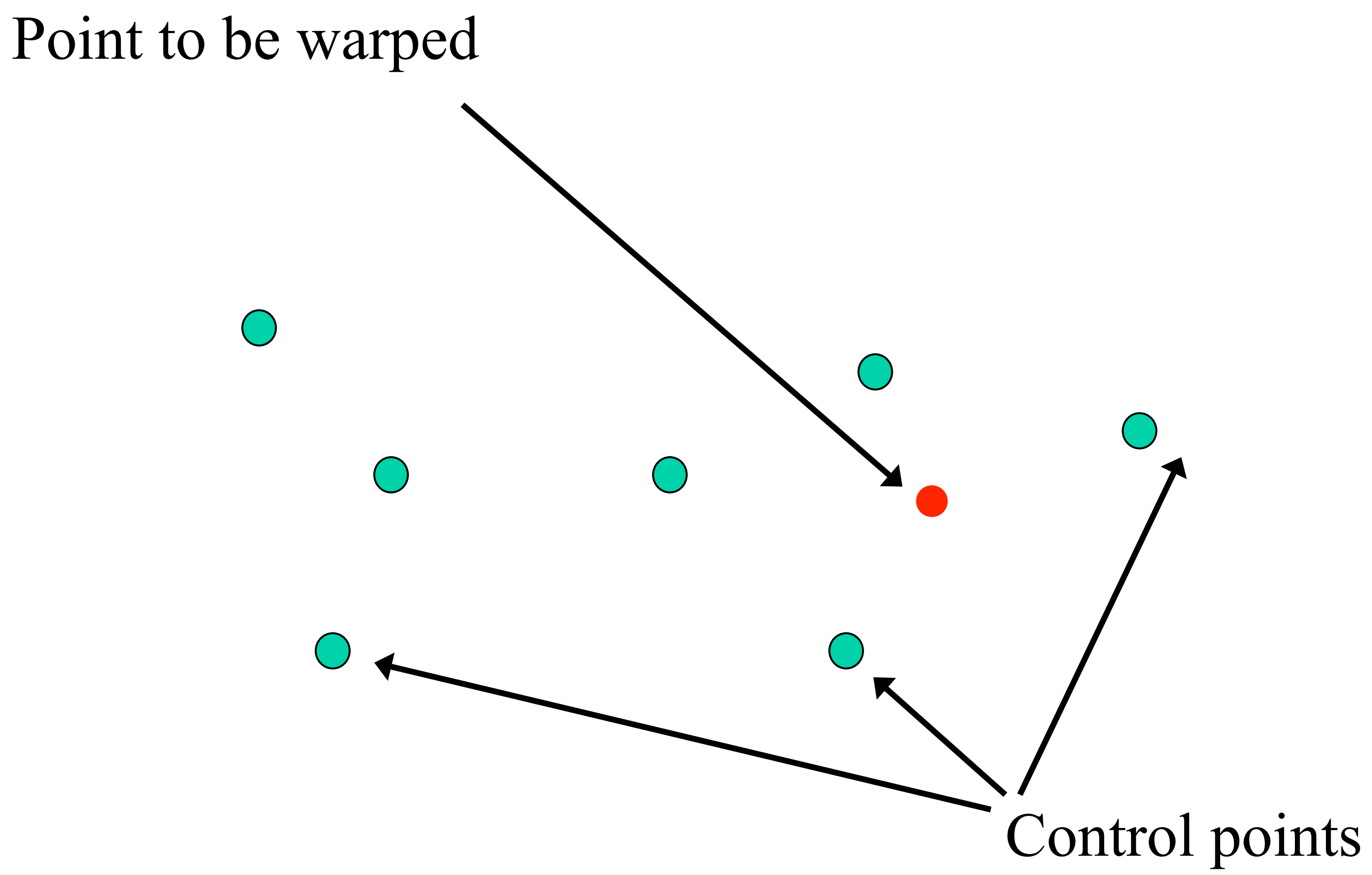
Interactive Computer Graphics: Lecture 16

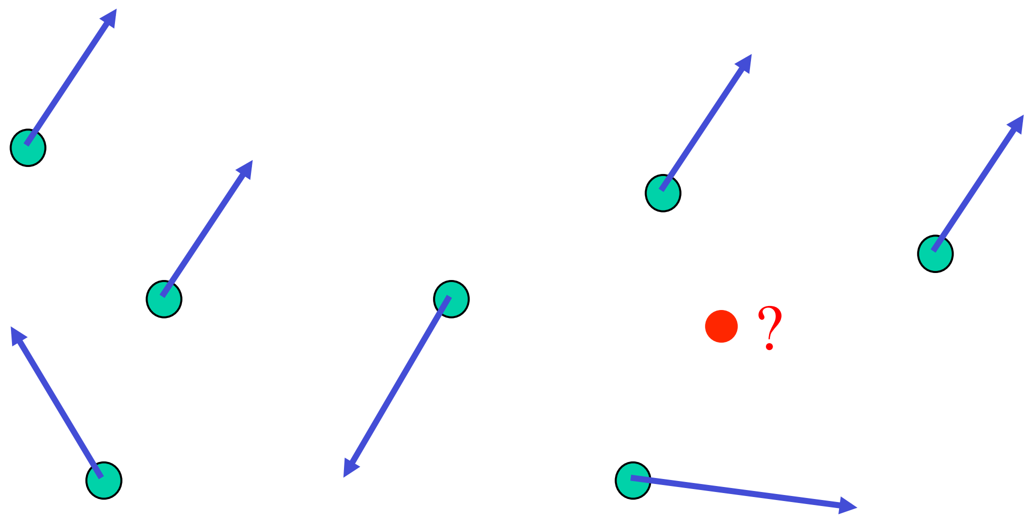
Warping and Morphing (cont.)

Non-rigid transformation



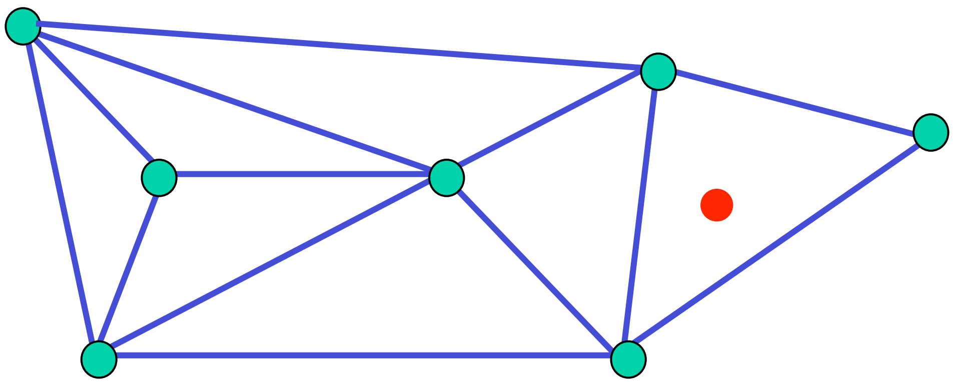
Non-rigid transformation

- For each control point we have a displacement vector
- How do we interpolate the displacement at a pixel?



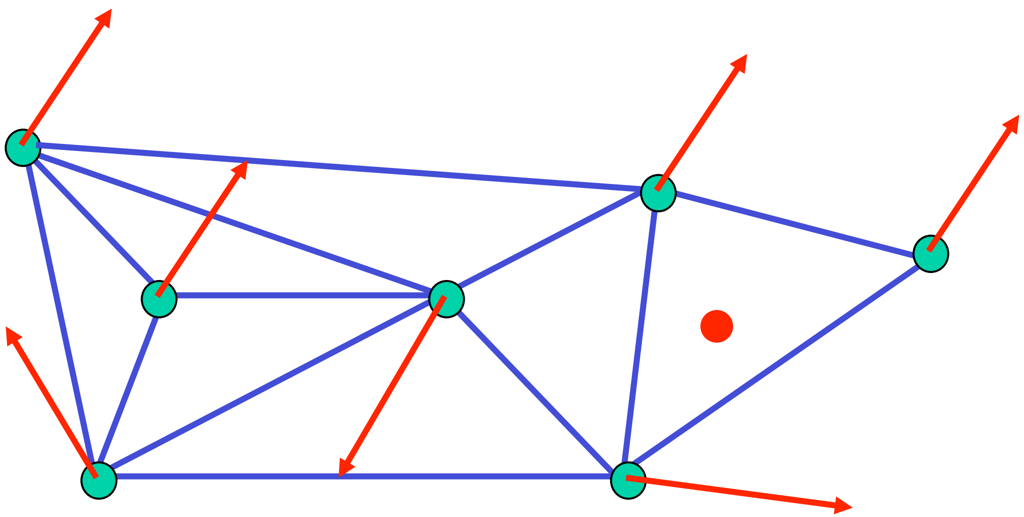
Non-rigid transformation: Piecewise affine

- Partition the convex hull of the control points into a set of triangles



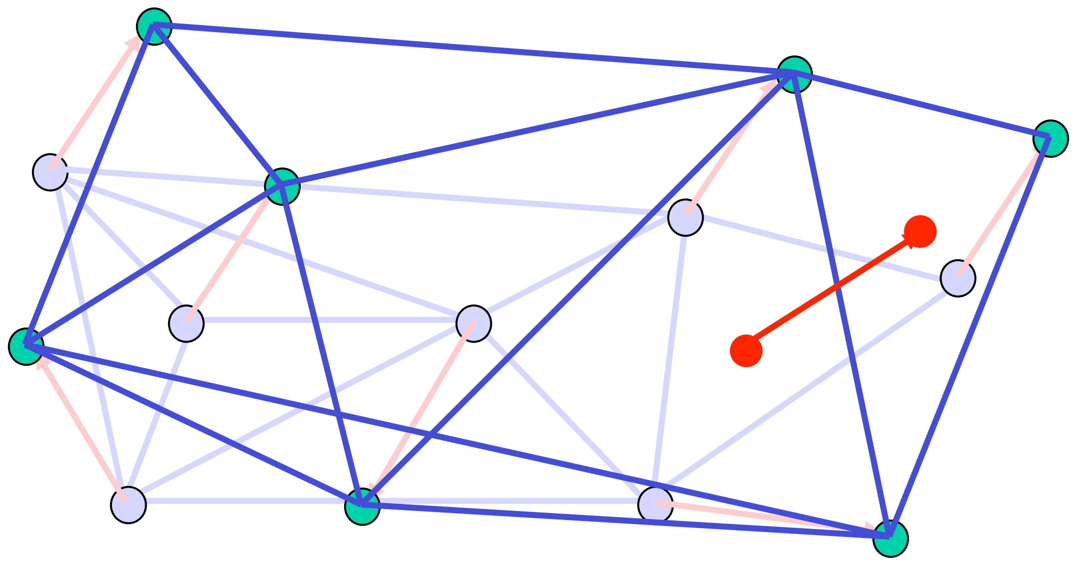
Non-rigid transformation: Piecewise affine

- Partition the convex hull of the control points into a set of triangles



Non-rigid transformation: Piecewise affine

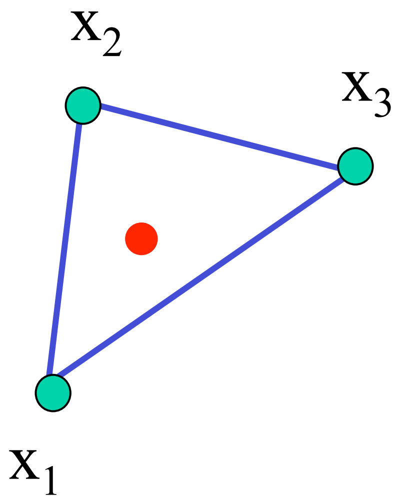
- Partition the convex hull of the control points into a set of triangles



Non-rigid transformation: Piecewise affine

- Find triangle which contains point **p** and express in terms of the vertices of the triangle:

$$\mathbf{p} = \mathbf{x}_1 + \alpha(\mathbf{x}_2 - \mathbf{x}_1) + \beta(\mathbf{x}_3 - \mathbf{x}_1)$$



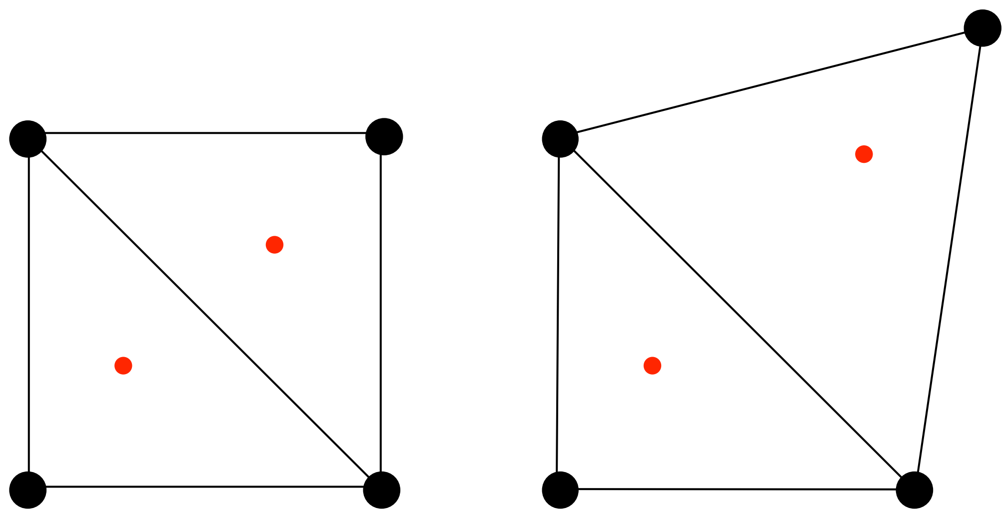
Non-rigid transformation: Piecewise affine

- Or $\mathbf{p} = \gamma\mathbf{x}_1 + \alpha\mathbf{x}_2 + \beta\mathbf{x}_3$ with $\gamma = 1 - (\alpha + \beta)$

- Under the affine transformation this point simply maps to

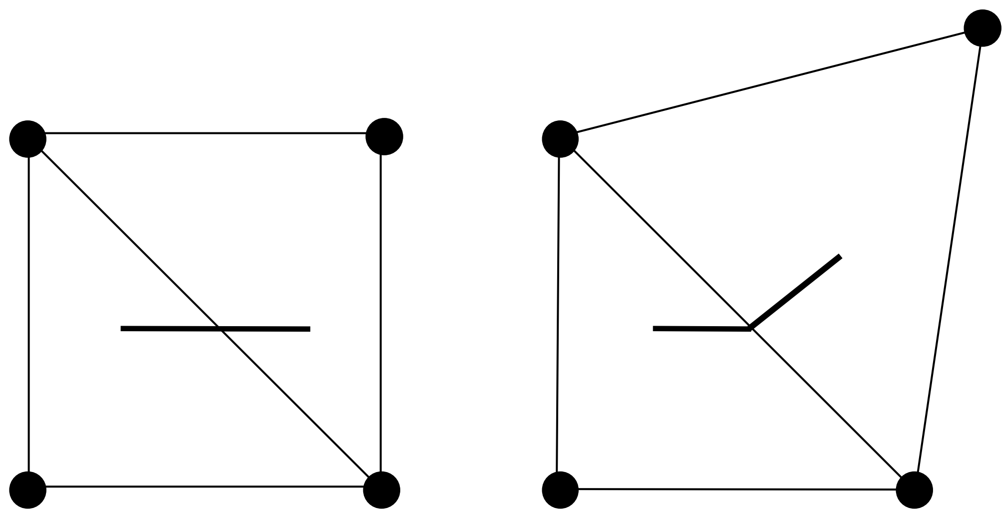
$$\mathbf{p}' = \gamma\mathbf{x}_1' + \alpha\mathbf{x}_2' + \beta\mathbf{x}_3'$$

Non-rigid transformation: Piecewise affine



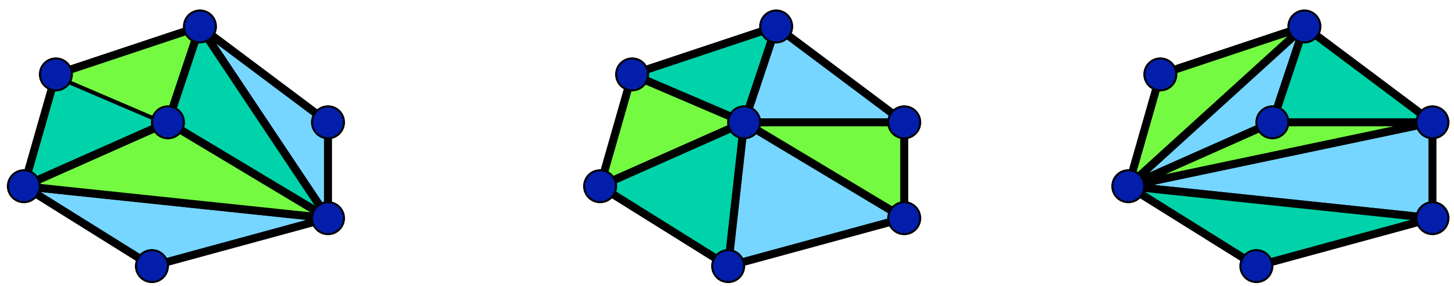
Non-rigid transformation: Piecewise affine

- Problem: Produces continuous deformations, but the deformation may not be smooth. Straight lines can be kinked across boundaries between triangles



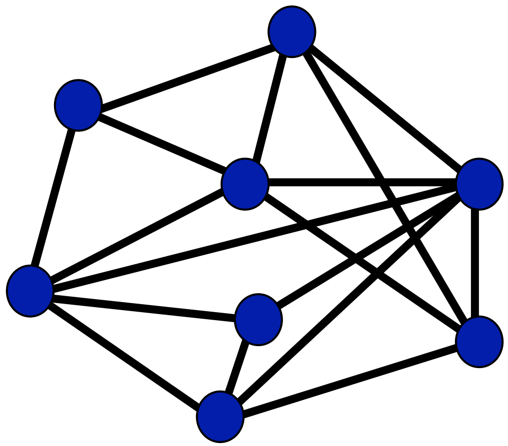
Triangulations

- A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.
- There are an exponential number of triangulations of a point set.



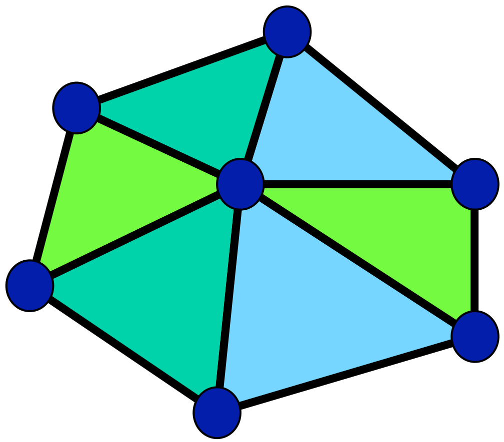
An O(n^3) Triangulation Algorithm

- Repeat until impossible:
  - Select two sites.
  - If the edge connecting them does not intersect previous edges, keep it.

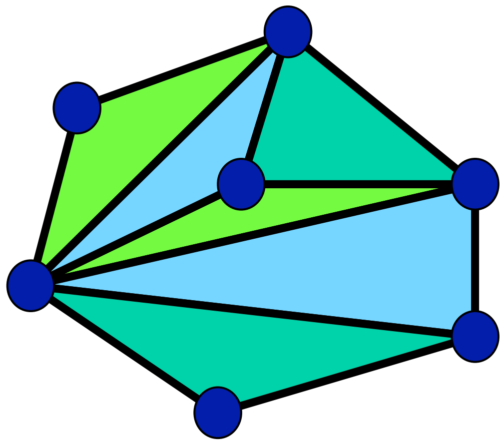


"Quality" Triangulations

- Let $\alpha(T) = (\alpha_1, \alpha_2, \ldots, \alpha_{3t})$ be the vector of angles in the triangulation T in increasing order.
- A triangulation T_1 will be "better" than T_2 if $\alpha(T_1) > \alpha(T_2)$ lexicographically.
- The Delaunay triangulation is the "best"
  - Maximizes smallest angles



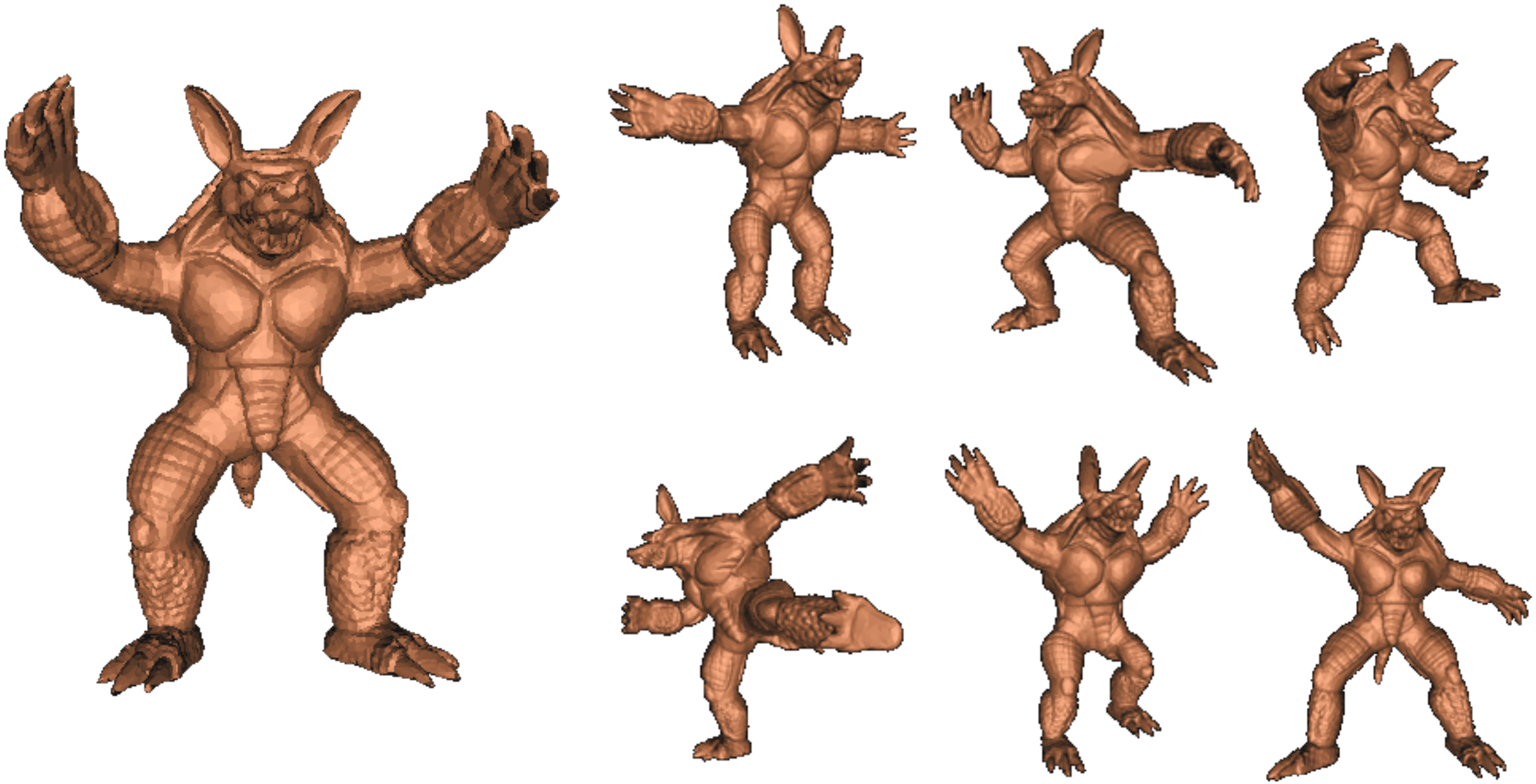
good



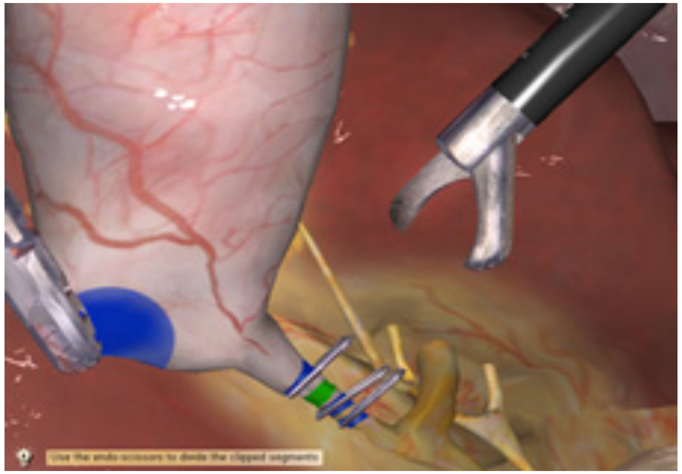
bad

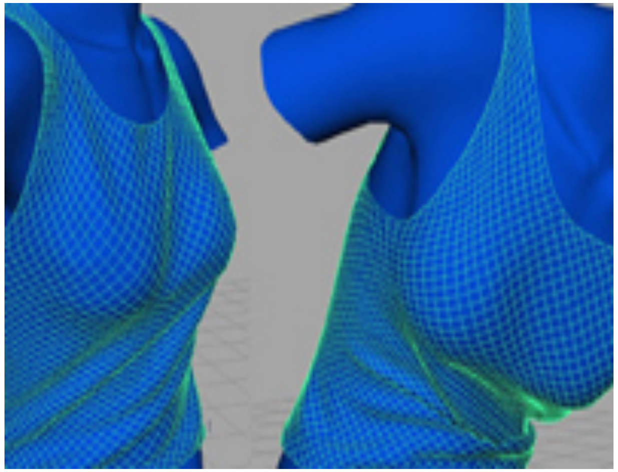
Modelling 3D Deformations

Modelling 3D Deformations



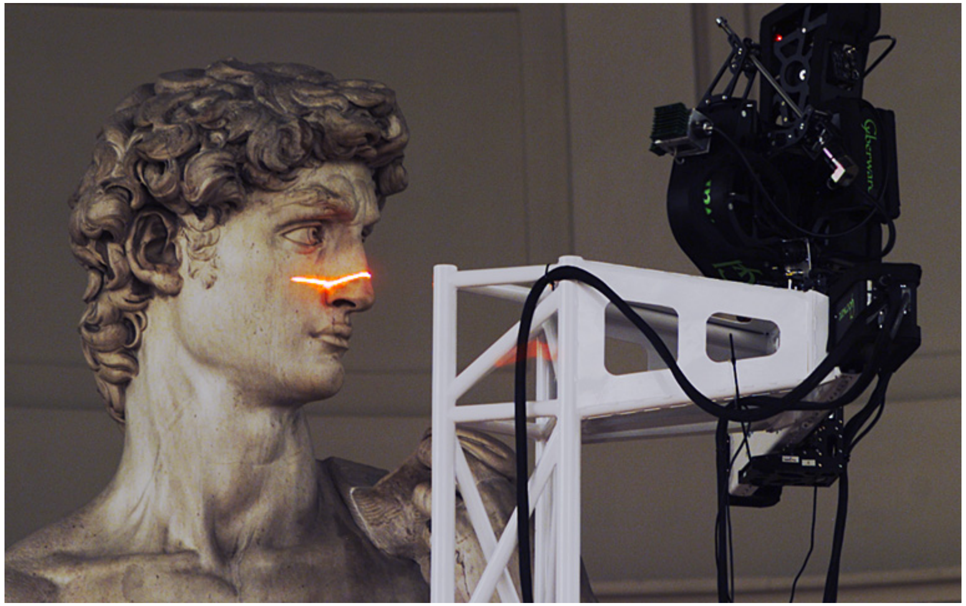
Modelling 3D Deformations: Applications





Challenges in Modelling 3D Deformations

- Large meshes – millions of polygons
- Need efficient techniques for computing and specifying the deformation



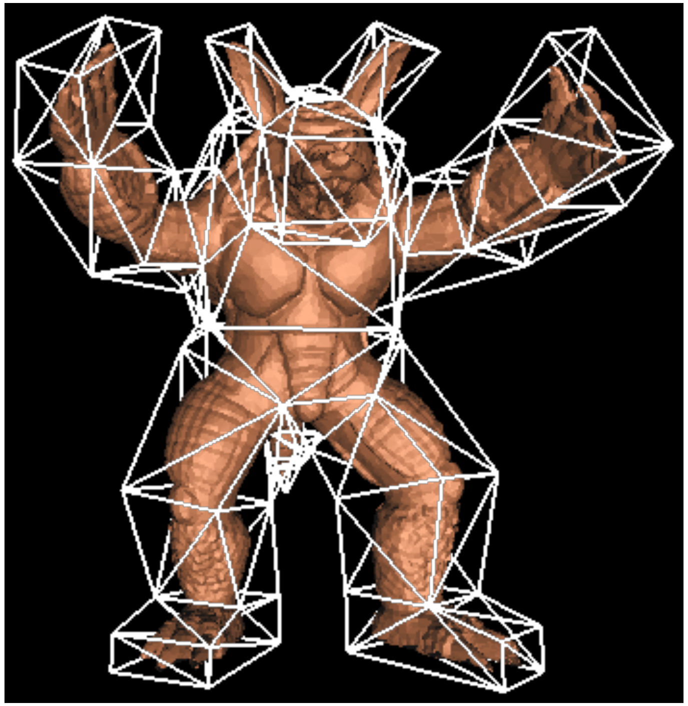
Digital Michelangelo Project

Deformation Handles

- Low-resolution auxiliary shape controls deformation of high-resolution model

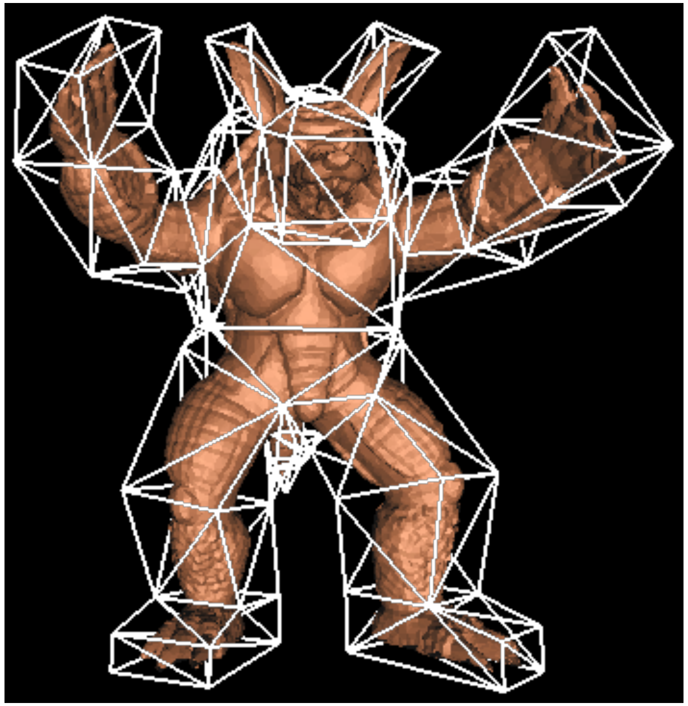
Deformation Handles

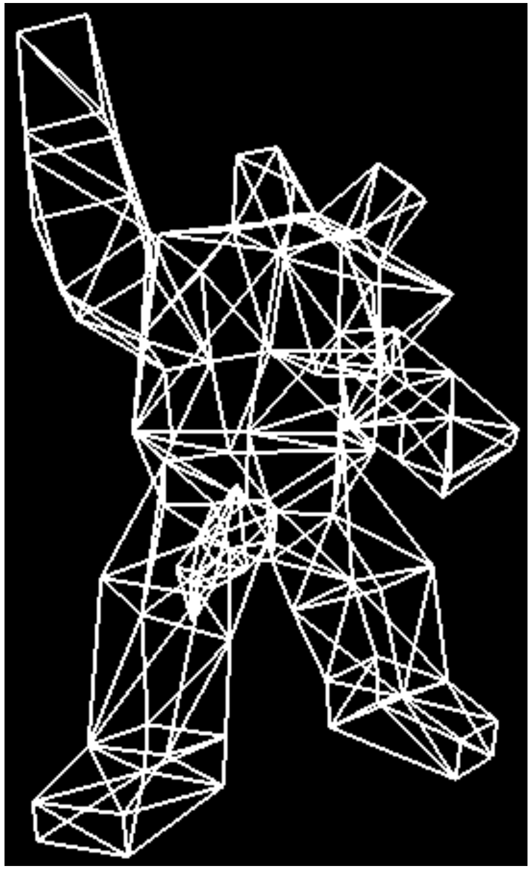
- Low-resolution auxiliary shape controls deformation of high-resolution model



Deformation Handles

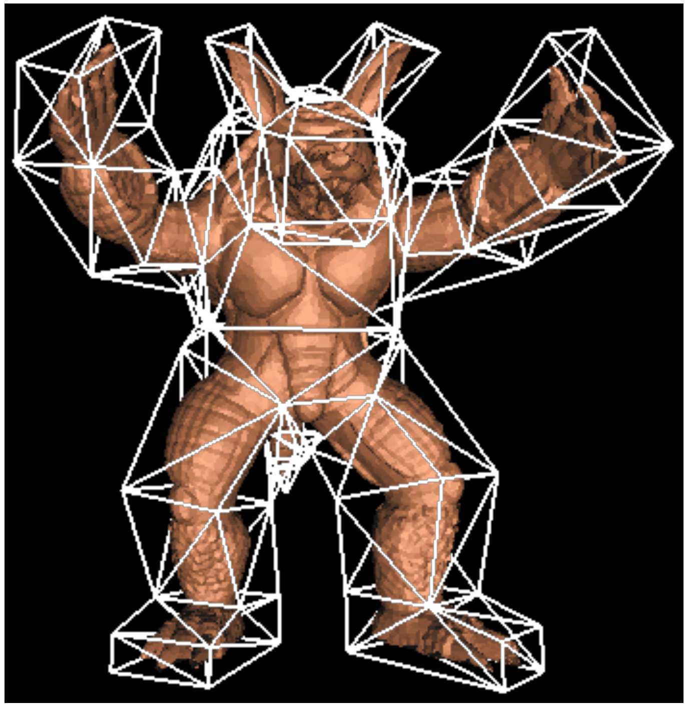
- Low-resolution auxiliary shape controls deformation of high-resolution model

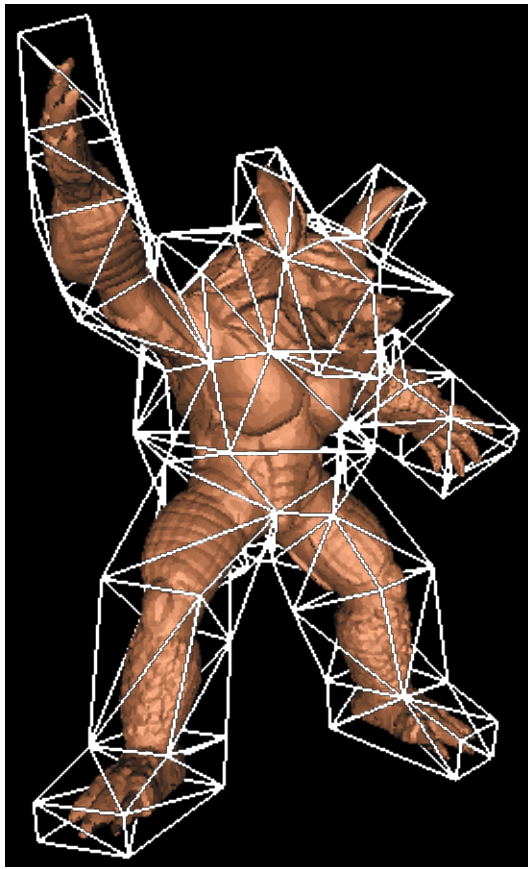




Deformation Handles

- Low-resolution auxiliary shape controls deformation of high-resolution model





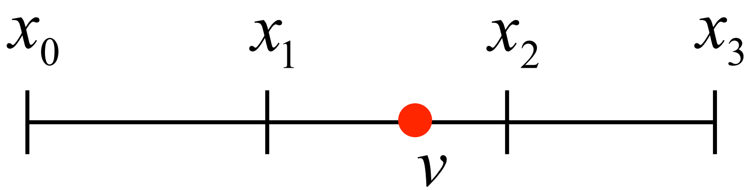
Free-Form Deformations

- Smooth deformations of arbitrary shapes
- Local control of deformation
- Performing deformation is fast
- Widely used
  - Game/Movie industry
  - Part of nearly every 3D modeler

Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices

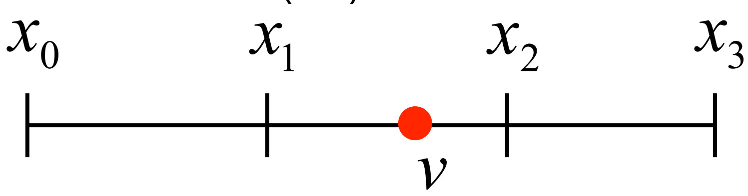
$$v = \sum_i w_i x_i$$



Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices
- Assume x_i are equally spaced and use Bernstein basis functions

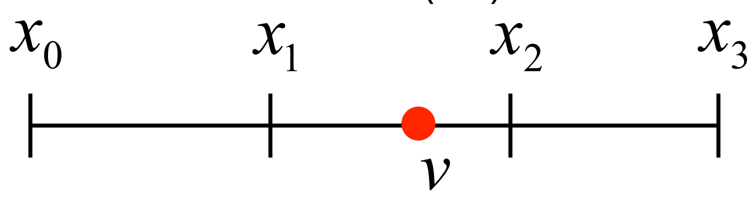
$$w_i = \binom{d}{i}(1-v)^{d-i}v^i$$



Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices
- Assume x_i are equally spaced and use Bernstein basis functions

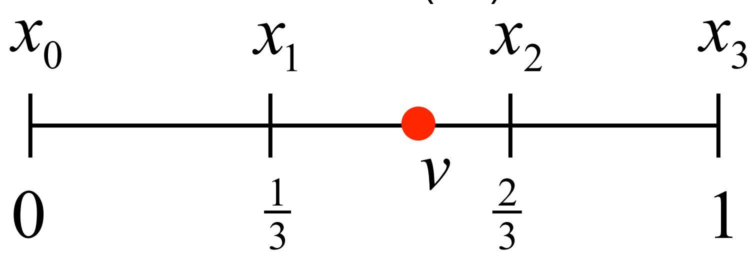
$$v = \sum_i w_i x_i = \sum_i \binom{d}{i} (1-t)^{d-i} t^i x_i$$



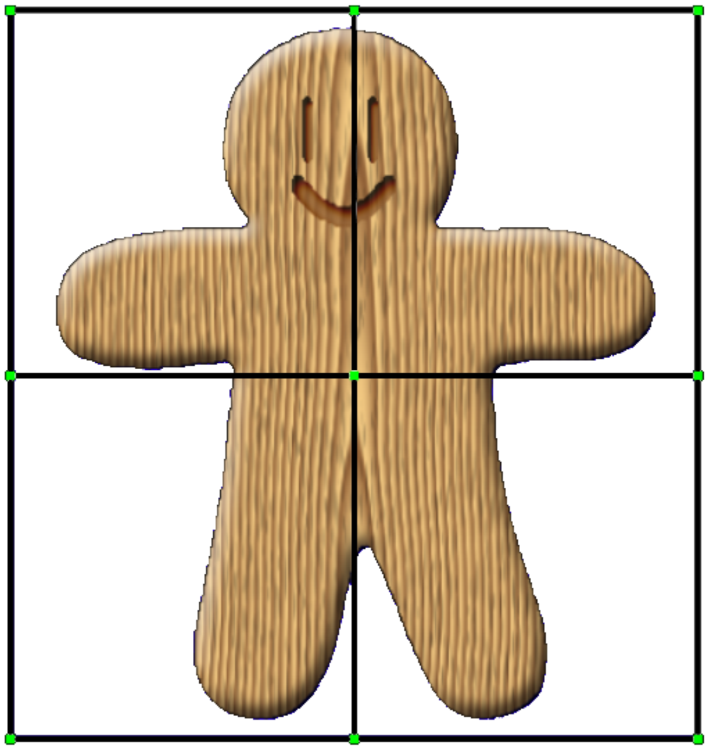
Free-Form Deformations

- Embed object in uniform grid
- Represent each point in space as a weighted combination of grid vertices
- Assume x_i are equally spaced and use Bernstein basis functions

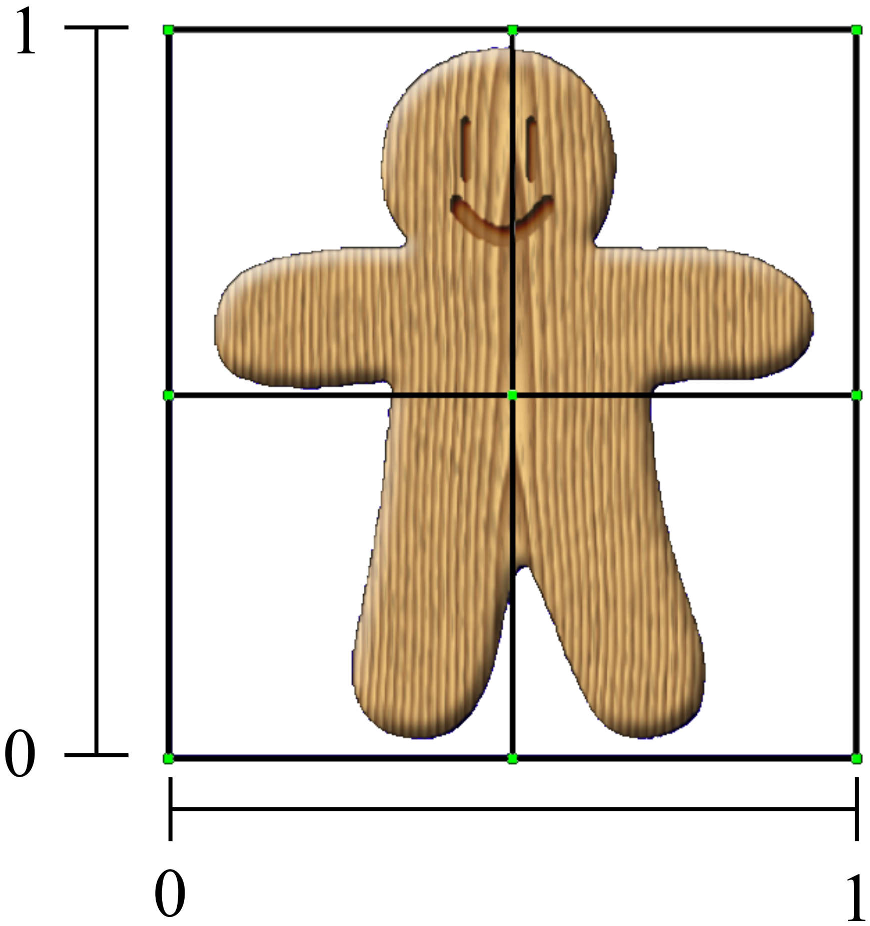
$$v = \sum_i w_i x_i = \sum_i \binom{d}{i} (1-t)^{d-i} t^i x_i$$



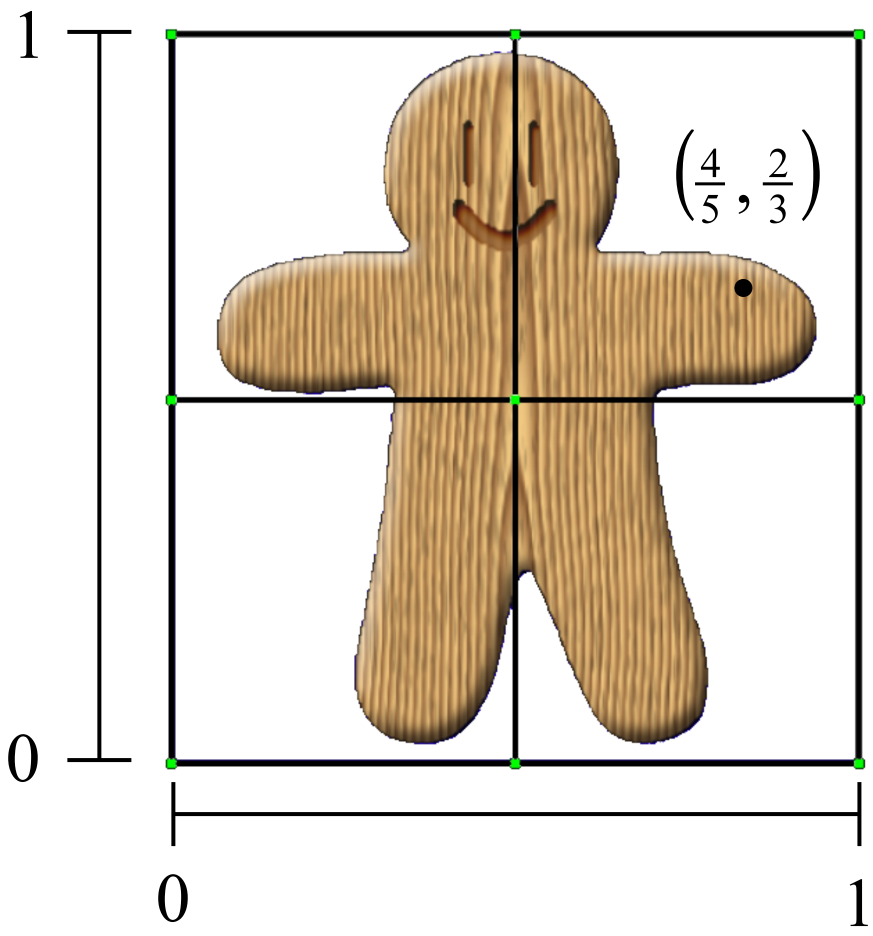
2D Example



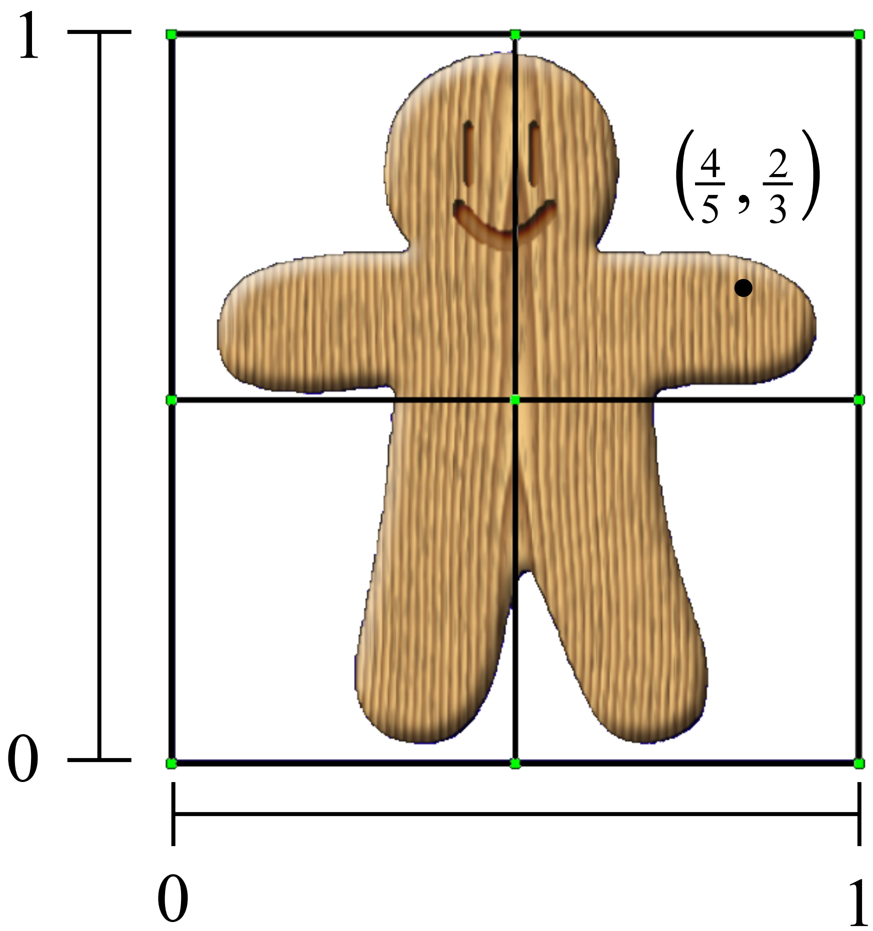
2D Example



2D Example



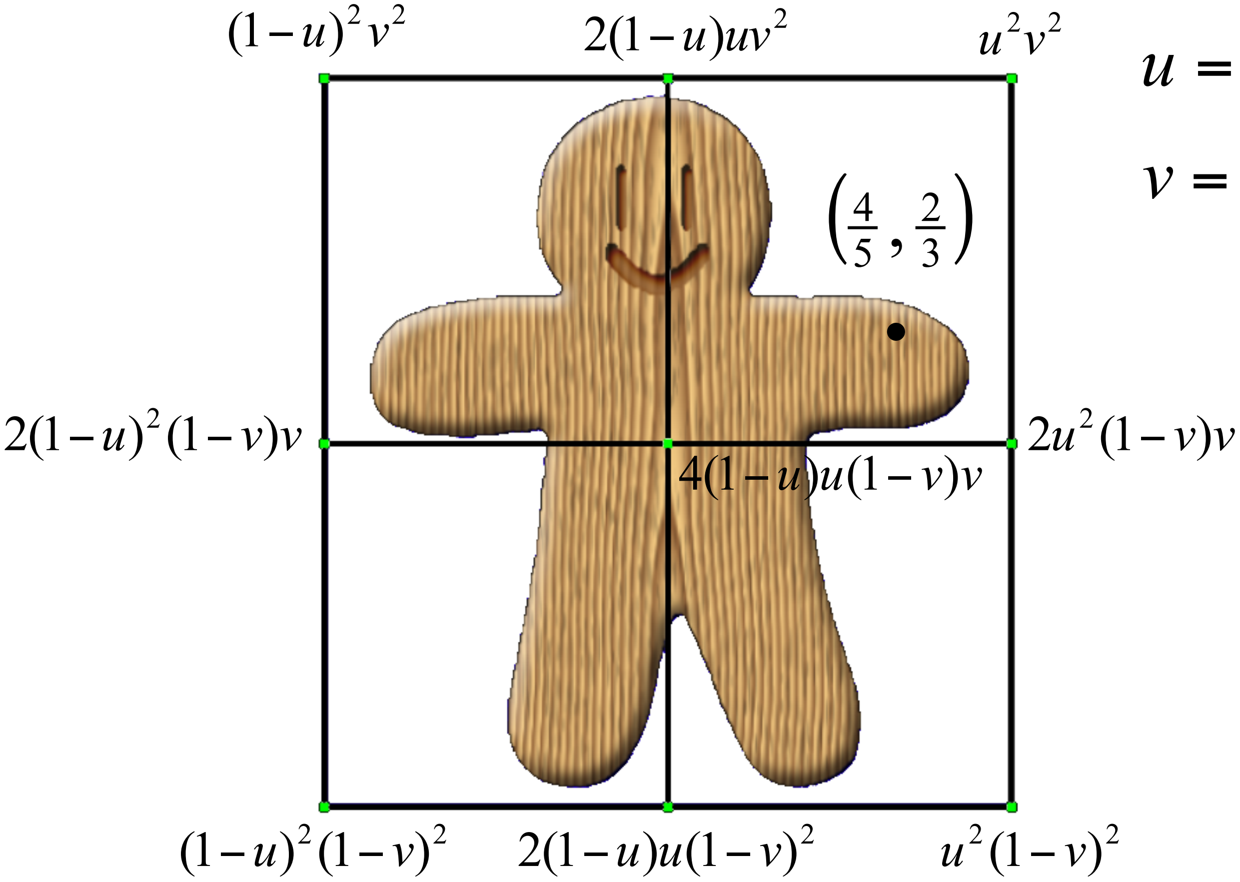
2D Example



$u = \frac{4}{5}$

$v = \frac{2}{3}$

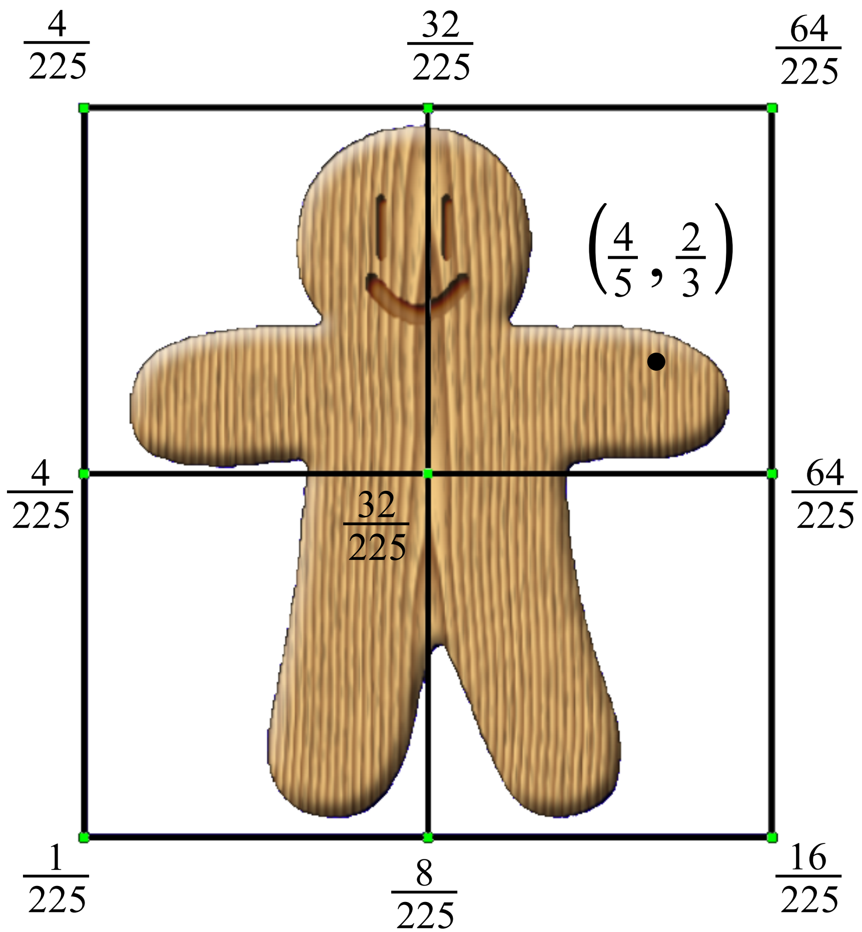
2D Example



$$u = \tfrac{4}{5}$$

$$v = \tfrac{2}{3}$$

2D Example

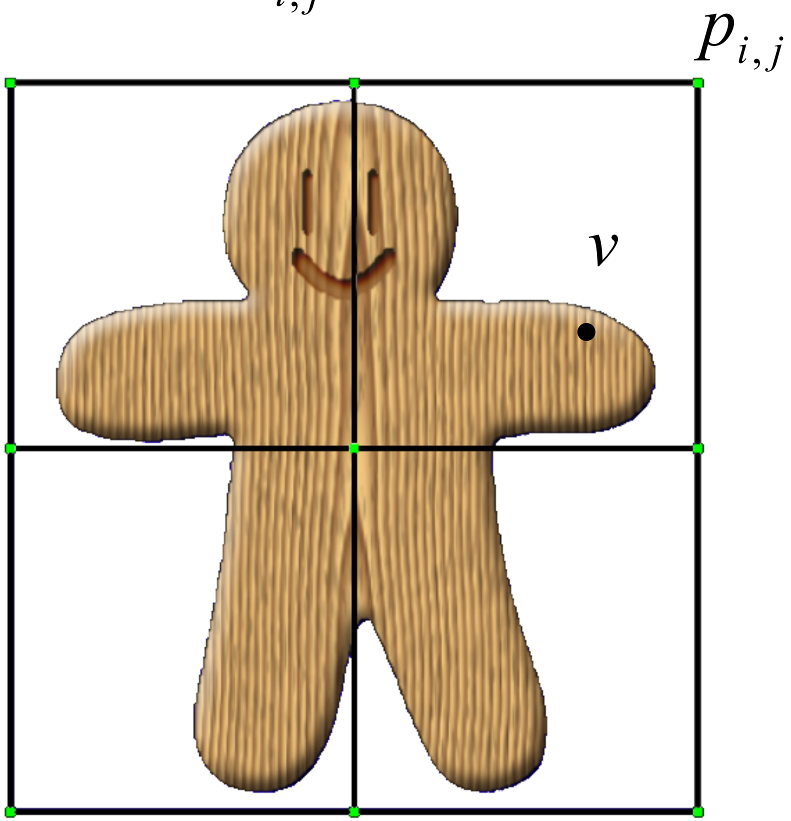


$u = \frac{4}{5}$

$v = \frac{2}{3}$

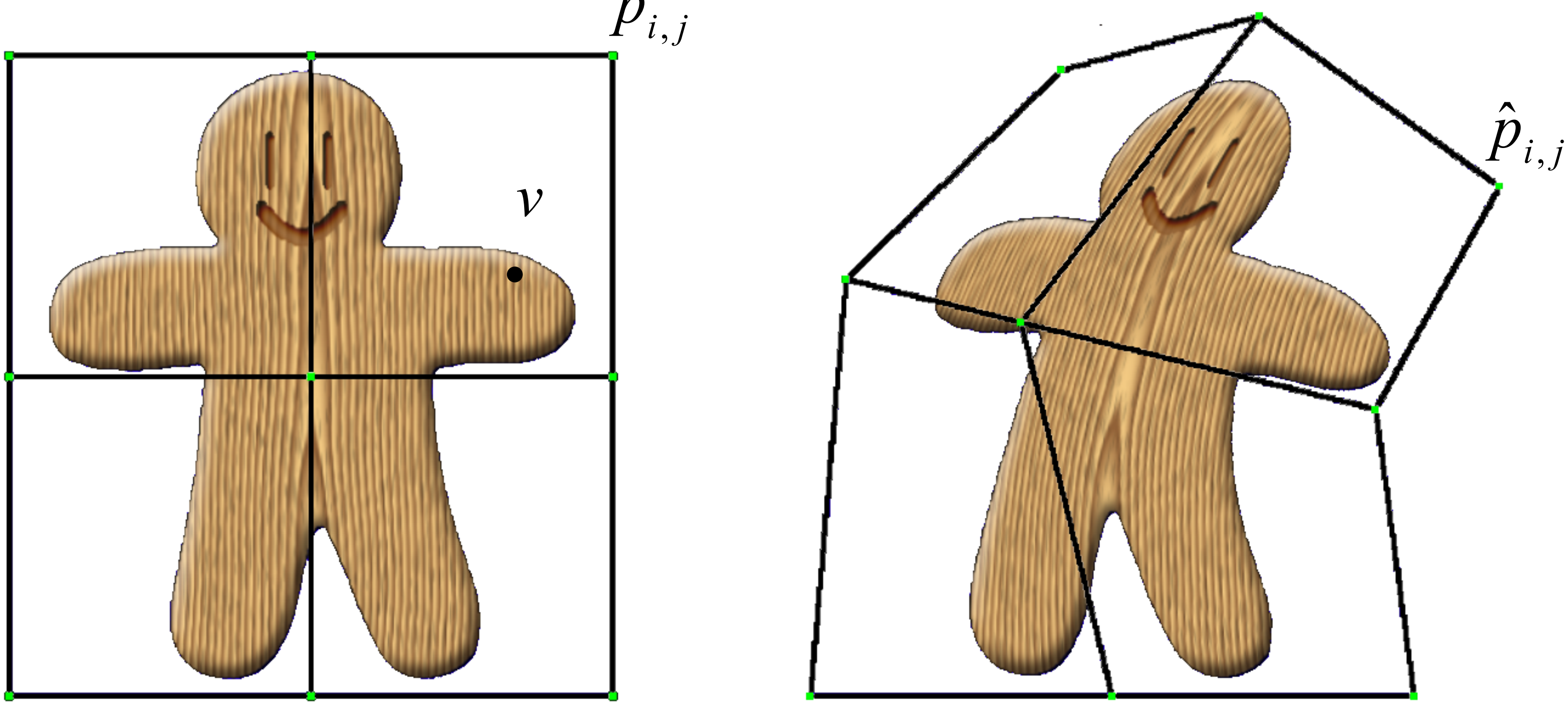
Applying the Deformation

$$v = \sum_{i,j} w_{i,j} p_{i,j}$$



Applying the Deformation

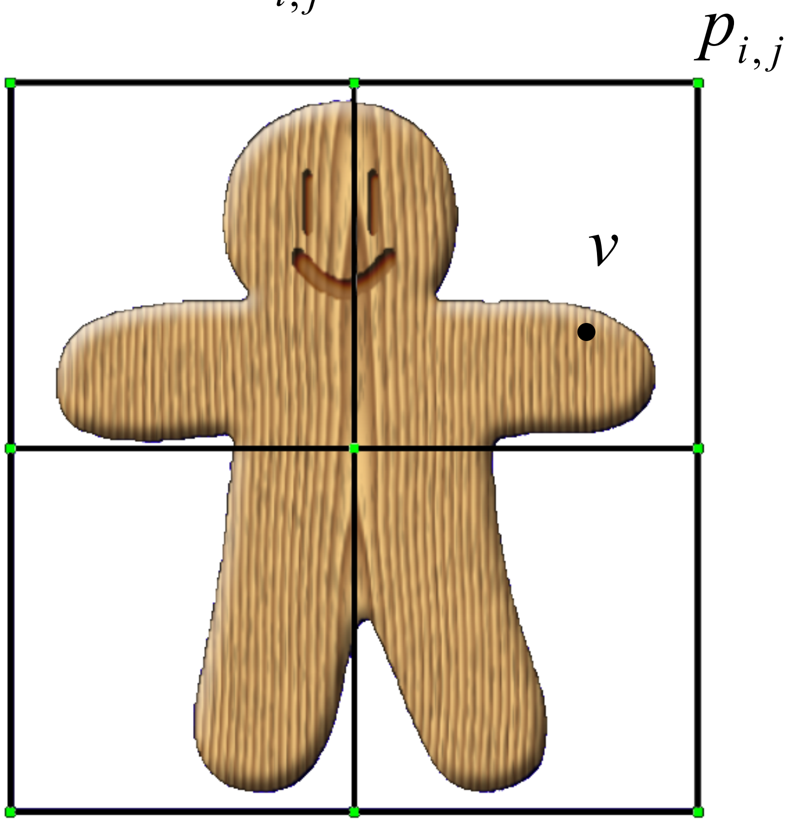
$$v = \sum_{i,j} w_{i,j} p_{i,j}$$

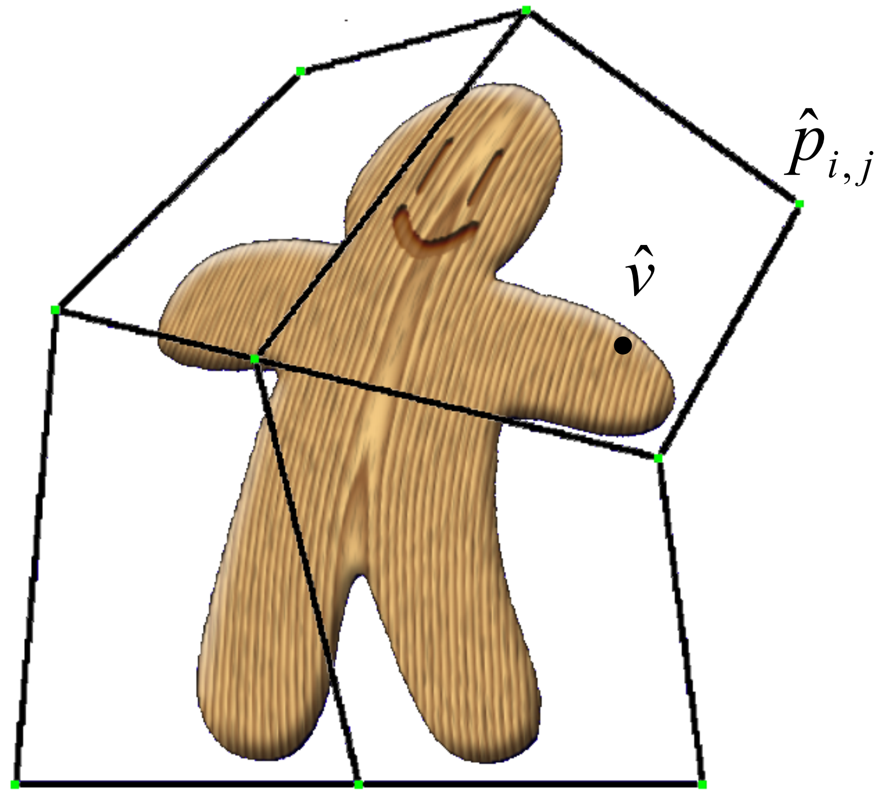


Applying the Deformation

$$v = \sum_{i,j} w_{i,j} p_{i,j}$$

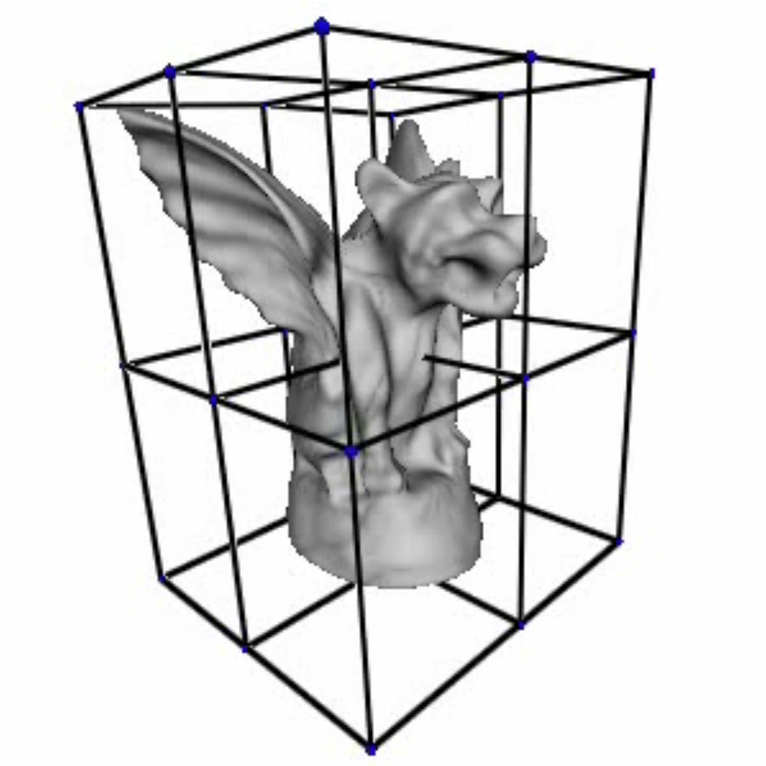
$$\hat{v} = \sum_{i,j} w_{i,j} \hat{p}_{i,j}$$





Advantages

- Smooth Deformation of arbitrary shapes
- Local control of deformations
- Computing the deformation is easy
- Deformations are very fast

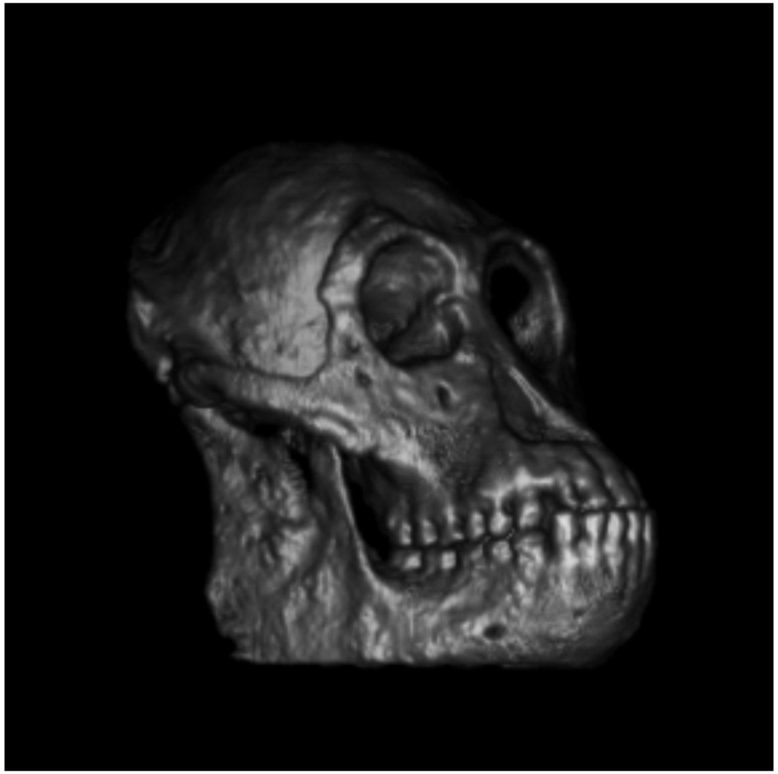


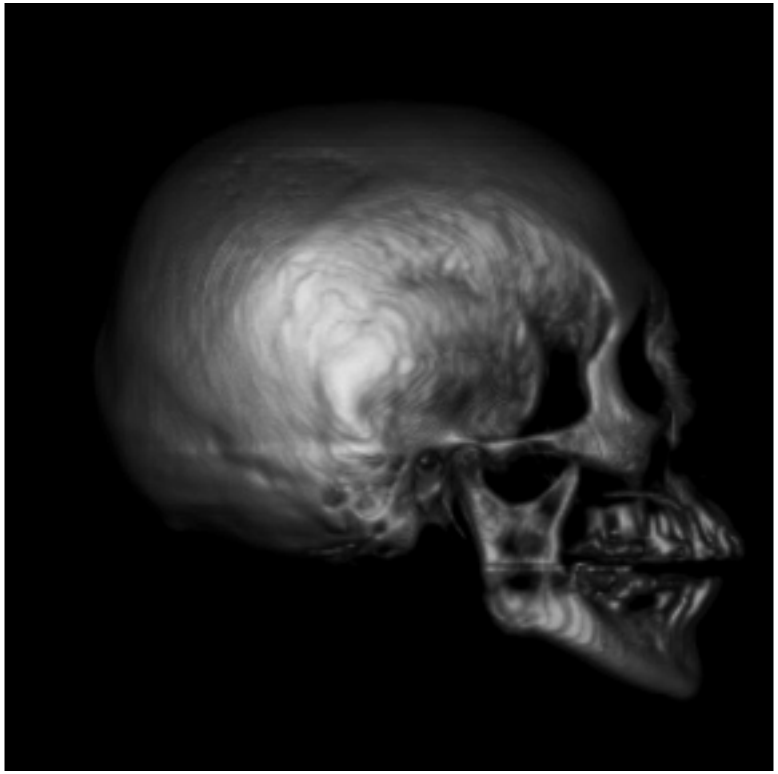
Disadvantages

- Must use cubical cells for deformation
- Restricted to uniform grid
- Deformation warps space… not surface
  - Does not take into account geometry/topology of surface
- May need many FFD's to achieve a simple deformation

Summary

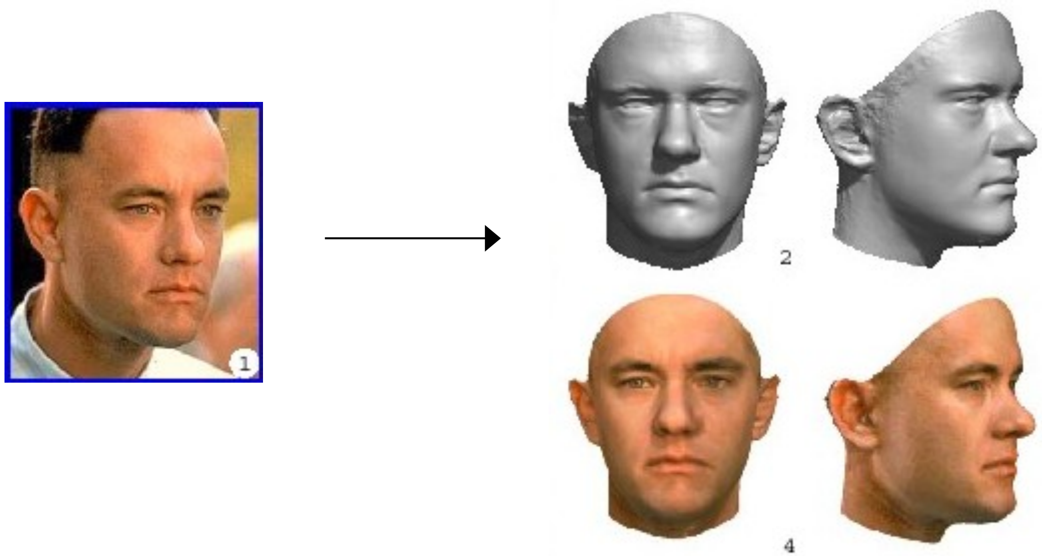
- Widely used deformation technique
- Fast, easy to compute
- Some control over volume preservation/smoothness
- Uniform grids are restrictive



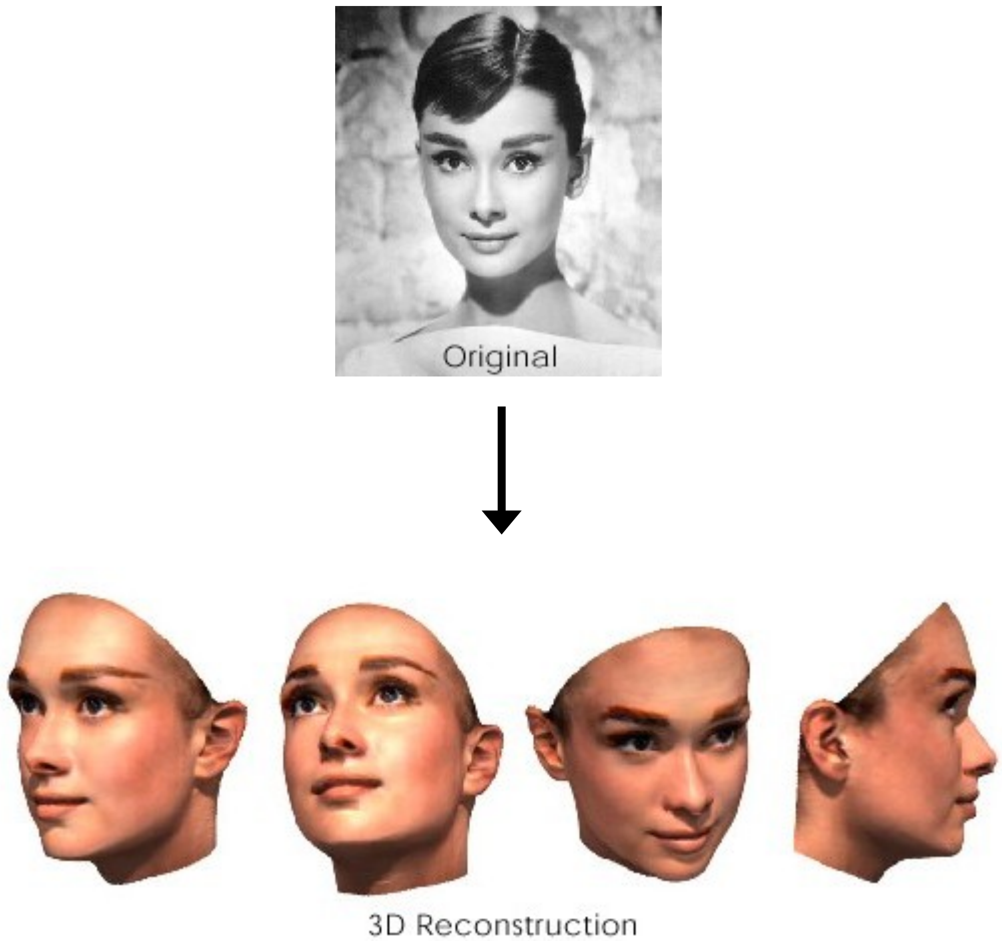


Morphable 3D Face Model

- SIGGRAPH 1999 by V. Blanz and T. Vetter
- Idea:
  - Learn a statistical shape and appearance models in 3D
  - Fit a morphable 3D face model to new 2D images

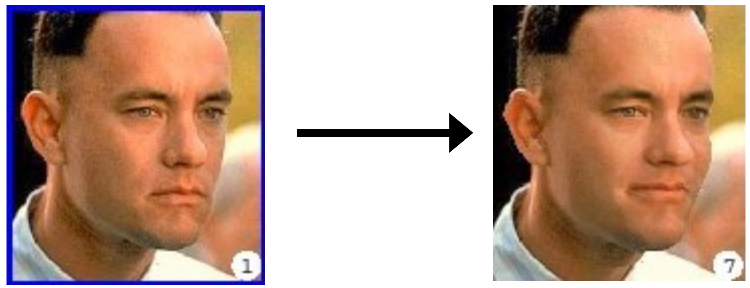


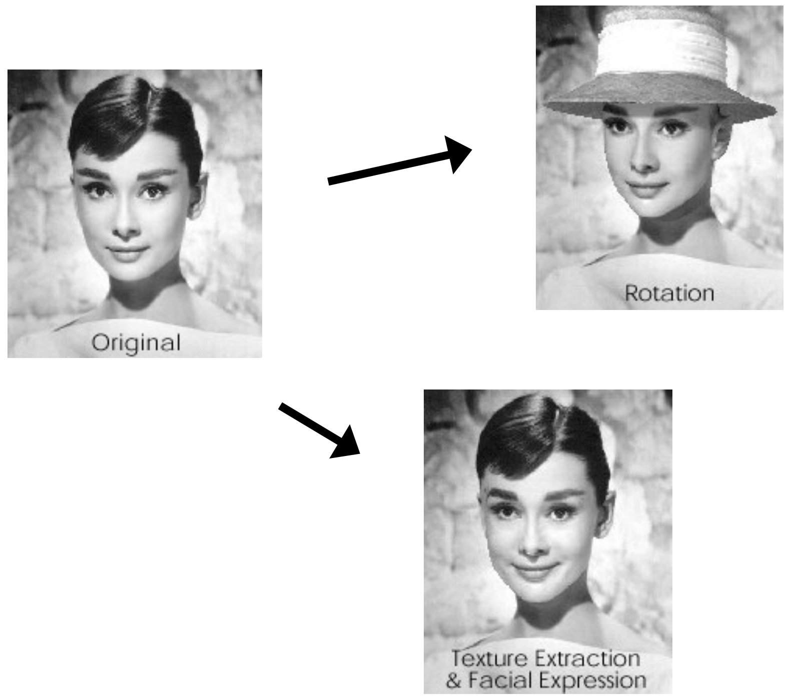
Morphable 3D Face Model



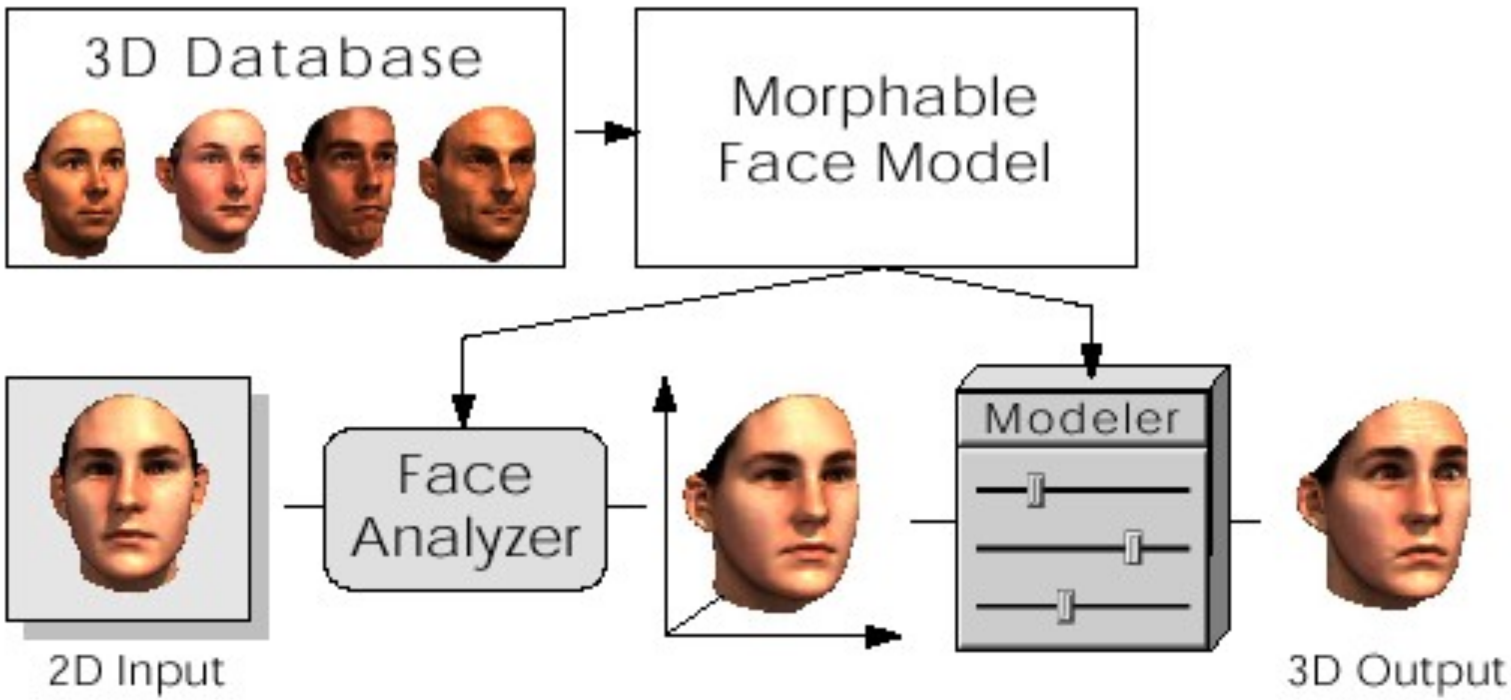
Morphable 3D Face Model

- Allows
  - facial expression manipulations
  - changes in pose
  - generation of new shadows and lighting conditions





Morphable 3D Face Model: Approach



Morphable 3D Face Model: Method

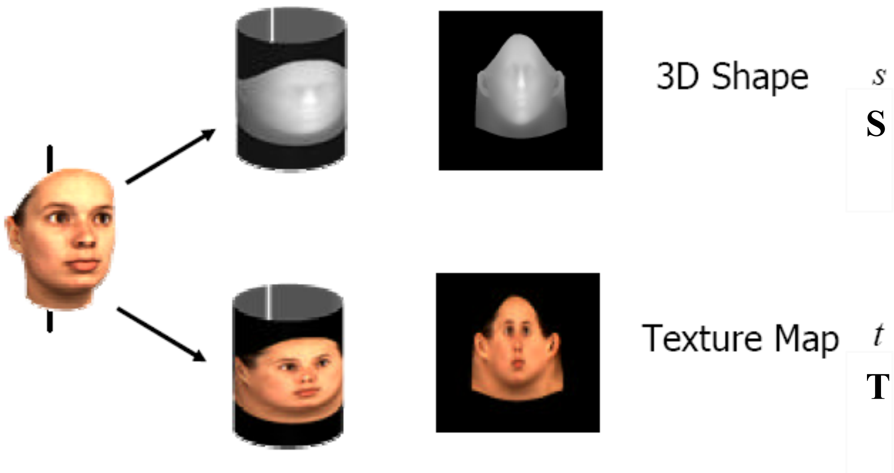
- The actual 3D structure of known faces is captured in the shape vector

$$\mathbf{S} = (x_1, y_1, z_1, x_2, \ldots, y_n, z_n)^T$$

containing the *(x, y, z)* coordinates of the n vertices of a face, and the texture vector

$$\mathbf{T} = (R_1, G_1, B_1, R_2, \ldots, G_n, B_n)^T$$

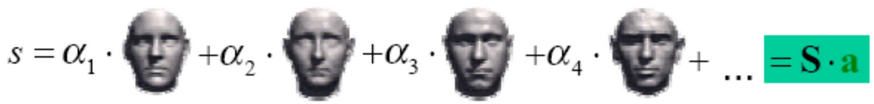
containing the color values at the corresponding vertices.

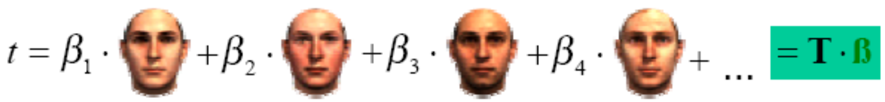


Morphable 3D Face Model: Method

- Assuming that we have m such vector pairs in full correspondence, we can form new shapes $\mathbf{S}_{\text{model}}$ and new textures $\mathbf{T}_{model}$ as:

$$\mathbf{S}_{model} = \sum_{i=1}^{m} \alpha_i \mathbf{S}_i \qquad \mathbf{T}_{model} = \sum_{i=1}^{m} \beta_i \mathbf{T}_i$$





Morphable 3D Face Model: Method

- In order to constrain the solution to lie close to our data cloud, we fit a normal distribution to a set of 200 sample faces, using PCA:
    1. Compute average shape and texture
    2. Compute covariance matrices $\mathbf{C_S}$ and $\mathbf{C_T}$ over the shape $\Delta\mathbf{S}_i = \overline{\mathbf{S}} - \mathbf{S}_i$ and texture differences $\Delta\mathbf{T}_i = \overline{\mathbf{T}} + \mathbf{T}_i$
    3. Compute eigenvectors of covariance matrices $\mathbf{C_S}$ and $\mathbf{C_T}$

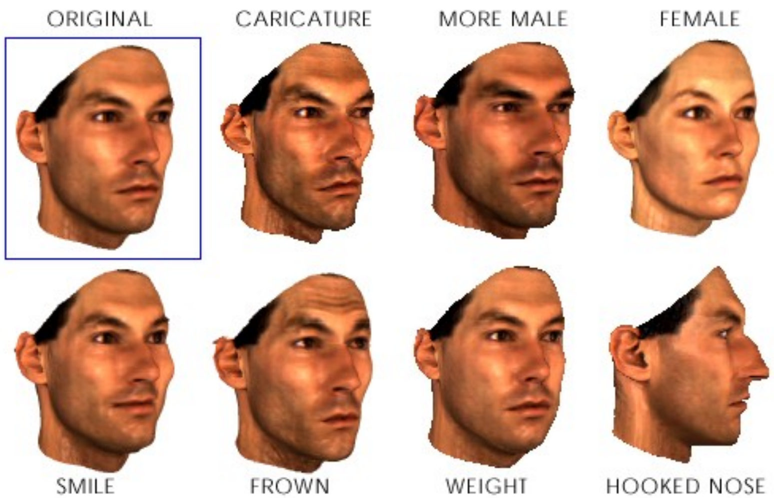
$$\mathbf{S}_{model} = \overline{\mathbf{S}} + \sum_{i=1}^{m-1} \alpha_i \mathbf{s}_i \qquad \mathbf{T}_{model} = \overline{\mathbf{T}} + \sum_{i=1}^{m-1} \beta_i \mathbf{t}_i$$

Morphable 3D Face Model: Facial attributes

- Given a set of faces $(\mathrm{S}_i, \mathrm{T}_i)$ with manually assigned labels μ_i compute

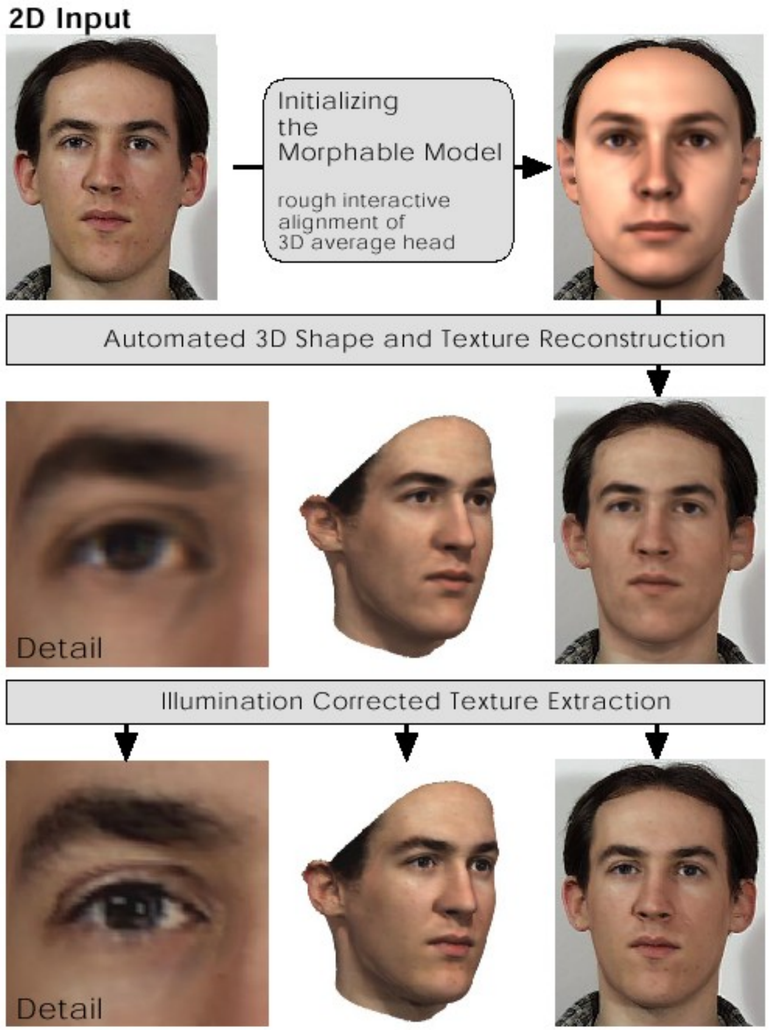
$$\Delta\mathbf{S} = \sum_i \mu_i\left(\overline{\mathbf{S}} - \mathbf{S}_i\right) \qquad \Delta\mathbf{T} = \sum_i \mu_i\left(\overline{\mathbf{T}} - \mathbf{T}_i\right)$$

- By adding multiples of $\Delta\mathbf{S}$ and $\Delta\mathbf{T}$ to a face one can emphasize facial features
- Labels can correspond to
  - sex
  - body weight
  - facial expression



Fitting the Morphable Model to an Image

- Rough initial manual alignment
- Reconstruction of 3D shape, texture and rendering parameters by fitting the model to image
- Extracting texture from image



Fitting the Model to an Image

- Coefficients of the 3D model

$$(\alpha_1, \alpha_2, \dots , \alpha_m)^T \quad \text{and} \quad (\beta_1, \beta_2, \dots , \beta_m)^T$$

are optimized together with the rendering parameters $\boldsymbol{\rho}$ such as camera position, object scale, image plane rotation and translation, intensity of ambient and directed light, etc.

Fitting the Model to an Image

- At every iteration the algorithm renders an image $\mathbf{I}_{model}$ using the current parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\rho}$ and updates them so as to minimize the residual norm summed over the pixels (x, y)

$$E_I = \sum_{x,y} \| \mathbf{I}_{input}(x, y) - \mathbf{I}_{model}(x, y) \|^2$$

where $\mathbf{I}_{input}$ is the input image.

Morphable 3D Face Model: Results

- After rough manual initialization, a gradient descent technique minimizes a functional that gives preference to reconstructed faces that are closer to the average face in the database.

Morphable 3D Face Model: Results

- After reconstruction, additional texture is extracted from the input image using the obtained shape, texture, and rendering parameters by looking at the residual at each pixel.
- New images are rendered modeling artificial lighting conditions, rotations, and facial attributes