Lecture -13

Making Faces

Guenter et al

SIGGARPH'98

Making Faces

- System for capturing 3D geometry and color and shading (texture map).
- Six cameras capture 182 color dots (six colors) on a face.
- 3D coordinates for each color dot are computed using pairs of images.
- Cyberware scanner is used to get dense wire frame model.

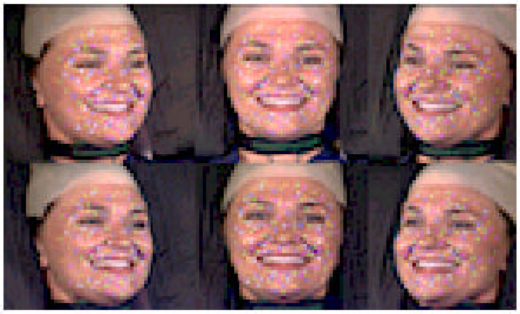
Making Faces

- Two models (cyberware and frame data) are related by a rigid transformation.
- Movement of each node in successive frames is computed by determining correspondence of nodes.

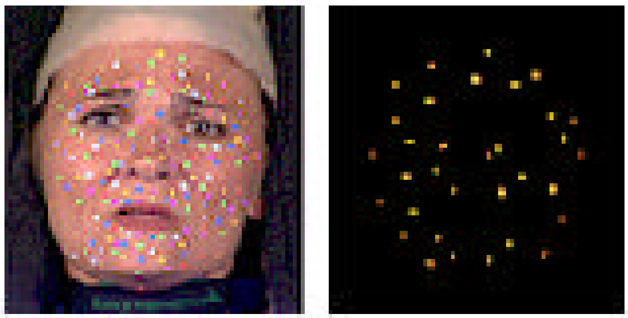
Applications

- Facial expressions
  - can be captured in a studio,
  - delivered via CDROM or internet to a user
  - reconstructed in real time on a user’s computer in a virtual 3D environment
- User can select
  - any arbitrary position for the face,
  - any virtual camera view point,
  - any size

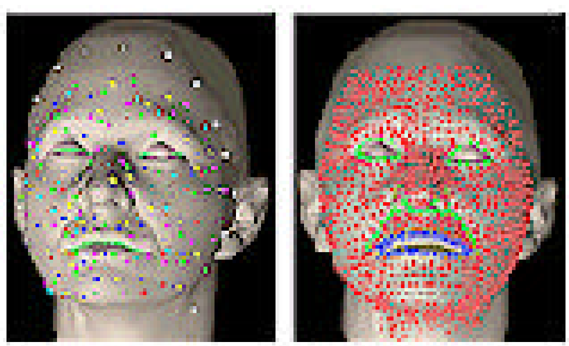
Six Views



Color Dots

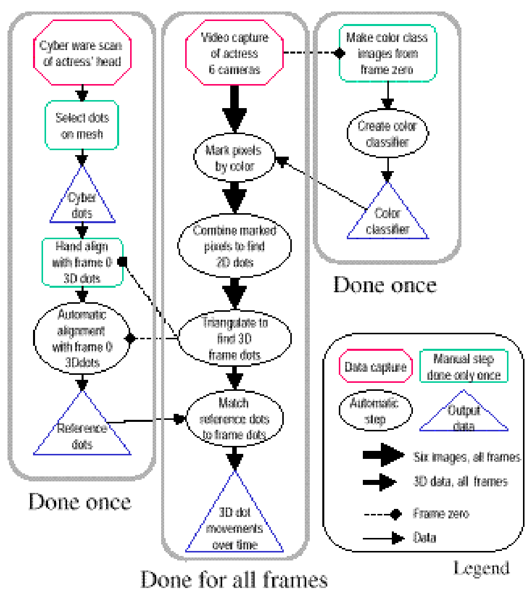


Wireframe Model

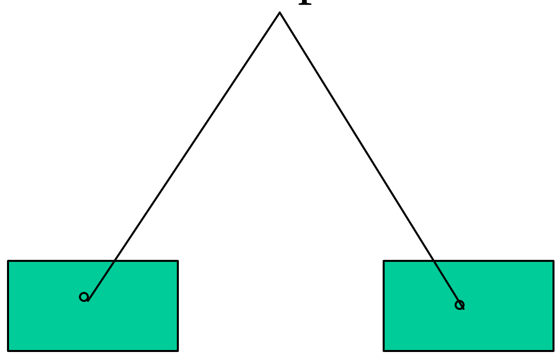


Main Steps

- 3-D reconstruction from 2-D dots
- Correspondence of Cyberware dots (reference) with 3-D frame dots
- Frame to frame dot correspondences
- Constructing the mesh
- Compression of Geometric Data



Intersection of two rays is 3-D point



3-D reconstruction from 2-D dots

- Generate all potential 2-D point correspondences for k cameras with n points in each camera: $\binom{k}{2} n^2$
- Each point correspondence gives rise to a 3-D candidate point defined as intersection of two rays cast from 2-D points.
- Project 3-D candidate point to each of two camera views,
  - if the projection is not within some bound from the centroid of either 2-D point then discard it as a potential 3-D candidate point.

3-D reconstruction from 2-D dots

- Each of the points in 3-D list is projected into a reference view, which is the camera with the best view of all points on the face.
  - If the projected point is not within a threshold distance from the centroid of 2-D dot it is deleted from the list
  - The remaining points constitute 3-D match list for that point
- For each 2-D point $\binom{m}{3}$ possible combinations of three points in the 3-D list are computed, and the combination with the smallest variance is chosen.
  - The average of three points in the best combination is the true 3-D position corresponding to a 2-D dot.

Correspondence of Cyberware dots (reference) with 3-D frame dots

- Obtain Cyberware scan of a face.
- Place reference dots on the Cyberware model by manually clicking on the dots.
- Align reference dots in Cyberware scan with the video frame dots.
  - Manually align frame dots in frame zero with the reference dots

Correspondence of Cyberware dots (reference) with 3-D frame dots

– Automatically align reference dots with frame dots in other frames by solving correspondence using graph matching
  - For each reference dot add an edge for every frame dot of the same color that is within a distance *e*.
  - Search for connected components of graph which has equal number of reference and frame dots
    – (most connected components will have two dots, one for reference and other from frame dots).

Correspondence of Cyberware dots (reference) with 3-D frame dots

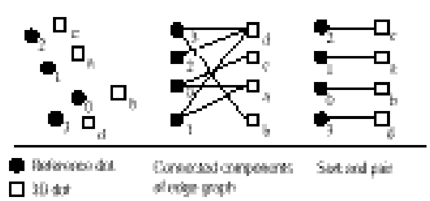
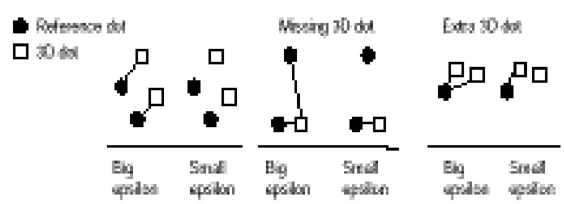


Figure 6: Matching dots.



Frame to frame dot correspondences

- Assume Cyberware scan as a reference nodes
- Solve correspondence between reference dots and frame dots for frame 0.
- For each frame *i>0* move the reference dots to the location in previous frame, then find the best match between the reference dot and neighboring frame dots.
- Move each reference dot to the location of its corresponding 3D location.

$$d_j^i = d_j + \vec{v}_j^i$$

Constructing The Mesh

- Move other vertices by a linear combination of the offsets of the nearest matching dots.

$$p_j^i = p_j + \sum_k a_k^j \| d_k^i - d_k \|$$

Compression of Geometric Data

- 182 3-D dots in each frame
- Use eigen vector approach to reduce dimensionality to only 45 principal components
- Need to transmit the coefficients and eigen vectors
- This reduces geometric data to 26kbps for coefficients, and 13kbps for eigen vectors

Compression

Figure 14: Left to Right: Mesh with uncompressed textures, compressed to 400 kbits/sec, and compressed to 200 kbits/sec

Original 400 kbps 200 kbps

Rendered Images

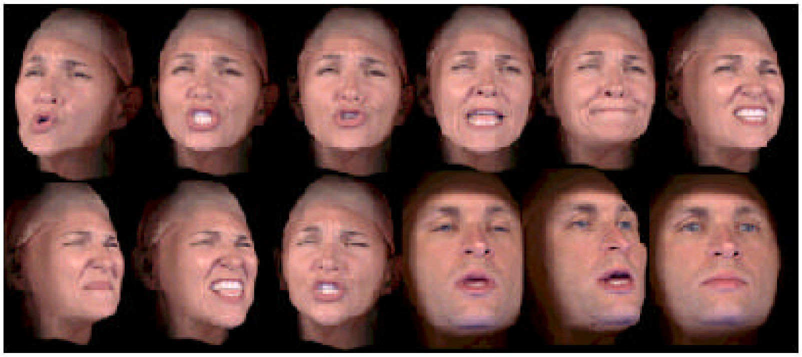


Figure 16: Sequence of rendered images of textured mesh.