Synthetic Images of Faces - An Approach to Model-Based Face Recognition

Andrew C. Aitchison and Ian Craw
Department of Mathematical Sciences, University of Aberdeen,
Aberdeen AB9 2TY, Scotland

Abstract

We present a method of generating realistic views of the head of any individual from a single photograph of the individual and a *generic* model of a human head.

The generic model is distorted so that it projects correctly onto the photograph. This is done by finding the least energy position of a spring-frame model, some of whose vertices are constrained to lie on particular features in the photograph.

This gives us a model of the individual, which is then texture mapped using the original photograph to generate the final view. This is perceived to be a realistic view of the individual, even if the model is quite crude.

1 Introduction

Model-based vision depends upon having models which describe the class of objects being sought. For simple geometric machined parts it is not hard to generate suitable parameterised 3D models, but natural objects such as human heads are more difficult to model.

Parke was the first to describe and implement a parameterised face model [1]. Waters [2] and Patel and Willis [3] both describe detailed face models which are can show a wide range of expressions. The images generated by these all models, impressive as they are, are not convincing images of any individual, principally because they are lacking in surface detail. Yau and Duffy [4] generated realistic views of individual people by *texture mapping* a photograph of a person onto a carefully constructed '$2\frac{1}{2}$-dimensional' model of the front of that person's head. They required a different model for each person they wished to draw.

We use a 3-dimensional model of the whole head. However this need not be a model of any particular individual - it is sufficient that it gives a good description of the general head. For this reason it is known as the *generic head model.*

2 The Generic Head Model

The generic model is defined using a triangulation, a wireframe model made entirely of triangles. This is one of the simplest representations for a 3-dimensional object.

BMVC 1991 doi:10.5244/C.5.29

We have a wireframe model of an individual's head, made up of nearly 10 000 points around the head. This was constructed by Linney [5] at University College Hospital, London by vertically scanning a laser range-finder up and down the face of a volunteer's head while she was seated on a rotating platform. This model is known as Vicki after the subject. It is too detailed for our purposes - 1000 triangles would be plenty - so we decided to build an approximation to it using only a subset of the vertices.

There are at least two approaches we could take in approximating a 3-dimensional triangulation such as Vicki. Both methods aim to find a triangulation with as few triangles as possible while ensuring that the error in approximating the original is below a given threshold. To use either method to generate a triangulation of a given size we must make several approximations at different thresholds, until a triangulation of an acceptable size results.

2.1 Faugeras' Algorithm

Faugeras et al. [6] describe an algorithm which generates a triangulation approximation to a given wireframe model of a 3-dimensional surface. The vertices of the approximation are all vertices of the original wireframe. The first approximating triangulation is just two triangles back to back. This approximating polyhedron is iteratively improved by adding more vertices of the original wireframe, until the approximation is sufficiently accurate everywhere.

At each step the wireframe is divided into regions as follows. For each triangle in the polyhedron find the shortest path on the wireframe between each pair of the triangle's vertices. The triangle is then used as the current approximation to all the vertices within these bounds. The approximating polyhedron is improved by repeatedly dividing each triangle into three new triangles by adding a new vertex from the corresponding region of the wireframe. The vertex added is the one which is furthest from the current triangular face.

Since an edge could occur in the polyhedron which was a poor approximation to the wire frame, a refinement is made to the approximating polyhedron after each step. Each adjacent pair of triangles is split into four by adding the vertex lying along the common edge of the approximated regions which is furthest from the common edge of the triangles.

This gives good approximations for machined parts, but the results are not very good for our heads. The problem is that while the new triangulation is a good approximation of the original *points* it is not always a good approximation of the original *surface*.

2.2 A Point-Removing Algorithm

We therefore tried the alternative approach - start with the whole model and remove vertices one at a time until no more can be removed without making the approximating error too large. Several variations on this approach have been tried. We will only describe the one used to generate the current generic head model.

We cannot forget about a vertex once it has been removed, since removing another vertex may remove the triangle that it was closest to. Whenever we remove a vertex we must find the new triangle which it is closest to, and indicate it as such. Therefore each triangle has a list of vertices which it approximates.

It would be terribly inefficient to calculate the error in approximating each vertex of a large given triangulation before determining whether we may remove a single vertex. Fortunately we don't need to. Removing a single vertex $\mathbf{v}$ from a triangulation T is a local operation - the new, reduced, triangulation will only differ from T in a neighbourhood of $\mathbf{v}$. The triangulation of this neighbourhood, which is a subtriangulation of T, is just the union of the triangles which have $\mathbf{v}$ as a vertex. We call this the *neighbourhood triangulation*. We split T into two: the neighbourhood triangulation and the *complementary triangulation* - the complement of the neighbourhood triangulation in T. We now form the *reduced neighbourhood triangulation* by removing the vertex $\mathbf{v}$. Finally we form the *reduced triangulation* by taking the union of the reduced neighbourhood triangulation with the complementary triangulation. This is the triangulation T with the vertex $\mathbf{v}$ deleted.

Assuming that T is an acceptable approximation to the given triangulation, the reduced triangulation will be acceptable if it is a sufficiently good approximation to all the vertices of the given triangulation which are best approximated in T by a triangle in the neighbourhood triangulation. This will certainly be true if the error in approximating each of these vertices by the reduced neighbourhood triangle is sufficiently small. Thus we can determine whether removing a vertex will result in an acceptable approximation to the given triangulation, without calculating the reduced triangulation. Neither do we need to calculate the error in approximation of all the given vertices. It is sufficient to calculate the reduced neighbourhood triangulation and the error in approximating the removed vertex and those vertices of the given triangulation which are approximated by triangles in the neighbourhood triangulation.

We now give the outline of the algorithm. To find a triangulation T which approximates triangulation $\mathcal{T}$ such that the error in approximating each vertex of $\mathcal{T}$ is at most ϵ, and T has as few vertices (and hence triangles) as possible:

- Set $T_0 = \mathcal{T}$. No triangle in T_0 approximates a vertex yet.
- For $n = 1, \ldots,$ number of vertices in $\mathcal{T}$ do:
  1. Choose a vertex $\mathbf{v}_n$ of T_{n-1} that has not already been chosen.
  2. Split T_n into two triangulations: N_n the neighbourhood triangulation of $\mathbf{v}_n$, and C_n, the complementary triangulation.
  3. Remove $\mathbf{v}_n$ from N_n and form a new triangulation R_n.
     Clear the lists of vertices approximated by each triangle in R_n.
  4. Let *ApproxVerts* = list of all vertices approximated by triangles in N_n, plus $\mathbf{v}_n$.
     Let *WorstError* = maximum of all the errors in approximating vertices in *ApproxVerts* by the triangulation R_n.
  5. If *WorstError* $> \epsilon$ then
     - set $T_n = T_{n-1}$

     else
     - set $T_n = C_n + R_n$
     - for each vertex in *ApproxVerts* find triangle in R_n which this vertex is closest too, and add it to the list of vertices this triangle approximates.

- T_n is the desired approximation. □

Our current generic head is a wireframe triangulation made up of around 5000 triangles and 2500 points.

3 The Specific Head Model

The generic head gives us a 3D description of the shape of a human head. We tune this to match the head of the individual in the photograph by moving the vertices appropriately. Each vertex of the model represents a point on the surface of a human head, and should be moved so that it matches this point when the model is projected onto the plane of the photograph.

We do this in three stages. A global transformation roughly matches the heads. Some of the points are then found on the photo, and depth values for these vertices are derived. The remaining vertices are found by interpolation.

3.1 Globally Matching the Model to the Photograph

First, a global affine transformation is applied to the model so that its general size and orientation matches the head in the photograph. The model is rotated to match the orientation of the head in the photograph, and shrunk or enlarged horizontally and vertically until the height and width match also. The depth is then scaled so that it remains in proportion to the first two directions. This scaling may involve different scale factors in each direction as some people have longer, thinner faces than others.

3.2 Finding Points on the Photograph

Once we have transformed the generic head model globally the individual vertices are moved locally to produce the specific model triangulation. It would be impractical to find the positions of a thousand or more vertices on the photograph manually. Instead we define a smaller set of the most important, or *major*, vertices. The positions of the points represented by these major vertices are found directly from the photograph. Ideally this is done using the feature finding systems developed by researchers at Aberdeen [7, 8], but they could be found relatively easily by hand if necessary. The positions of the remaining, *minor*, vertices are found by interpolating between the major vertices. Individual major vertices can be 'demoted' to minor vertices if they cannot be found on a particular image.

Each vertex of the specific model has three components, two in the plane of the photograph, and one, the depth, perpendicular to this plane. The two components in the plane of the photograph are just the components of the position of the corresponding point in the photograph. This ensures that the vertices of the specific model match their respective points when projected onto the plane of the photograph.

Since the general shape and size of the scaled and rotated generic model and the head in the photograph match in at least their grossest form, it seems reasonable to use the depth component of the scaled generic model, without further modification, as the depth component of the specific model. The generic

head was introduced especially to give the system knowledge of the shape of human heads. This knowledge will always be available and will, we believe, usually give good results. It is likely to fail only when the individual is markedly different from the norm represented by the generic head. For example a front view of a person will give little indication of the size of their nose.

3.3 Interpolation

Finally the positions of the minor vertices are found by imagining each edge of the model to be a spring. These springs all work in tension and in compression and have the same stiffness ϵ. The spring lengths are chosen so that if no external forces are applied the spring model takes up the shape of the globally transformed model.

The major vertices are fixed in their desired positions, and the model is then allowed to reach equilibrium. This least-energy position is our final model for the head of the person in the photograph.

The equilibrium position of the spring model is found using the following iterative algorithm. I make no claims for the stability or efficiency of this algorithm, but merely report that it works.

Apply global transformation to model.
Calculate rest length of each spring.
    Spring with ends at vertices i and j has rest length
    $L_{ij} = | \mathbf{v}_j - \mathbf{v}_i |$
    where $\mathbf{v}_i$ is position of vertex i.
Displace major vertices to desired positions on photo.
repeat
    for each vertex $\mathbf{v}_i$
        Force at $\mathbf{v}_i$ towards $\mathbf{v}_j$ caused by connecting spring:
        $\mathbf{F}_{ij} = \epsilon . (1 - \frac{L_{ij}}{|\mathbf{v}_j - \mathbf{v}_i|}) . (\mathbf{v}_j - \mathbf{v}_i)$.
        Displace vertex $\mathbf{v}_i$ in proportion to the *total* force acting upon it.
    endfor
until total displacement in one step is sufficiently small.

As an extension, instead of fixing a major vertex to a particular location we could allow it to travel along a fixed wire. This might be appropriate when we know the *outline* of the head in the photograph but do not know exact positions of the major vertices which lie along it.

4 Display

We now have a 3D model of an individual and a photograph of the same person in which the vertices of the model are located. The model can easily be rotated to give the impression of viewing from a new viewpoint. It should also be possible to distort the model further, to change the expression of the face, perhaps changing a smile into a frown, or shutting an eyelid, while keeping the particular shape of the subject's head.

We then display the specific head, using the original photograph to provide colour and texture information, so that the head looks like the person in the

photograph. This is called *texture mapping.* We implement this by considering the triangles which make up the model, one at a time. The corners of each triangle are vertices of the model so we have been given or have interpolated the position on the photograph which each corner represents. We find the position of each corner on the displayed image by applying the viewing transformation to each vertex. This gives us the position of the corner in the viewing plane, and its depth relative to this plane. We now have two triangles, one on the photograph and one on the viewing plane. For 2 triangles with identified corners (as these are) there is a unique affine mapping from one to the other. We use this mapping to copy the pattern from the first triangle onto the second. We ensure that parts of the model which are obscured by nearer parts are not displayed, by using a z-buffer hidden surface removal algorithm. This triangle-based texture mapping has also been used successfully to distort 2 dimensional faces. In particular Phil Benson [9] has used the same technique to generate photographic quality caricatures for psychological experiments.

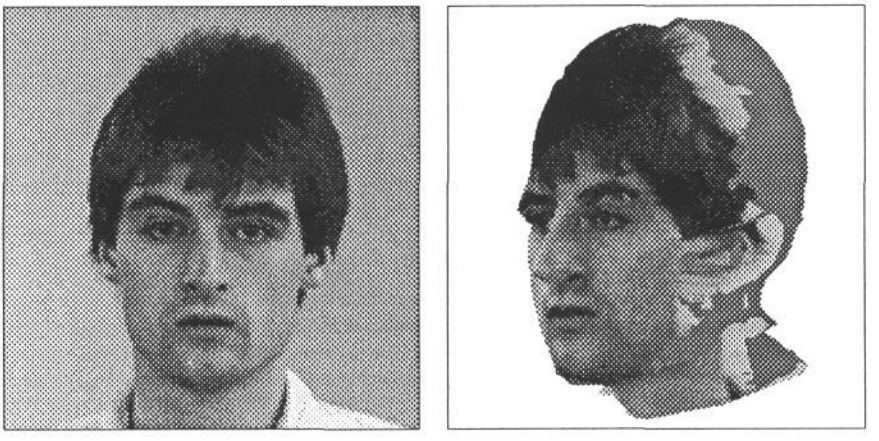

Figure 1: Applying the method to the left image gives the right image.

5 Applications

In their work on eigenfaces, Craw and Cameron [10] found that reconstruction was greatly improved by *standardisation.* This is just a 2-dimensional texture-mapping onto an average face. Eigenfaces have also been used for face recognition (see [11]). It would be useful to recognise faces seen through security cameras. Typically these are mounted high up and look down on a face at a significant angle - unlike the full-face images used in current recognition work and the photo-ids many of us carry today. Our system would facilitate the conversion between these views.

If we can generate synthetic views of a real person, what about creating imaginary people ? If we take the average of the models of two people and the average of the textures we get a realistic picture of the average person. If we take several weighted averages we get a smoothly flowing sequence which starts with one person and finishes with another. Each still frame looks like a

photograph of a person. The sequence shown at BMVC last year was generated this way.

6 Acknowledgements

The first author would like to thank British Telecom, Martlesham Heath and the SERC for the CASE studentship under which this work was done.

References

[1] Frederic I. Parke. Parameterized models for facial animation. *IEEE CG&A*, pages 61–68, November 1982.

[2] Keith Waters. A muscle model for animating three-dimensional facial expression. *Computer Graphics*, 21(4):17–24, 1987. Proc. ACM SIGGRAPH '87.

[3] Patel and Willis. Faces: Facial animation, construction and editing system. In *EUROGRAPHICS '91*, Vienna, September 1991.

[4] Neil D. Duffy and John F. S. Yau. Facial image reconstruction and manipulation from measurements obtained using a structured lighting technique. *Pattern Recognition Letters*, 7(4):239–243, April 1988.

[5] Alf D. Linney. The use of 3-D computer graphics for the simulation and prediction of facial surgery. In Vicki Bruce and Mike Burton, editors, *Processing Images of Faces.* Ablex, Norwood, NJ. To Appear 1991 ?

[6] O. D. Faugeras, M. Hebert, P. Mussi, and J. D. Boissonnat. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics and Image Processing*, 25:169–183, 1984.

[7] David Tock, Ian Craw, and Roly Lishman. A knowledge based system for measuring faces. In *BMVC90 Proceedings of the British Machine Vision Conference*, pages 401–407, University of Oxford, 24–27 September 1990. British Machine Vision Association.

[8] Alan D. Bennett and Ian Craw. Finding facial features using deformable templates and detailed prior statistical knowledge. In this volume.

[9] P. J. Benson and D. I. Perrett. Perception and recognition of photographic quality facial caricatures: Implications for the recognition of natural images. *European Journal of Cognitive Psychology*, 3(1), 1991.

[10] Ian Craw and Peter J. Cameron. Parameterising images for recognition and reconstruction. In this volume.

[11] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.