

Model-Based Video Coding

Model-Based Compression

- Object-based
- Knowledge-based
- Semantic-based

Model-Based Compression

- Analysis
- Synthesis
- Coding

Video Compression

- MC/DCT
 - Source Model: translation motion only
 - Encoded Information: Motion vectors and color of blocks
- Object-Based
 - Source Model: moving **unknown** objects
 - translation only
 - affine
 - affine with triangular mesh
 - Encoded Information: Shape, motion, color of each moving object

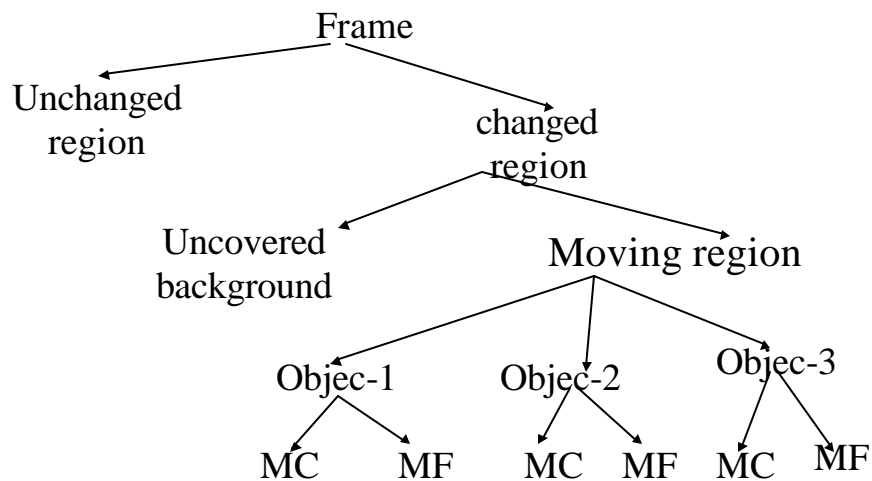
Video Compression

- Knowledge-Based
 - Source Model: Moving **known** objects
 - Encoded Information: Shape, motion and color of known objects
- Semantic
 - Source Model: Facial Expressions
 - Encoded Information: Action units

Expected bits per CIF (352 x 288) frame

Source Model	Motion	Shape	Color
2-D Rigid affine motion	600	1300	15000
2-D flexible translation motion	1100	900	4000
3-D rigid 3-D motion	200	1640	4000

Object-Based Coding

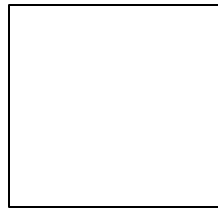


2-D objects With Affine Motion

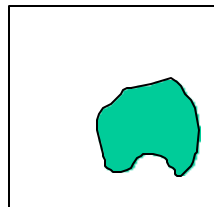
- Analysis
 - Segment image by change detection
 - Compute motion parameters, e.g. affine ($x' = Ax + b$)
 - Synthesize the region in the current frame using previous frame and the motion parameters
 - if the difference between actual and synthesized region is significant, recursively segment the region into small regions

2-D objects With Affine Motion

- Synthesis
 - Using final segmented regions and the motion parameters synthesize the frame, and compute the synthesis error.
- Coding
 - Code motion parameters (using 6 to 7 bits each)
 - Code region shapes
 - Code prediction errors



Frame k-1



Frame k

2-D Objects With Translation Motion

- Compute change detection mask
- Estimate motion vectors for pixels in the changed region using block matching method.
 - Estimate motion on a sparse grid of pixels using block matching
 - Compute dense motion field using bilinear interpolation
- Compute ternary mask: unchanged areas, moving areas, uncovered background

2-D Objects With Translation Motion

- Approximate shape of moving area using polygon/splines.
- Synthesize moving regions using motion and shape information.
- Determine Model Failure (MF) regions
- Code pixel data in MF and uncovered background regions using shape & color.
- Code motion and shape for MC regions

Affine Transformation With Triangular Meshes

- Partition the current frame into triangular patches.
- Estimate rough motion vectors at vertices of triangles using block matching.
- Fit affine transformation to three vertices of each triangle using the motion vectors from block matching.
- Synthesize the current frame using affine transformation. Compute synthesized error.

Affine Transformation With Triangular Meshes

- Encode at each grid point
 - motion vectors
 - synthesis error
 - no shape information needs to be coded

Contents

- Estimation using rigid+non-rigid motion model
- Making Faces (SIGGRAPH-98)
- Synthesizing Realistic Facial Expressions from Photographs (SIGGRAPH-98)
- MPEG-4
- MPEG-7

Model-Based Image Coding

- The transmitter and receiver both possess the same 3D face model and texture images.
- During the session, at the transmitter the facial motion parameters: global and local, are extracted.
- At the receiver the image is synthesized using estimated motion parameters.
- The difference between synthesized and actual image can be transmitted as residuals.

Face Model

- Candide model has 108 nodes, 184 polygons.
- Candide is a generic head and shoulder model. It needs to be conformed to a particular person's face.
- Cyberware scan gives head model consisting of 460,000 polygons.
- Another face model was created by sticking 182 color dots on the face, and capturing dots by six cameras.

Wireframe Model Fitting

- Fit orthographic projection of wireframe to the frontal view of speaker using Affine transformation.
- Locate four features in the image and the projection of model.
- Find parameters of Affine using least squares fit.
- Apply Affine to all vertices, and scale depth.

Synthesis

- Collapse initial wire frame onto the image to obtain a collection of triangles.
- Map observed texture in the first frame into respective triangles.
- Rotate and translate the initial wire frame according to global and local motion, and collapse onto the next frame.
- Map texture within each triangle from first frame to the next frame by interpolation.

Video Phones

Motion Estimation

Perspective Projection (optical flow)

$$u = f\left(\frac{V_1}{Z} + \Omega_2\right) - \frac{V_3}{Z}x - \Omega_3y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f\left(\frac{V_2}{Z} - \Omega_1\right) + \Omega_3x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

Optical Flow Constraint Eq

$$f_x u + f_y v + f_t = 0$$

$$\begin{aligned}
& f_x \left(f \left(\frac{V_1}{Z} + \Omega_2 \right) - \frac{V_3}{Z} x - \Omega_3 y - \frac{\Omega_1}{f} xy + \frac{\Omega_2}{f} x^2 \right) + f_y \\
& \left(f \left(\frac{V_2}{Z} - \Omega_1 \right) + \Omega_3 x - \frac{V_3}{Z} y + \frac{\Omega_2}{f} xy - \frac{\Omega_1}{f} y^2 \right) + f_t = 0 \\
& \left(f_x \frac{f}{Z} \right) V_1 + \left(f_y \frac{f}{Z} \right) V_2 + \left(\frac{f}{Z} (f_x x - f_y y) \right) V_3 + \\
& \left(-f_x \frac{xy}{f} + f_y \frac{y^2}{f} - f_y f \right) \Omega_1 + \left(f_x f + f_x \frac{x^2}{f} + f_y \frac{xy}{f} \right) \Omega_2 + \\
& (f_x y + f_y x) \Omega_3 = -f_t
\end{aligned}$$

$$\begin{aligned}
& \left(f_x \frac{f}{Z} \right) V_1 + \left(f_y \frac{f}{Z} \right) V_2 + \left(\frac{f}{Z} (f_x x - f_y y) \right) V_3 + \\
& \left(-f_x \frac{xy}{f} + f_y \frac{y^2}{f} - f_y f \right) \Omega_1 + \left(f_x f + f_x \frac{x^2}{f} + f_y \frac{xy}{f} \right) \Omega_2 + \\
& (f_x y + f_y x) \Omega_3 = -f_t
\end{aligned}$$

$$\mathbf{Ax} = \mathbf{b} \quad \text{Solve by Least Squares}$$

$$\mathbf{x} = (V_1, V_2, V_3, \Omega_1, \Omega_2, \Omega_3)$$

$$A = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (f_x \frac{f}{Z}) & (f_y \frac{f}{Z}) & (\frac{f}{Z}(f_x x - f_y y)) & (-f_x \frac{xy}{f} + f_y \frac{y^2}{f} - f_y f) & (f_x f + f_x \frac{x^2}{f} + f_y \frac{xy}{f}) & (f_x y + f_y x) \end{bmatrix}$$

Comments

- This is a simpler (linear) problem than sfm because depth is assumed to be known.
- Since no optical flow is computed, this is called “direct method”.
- Only spatiotemporal derivatives are computed from the images.

Problem

- We have used 3D rigid motion, but face is not purely rigid!
- Facial expressions produce non-rigid motion.
- Use global rigid motion and non-rigid deformations.

3-D Rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{a} & \mathbf{b} \\ \mathbf{a} & 1 & -\mathbf{g} \\ -\mathbf{b} & \mathbf{g} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\mathbf{a} & \mathbf{b} \\ \mathbf{a} & 0 & -\mathbf{g} \\ -\mathbf{b} & \mathbf{g} & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

3-D Rigid Motion

$$\begin{bmatrix} X' - X \\ Y' - Y \\ Z' - Z \end{bmatrix} = \begin{bmatrix} 0 & -\mathbf{a} & \mathbf{b} \\ \mathbf{a} & 0 & -\mathbf{g} \\ -\mathbf{b} & \mathbf{g} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

$$\dot{\mathbf{X}} = \boldsymbol{\Omega} \times \mathbf{X} + \mathbf{V}$$

3-D Rigid+Non-rigid Motion

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T} + \mathbf{E}\Phi$$

$$\mathbf{E} = \begin{bmatrix} e_{11} & e_{12} & \dots & e_{1m} \\ e_{21} & e_{22} & \dots & e_{2m} \\ e_{31} & e_{32} & \dots & e_{3m} \end{bmatrix}$$

Facial expressions

Action Units:

- opening of a mouth
- closing of eyes
- raising of eyebrows

$$\Phi = (\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_m)^T$$

3-D Rigid+Non-rigid Motion

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{a} & \mathbf{b} \\ \mathbf{a} & 1 & -\mathbf{g} \\ -\mathbf{b} & \mathbf{g} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^m e_{1i} \mathbf{f}_i \\ T_Y + \sum_{i=1}^m e_{2i} \mathbf{f}_i \\ T_Z + \sum_{i=1}^m e_{3i} \mathbf{f}_i \end{bmatrix}$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \left(\begin{bmatrix} 0 & -\mathbf{a} & \mathbf{b} \\ \mathbf{a} & 0 & -\mathbf{g} \\ -\mathbf{b} & \mathbf{g} & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^m e_{1i} \mathbf{f}_i \\ T_Y + \sum_{i=1}^m e_{2i} \mathbf{f}_i \\ T_Z + \sum_{i=1}^m e_{3i} \mathbf{f}_i \end{bmatrix}$$

3-D Rigid+Non-rigid Motion

$$\begin{bmatrix} X' - X \\ Y' - Y \\ Z' - Z \end{bmatrix} = \begin{bmatrix} 0 & -\mathbf{a} & \mathbf{b} \\ \mathbf{a} & 0 & -\mathbf{g} \\ -\mathbf{b} & \mathbf{g} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^m e_{1i} \mathbf{f}_i \\ T_Y + \sum_{i=1}^m e_{2i} \mathbf{f}_i \\ T_Z + \sum_{i=1}^m e_{3i} \mathbf{f}_i \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X + \sum_{i=1}^m e_{1i} \mathbf{f}_i \\ T_Y + \sum_{i=1}^m e_{2i} \mathbf{f}_i \\ T_Z + \sum_{i=1}^m e_{3i} \mathbf{f}_i \end{bmatrix}$$

$$\dot{\mathbf{X}} = \boldsymbol{\Omega} \times \mathbf{X} + \mathbf{D}$$

3-D Rigid+Non-rigid Motion

$$\dot{X} = -\Omega_3 Y + \Omega_2 Z + V_1 + \sum_{i=1}^m e_{1i} \mathbf{f}_i$$

$$\dot{Y} = \Omega_3 X - \Omega_1 Z + V_2 + \sum_{i=1}^m e_{2i} \mathbf{f}_i$$

$$\dot{Z} = -\Omega_2 X + \Omega_1 Y + V_3 + \sum_{i=1}^m e_{3i} \mathbf{f}_i$$

Perspective Projection (arbitrary flow)

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f \frac{\dot{X}}{Z} - x \frac{\dot{Z}}{Z}$$

$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f \frac{\dot{Y}}{Z} - y \frac{\dot{Z}}{Z}$$

Perspective Projection (arbitrary flow)

$$u = \dot{x} = \frac{fZ\dot{X} - fX\dot{Z}}{Z^2} = f \frac{\dot{X}}{Z} - x \frac{\dot{Z}}{Z}$$

$$v = \dot{y} = \frac{fZ\dot{Y} - fY\dot{Z}}{Z^2} = f \frac{\dot{Y}}{Z} - y \frac{\dot{Z}}{Z}$$

$$u = f \left(\frac{V_1 + \sum_{i=1}^m e_{1i} f_i}{Z} + \Omega_2 \right) - \frac{V_3 + \sum_{i=1}^m e_{3i} f_i}{Z} x - \Omega_3 y - \frac{\Omega_1}{f} xy + \frac{\Omega_2}{f} x^2$$

$$v = f \left(\frac{V_2 + \sum_{i=1}^m e_{2i} f_i}{Z} - \Omega_1 \right) + \Omega_3 x - \frac{V_3 + \sum_{i=1}^m e_{3i} f_i}{Z} y + \frac{\Omega_2}{f} xy - \frac{\Omega_1}{f} y^2$$

Optical Flow Constraint Eq

$$f_x u + f_y v + f_t = 0$$

$$\mathbf{Ax} = \mathbf{b}$$

Estimation Using Flexible
Wireframe Model

- Model photometric effects
- Simultaneously compute 3D motion and adapt the wireframe model.

Generalized Optical Flow Constraint

$$f(x, y, t) = \mathbf{r}N(t).L \quad \text{Lambertian Model}$$

$$\frac{df(x, y, t)}{dt} = \mathbf{r}L \cdot \frac{dN}{dt}$$

$$f_x u + f_y v + f_t = \mathbf{r}L \cdot \frac{dN}{dt}$$

Perspective Projection (optical flow)

$$u = f\left(\frac{V_1}{Z} + \Omega_2\right) - \frac{V_3}{Z}x - \Omega_3y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2$$

$$v = f\left(\frac{V_2}{Z} - \Omega_1\right) + \Omega_3x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2$$

$$\begin{aligned} & f_x\left(f\left(\frac{V_1}{Z} + \Omega_2\right) - \frac{V_3}{Z}x - \Omega_3y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2\right) + f_y \\ & \left(f\left(\frac{V_2}{Z} - \Omega_1\right) + \Omega_3x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2\right) + f_t = \mathbf{rL} \cdot \frac{dN}{dt} \\ & f_x\left(f\left(\frac{V_1}{Z} + \Omega_2\right) - \frac{V_3}{Z}x - \Omega_3y - \frac{\Omega_1}{f}xy + \frac{\Omega_2}{f}x^2\right) + f_y \\ & \left(f\left(\frac{V_2}{Z} - \Omega_1\right) + \Omega_3x - \frac{V_3}{Z}y + \frac{\Omega_2}{f}xy - \frac{\Omega_1}{f}y^2\right) \\ & + f_t = \mathbf{rL} \cdot \left[\frac{(-p', -q', 1)^T}{\sqrt{p'^2 + q'^2 + 1}} - \frac{(-p, -q, 1)^T}{\sqrt{p^2 + q^2 + 1}} \right] \end{aligned}$$

$$\begin{aligned}
& f_x \left(f \left(\frac{V_1}{Z} + \Omega_2 \right) - \frac{V_3}{Z} x - \Omega_3 y - \frac{\Omega_1}{f} xy + \frac{\Omega_2}{f} x^2 \right) + f_y \\
& \left(f \left(\frac{V_2}{Z} - \Omega_1 \right) + \Omega_3 x - \frac{V_3}{Z} y + \frac{\Omega_2}{f} xy - \frac{\Omega_1}{f} y^2 \right) + f_t = \\
& \mathbf{nL} \cdot \left[\frac{(-p', -q', 1)^T}{\sqrt{p'^2 + q'^2 + 1}} - \frac{(-p, -q, 1)^T}{\sqrt{p^2 + q^2 + 1}} \right]
\end{aligned}$$

$$P_1^{(i)} = (X_1^{(i)}, Y_1^{(i)}, Z_1^{(i)})$$

$$P_2^{(i)} = (X_2^{(i)}, Y_2^{(i)}, Z_2^{(i)})$$

$$P_3^{(i)} = (X_3^{(i)}, Y_3^{(i)}, Z_3^{(i)})$$

$$P^{(i)} = (X^{(i)}, Y^{(i)}, Z^{(i)})$$

$$\overline{P^{(i)} P_1^{(i)}} \cdot \overline{(P_2^{(i)} P_1^{(i)} \times P_3^{(i)} P_1^{(i)})} = 0$$

$$Z^{(i)} = p_i X^{(i)} + q_i Y^{(i)} + c_i$$

$$p_i = \frac{(Y_2^{(i)} - Y_1^{(i)})(Z_3^{(i)} - Z_1^{(i)}) - (Z_2^{(i)} - Z_1^{(i)})(Y_3^{(i)} - Y_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

$$q_i = \frac{(Z_2^{(i)} - Z_1^{(i)})(X_3^{(i)} - X_1^{(i)}) - (X_2^{(i)} - X_1^{(i)})(Z_3^{(i)} - Z_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

$$c_i = Z_1^{(i)} + X_1^{(i)} \frac{(Y_2^{(i)} - Y_1^{(i)})(Z_3^{(i)} - Z_1^{(i)}) - (Z_2^{(i)} - Z_1^{(i)})(Y_3^{(i)} - Y_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})} +$$

$$Y_1^{(i)} \frac{(Z_2^{(i)} - Z_1^{(i)})(X_3^{(i)} - X_1^{(i)}) - (X_2^{(i)} - X_1^{(i)})(Z_3^{(i)} - Z_1^{(i)})}{(X_2^{(i)} - X_1^{(i)})(Y_3^{(i)} - Y_1^{(i)}) - (Y_2^{(i)} - Y_1^{(i)})(X_3^{(i)} - X_1^{(i)})}$$

$$E = \sum_i \sum_{(x,y) \in \text{ithpatch}} e_i^2$$

$$e_i(x, y) = f_x(\Omega_3 y - \Omega_2(p_i x + q_i y + c_i) + V_1)$$

$$+ f_y(-\Omega_3 x + \Omega_1(p_i x + q_i y + c_i) + V_2) + f_t$$

$$- \mathbf{r}(L_1, L_2, L_3) \cdot \left(\frac{-\Omega_2 + p_i}{1 + \Omega_2 p_i}, \frac{-\Omega_1 + q_i}{1 - \Omega_1 q_i} \right) -$$

$$\left(\left(\frac{-\Omega_2 + p_i}{1 + \Omega_2 p_i} \right)^2 + \left(\frac{-\Omega_1 + q_i}{1 - \Omega_1 q_i} \right)^2 + 1 \right)^{1/2}$$

$$\frac{(-p_i, -q_i, 1)}{(p_i^2 + q_i^2 + 1)^{1/2}}$$

Algorithm

- Estimate light source direction
- Initialize coordinates of all nodes using approximately scaled wireframe model.
Determine initial values of surface normals for each triangle.
- Determine initial motion parameters based on selected feature correspondences and their depth values from wireframe model.
- Compute the adjustments to motion parameters and depth values by minimizing the error function E .

Making Faces

Guenter et al
SIGGRAPH'98

Making Faces

- System for capturing 3D geometry and color and shading (texture map).
- Six cameras capture 182 color dots 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 are related by a rigid transformation.
- Movement of each node in successive frames is computed by determining correspondence of nodes.

Synthesizing Realistic Facial Expressions from Photographs

Pighin et al
SIGGRAPH'98

Synthesizing Realistic Facial Expressions

- Select 13 feature points manually in face image corresponding to points in face model created with Alias.
- Estimate camera poses and deformed 3d model points.
- Use these deformed values to deform the remaining points on the mesh using interpolation.

Synthesizing Realistic Facial Expressions

- Extract texture.
- Create new expressions using morphing.

3D Rigid Transformation

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}$$

Camera coordinates

Wireframe coordinates

$$x_i'^k = f^k \frac{X_i'^k}{Z_i'^k}, y_i'^k = f^k \frac{Y_i'^k}{Z_i'^k} \quad \text{perspective}$$

3D Rigid Transformation

$$x_i'^k = f^k \frac{X_i'^k}{Z_i'^k}, y_i'^k = f^k \frac{Y_i'^k}{Z_i'^k}$$

$$x_i'^k = f_k \frac{r_{11}^k X_i + r_{12}^k Y_i + r_{13}^k Z_i + T_X^k}{r_{31}^k X_i + r_{32}^k Y_i + r_{33}^k Z_i + T_Z^k}$$

$$y_i'^k = f_k \frac{r_{21}^k X_i + r_{22}^k Y_i + r_{23}^k Z_i + T_Y^k}{r_{31}^k X_i + r_{32}^k Y_i + r_{33}^k Z_i + T_Z^k}$$

Model Fitting

$$x_i'^k = f_k \frac{\mathbf{r}_x^k \mathbf{p}_i + T_X^k}{\mathbf{r}_z^k \mathbf{p}_i + T_Z^k}$$

$$y_i'^k = f_k \frac{\mathbf{r}_y^k \mathbf{p}_i + T_Y^k}{\mathbf{r}_z^k \mathbf{p}_i + T_Z^k}$$

Model Fitting

$$x_i'^k = f_k \frac{\mathbf{r}_x^k \mathbf{p}_i + T_X^k}{\mathbf{r}_z^k \mathbf{p}_i + T_Z^k}$$

$$y_i'^k = f_k \frac{\mathbf{r}_y^k \mathbf{p}_i + T_Y^k}{\mathbf{r}_z^k \mathbf{p}_i + T_Z^k}$$

$$x_i'^k = s_k \frac{\mathbf{r}_x^k \mathbf{p}_i + T_X^k}{1 + \mathbf{h}^k \mathbf{r}_z^k \mathbf{p}_i}$$

$$y_i'^k = s_k \frac{\mathbf{r}_y^k \mathbf{p}_i + T_Y^k}{1 + \mathbf{h}^k \mathbf{r}_z^k \mathbf{p}_i}$$

Model Fitting

$$x_i'^k = s_k \frac{\mathbf{r}_x^k \mathbf{p}_i + T_X^k}{1 + \mathbf{h}^k \mathbf{r}_z^k \mathbf{p}_i}$$

$$y_i'^k = s_k \frac{\mathbf{r}_y^k \mathbf{p}_i + T_Y^k}{1 + \mathbf{h}^k \mathbf{r}_z^k \mathbf{p}_i}$$

$$w_i^k (x_i'^k + x_i'^k \mathbf{h}^k (\mathbf{r}_z^k \cdot \mathbf{p}_i)) - s^k (\mathbf{r}_X^k \cdot \mathbf{p}_i + T_X^k) = 0$$

$$w_i^k (y_i'^k + y_i'^k \mathbf{h}^k (\mathbf{r}_z^k \cdot \mathbf{p}_i)) - s^k (\mathbf{r}_Y^k \cdot \mathbf{p}_i + T_Y^k) = 0$$

Model Fitting

- Solve for unknowns in five steps:

$$s^k; \mathbf{p}_i; \mathbf{R}^k; T_X^k, T_Y^k; \mathbf{h}^k$$

- Use linear least squares fit.
- When solving for an unknown, assume other parameters are known.

Interpolation

- Use initial set of coordinates for the feature points (13 points), to deform the remaining vertices using interpolation.

Interpolation

$$f(\mathbf{p}) = \sum_i c_i f(\|\mathbf{p} - \mathbf{p}_i\|) + \mathbf{M}\mathbf{p} + \mathbf{t}$$

$$u_i = \mathbf{p}_i - \mathbf{p}_i^0, u_i = f(\mathbf{p}_i)$$

$$\sum_i c_i = 0, \sum_i c_i \mathbf{p}_i = 0$$

$$f(r) = e^{\frac{-r}{64}}$$

Texture Extraction

$$T(\mathbf{p}) = \frac{\sum_k m^k(\mathbf{p}) I^k(x^k, y^k)}{\sum_k m^k(\mathbf{p})}$$

I^k is k-th image

m^k is weight

Weights

- Self-occlusion
- Smoothness
- Positional certainty
- View similarity

Texture Extraction

- Visibility map $F^k(u, v)$ is set to 1 if the corresponding point p is visible in k -th image, and zero otherwise.
- Positional certainty, $P^k(\mathbf{p})$ is define as a dot product of surface normal at p and the k -th direction of projection.

Texture Extraction

- View-independent texture mapping:

$$m^k(u, v) = F^k(u, v)P^k(\mathbf{p})$$

- View-dependent texture mapping:

$$m^k(u, v) = F^k(x^k, y^k)P^k(\mathbf{p})V^k(d)$$

$$V^k(\mathbf{d}) = \mathbf{d} \cdot \mathbf{d}^k - \mathbf{d}^l \cdot \mathbf{d}^{l+1}$$