy function over all possible correspondence relations for which smaller values indicate better correspondences. We will then give an algorithm for finding a minimum of this energy function which will hopefully correspond to a good match according to the metric encoded in the energy function. Crucial to the practical success of this minimization is the mesh simplification described in section 2.4.

2.1. Energy Function

Let C be a function which maps points on one surface, $\mathcal{A}$, to arbitrary points in space. In order for C to be a good correspondence from $\mathcal{A}$ to $\mathcal{B}$ we propose it must satisfy three properties:

1. Similarity: For every point a on the surface $\mathcal{A}$, $C(a)$ should be near or on the surface $\mathcal{B}$.
2. Structure: C should distort the surface $\mathcal{A}$ as little as possible. Put differently, $C(\mathcal{A})$ should be as structurally similar to $\mathcal{A}$ as possible.
3. Prior Information: C should represent a plausible deformation of the surface.

The first property states that C should be a correspondence and actually match points on $\mathcal{A}$ to points on $\mathcal{B}$. The second says that such a matching should not be arbitrary, but rather should attempt to keep the structure of the first surface. This will hopefully force matching of similar substructures from $\mathcal{A}$ to those of $\mathcal{B}$ and preserve our intuitive notion of correspondence. The last term serves to enforce prior knowledge about valid shapes (for example, we may wish to penalize surfaces which cross a lot and are very unsmooth as being unlikely results from any correspondence).

Our energy function will therefore have the following form:

$$E(C) = E_{sim}(C) + \alpha E_{str}(C) + \beta E_{pri}(C) \tag{1}$$

E_{sim} measures how closely C matches points on $\mathcal{A}$ to points on $\mathcal{B}$. To ease notation, let us define $P_{\mathcal{X}}(q)$ to be the point on the surface $\mathcal{X}$ closest to the point q. We will then define E_{sim} as

$$E_{sim}(C) = \int_{C(\mathcal{A})} \|c - P_{\mathcal{B}}(c)\|^2 \, dc + \int_{\mathcal{B}} \left\|b - P_{C(\mathcal{A})}(b)\right\|^2 \, db \tag{2}$$

This is the sum of the integrals over each surface of the squared distance from points on that surface to the other surface.

The definitions of the remaining two terms of the energy function will depend on prior information we have about surfaces. More sophisticated methods of measuring the distortion of the surface could be developed and used. However, we have found that connecting springs between points on the surface to be sufficient. In particular we will use "directional" springs that prefer their original orientation (not just their original length). There are two reasons for this choice: First, they tend to minimize buckling of the surface more than "regular" springs (which actually encourage it) and second, they lead to a nice form for the minimization algorithm since they are quadratic. For a directional spring connecting the two points p and q, the energy of that spring under the correspondence C is

$$E_{ds}(p, q, C) = \frac{\|(p - q) - (C(p) - C(q))\|^2}{\|p - q\|} \tag{3}$$

To create a tight surface and an energy function which does not depend on the parameterization of the surface, we let

$$E_{str}(C) = \int_{\mathcal{A}} E_{ds}(a, a + da, C) \tag{4}$$

This corresponds to placing directional springs continuously over the entire surface.

If we let $E_s(p, q, C)$ be similarly defined for a normal spring of rest length 0,

$$E_s(p, q, C) = \frac{\|C(p) - C(q)\|^2}{\|p - q\|} \tag{5}$$

in order to penalize discontinuous surfaces, we define E_{pri} to be

$$E_{pri}(C) = \int_{\mathcal{A}} E_s(a, a + da, C) \tag{6}$$

The three terms of $E(C)$ do not play equal roles. The first, E_{sim}, must be zero for C to be a true correspondence. The second, E_{str}, exists to guide and select among C functions which set the first term to zero. The last term, E_{pri}, serves to smooth out any noise which may arise due to inaccurate models or an inability to find the true correspondence; as a prior over surfaces, it models the fact that very unlikely correspondences should be discarded in favor of more likely ones which perhaps do not satisfy the other requirements as well.

Thus, the weighting of the various terms in $E(C)$ will be set as follows: α will be initially large to enforce a good correspondence. Over time, it will be reduced to allow the algorithm to drive E_{sim} to zero. β will be set to a small constant value which is inversely proportional to our confidence in our model and ability to find the correct correspondence.

2.2. Practical Instantiation of the Energy Function

The energy function as described in the previous section is not practical to use. The integrals, for most surfaces, are intractable to compute. In this paper, we consider only piece-wise linear surfaces and can therefore make a number of useful simplifications. For a d-dimensional surface (meaning that there are d orthogonal tangent vectors at every point on the surface), the surface can be defined as an ordered collection of vectors (the positions of the vertices of the surface) and a set of linear patches connecting $d+1$ vectors from the collection. Hoppe et al. (1993) gives a more mathematically concrete definition of such a structure (a tuple of the vertex positions and a simplicial complex). A triangulated mesh is an example of such a 2-dimensional surface.

The integrals in E_{sim} are easily and well approximated by a uniform stochastic sampling of their respective surfaces. Thus, we have

$$\hat{E}_{sim} = \frac{1}{n} \sum_{c \in S_n(C(\mathcal{A}))} \|c - P_{\mathcal{B}}(c)\|^2 + \frac{1}{n} \sum_{b \in S_n(\mathcal{B})} \left\|b - P_{C(\mathcal{A})}(b)\right\|^2 \qquad (7)$$

where $S_n(\mathcal{X})$ is a set of n points sampled from the surface $\mathcal{X}$.

$\hat{E}_{sim}$ is therefore independent of the parameterization of the surface. In other words, given two different tessellations of the same surface, $\hat{E}_{sim}$ will not change. We have not yet found similarly parameter-independent approximations of E_{str} and E_{pri} which yield simple algorithms. Instead we approximate both by connecting adjacent vertices in the surface description with the appropriate springs. Since the springs defined above are normalized by length, a given straight length of spring will be independent of how it is cut (*i.e.* how many sections are used to describe it). Yet, past 1-dimensional surfaces, the energy terms as

a whole will not be independent of the tessellation used. However, we have found it to work well in practice.

$$\hat{E}_{str} = \frac{1}{|Ad(\mathcal{A})|} \sum_{(p,q)\in Ad(\mathcal{A})} E_{ds}(p,q,C) \tag{8}$$

$$\hat{E}_{pri} = \frac{1}{|Ad(\mathcal{A})|} \sum_{(p,q)\in Ad(\mathcal{A})} E_{s}(p,q,C) \tag{9}$$

where $Ad(\mathcal{A})$ is the set of all pairs of adjacent vertices in the surface tessellation $\mathcal{A}$.

2.3. Energy Minimization

Just as we limited the surface to be piece-wise linear, we will limit the correspondence function to be piece-wise linear. In particular, we will allow C to take arbitrary values at the vertices of $\mathcal{A}$ and require it to be a linear interpolation of the vertex values along the faces of $\mathcal{A}$. This means that applying C to $\mathcal{A}$ simply involves applying the elements of C to their respective vertices of $\mathcal{A}$: no structural or topological changes need be made to $\mathcal{A}$.

With this parameterization of C, $\hat{E}_{str}$ and $\hat{E}_{pri}$ are clearly quadratic in the elements of C. However, $\hat{E}_{sim}$ is not so well behaved due to the unsmooth nature of the function P (whose first derivative is discontinuous). We therefore turn the algorithm into a dual optimization. First, we compute $P_{\mathcal{B}}(c)$ and $P_{C(\mathcal{A})}(b)$ for all of the a and b samples. We then fix them at which point $\hat{E}_{sim}$ becomes quadratic (and thus $\hat{E}(C)$ is quadratic) and we can solve this minimization for C in closed form. This gives a new set of positions for the vertices from which we can repeat the sampling, calculation of the closest points, and values of C.

To be more concrete about the algorithm, let us first note that any point on the surface of $\mathcal{X}$ can be described as a convex combination of the positions of the vertices of $\mathcal{X}$. For vertices (such as p and q in $\hat{E}_{str}$ and $\hat{E}_{pri}$), the combination coefficients are all 0 except for a single 1. For points internal to a face (such as c or $P_{C(\mathcal{A})}(b)$ in $\hat{E}_{sim}$), the coefficients are all zero except for $d+1$ non-zero elements (which sum to 1 due to convexity) corresponding to the $d+1$ vertices of the face on which the point is located. These coefficients are sometimes called the barycentric coordinates of the point (Hoppe et al., 1993). The barycentric coordinates are invariant to the transformations we are allowing for C. That is, the barycentric coordinates of a in $\mathcal{A}$ are the same as the barycentric coordinates of $C(a)$ in $C(\mathcal{A})$.

We will now add the notation that any variable with an overscore is not a Euclidean vector in the embedded space of the surface, but a

barycentric coordinate vector (a vector of the coefficients of the previous paragraph) with respect to the appropriate surface. Furthermore, we will let the matrix A be the matrix whose columns are the vectors of the positions of the vertices of $\mathcal{A}$. B will be similarly defined for $\mathcal{B}$ and C will be the matrix whose columns are the positions of the vertices of $\mathcal{A}$ under the correspondence. Thus if $\mathcal{A}$ has v_A vertices, $\mathcal{B}$ has v_B vertices, and they are embedded in a D-dimensional space, A and C are $D \times v_A$ and B is $D \times v_B$.

By expanding $\hat{E}(C)$ by using the relationships $b = B\overline{b}$, $a = A\overline{a}$, and $C(a) = C\overline{a}$, we can then write the whole energy function as

$$\begin{aligned}\hat{E}(C) = \frac{1}{n} &\sum_{\overline{c}\in\overline{S}_n(C(\mathcal{A}))} \left\|C\overline{c} - B\overline{P}_{\mathcal{B}}(C\overline{c})\right\|^2 \\ &+ \frac{1}{n} \sum_{\overline{b}\in\overline{S}_n(\mathcal{B})} \left\|B\overline{b} - C\overline{P}_{C(\mathcal{A})}(B\overline{b})\right\|^2 \\ &+ \frac{\alpha}{|Ad(\mathcal{A})|} \sum_{(\overline{p},\overline{q})\in Ad(\mathcal{A})} \frac{1}{\|A(\overline{p}-\overline{q})\|} \|(C-A)(\overline{p}-\overline{q})\|^2 \\ &+ \frac{\beta}{|Ad(\mathcal{A})|} \sum_{(\overline{p},\overline{q})\in Ad(\mathcal{A})} \frac{1}{\|A(\overline{p}-\overline{q})\|} \|C(\overline{p}-\overline{q})\|^2\end{aligned} \tag{10}$$

where, to carry the overscore notation further, $\overline{S}$ is a set of barycentric coordinate vectors (of points sampled uniformly over the surface as before) and $\overline{P}$ is the barycentric coordinate vector of the closest point on the surface. Note that if we fix a sampling over each surface and we fix the values of $\overline{P}$ (they actually depend on C, but we are fixing them), then every term of $\hat{E}$ is of the form $w_i \|Cc_i - d_i\|^2$ (w_i is a scalar, C is a matrix of vertex positions, c_i is the barycentric coordinate vector for the constraint, and d_i is the target position).

This optimization can be solved by converting it to a sparse linear-least squares problem separately for each dimension (Hoppe et al., 1993). However, we have found that an improvement can be made at this stage before conversion to a linear system. At each step of the minimization, points are sampled from each surface and for each point, the closest point on the other surface is found. Then, $\hat{E}$ is minimized assuming that the goal is to minimize the distance from each sampled point to the found closest point. However, a better match might be found if, instead of insisting that the point match the closest point, we relax and allow the point to match any position on the closest face. This is closer to our goal of allowing the point to match anywhere on the surface. Since the closest face is bounded, we have to approximate this

by minimizing the distance from the point to the plane of the closest face.

Thus, instead of minimizing $w_i \left\| C\overline{a}_i - B\overline{P}_i \right\|^2$ for points sampled from $\mathcal{A}$ or $w_i \left\| C\overline{P}_i - B\overline{b}_i \right\|^2$ for points sampled from $\mathcal{B}$, we are minimizing $w_i \left\| T_i C\overline{a}_i - T_i B\overline{P}_i \right\|^2$ and $w_i \left\| T_i C\overline{P}_i - T_i B\overline{b}_i \right\|^2$ where T_i is a matrix which projects out the tangent directions for the face on $\mathcal{B}$ involved in the ith constraint. Depending on whether the sampled or projected point on $\mathcal{B}$ falls on a vertex, edge, or face, there could be from 0 to d orthogonal tangent vectors at that point on $\mathcal{B}$. If we enumerate them as the unit length vectors $t_{i,1}, \ldots, t_{i,m}$,

$$T_i = I - \sum_m t_{i,m} t_{i,m}^T \tag{11}$$

(If instead of allowing the point to be anywhere on the plane of the face, we wish to penalize moving from the closest point slightly, we can place a coefficient between 0 and 1 in front of the sum in the above equation thus resulting in a slightly different distance metric.)

Since the constraints now have terms of the form $(T_i C a_i)^T (T_i C a_i)$, they cannot be converted into a sparse linear least-squares system independently for each dimension. Instead, we must solve for all dimensions simultaneously (unless all T_i matrices are diagonal). However, with a bit of careful manipulation, we can derive a different sparse system to solve.

Our energy function now has the form

$$\hat{E} = \sum_i w_i (T_i C a_i - q_i)^2 \tag{12}$$

where w_i, T_i, a_i, and q_i are different depending on the constraint: for the first n constraints, $w_i = \frac{1}{n}$, a_i is the barycentric coordinate vector of a point sampled from $\mathcal{A}$, q_i is the closest point on $\mathcal{B}$ to this sampled point, and T_i is the tangent matrix at the point q_i. For the second n constraints, $w_i = \frac{1}{n}$, q_i is a sampled point on $\mathcal{B}$, T_i is the tangent matrix to this point, and a_i is the barycentric coordinate vector of the point on $\mathcal{A}$ closest to q_i. For the next $|Ad(\mathcal{A})|$ constraints, $w_i = \frac{\alpha}{|Ad(\mathcal{A})| \|A(\overline{p}-\overline{q})\|}$, $T_i = I$, q_i is the difference between the positions of the two vertices in $\mathcal{A}$, and a_i is the difference between the barycentric coordinate vectors of the two vertices (a vector with one 1 and one -1). Finally, the last $|Ad(\mathcal{A})|$ constraints are exactly the same as the previous $|Ad(\mathcal{A})|$ except the α is replaced by β in w_i and $q_i = 0$.

For the proper definition of the matrix R_i (a function of T_i and a_i) and the vector c (a function of C) as derived below, this can be

converted into

$$\hat{E} = \sum_i (R_i c - q_i)^2 \tag{13}$$

which is a classic least-squares problem with the solution

$$c = \left(\sum_i R_i^T R_i \right)^{-1} \sum_i R_i^T q_i \tag{14}$$

First we will define c as shown pictorially below to be the vector of the components of the matrix C:

$$c^T = \begin{bmatrix} C_{1,1} & \cdots & C_{D,1} & C_{1,2} & \cdots & C_{D,2} & \cdots & \cdots & C_{1,N} & \cdots & C_{D,N} \end{bmatrix} \tag{15}$$

where N is the total number of constraints.

We now define R_i as:

$$(R_i)_{j,D(k-1)+l} = (T_i)_{j,l} (a_i)_k \tag{16}$$

which means that R_i has the block form:

$$R_i = \begin{bmatrix} (a_i)_1 T_i & (a_i)_2 T_i & \cdots & (a_i)_N T_i \end{bmatrix} \tag{17}$$

(remember that $(a_i)_k$ is the kth element of the vector a_i and thus a scalar). It can easily be shown from here that $R_i c = T_i C a_i$. We now have a simple linear least-squares problem with the solution shown in equation 14.

A nice property of this formulation is that R_i is sparse. Only up to $d+1$ elements of a_i may be non-zero and thus only a maximum of $d+1$ of the blocks of R_i are non-zero. Many of the T_i matrices are the identity matrix (for all of the spring constraints) and thus many more of the elements of R_i will be zero for these constraints. So, while $R_i^T R_i$ is a $DN \times DN$ matrix, only a maximum of $D^2(d+1)^2$ elements of it will be non-zero. Furthermore, these $D \times D$ blocks that make up $R_i^T R_i$ can only be non-zero for blocks that correspond to adjacent vertices in $\mathcal{A}$. Thus the matrix $\sum_i R_i^T R_i$ that needs to be inverted (or at least for which an LU-decomposition needs to be found) in equation 14 is very sparse. It is symmetric and on every row, a maximum of Dg elements are non-zero (where g is the maximum outgoing degree of any vertex in $\mathcal{A}$). The solution therefore can be computed efficiently using sparse inversion techniques such as conjugate gradient descent (Press et al., 1992).

2.4. Surface Simplification

A vital component in our minimization algorithm is the simplification of the surface descriptions. Attempting to minimize the energy function as described in the previous section directly on the original models $\mathcal{A}$ and $\mathcal{B}$ can result in severely suboptimal local minima and long running times. To combat both of these problems, we first perform a sequence of surface simplifications. For $\mathcal{A}$ we produce the sequence of surfaces $\{\mathcal{A}^0, \mathcal{A}^1, \ldots, \mathcal{A}^{l_A}\}$ and for $\mathcal{B}$ we produce the sequence $\{\mathcal{B}^0, \mathcal{B}^1, \ldots, \mathcal{B}^{l_B}\}$ where $\mathcal{A}^0$ and $\mathcal{B}^0$ are the original input surfaces and $\mathcal{X}^{i+1}$ is a mesh approximating $\mathcal{X}$ with half the complexity of $\mathcal{X}^i$. Complexity can be any measure; in this case we use the number of vertices. l_A and l_B are determined by setting a threshold on the maximum distance from the simplified surface to the original surface. A single threshold works well for all similar models.

The computer graphics literature has a number of algorithms that can be used to produce simpler meshes approximating an input mesh. Heckbert and Garland (1997) gives a good review of the major algorithms of the field. We have implemented and used progressive meshes (Hoppe, 1996) and quadric error metrics (Garland and Heckbert, 1997) and both provide qualitatively the same performance when used in our algorithm on the types of surfaces described in the experimental section of this paper. However, our implementation of Garland's and Heckbert's quadric error metrics runs faster and so the results reported in this paper use that algorithm.

Once we have the sequence of simplified surfaces, we begin with the two simplest surfaces ($\mathcal{A}^{l_A}$ and $\mathcal{B}^{l_B}$) and use the energy minimization technique from the previous section to find a correspondence between the two surfaces. We then use the found correspondence to initialize the starting position for finding the correspondence between the next two surfaces in each sequence (*i.e.* $\mathcal{A}^{l_A-1}$ and $\mathcal{B}^{l_B-1}$). We continue in this manner until we finally compute the correspondence between $\mathcal{A}^0$ and $\mathcal{B}^0$ (the original input surfaces). If $l_A \neq l_B$, then at some point one of the two sequences will run out of more complex surfaces. At that point, we just continue to use the most complex surface for the shorter sequence until the algorithm has matched all of the surfaces in the other sequence and finished.

At the base level, the correspondence is initialized to be the vertex positions of $\mathcal{A}^{l_A}$: no movement of $\mathcal{A}$. At each new step when a more complex model is introduced for $\mathcal{A}$, we must find a position of the vertices of $\mathcal{A}^i$ based on the correspondence found for $\mathcal{A}^{i+1}$. Because $\mathcal{A}^i$ and $\mathcal{A}^{i+1}$ could have arbitrarily different geometries or

even topologies[2], for every vertex a of the surface $\mathcal{A}^i$, we let $C_{\mathcal{A}^i}(a) = a + C_{\mathcal{A}^{i+1}}(P_{\mathcal{A}^{i+1}}(a)) - P_{\mathcal{A}^{i+1}}(a)$. More informally, we take every vertex of the new, more complex surface, and project it to the simpler surface. We compute the corresponding point to that projection point based on the old coarse correspondence. We then take the implied motion of that point (the difference between the projected point the correspondence at the projected point) and add that to the vertex to give the initial correspondence at the vertex of the more complex surface.

2.5. Polygon Reduction verses Image Pyramids

In some optical flow and other image matching algorithms, a similar simplification is used on the images to produce a pyramid of images from which the correspondence can be computed in a coarse-to-fine manner. Gaussian or Laplacian pyramids are usually the method of choice where each simpler layer in the set is a blurred down-sampled version of the previous (or possibly the derivative thereof). We argue that the polygon reduction described above is fundamentally different in that the surface is simplified in a *data-dependent* fashion. Gaussian and Laplacian pyramids are constructed by applying the same linear filters to the image regardless of the image. Polygon reduction produces simpler surfaces by careful consideration of the shape of the surface. This insures a more faithful representation of the surface especially at the simplest levels which means that the computations done at coarsest levels have much more bearing on the final output thereby leading to fewer local minima and faster computation because less work needs to be undone.

As pointed out in Hoppe (1996), mesh simplification can be seen as a (potentially lossless) compression technique. It is not surprising that compression can be used to obtain a good representation for matching. If we assume that both surfaces are collections of features drawn from a random (but structured) distribution, we would expect a compression algorithm to be able to find common substructures and reduce them in the same way thus allowing for easier matching. Finding the best method for compression and the best features for learning are often similar problems. We can view the polygon reduction stages of the algorithm as implicit feature detectors.

[2] For the two algorithms we tested, progressive meshes and quadric error metrics, and many other mesh simplification algorithms, the simplification is done by collapsing edges and thus the vertices of $\mathcal{A}^i$ can be mapped to the vertices of $\mathcal{A}^{i+1}$. However, after removing half of the vertices such a mapping can be misleading and we would like this algorithm to be independent of the method used to obtain the simplified surfaces.

3. Morphable Surface Models

Now that we have an algorithm for finding correspondences between surfaces, we turn to building Morphable Surface Models. We are given m surfaces, $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\}$ and we fix one of them as the base surface (without loss of generality, assume this is the one labeled $\mathcal{A}_1$). We now find the correspondences between $\mathcal{A}_1$ and all of the other surfaces. This produces a set of correspondences $\{C_1, C_2, \ldots, C_m\}$ (where C_1 is trivially computed and the rest describe how to warp $\mathcal{A}_1$ to each of the other surfaces in the set).

These correspondences can be combined in a convex combination to produce a new warping of the base surface. If $\Xi = \{\xi_1, \xi_2, \ldots, \xi_m\}$ are the convex combination coefficients ($\sum_i \xi_i = 1$), then we define this new warping in the obvious way,

$$W_\Xi(a) = \sum_i \xi_i C_i(a) \tag{18}$$

where $W_\Xi(a)$ is the new position of the point a from the base mesh in the model whose parameters have been set to Ξ. This restricts the new warping to lie in the $m-1$-dimensional space of these m warpings. Clearly there is a slight bias in this model relative to the arbitrary choice of the base surface (since correspondence calculations as described above are not invariant to the ordering of the two surfaces). However, for classes of similar objects with the same or similar topologies, this bias is small.

To remove the condition that the sum of the ξ_i parameters must be 1, we will assume that the model parameters have been centered (we will translate the space to a "center point" making any scaling of a warping a valid warping). By this we mean that an origin model has been set to warping when all $\xi_i = \frac{1}{m}$ and the correspondence vectors have been changed from the absolute positions of the vertices to the relative difference between the vertex position for the correspondence and the vertex position of this origin model. Mathematically, we define an origin as $O(a) = \sum_i \frac{1}{m} C_i(a)$ and then define new "displacement" correspondences $C'_i(a) = C_i(a) - O(a)$ and associated parameters ξ'_i which do not now need to sum to 1. This means that W is now

$$W'_\Xi(a) = O(a) + \sum_i \xi'_i C'_i(a) \tag{19}$$

From now on, we will drop the prime notation and assume that all models have been described in this fashion to eliminate the convex constraint on the ξ's.

3.1. Matching Models to Surfaces

We can now formulate an algorithm for finding the parameters of a morphable model so that the shape of the model best matches a given surface, $\mathcal{B}$ (this surface is not necessarily one of the ones used to construct the model). We again pose this as an energy minimization problem with the following energy term:

$$E_m(\Xi) = \int_{\mathcal{A}_1} \|W_\Xi(a) - P_{\mathcal{B}}(a)\|^2 \, da + \int_{\mathcal{B}} \left\|b - P_{W_\Xi(a)}(b)\right\|^2 \, db + \lambda \sum_i \xi_i \tag{20}$$

This minimizes the difference between points from the model's surface and their closest points on the target surface and vice-versa. We have also added a term penalizing high coefficient values. This reflects our intuition that the model will not be a good one far away from the surfaces used to construct it. If our model has been whitened as described in section 3.3, then this penalty is analogous to assuming a Gaussian prior in correspondence space.

Again, the minimization of E_m is difficult due to the integrals, so we will approximate them with sums of sampled points yielding

$$\begin{aligned} \hat{E}_m(\Xi) = \frac{1}{n} \sum_{a \in S_n(O(\mathcal{A}_1))} & \|W(a) - P_{\mathcal{B}}(W(a))\|^2 \\ + \frac{1}{n} \sum_{b \in S_n(\mathcal{B})} & \left\|P_{W(\mathcal{A})}(b) - b\right\|^2 + \lambda \sum_i \xi_i \end{aligned} \tag{21}$$

Just as before, we can minimize this function by alternating between finding the closest points to sampled points from the opposite surface and fixing these points to find Ξ by solving a linear-least squares problem. In this case, we do not minimize the distance to the plane, but just the distance to the projected point for simplicity. By taking derivatives of $\hat{E}_m$ with respect to Ξ while keeping the values of $P_{\mathcal{B}}(W(a))$ constants (pretending they don't depend on Ξ) and the values of $P_{W(\mathcal{A})}(b)$ as linear functions of Ξ (pretending their barycentric coordinates are fixed), the second half of the minimization has the solution:

$$\Xi = (A + 2n\lambda I)^{-1} q \tag{22}$$

where

$$A_{ij} = \sum_k C_i(a_k)^T C_j(a_k) \tag{23}$$

$$q_i = \sum_k C_i(a_k)^T b_k \tag{24}$$

for k ranging over the terms of both sums of equation 21. a_k is the point on model's surface (sampled for the first n values of k and projected from a point on $\mathcal{B}$ for the second n) and b_k is the associated point on $\mathcal{B}$ (projected from a point on the model's surface for the first n values of k and sampled for the second n).

3.2. Inner Products of Correspondence

In order to use bootstrapping (Vetter et al., 1997; Jones and Poggio, 1998) for building models in a more robust manner (section 3.3), we need to define an inner product over the space of correspondences. It might seem that if we simply stack all of the components of the correspondence vectors at each vertex into a large vector, we could use the dot product of these vectors as the inner product of the correspondence space. However, this definition is sensitive to the parameterization of the base surface. As an example, if we keep the base surface the same and the correspondences the same but subdivide some of the faces of the base surface (so we have the same surface described in a different way), we end up with a different inner product: we have lengthened the vector of vertex positions (although the new components are dependent on the old ones) and therefore magnified the importance of the correspondence at the faces we subdivided.

To solve this problem, we define the inner product between two correspondences in a parameter-independent fashion,

$$< C_i, C_j >= \int_{\mathcal{A}_1} C_i(a)^T C_j(a)\, da \tag{25}$$

We would now like to find a tractable method for computing this integral for piece-wise linear surfaces. For concreteness, we will take the example of 2-dimensional surfaces which are therefore composed of triangles. The analysis and resulting answer generalize to arbitrary dimensional surfaces.

The above integral, for the case of 2-dimensional piece-wise linear surfaces, becomes the sum of integrals of $C_i(a)^T C_j(a)$ over triangles. We know that $C_i(a)$ is a linear function of a over the triangle patch (and similarly with $C_j(a)$). Therefore, the integral of the dot product of these two vectors is a quadratic function of position taken over the

triangle. The following equality holds and can be verified easily.

$$\int_{\triangle x_0 x_1 x_2} C_i(x)^T C_j(x)\, dx = \frac{1}{12}\text{area}(\triangle x_0 x_1 x_2)\left[\sum_{k=0}^{2} C_i(x_k)^T C_j(x_k) + \sum_{k=0}^{2}\sum_{l=0}^{2} C_i(x_k)^T C_j(x_l)\right] \tag{26}$$

To allow us to use normal vector packages, we derive a new vector representation for a correspondence whose dot-product in the usual vector sense is exactly this inner product. Let $V_{\mathcal{A}}$ be the set of all vertices of $\mathcal{A}$, $F_{\mathcal{A}}$ be the set of all faces of $\mathcal{A}$, $\{f_0, f_1, f_2\}$ be the vertices of the face f, and $F(v)$ be the set of faces adjacent to the vertex v.

$$\begin{aligned}
< C_i, C_j > &= \sum_{f\in F_{\mathcal{A}}} \int_f C_i(x)^T C_j(x)\, dx \\
&= \sum_{f\in F_{\mathcal{A}}} \left[\frac{1}{12}\text{area}(f)\left[\sum_{k=0}^{2} C_i(f_k)^T C_j(f_k) + \sum_{k=0}^{2}\sum_{l=0}^{2} C_i(f_k)^T C_j(f_l)\right]\right] \\
&= \sum_{v\in V_{\mathcal{A}}} \sum_{f\in F(v)} \frac{1}{12}\text{area}(f) C_i(v)^T C_j(v) \\
&\quad + \sum_{f\in F_{\mathcal{A}}} \frac{1}{12}\text{area}(f) \sum_{k=0}^{2} C_i(f_k)^T \sum_{k=0}^{2} C_j(f_k) \\
&= \sum_{v\in V_{\mathcal{A}}} \left[\sqrt{\frac{\sum_{f\in F(v)}\text{area}(f)}{12}} C_i(v)\right]^T \left[\sqrt{\frac{\sum_{f\in F(v)}\text{area}(f)}{12}} C_j(v)\right] \\
&\quad + \sum_{f\in F_{\mathcal{A}}} \left[\sqrt{\frac{\text{area}(f)}{12}} \sum_{k=0}^{2} C_i(f_k)\right]^T \left[\sqrt{\frac{\text{area}(f)}{12}} \sum_{k=0}^{2} C_j(f_k)\right]
\end{aligned} \tag{27}$$

which implies that, if D is the dimensionality of the embedded space, the inner product can be represented as a dot product between two vectors whose first $D \times V_{\mathcal{A}}$ components are the positions of the vertices of the surface multiplied by the square root of the sum of the areas of the adjacent faces and whose last $D \times F_{\mathcal{A}}$ components are the sums of the positions of the vertices of each of the $f_{\mathcal{A}}$ faces multiplied by the square root of the area of the face.

3.3. Building a Model

Now that we have an inner product for correspondences, we can use the bootstrapping algorithm described in Vetter et al., Jones and Poggio (1997, 1998) to build the model robustly. Usually the method described above (where we take each of the surfaces and compute its correspondence to the base surface) will work fine. However, since the correspondence algorithm is not perfect, occasionally some parts of the correspondence will be off. If we have a large number of surfaces to be incorporated, we can correct for this problem in many cases. The bootstrapping algorithm is

1. Let M be the model. Initialize it as just the base surface (thus only one correspondence)

2. Repeat the following until the model has reach the desired level of flexibility:

   a) for each input surface:

      i) Match the current model to the surface (as in section 3.1) yielding an approximation to the input surface: $W(a)$

      ii) Find the correspondence from the base surface to the approximation, $W(a)$ (as in section 2): $C(a)$

      iii) Let $C_i(a) = W(C(a))$.

   b) Center the correspondences found in the previous step (as described in the beginning of section 3) and create the data matrix M whose columns are the centered correspondences (in the vector representation of section 3.2).

   c) Perform singular value decomposition on M yielding UDV^T.

   d) Retain only those columns of U with the largest singular values, increasing the number of parameters of the model (each column of U is an axis of correspondence space). Scale each column by its singular value.

In this algorithm, we successively build a more and more flexible model. Each time we use the old model to match the input surfaces and give an easier starting point for the correspondence routines. We then find and keep, via singular value decomposition, the axes of our space with largest variance since they are likely to be "real" surface changes and not due to noise in our correspondence algorithm.

Since we are scaling the axes by the standard deviation of the correspondences (in step (d)), we have whitened the data. This supports the $\sum_i \xi_i$ log-prior term in the matching energy function of section 3.1.

4. Experimental Results

We report results of running the model building algorithm on three collections of surfaces. Figure 1 shows a few examples from a collection of hand-drawn "smiley faces." These surfaces are (one-dimensional) lines embedded in the (two-dimensional) plane. They were created using a tablet input device which sampled the pen position over time. The example faces have between 66 and 407 line segments each, with most surfaces having about 100 segments. All faces were composed of four separate open curves and thus had the same topology; however this topological equivalence was not used in the matching.

These types of line drawings are very difficult for conventional optical flow algorithms: if the surfaces were converted to images, the spatial derivatives would be zero almost everywhere. A simple matching of closest points also fails dramatically because of the variation in the position of the eyes and mouth. It results in matching portions of some eyes and mouths to the large head circle. The structure term plays a crucial role.

We ran the bootstrapping algorithm on 57 faces for 23 iterations producing a model with 23 parameters. Figure 2 shows the major axes of variation captured by the model (the principal components of the last iteration of the bootstrapping algorithm).

We then asked two people to rank each face on four attributes (friendly vs. sinister, ugly vs. cute, left-looking vs. right-looking, and down-looking vs. up-looking). We created a radial-basis function (RBF) regressor (Bishop, 1995) mapping those four attributes to the dimensions of the face model. From that mapping, we automatically generated the results shown in figure 3. Similarly, we used an RBF to create the reverse mapping (from parameters to attributes). Figure 4 shows the results of matching eight faces not used in creating the model and then estimating the attributes from the resulting model parameters.

Secondly, we constructed a surface model of cars from 15 computer graphics surfaces (6 of which are shown in figure 5). These models have different topologies. All of the cars have five closed surfaces representing the body and tires. Some additionally have either open or closed surfaces representing the bumpers and side mirrors. Between 1270 and 10568 triangles define each surface. Color is represented by three extra dimensions (red, green, and blue). Thus, these shapes are 2D surfaces embedded in six-dimensional space. This 6D representation allows the algorithm to trade-off matching color verses matching shape. In previous Morphable Model work with images, the variations in shape had different model parameters than those of texture (or color). This would be analogous to separating every correspondence in the model into two

correspondences (one that had only non-zero spatial components and one that had only non-zero color components). We have chosen not to do so here and instead our results insist that the color and surface changes be dependent. Thus a given correspondence represents a change in the surface shape and surface color.

We ran the bootstrapping algorithm to completion (15 rounds, adding one additional parameter per round). The resulting principle components are shown in figure 6. Given that only 15 example surfaces were used and the high dimensionality of the problem (over 6000 numbers to represent one surface), we feel these capture the natural variations in car bodies well. The bounding boxes on the cars were normalized, so the first eigenvector well captures the difference between small and large car shapes. The second could be characterized as a "sporty" verses "boxy" axes. The third and fourth make distinctions in the tail-lights and rear windows.

The results from the matching procedure from section 3.1 are shown in figure 7. The first line is the input car to be matched. The second line is the model matched to the input. The third line is the model with the input car removed matched to the input. The difference between the second and third lines is mainly due to limited size of the model. As there are only 14 other cars, they do not completely span the space of all cars and thus were not able to exactly match the input. There were only two other "sporty" two-seater cars and no other car had such large side or rear windows. Thus it was impossible for the model to represent the input shape as the learned examples did not cover that portion of "car shapes."

The wheels in most of the car correspondences are a problem. Because they are round, the matching algorithm tends to add a bit of random rotation to the correspondences. Rotation correspondence fields do not add linearly and thus the wheel representations are not as crisp as the rest of the vehicle (the wheels tend to collapse). This is essentially a mismatch between the polygonal representation and the round surface.

To visually demonstrate the correspondence algorithm, figure 8 shows the results of each step of the algorithm run on two car surfaces. We think this clearly shows the iterative refining of the solution from coarse to fine resolution. Notice that most of the gross changes are done at the coarse level where the number of free variables (positions of vertices) are minimal. This helps to prevent overfitting.

Finally, to demonstrate the use of these correspondences as a metric-space, we took the animal surfaces of figure 9 and built a morphable model. Using the distance metric implied by the vector representation of section 3.2, we clustered the examples using the k-means algorithm (Bishop, 1995). This produced one cluster of the cat surfaces and one

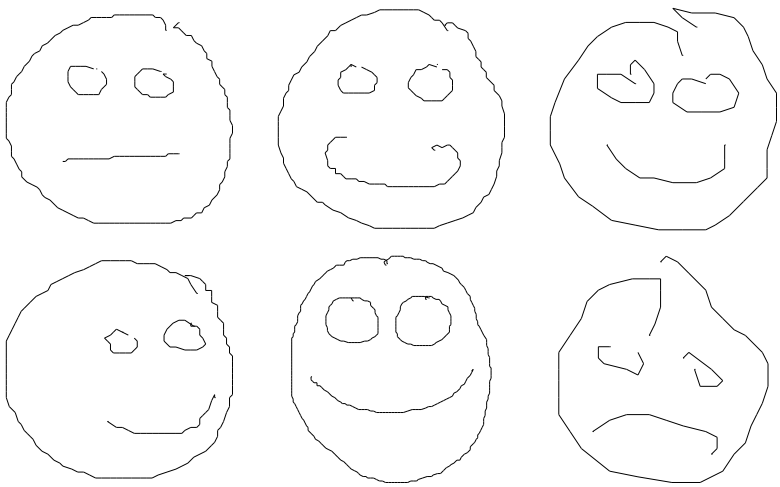

Figure 1. Six of the 57 example line-drawings used to construct the face model.

of the dog surfaces without a supervision. The projection of the points and cluster centers onto the first two principle components is shown in figure 10. The color and spatial positioning differences among the surfaces were eliminated by projecting out the correspondence directions of pure translation (in both spatial and color coordinates).

The goal in all of these experiments is to show that the correspondence technique and model building algorithms produce models which are robust and good at capturing the class of surfaces either for analysis or synthesis. The true test of a correspondence (especially in a domain where there isn't a ground truth) is in its use. We feel that these figures demonstrate the usefulness of the models built.

5. Extensions and Conclusions

No user intervention was required to build the models in this paper. The surfaces were input to the model building algorithm which ran automatically. They were all roughly aligned (centered, scaled, and rotated approximately the same). However, such rough alignment could have easily been done by comparing the first and second moments of the surface. Additionally, a few parameters[3] needed to be set (though not as many as in Shelton (1998) where gradient descent was used for

[3] specifically: α and *beta* (the weights of the terms of the energy function), n (number of sampled points), t (number of iterations of the minimization), γ (the ratio of color to spatial coordinates), λ (the prior's weight in the model matching algorithm), and the annealing schedule for α. None of these were sensitive in the

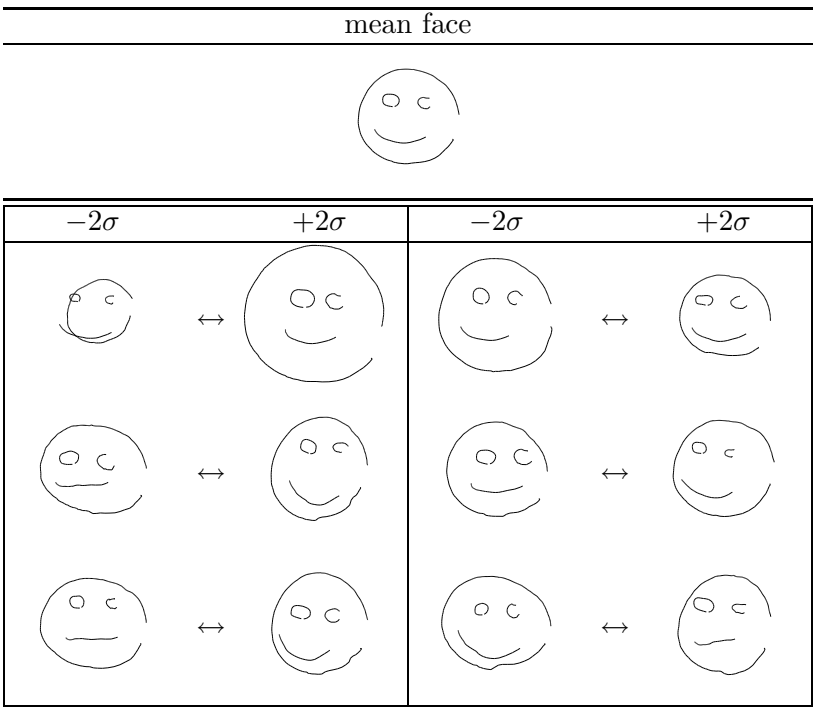

Figure 2. The first 6 eigenvectors of the face model scaled by ±2 standard deviations shown applied to the average face.

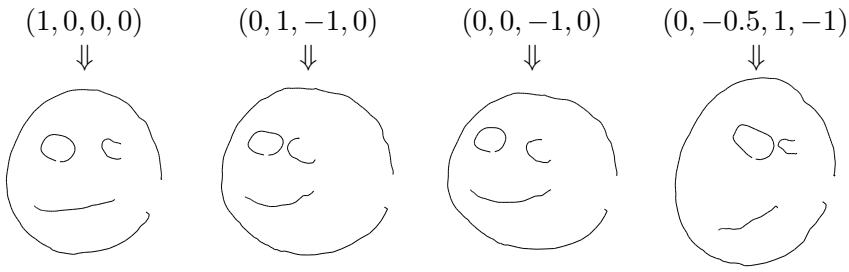

Figure 3. Results of fitting a radial-basis function network to mapping from four attributes (sinisterness, cuteness, left-right orientation, up-down orientation) to morphable model parameters. Four example outputs from settings of the attributes are shown.

minimization). We found that running a few test correspondences to find the correct order of magnitude for the parameters was all that was necessary.

For its generality, we feel that the energy function described produces good results. However, in domains where prior knowledge about

least and very easy to set except for the annealing schedule which we had to tune differently for each set of surfaces.

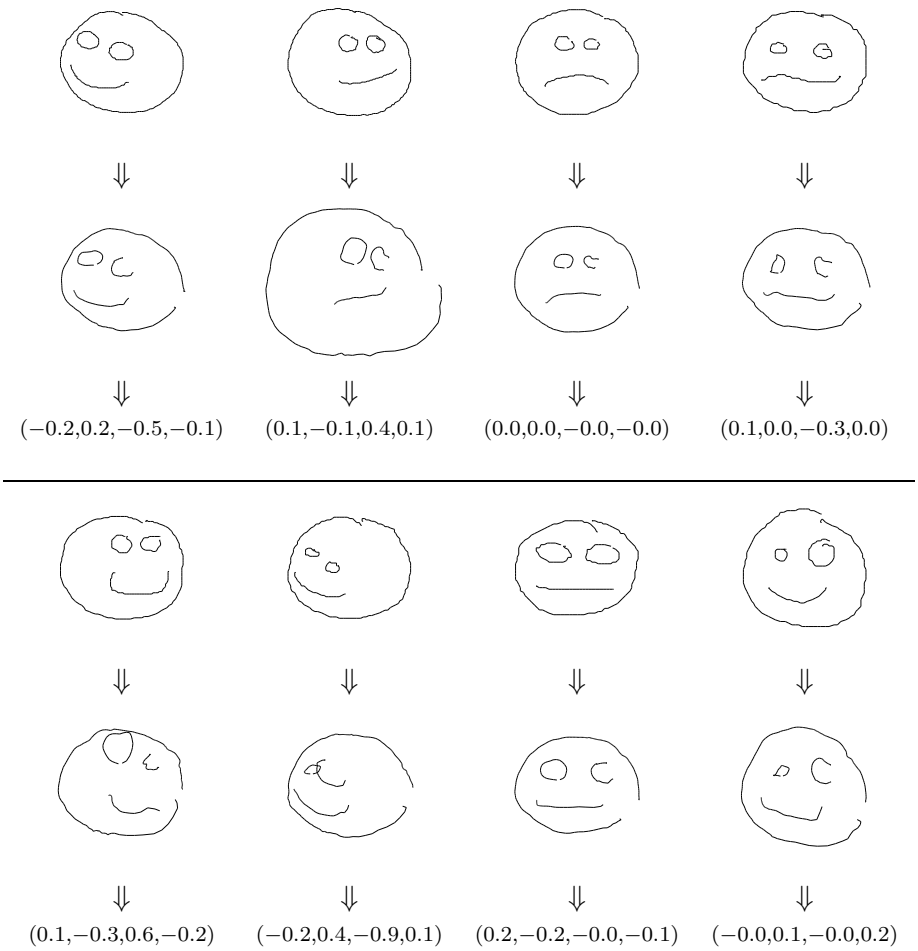

Figure 4. Analysis results for faces. The first row is the input faces. The second row is the model after matching. The third row of numbers are the output from the regressor (sinisterness, cuteness, left-right orientation, up-down orientation).

Figure 5. Six of the fifteen 3D surfaces used to construct the car morphable model.

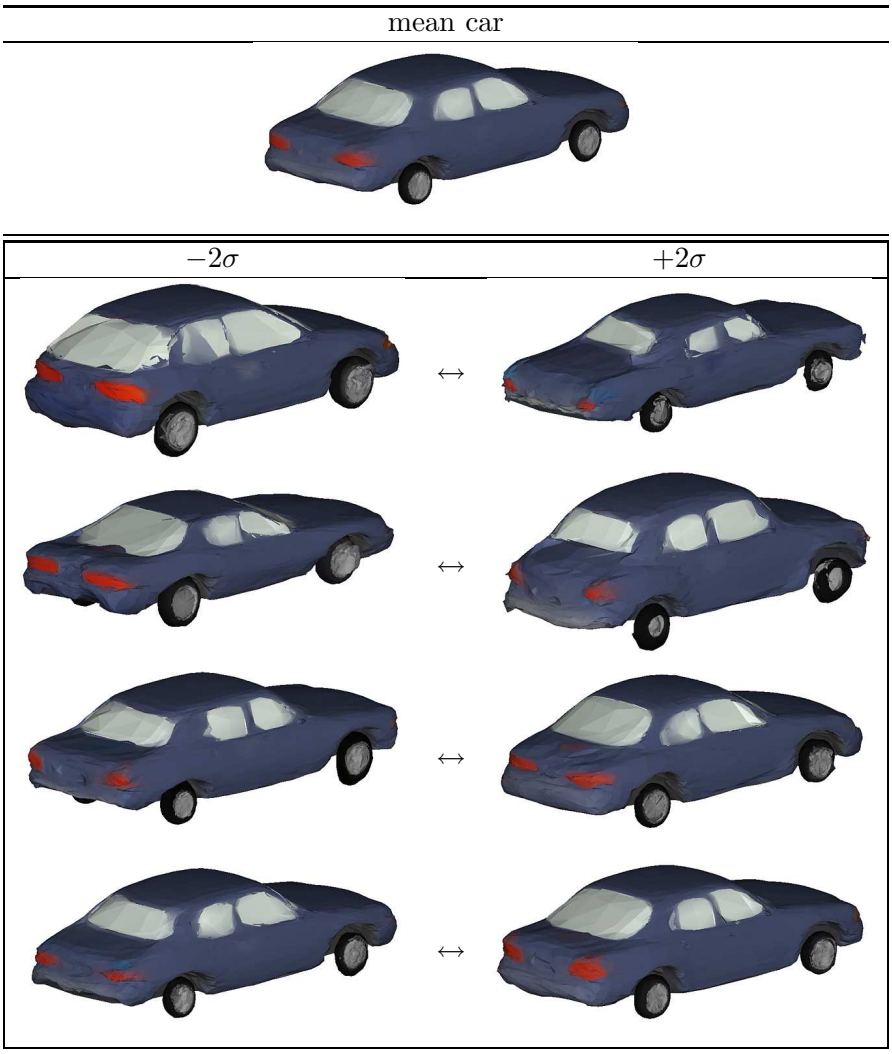

Figure 6. The first four eigenvectors of the car model. Each eigenvector is shown applied to the average model scaled by ± 2 standard deviations. The other eigenvectors have similar forms. We show the cars from this view since the back of the car tends to have the most variation.

surface deformations is available, better results could certainly be obtained by modifying the E_{str} and E_{pri} terms. Preliminary results with using this algorithm as a replacement for optical flow are promising (Shelton, 1998): the polygon reduction produces large triangles in textureless areas leading to easy matching where traditionally optical flow algorithms have had problems. Yet, this algorithm makes no assumptions about the images matched.

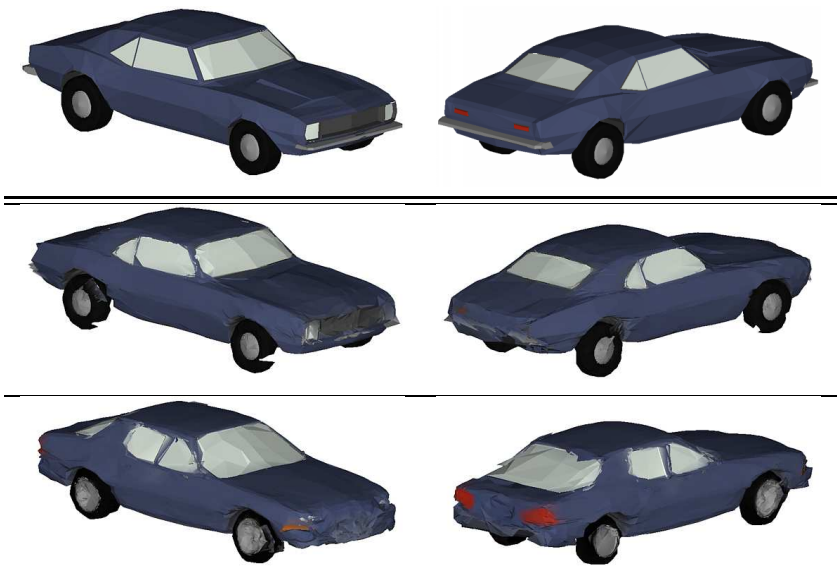

Figure 7. An example of matching a surface to a model. The top line is two renderings of the same input surface. The second line is the result of matching the top surface to a model of all 15 cars. The last line is the result of matching the surface to the model of only 14 cars (where the input car has been removed). Note that in this final case, the 14 cars used to construct the model do not span the space of all cars sufficiently enough to reconstruct the input car. The poor match in this case is mainly a result of insufficient flexibility in the model. For instance, in our set of car surface, there were no other cars with a small rear window and a similarly-shaped trunk.

We would like to replace the hard-threshold of the maximum operation in the P function (P takes a point and finds its closest point on another surface) with a soft-max function as is done in elastic networks (Durbin and Willshaw, 1987) since this leads to a smoother energy landscape (Durbin et al., 1989) and fewer local minima. However, so far we have not found a tractable method for doing so.

Although not shown here, user-defined correspondences can be used to improve the match by adding additional spring terms connecting the points on one surface to their matched points on the other surface. We have found such extra springs to be useful in matching shapes which differ greatly (*i.e.* in building a more general model of animals which included a rhinoceros, a camel, a giraffe, and a buffalo) or in applications where certain fairly undistinguished features are highly important for appearance to human observers (*i.e.* the outline of the lips can be difficult to match because of limited contrast in some images).

We would like to add the ability for the topology of the surface to change. Currently, example surfaces of different topologies can be

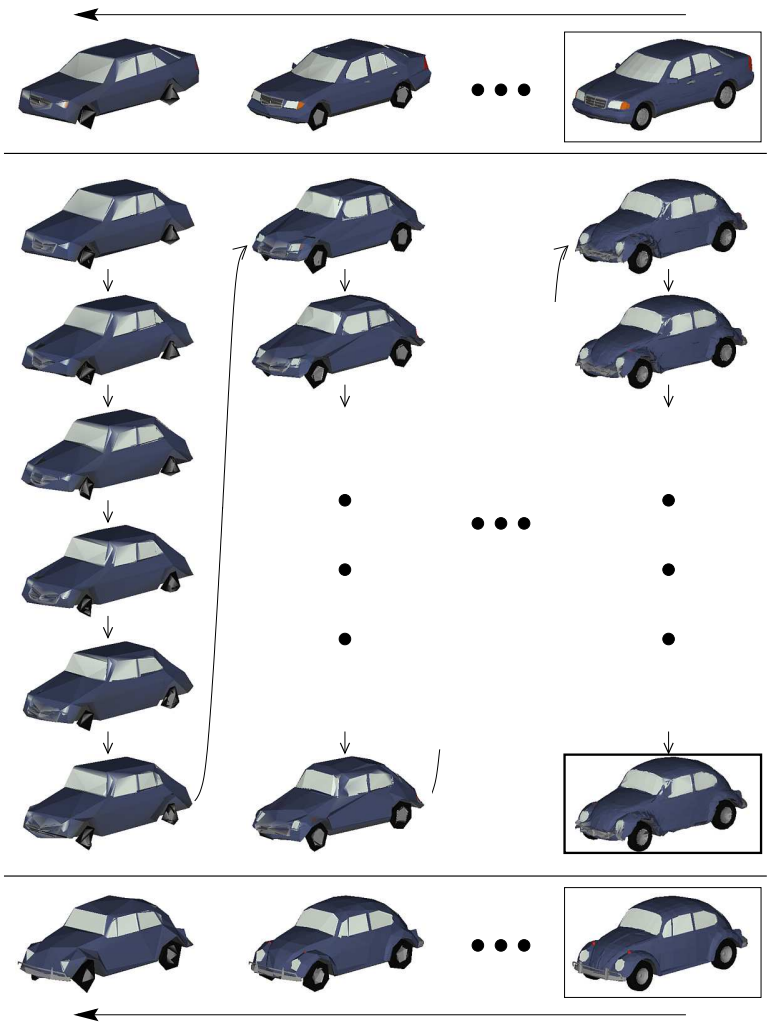

Figure 8. An example of the process of finding correspondence. The upper right surface is the input surface $\mathcal{A}$. The lower right surface is the input surface $\mathcal{B}$. The top line is the sequence of surfaces $\mathcal{A}^5$ through $\mathcal{A}^0$ (left to right). Similarly, the bottom line is the sequence of surfaces $\mathcal{B}^5$ through $\mathcal{B}^0$. These surfaces are produced from the input boxed surfaces as shown by the arrows at the top and bottom. Chronologically, the correspondence continues from the upper left corner and proceeds down each column and then across from left to right: First $\mathcal{A}^5$ is matched to $\mathcal{B}^5$ by iteratively solving the dual-minimization. The successive resulting correspondences, as applied to $\mathcal{A}^5$, are shown top-to-bottom in the first column (produced in the order shown by arrows). The final correspondence is then converted to a correspondence on $\mathcal{A}^4$ (the top surface in the column to the left) and the process is repeated in the next column (now matching to $\mathcal{B}^4$). Finally, the rightmost column shows the algorithm's steps on the last coarse-to-fine level with the figure outlined in bold being the final correspondence applied to $\mathcal{A}$.

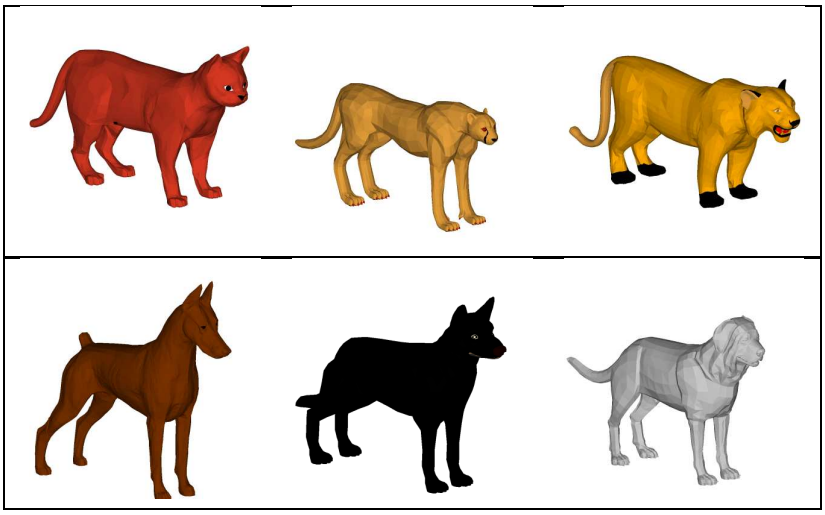

Figure 9. Surfaces used for constructing an animal model (this model has also been used for psychophysics experiments (Riesenhuber and Poggio, 1999; Riesenhuber and Poggio, 2000)). Clustering using the k-means algorithms (for $k = 2$) produced the top row as one cluster and the bottom row as another cluster. This nicely corresponds to cats and dogs.

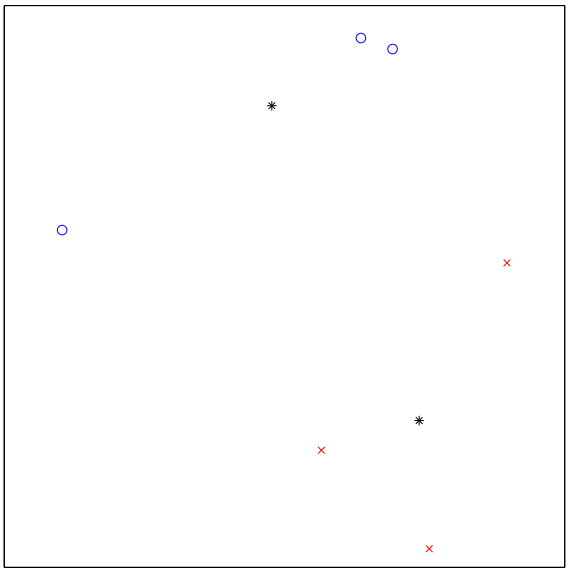

Figure 10. Plotting the six surfaces in correspondence space (projected onto the two axes of largest variance). The $\times$ markers are the cat surfaces and the circles are the dogs. The stars represent the cluster centers found.

used to create a model. However, all of the surfaces produced by that model will all have the topology of the base surface. McInerney and Terzopoulos (1995) describe a method for allowing the topology of snakes to change during the matching process. Combining this idea with the mesh reduction algorithm of Popović and Hoppe (1997), which allows topology changes, might provide for a more flexible surface model.

References

Beymer, D. and T. Poggio: 1996, 'Image Representations for Visual Learning'. *Science* **272**, 1905–1909.

Beymer, D., A. Shashua, and T. Poggio: 1993, 'Example Based Image Analysis and Synthesis'. A.I. Memo 1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Bishop, C. M.: 1995, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press.

Blanz, V. and T. Vetter: 1999, 'A Morphable Model For The Syntheis Of 3D Faces'. In: *Computer Graphics Proceedings, SIGGRAPH'99*. pp. 187–194.

Cootes, T. F., G. J. Edwards, and C. J. Taylor: 1998, 'Active Appearance Models'. In: *Proceedings of the European Conference on Computer Vision*, Vol. 2. pp. 484–498.

Durbin, R., R. Szeliski, and A. Yuille: 1989, 'An Analysis of the Elastic Net Approach to the Traveling Salesman Problem'. *Neural Computation* **1**, 348–358.

Durbin, R. and D. Willshaw: 1987, 'An analogue approach to the travelling salesman problem using an elastic net method'. *Nature* **326**(16), 689–691.

Garland, M. and P. S. Heckbert: 1997, 'Surface Simplification Using Quadric Error Metrics'. In: *Computer Graphics Proceedings, SIGGRAPH'97*. pp. 209–216.

Heckbert, P. S. and M. Garland: 1997, 'Survey of Polygonal Surface Simplification Algorithms'. Technical report, Carnegie Mellon University. to appear.

Hoppe, H.: 1996, 'Progressive Meshes'. In: *Computer Graphics Proceedings, SIGGRAPH'96*. pp. 99–108.

Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle: 1992, 'Surface Reconstruction from Unorganized Points'. In: *Computer Graphics Proceedings, SIGGRAPH'92*. pp. 71–78.

Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle: 1993, 'Mesh Optimization'. In: *Computer Graphics Proceedings, SIGGRAPH'93*. pp. 19–26.

Jones, M. J. and T. Poggio: 1998, 'Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes'. *International Journal of Computer Vision* **29**(2), 107–131.

Kang, S. and M. Jones, 'Appearance-based Structure from Motion Using Linear Classes of 3-D Models'. submitted to IJCV.

Kass, M., A. Witkin, and D. Terzopoulos: 1988, 'Snakes: Active Contour Models'. *International Journal of Computer Vision* **1**(4), 321–331.

McInerney, T. and D. Terzopoulos: 1995, 'Topologically Adaptable Snakes'. In: *Proceedings of the Fifth Internation Conference on Computer Vision (ICCV '95)*. pp. 840–845.

Nastar, C., B. Moghaddam, and A. Pentland: 1996, 'Generalized Image Matching: Statistical Learning of Physically-Based Deformations'. In: *Proceedings of Fourth European Conference on Computer Vision*.

Poggio, T. and T. Vetter: 1992, 'Recognition and Structure from one 2D Model View'. A.I. Memo 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Popović, J. and H. Hoppe: 1997, 'Progressive Simplicial Complexes'. In: *Computer Graphics Proceedings, SIGGRAPH'97*. pp. 217–224.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery: 1992, *Numerical Recipes in C*. Cambridge University Press, second edition.

Riesenhuber, M. and T. Poggio: 1999, 'A note on object class representation and categorical perception'. Technical Report AI Memo 1679, CBCL Paper 183, MIT AI Lab and CBCL, Cambridge, MA.

Riesenhuber, M. and T. Poggio: 2000, 'CBF: A New Framework for Object Categorization in Cortex'. In: M.-H. Yoo (ed.): *Proceedings of BMCV2000*. New York. To appear.

Shashua, A.: 1992, 'Projective Structure from Two Uncalibrated Images: Structure from Motion and Recognition'. A.I. Memo 1363, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Shelton, C. R.: 1998, 'Three-Dimensional Correspondence'. Master's thesis, Massachusetts Institute of Technology. Also available as AITR-1650.

Ullman, S. and R. Basri: 1991, 'Recognition by Linear Combinations of Models'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**, 992–1006.

Vetter, T., M. J. Jones, and T. Poggio: 1997, 'A Bootstrapping Algorithm for Learning Linear Models of Object Classes'. A.I. Memo 1600, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Vetter, T. and T. Poggio: 1995, 'Linear Object Classes and Image Synthesis from a Single Example Image'. A.I. Memo 1531, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.