Model-based Face Capture from Orthogonal Images

Douglas R. Lanman

August 6th, 2001

Outline

- Introduce face capture problem.
- Overview of model deformation methods.
- Orthogonal images simplification.
- Outline of texture mapping.
- Cylindrical texture mapping algorithm.
- Results

Introduction to Facial Modeling

Realistic facial synthesis is one of the most difficult problems in computer graphics.

Structural modeling challenges:

- The human face is an extremely complex geometric form.
- Fine detail and variations in color and texture are essential for recognition of identity and interpretation of expressions.

Example: Facial models in Pixar's *Toy Story* utilized several thousand control points.

Expression animating challenges:

- Facial movement is determined by mechanical properties of the underlying skeletal and muscular forms.
- Mechanical properties of skin and subcutaneous layers are difficult to mimic.

Methods of Facial Modeling

Digital scanners:

Laser-based cylindrical scanners can be used to directly model facial structure.

Advantages:

- Models are produced automatically (no user intervention).

Limitations:

- Scanner equipment is expensive.
- Laser-based scanners typically produce low resolution models. (Cyberware scanners are limited to a cylindrical grid of 512x256 samples.)

Photogrammetric methods:

A series of images can be used to create a precise geometry estimate.

Advantages:

- Low cost (only requires regular cameras).
- High resolution models can be created.

Limitations:

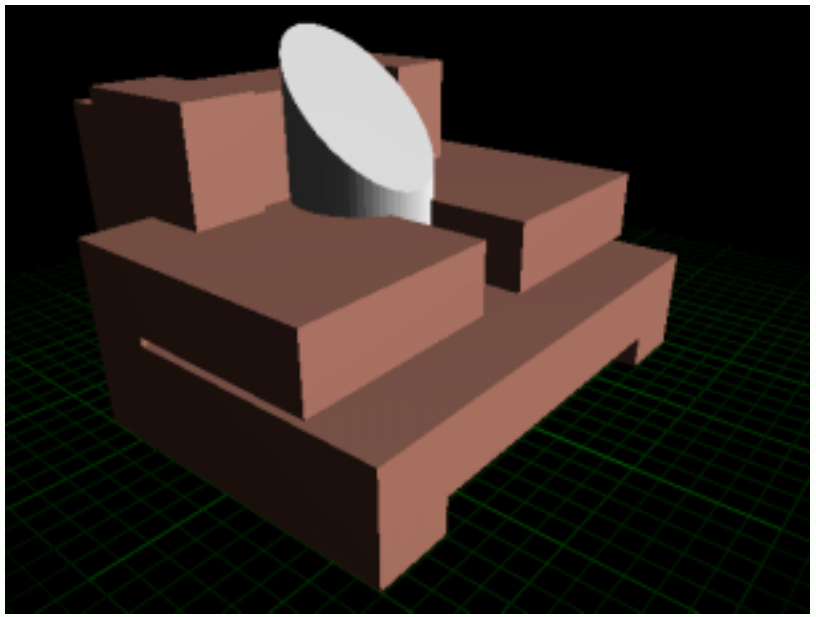
- User interaction is typically required.

Model Deformation

For objects that have similar structures, a generic model can be deformed to fit the features located in a series of photographs.

Examples:

- Architectural modeling [Debevec, 1996]
- Facial animation [Pighin, 1998]
- Medical imaging (CT, MRI, ultrasound) [Boes, 1995]



Facial Model Deformation

Prior art:

Pighin et al., 1998:
Introduced robust fitting algorithm for use in facial modeling. Method utilized multiple cameras with arbitrary intrinsic (focal length, radial distortion) and extrinsic (position, rotation) parameters.

Problem:
Initial estimates of camera parameters were required so minimal calibration was necessary.

Proposed simplification:

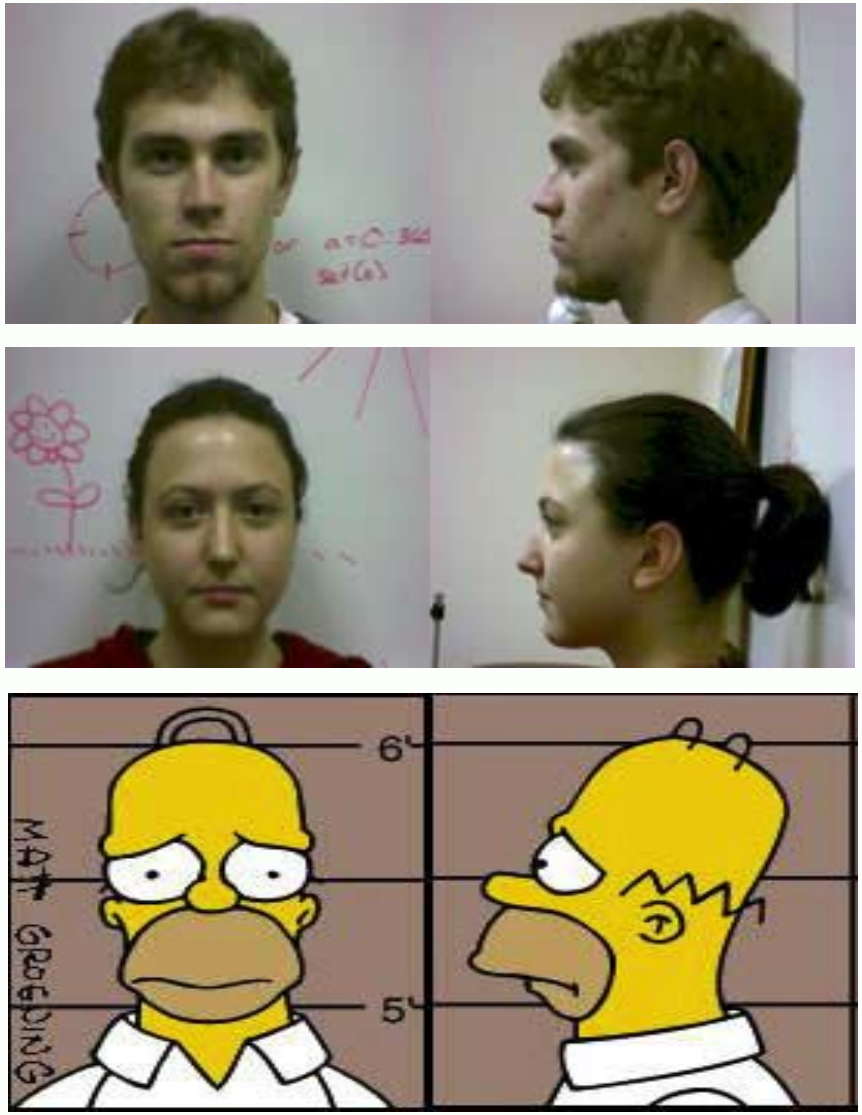
Use only two orthogonal images (front and profile).

Advantages:
- Appropriate viewing angles are easy to estimate.
- A single camera can be used to produce facial models.

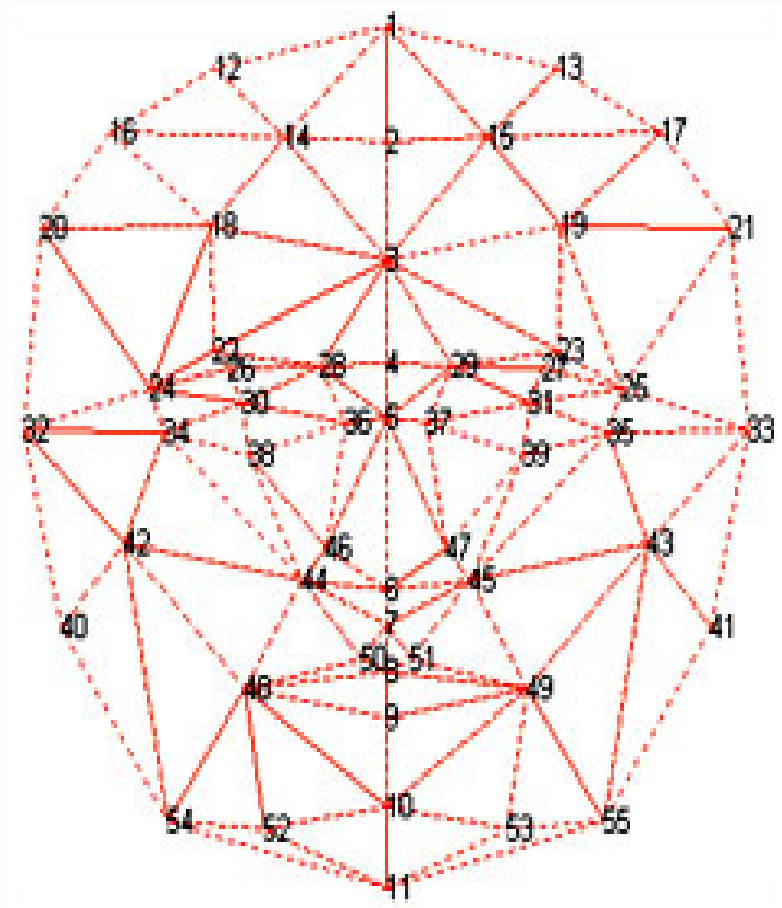
Limitations:
Models have reduced accuracy when compared to Pighin's.

Deformation Algorithm (1)

Input image sets:



Generic face model:



Deformation Algorithm (2)

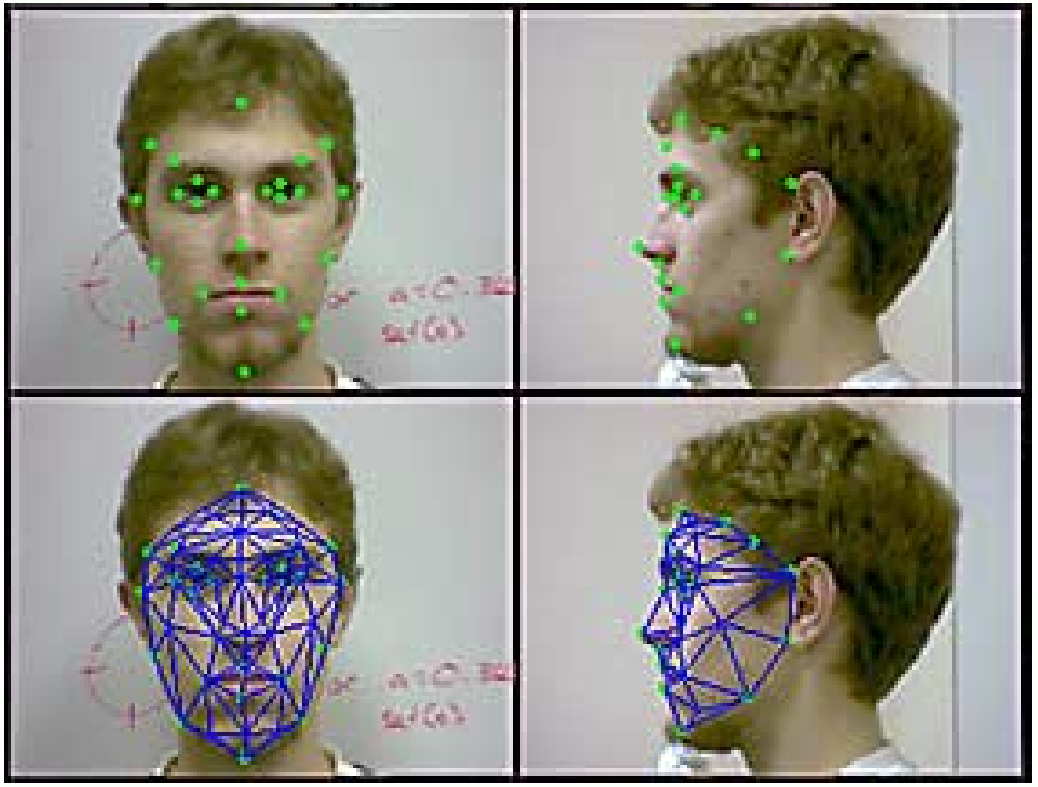
Goal:

Deform the generic head model to fit each image.

Technique:

Locate set of features in image set that correspond to subset of model vertices.

Use correspondences (displacement vectors) to create an interpolation function.



Interpolation Function (1)

user-selected features:

$\vec{p_i}$

model feature vertices:

$\vec{p_i}^0$

vector displacements:

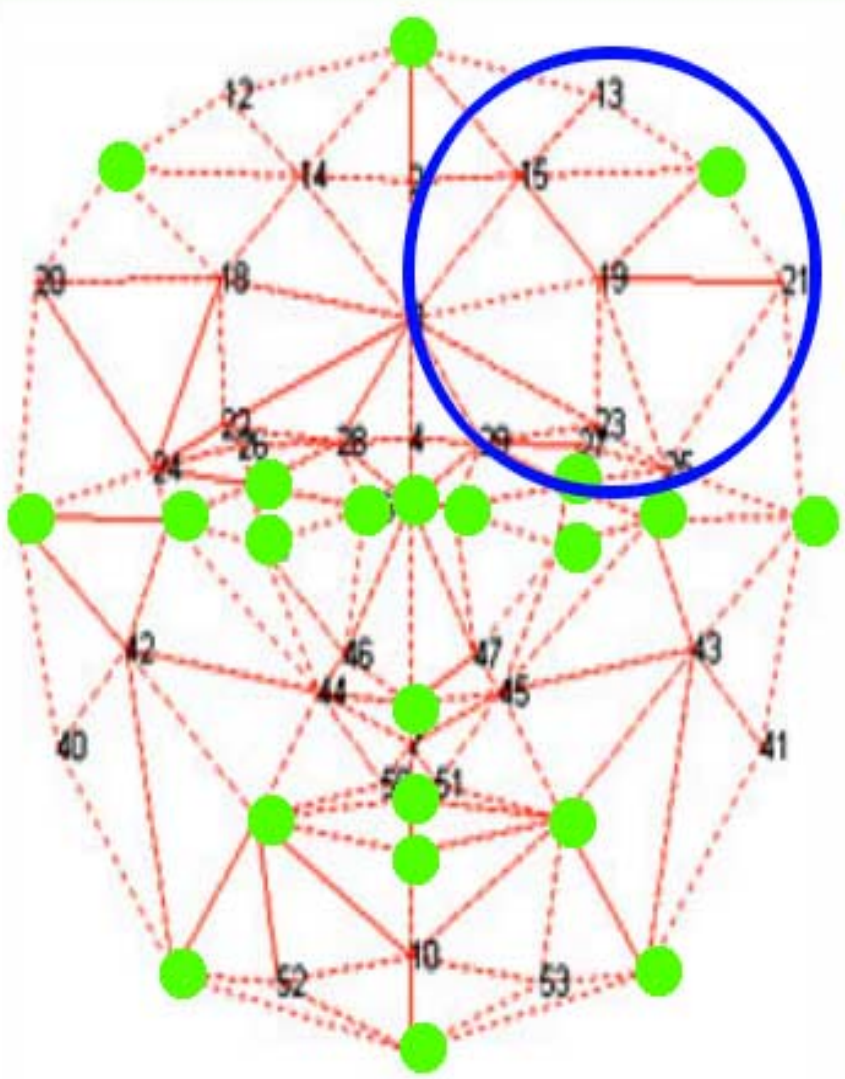
$\vec{u_i} = \vec{p_i} - \vec{p_i}^0$

general interpolation function:

$$f(\vec{p}) = \sum_{i=1}^{N} \vec{c_i} \phi(\| \vec{p} - \vec{p_i}^0 \|)$$

radial basis function:

$\phi_{rb}(r;\lambda) = e^{-\frac{r}{\lambda}}$, for $\lambda > 0$.



Interpolation Function (2)

specific interpolation function:

$$\text{choose } f(\vec{p_i}^0) = \vec{u_i} = \sum_{j=1}^{N} \vec{c_j} \phi_{rb}(\| \vec{p_i}^0 - \vec{p_j}^0 \|; \lambda)$$

in matrix form:

$$\begin{pmatrix} u_{x,1} & \cdots & u_{x,N} \\ u_{y,1} & \cdots & u_{y,N} \end{pmatrix} = \begin{pmatrix} c_{x,1} & \cdots & c_{x,N} \\ c_{y,1} & \cdots & c_{y,N} \end{pmatrix} \begin{pmatrix} \phi_{rb}(\| \vec{p_1}^0 - \vec{p_1}^0 \|; \lambda) & \cdots & \phi_{rb}(\| \vec{p_1}^0 - \vec{p_N}^0 \|; \lambda) \\ \vdots & \ddots & \vdots \\ \phi_{rb}(\| \vec{p_N}^0 - \vec{p_1}^0 \|; \lambda) & \cdots & \phi_{rb}(\| \vec{p_N}^0 - \vec{p_N}^0 \|; \lambda) \end{pmatrix}$$

solve for $\vec{c_j}$:

$$\begin{pmatrix} c_{x,1} & \cdots & c_{x,N} \\ c_{y,1} & \cdots & c_{y,N} \end{pmatrix} = \begin{pmatrix} u_{x,1} & \cdots & u_{x,N} \\ u_{y,1} & \cdots & u_{y,N} \end{pmatrix} \begin{pmatrix} \phi_{rb}(\| \vec{p_1}^0 - \vec{p_1}^0 \|; \lambda) & \cdots & \phi_{rb}(\| \vec{p_1}^0 - \vec{p_N}^0 \|; \lambda) \\ \vdots & \ddots & \vdots \\ \phi_{rb}(\| \vec{p_N}^0 - \vec{p_1}^0 \|; \lambda) & \cdots & \phi_{rb}(\| \vec{p_N}^0 - \vec{p_N}^0 \|; \lambda) \end{pmatrix}^{-1}$$

combine models:

$$(x, y, z) = (x_f, \frac{y_f + y_s}{2}, z_s)$$

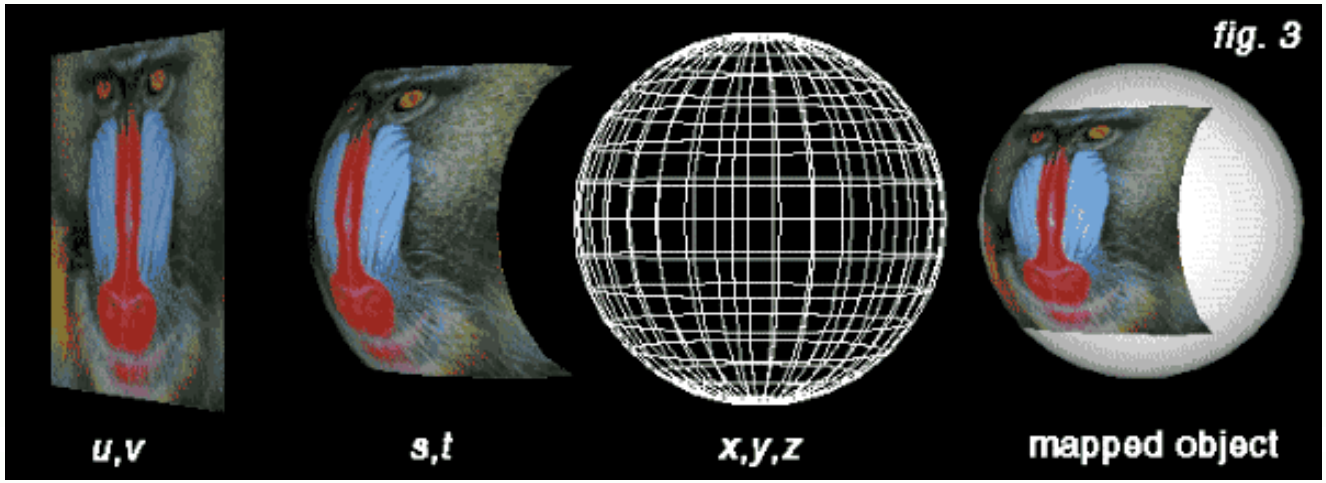
Texture Mapping

Definition:

A texture map is a way of controlling the diffuse color on a surface on a pixel-by-pixel basis, rather than by assigning a single overall value. Texture mapping is achieved by "wrapping" a 2D image (the texture map) around a 3D object.

Texture maps are really *color maps*, however the term has become standardized in the computer graphics industry.

Texture mapping is commonly achieved by applying a color bitmap image to a surface.



Cylindrical Projection

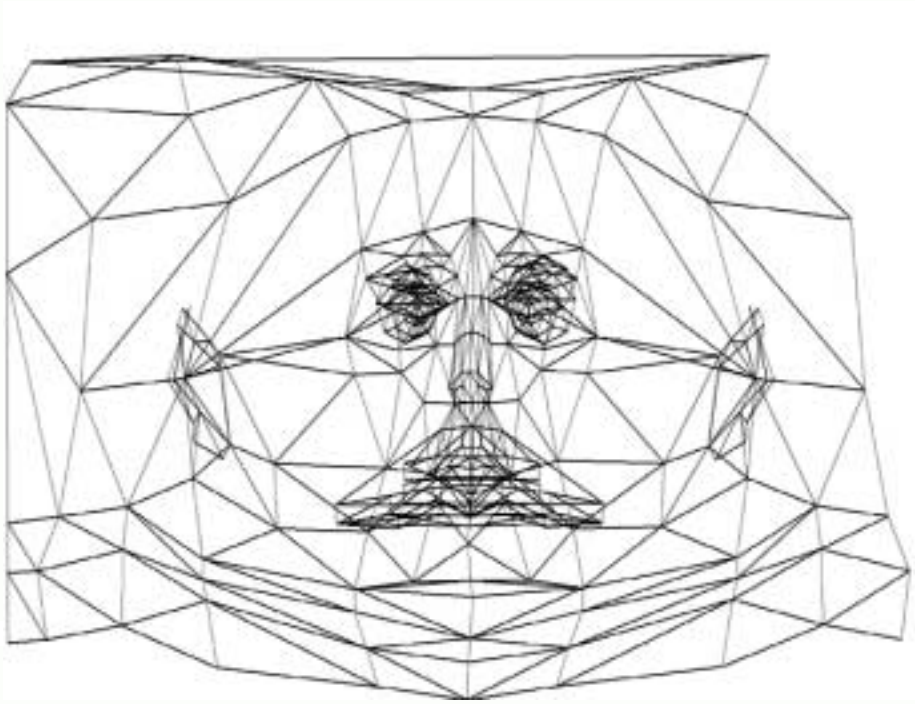
Goal:

Produce a 2D representation of the generic face/head model.

The projected coordinates of the model will serve as the texture coordinates (s,t) for the final texture map.

Technique:

Project model onto cylindrical surface.



Cylindrical Texture Map

Goal:

Sample pixels from the input images and assign them to the cylindrical texture map in a manner that maximizes resolution.

Algorithm:

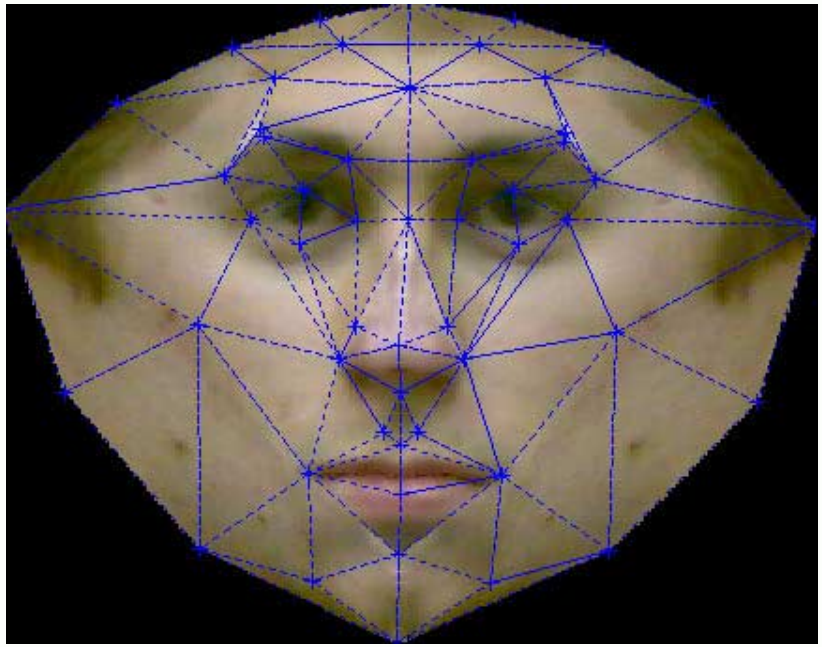
Set the sampling rate for the texture map.

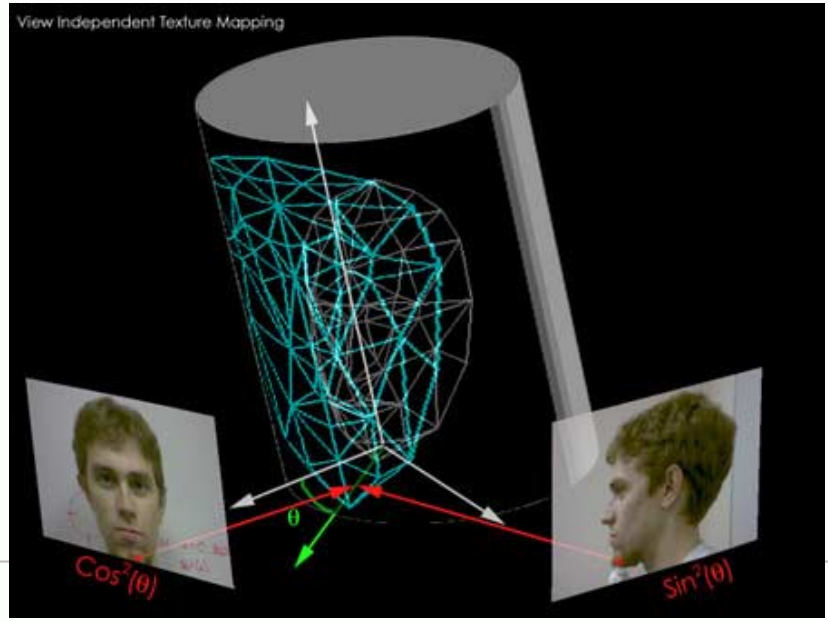
For each triangle:

Calculate affine coordinates of enclosed pixels within bounding box and triangle.

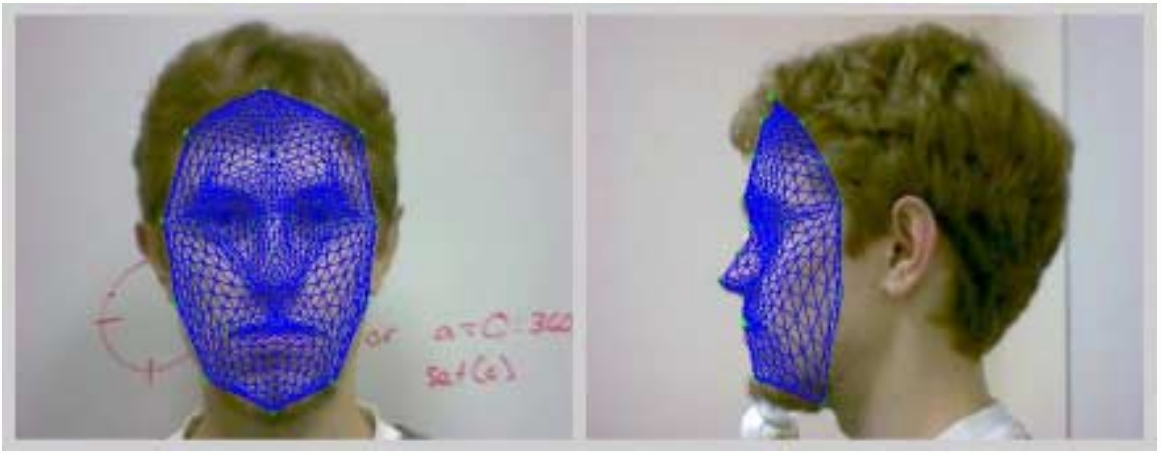
Use affine coordinates to locate corresponding pixels in front and side images.

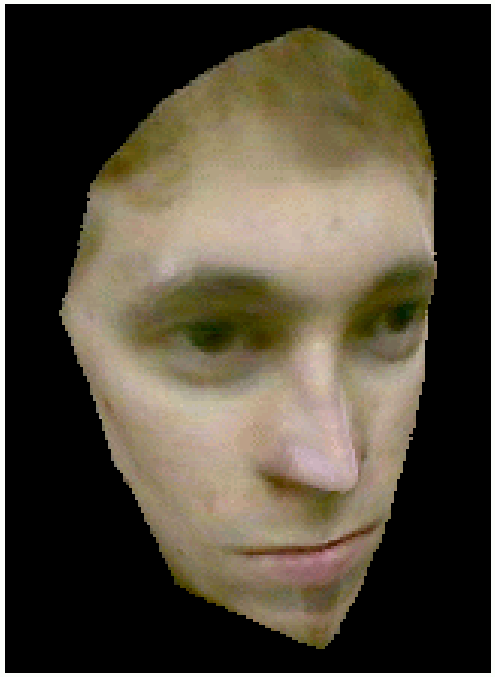
Combine color channels, weighted by the angle of the triangle surface normal.

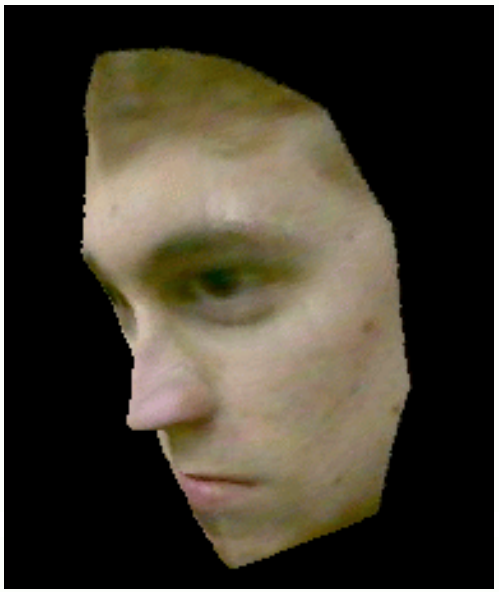




Results



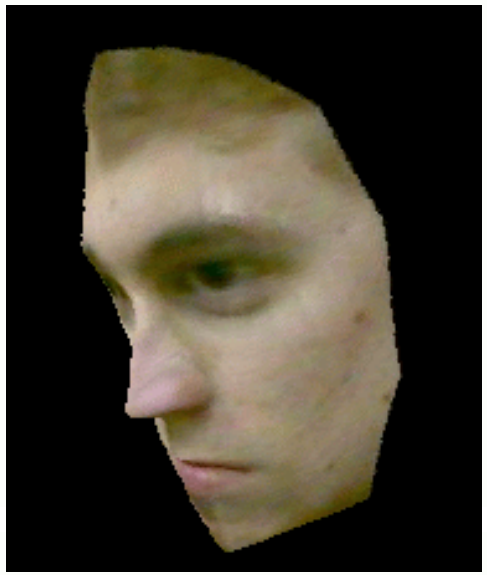
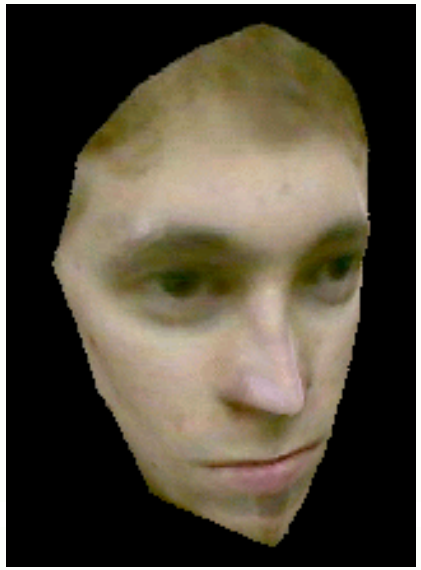




Results: Face Models





Results: Head Models

Conclusion

Photogrammetric facial modeling was achieved.

- Demonstrated model deformation process
- Implemented texture mapping algorithm

Extensions:

- Full body capture
- Expression morphing
- Automatic caricatures

Applications:

- Low-bandwidth video conferencing
- Computer gaming (avatar creation)
- Computer animation
- Multiple-view face recognition