Facial modeling

Hyewon SEO
ICube-CNRS

Facial modeling

Kinematic methods

Performance-based methods

Example based methods

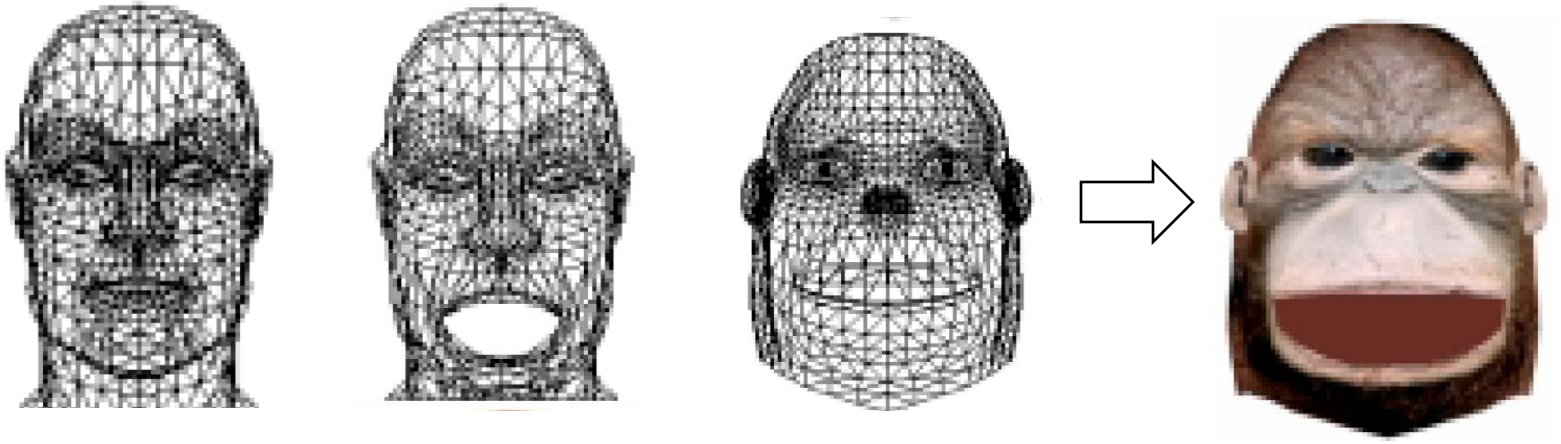
Data-driven methods

Example-based facial pose transfer

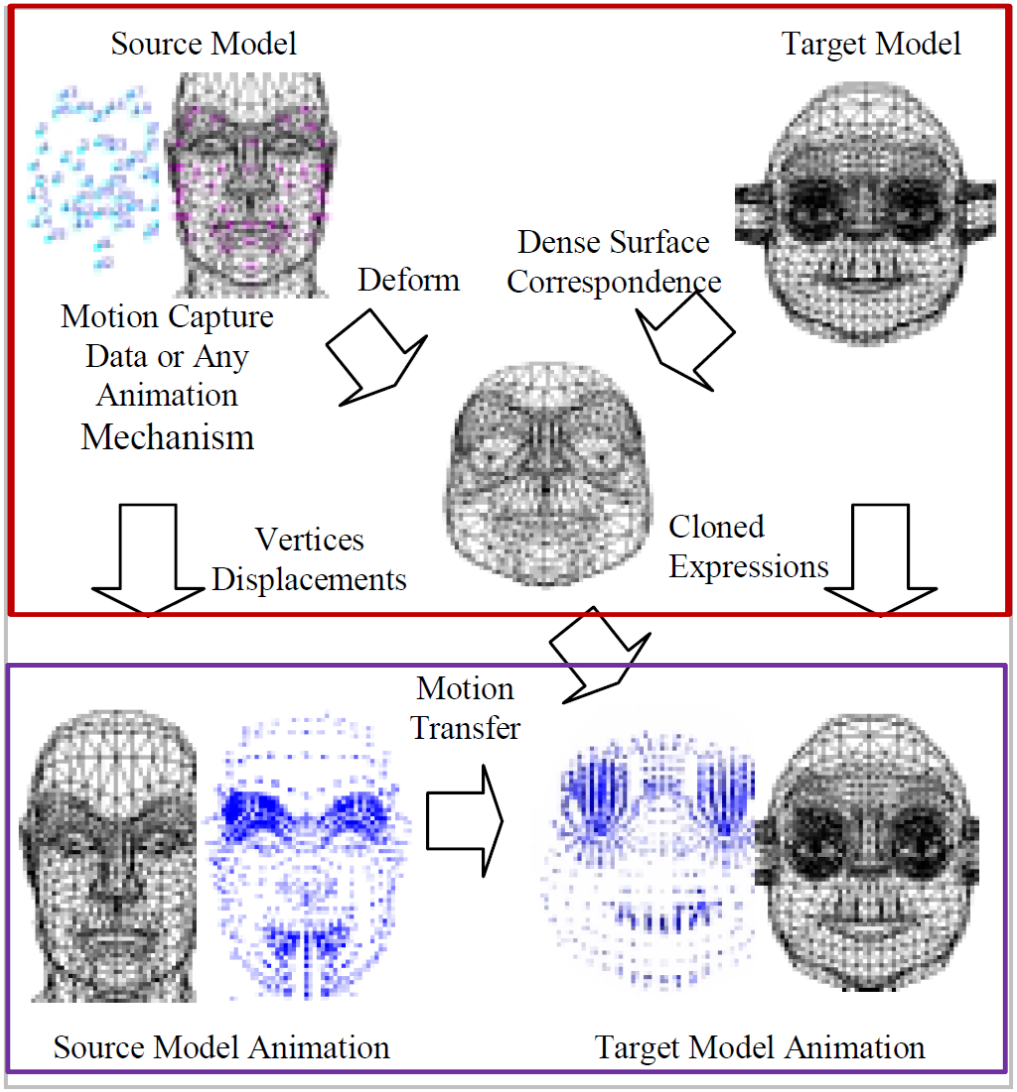
- What is it?
  - Take the geometrical element contributing to a specific pose from an existing model to a new one
  - Retargetting, cloning,...
- Why?
  - Easily create/reuse facial animations for character models

Example-based facial pose transfer

- How?
  - Determine the point-wise correspondence on the meshes
  - Transfer vertex displacement vectors from a source model to a target



Example-based facial pose transfer

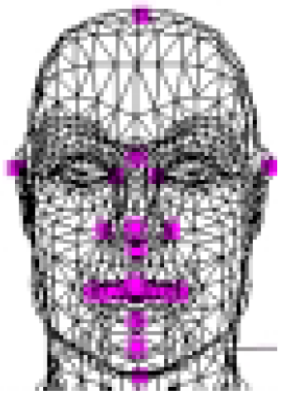


step1

step2

Dense surface correspondence

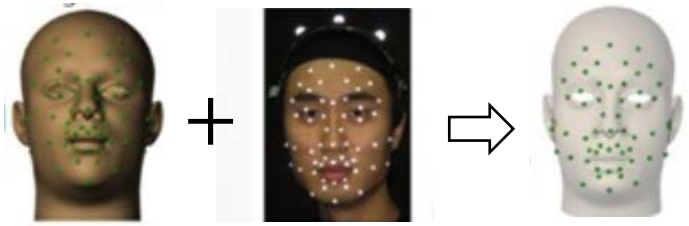
- Determine which surface points in the source correspond to vertices in the target model
  - The models come in different number of vertices or connectivity
  - Small set of initial correspondences to establish an approximate relationship



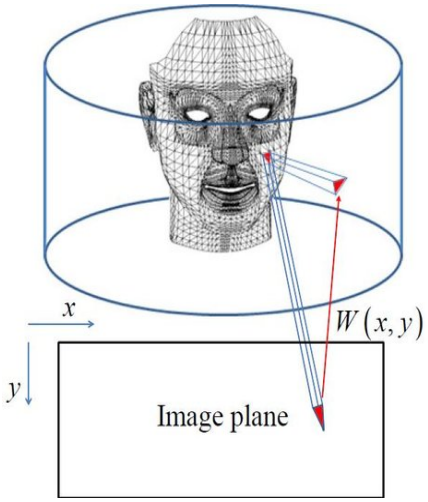
23 automatically found feature points including 9 lip contact points

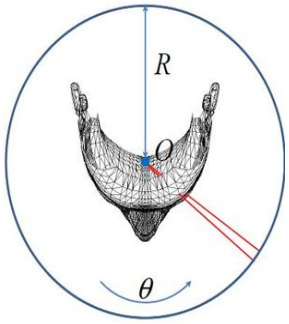
Dense surface correspondence

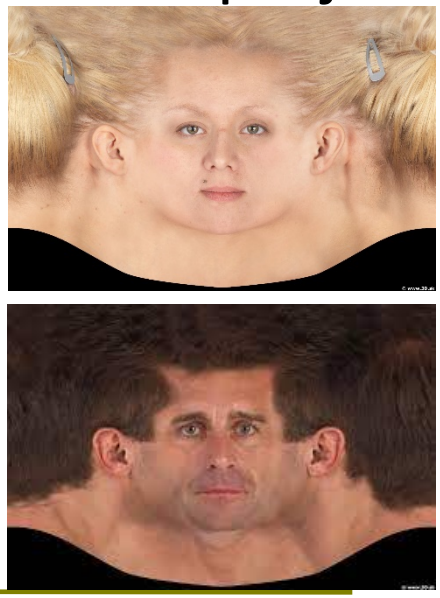
- Radial Basis Functions
  - Roughly project vertices in the source model onto the target model



- Dense mapping after the cylindrical projection

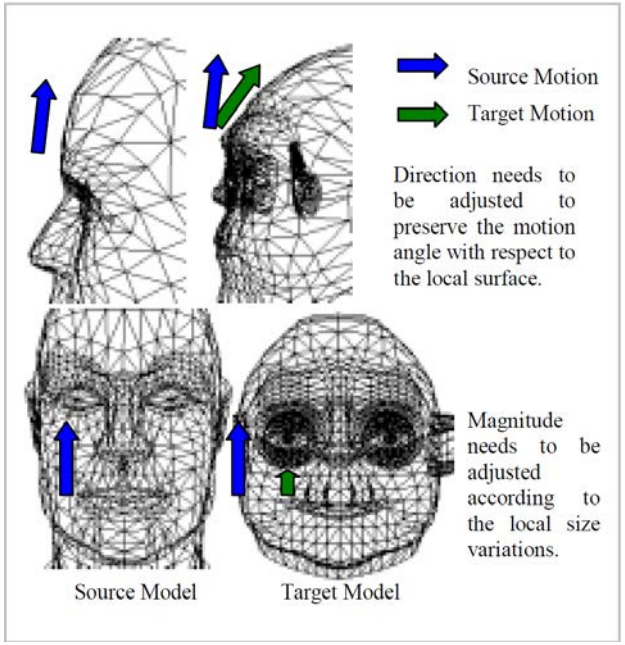






Displacement vector transfer

- Displace each source vertex to match the displacement of a corresponding target surface point
  - Direction and magnitude of a displacement vector must be altered and scaled



Example-based facial pose transfer: conclusion

- Produce different models with similar poses

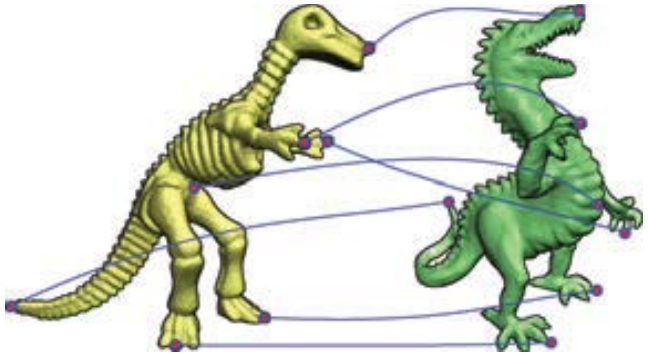


- Fast and easy to implement
- Expressions differ across individuals. Still:
  - *Simple transfer of 3D vertex displacements causes no obvious artifacts.*
  - *More sophisticated methods may improve results.*

Shape correspondence (matching)

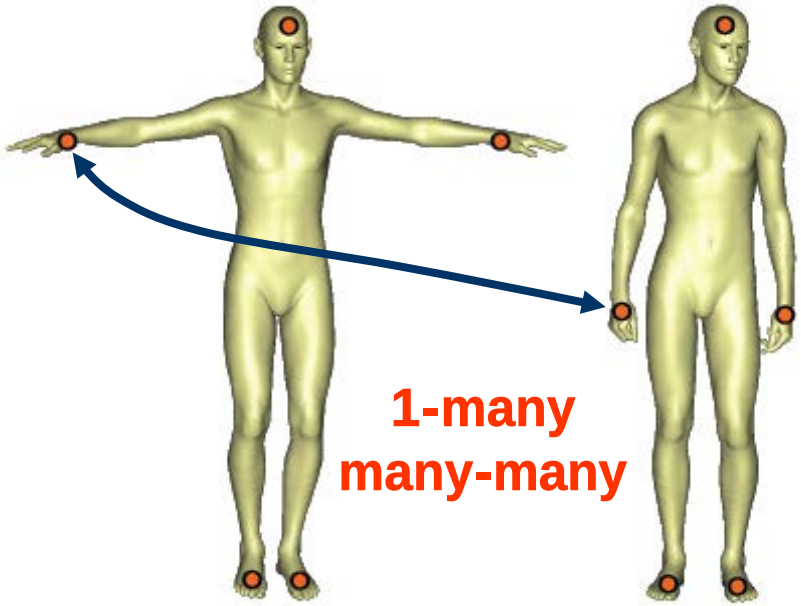
Correspondence vs. registration

- Correspondence: to **find a relation** between sets in which *each member* of one set is associated with *one or more members* of the other
- Registration: to bring several (partial) shapes to (almost) exact alignment, rigidly or non-rigidly



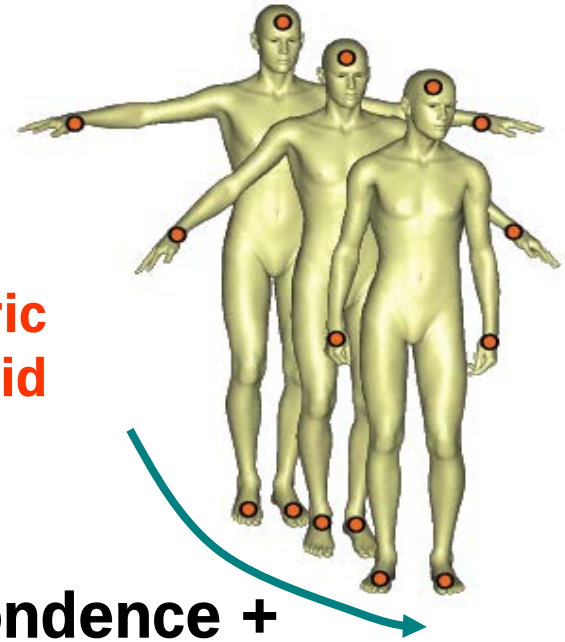
Correspondence vs. registration

Correspondence only



1-many
many-many

rigid
isometric
non-rigid

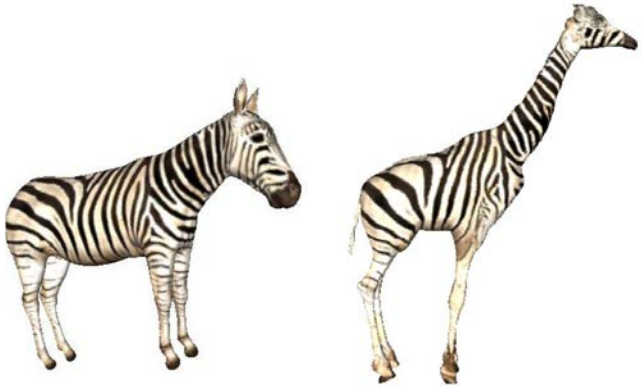


Correspondence +
aligning transformation

Applications

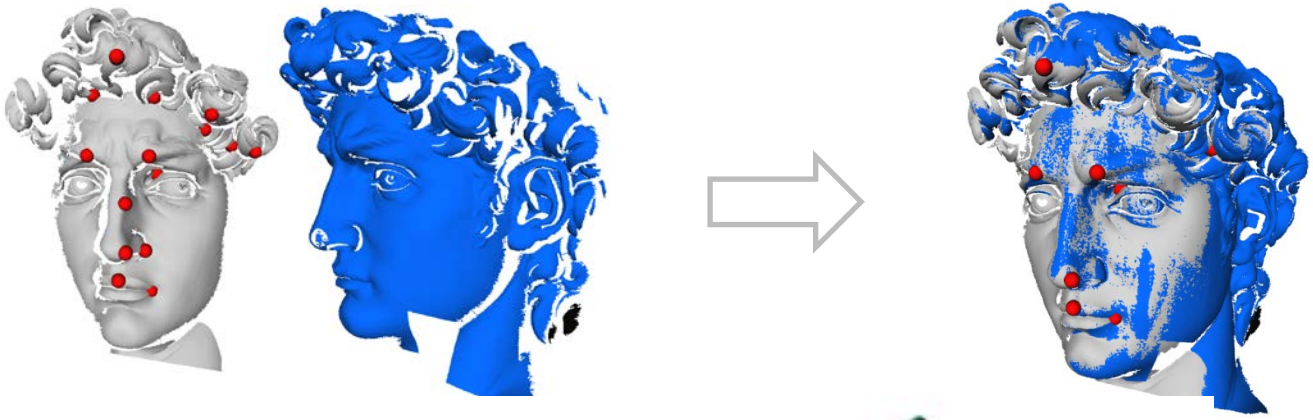
- Fundamental problem in CG, IP, CV

- **Key element** in higher level algorithms
  - Recognition, retrieval, **compact representation**
  - Attribute/motion transfer
  - **Shape interpolation**
  - Space-time reconstruction
  - **Statistical modeling**



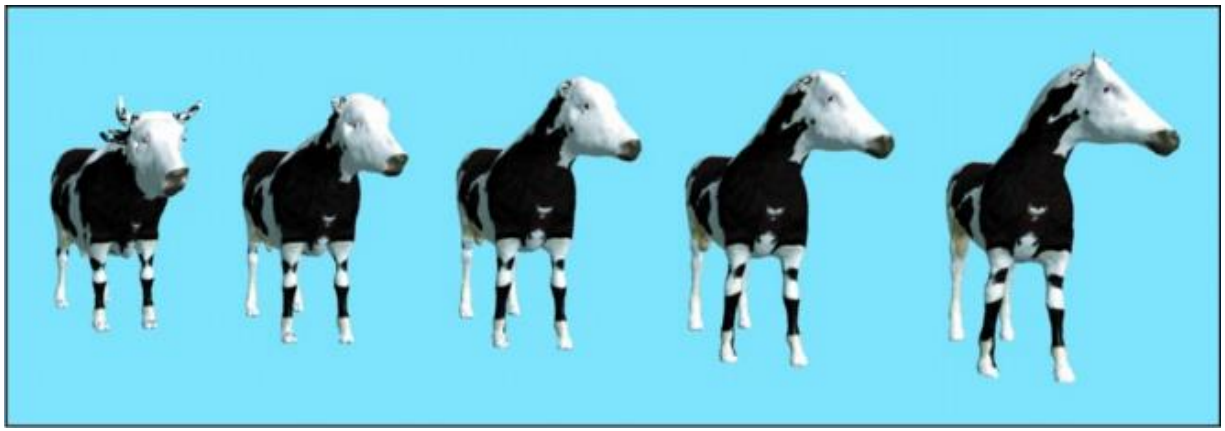
Applications

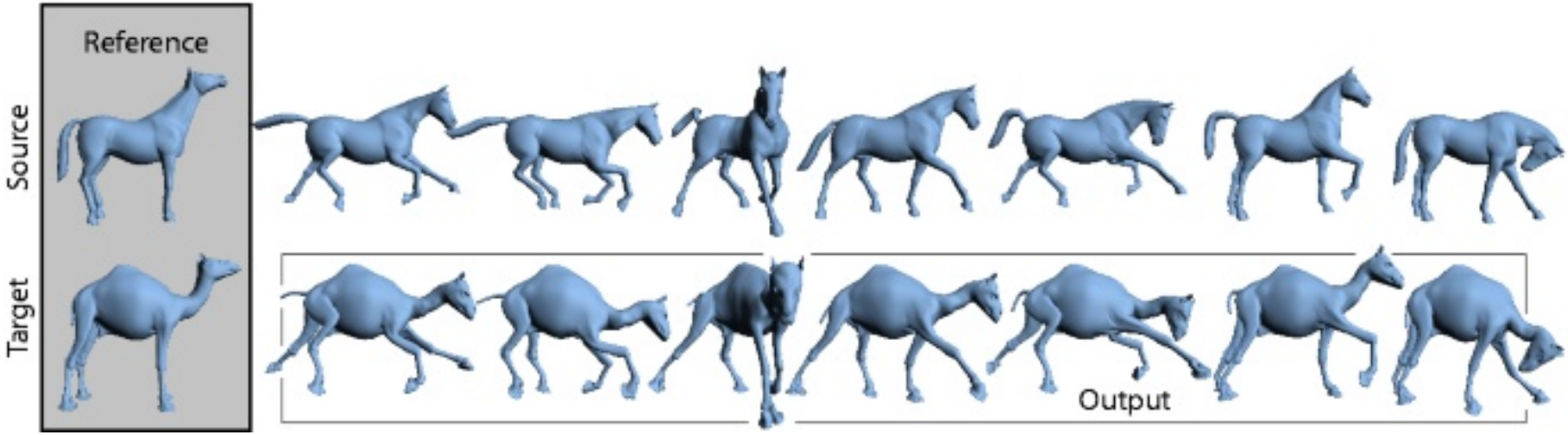
- 3D scan alignment for the reconstruction



Applications

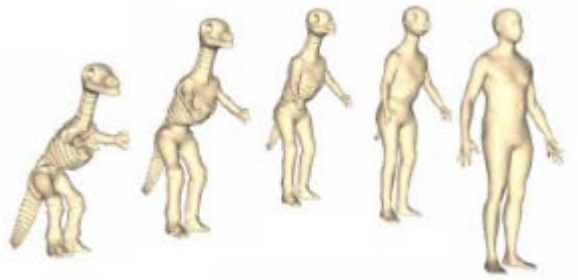
- attribute transfer



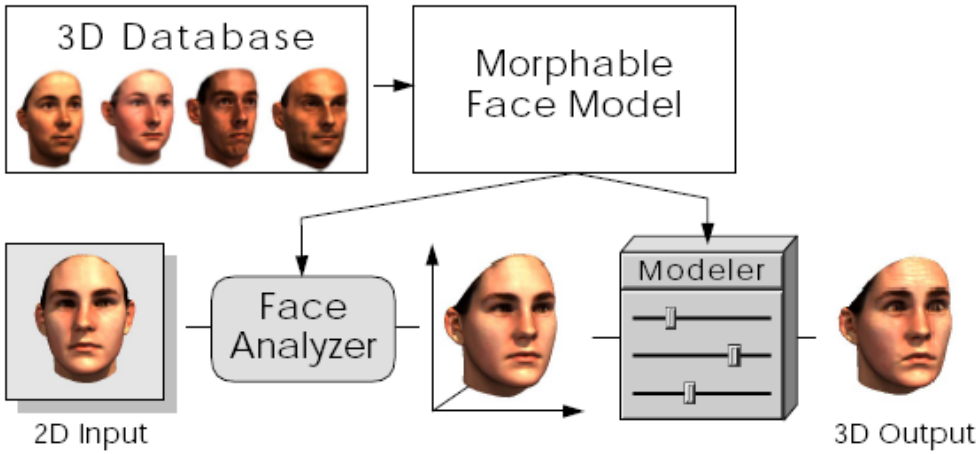


Applications

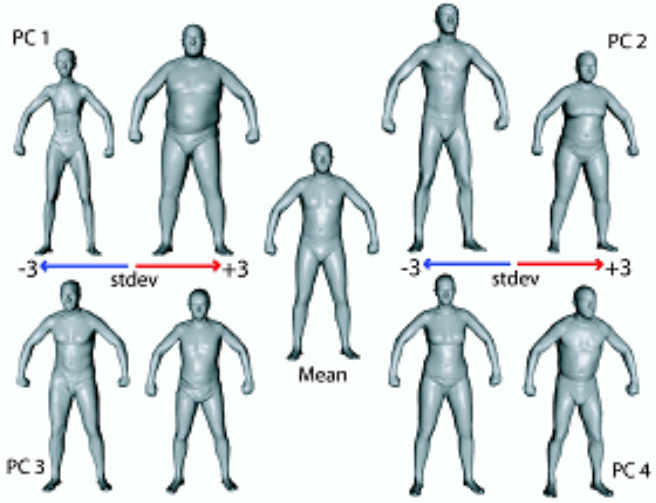
- Synthesis



Morphing [KS04]



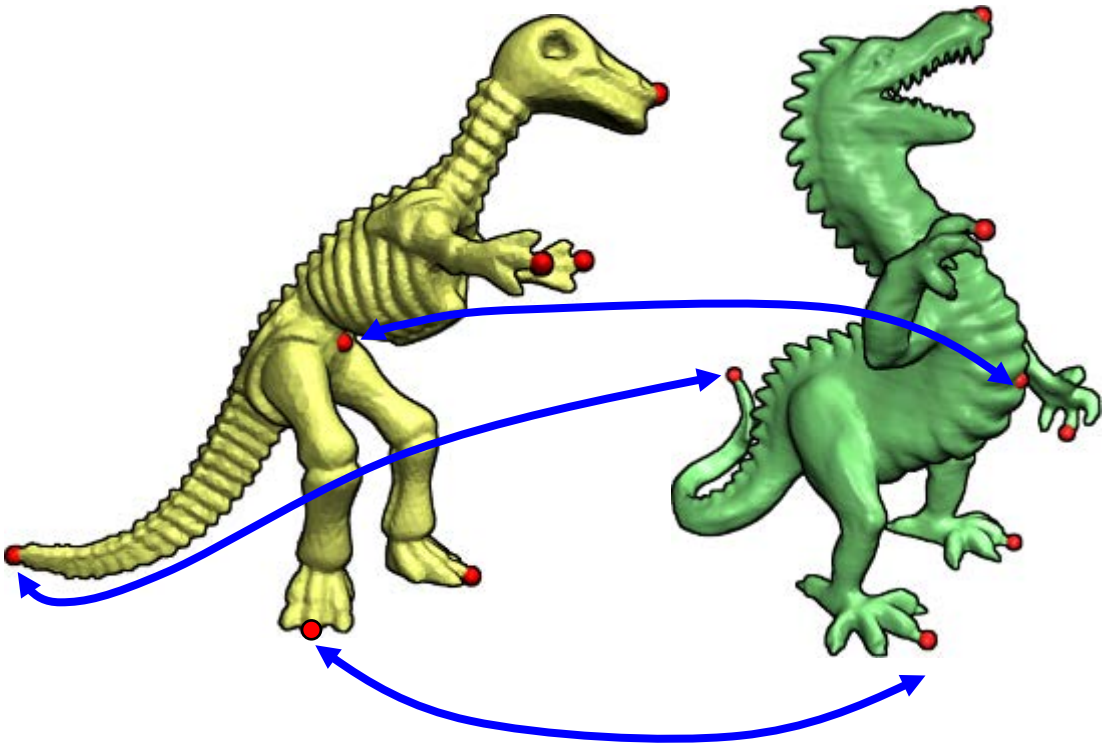
Morphable faces [BV99]



Human modeling [ASK*05]

Why is correspondence hard?

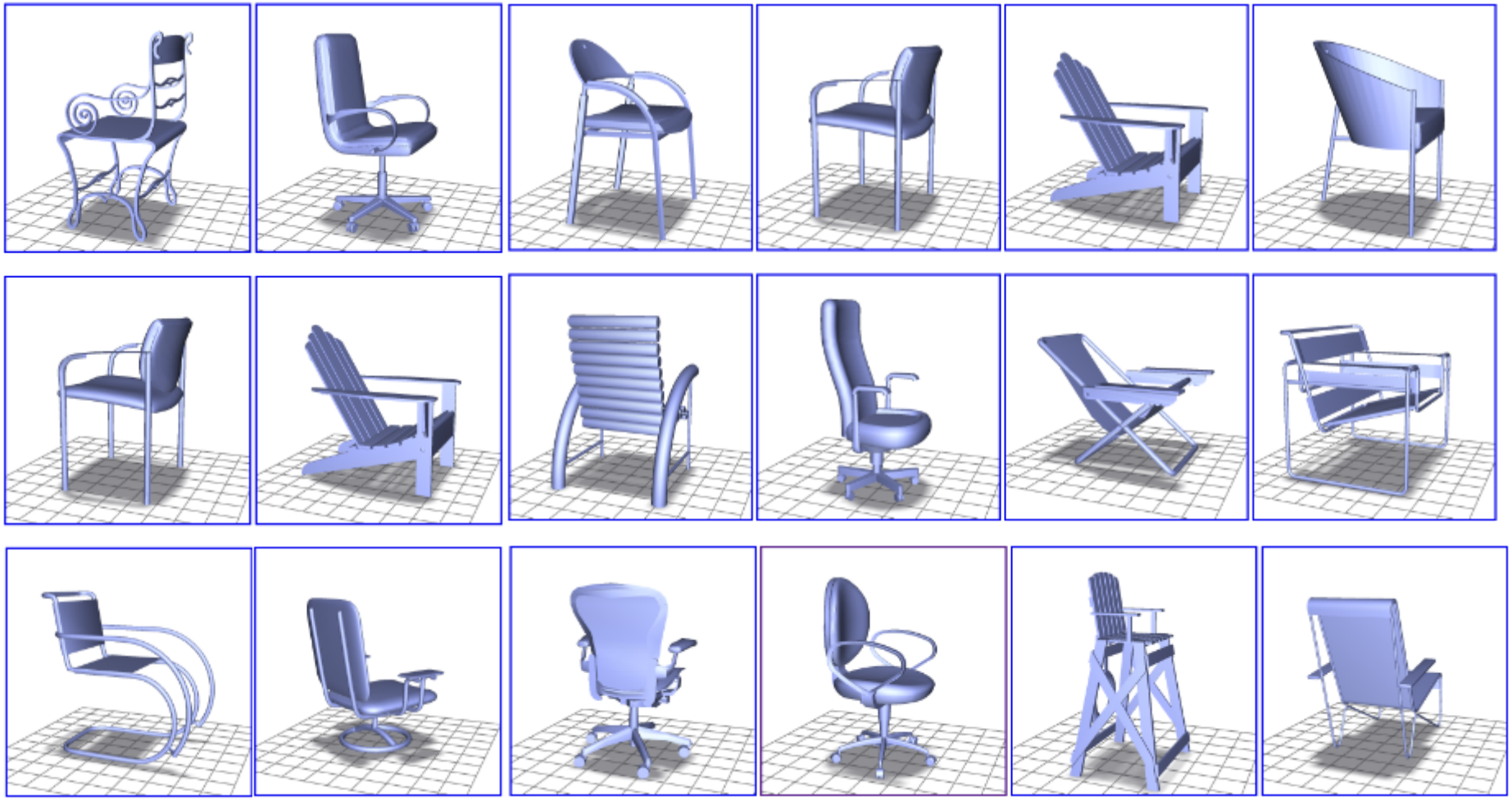
- Given *n* input shapes, search for a meaningful relation *R* between their elements



Challenges

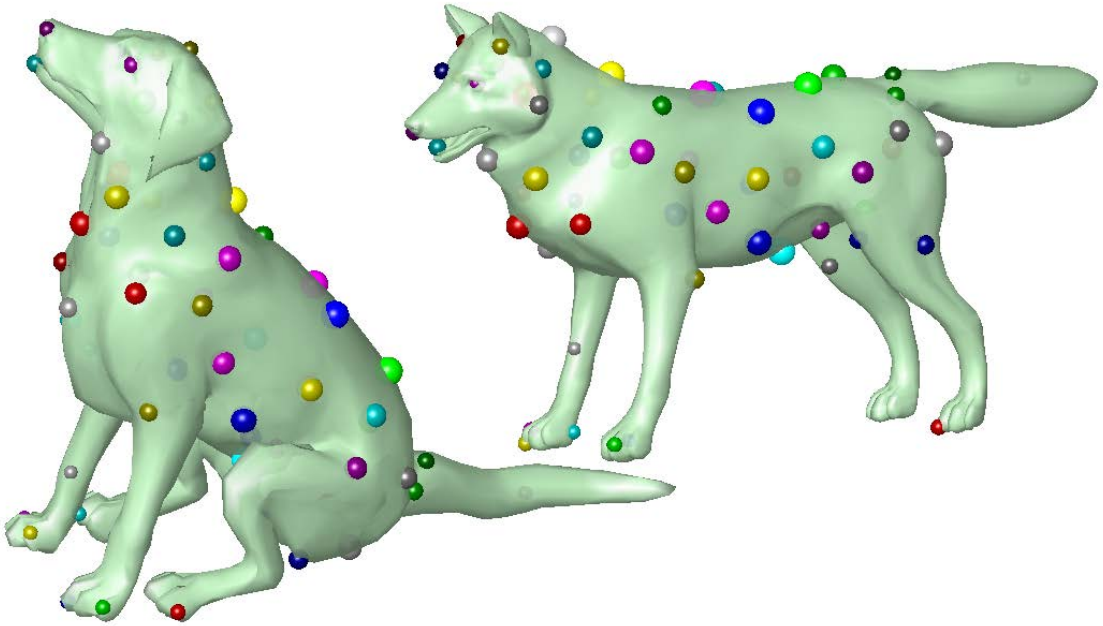
- Exponential complexity (NP hard)
  - $P\ {}_{N}^{M} = \frac{M!}{(M-N)!}$, $M > N$
- Requires understanding the **structure** of the shapes both at local and global level
- Requires understanding the **functionality** of shapes

Diffiulties: Semantic shape analysis



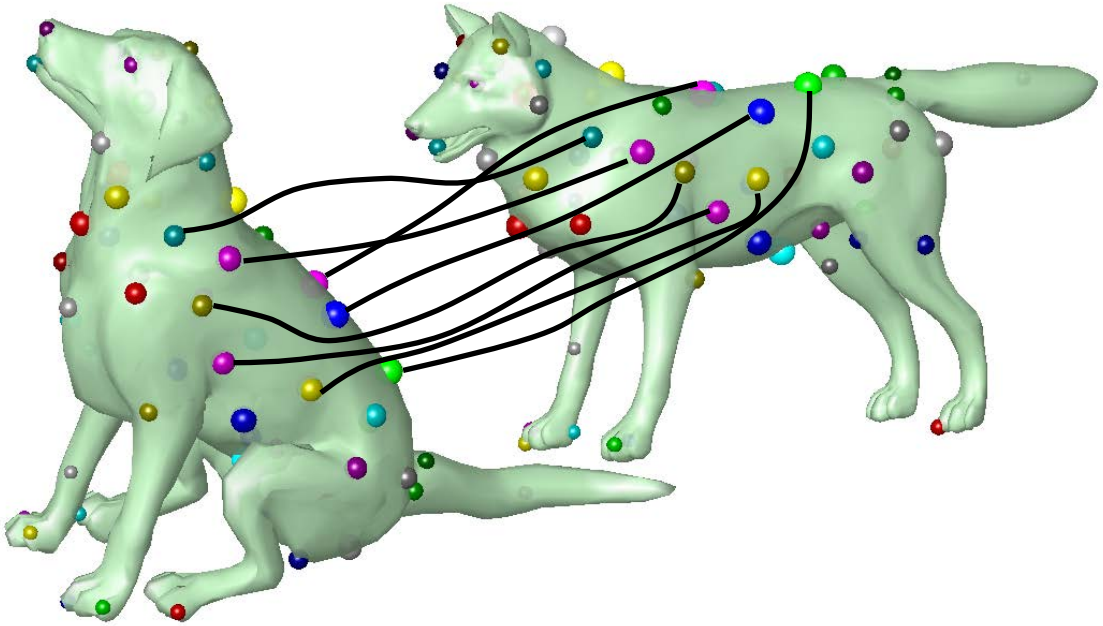
Aside: visualizing correspondence

- Matching color tags



Visualizing correspondence

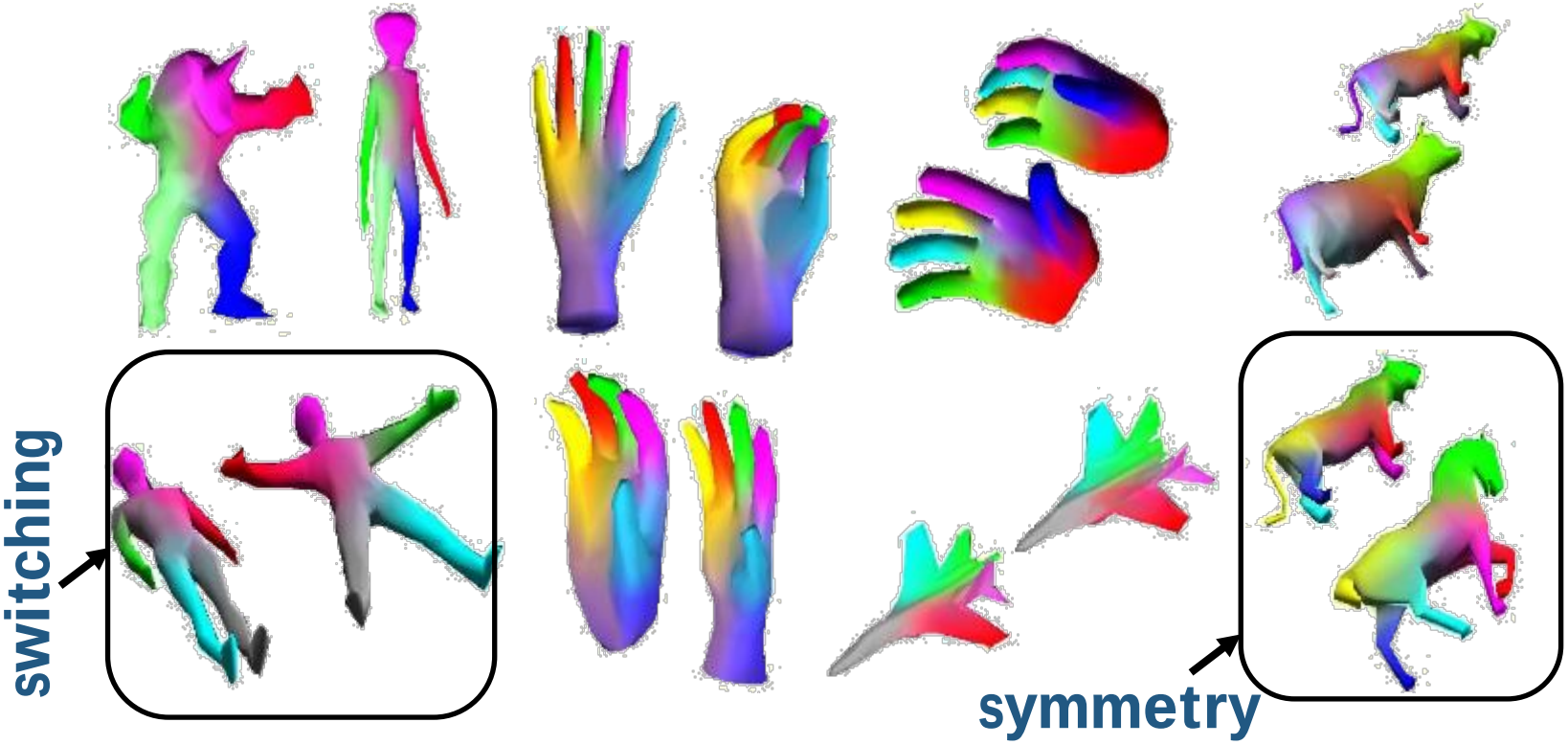
- Edge links: becoming cluttered quickly



Question: effective visualization of correspondence?

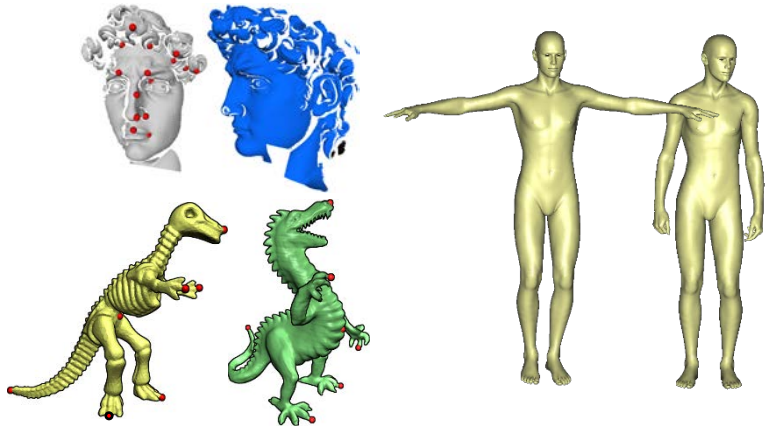
Visualizing correspondence

- Color map



Taxonomy

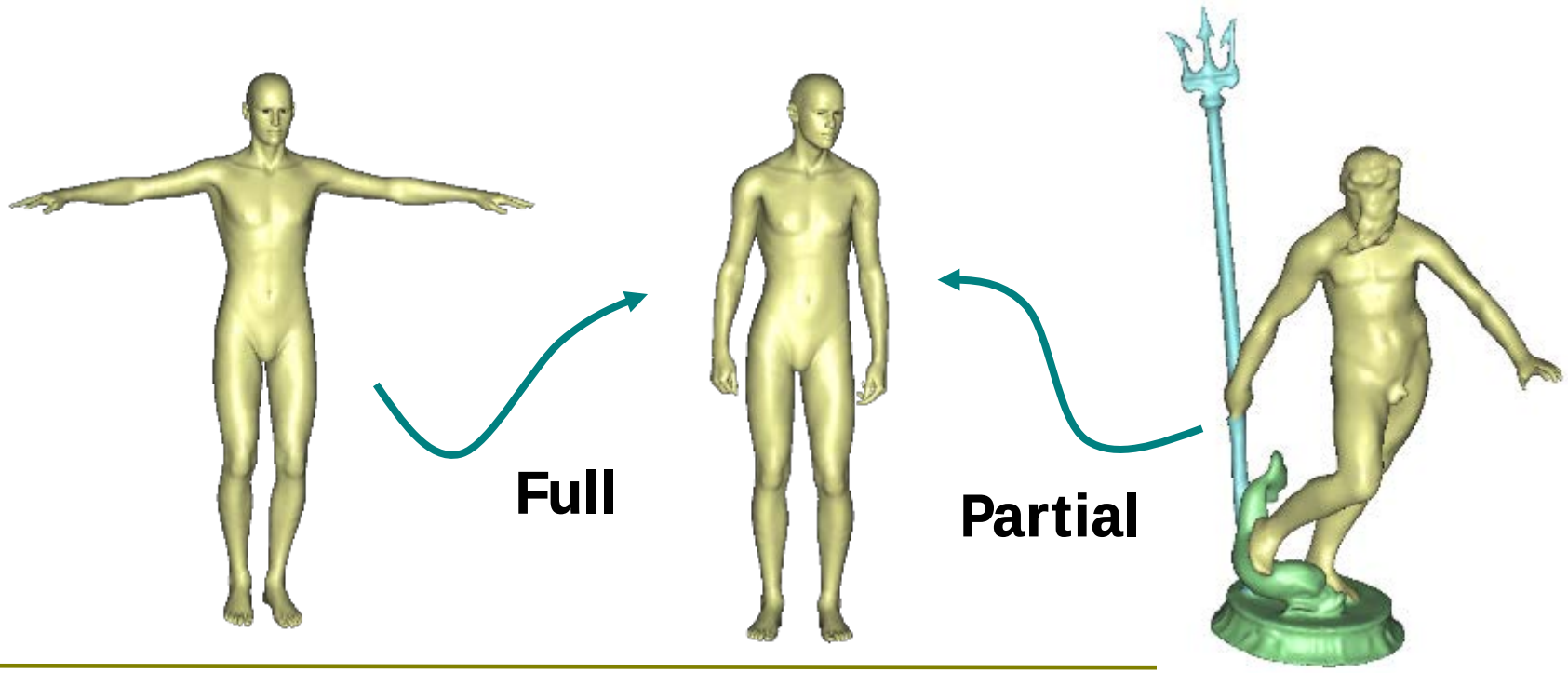
- Correspondence over the entire surface?
  - Yes: Full
  - No: Partial
- Initial registration available?
  - Yes: Local optimization methods
  - No: Global methods
- Class of transformations?
  - Rotation and translation: rigid
  - piece-wise rigid: Isometric
  - Large deformation



Also noted by : inter- vs. intra-subject registration

Output: full vs. partial

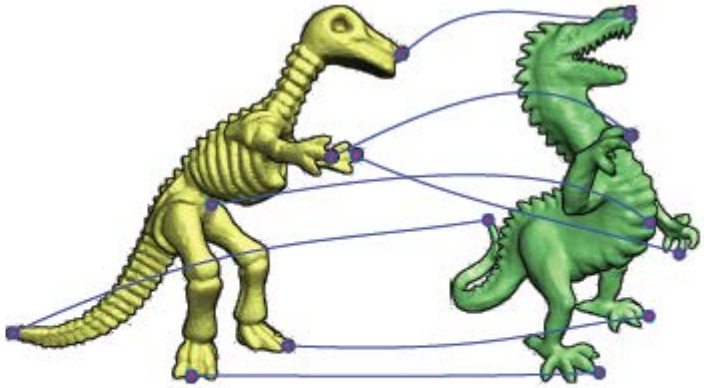
- Matching partial shapes —— adds more challenge
  - Larger search space: also need to find the subsets
  - many more possible solutions!



Output: dense vs. sparse

- Given *n* input shapes, search for a meaningful relation *R* between their elements

Dense [DYT05]



Sparse [ZSCO*08]

Traditional approaches

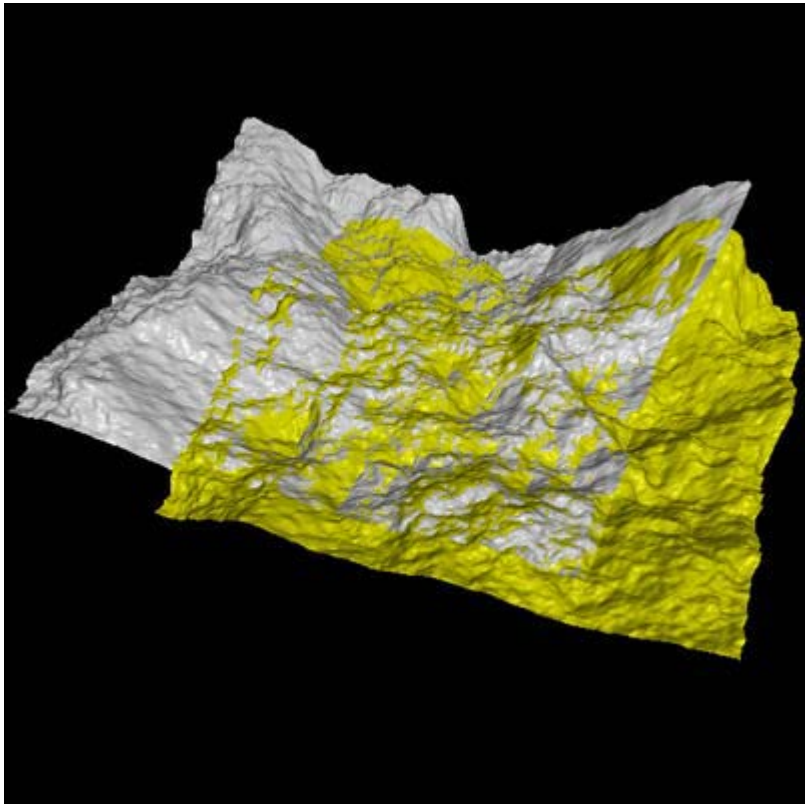
- Rigid alignment
  - ICP
- Nonrigid transformation: via feature matching
  - Similarity + regularization (distance preservation,..)

Iterative closest point (ICP) [Besl and Mckay 92]

- One of the most classic correspondence schemes
- Input: source (data) and target shapes
- Objective:
  - Rigid transform = rotation + translation
  - Minimize mean squared error from source points to closest points in target
- Correspondence obtained by Euclidean proximity

Rigid Registration : Goal

- Align two partially-overlapping meshes given initial guess for relative transform

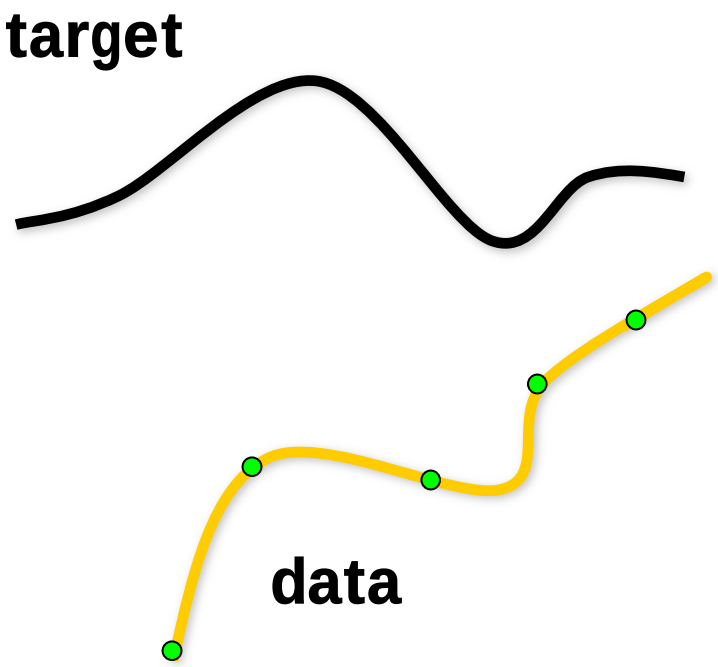


Basic ICP

- Select e.g. 1000 random points
- Match each to closest point on other scan, using data structure such as *k*-d tree
- Reject pairs with distance > *c* times median
- Construct error function and solve for it:
  - Minimize (closed form solution in [Horn 87])

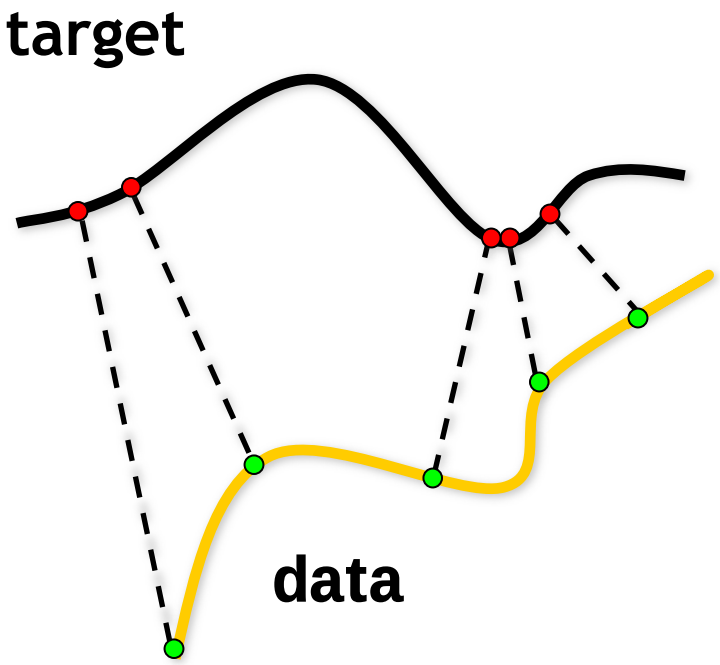
$$E \equiv \sum \left| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i \right|^2$$

- Repeat aboves



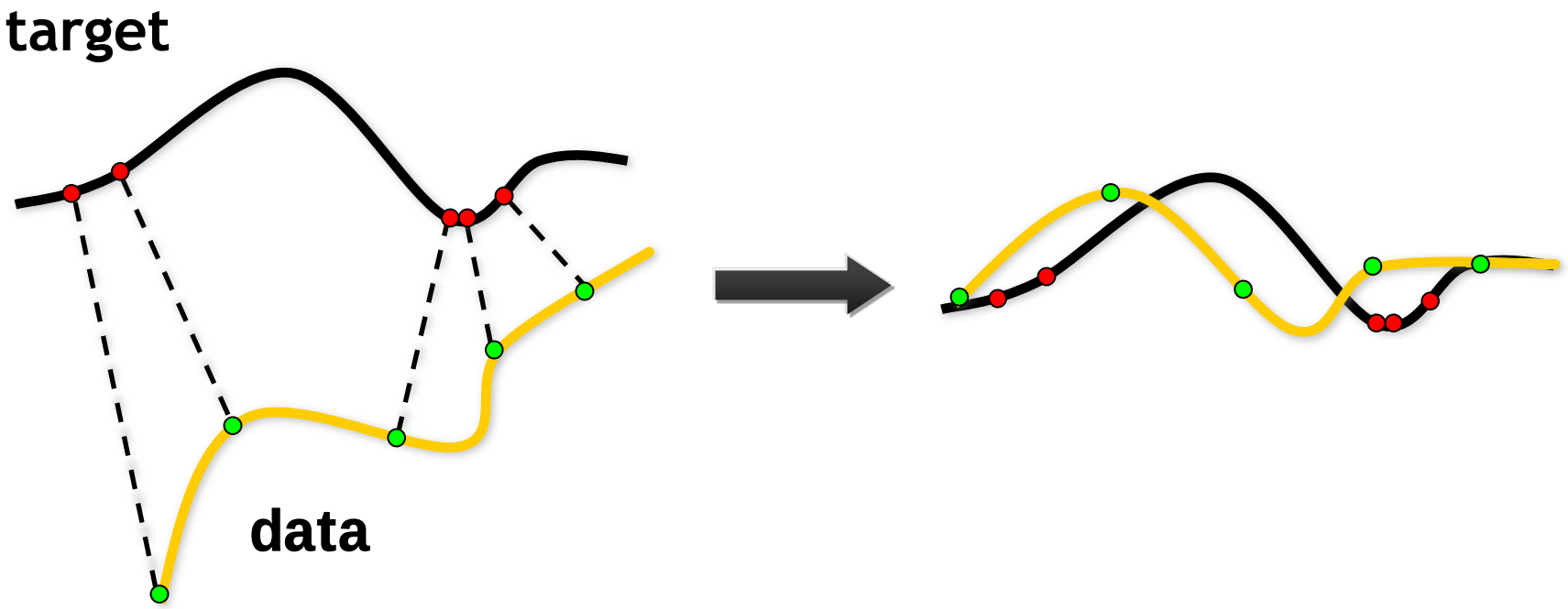
Model and data shapes (point samples)

[Besl and Mckay 92]



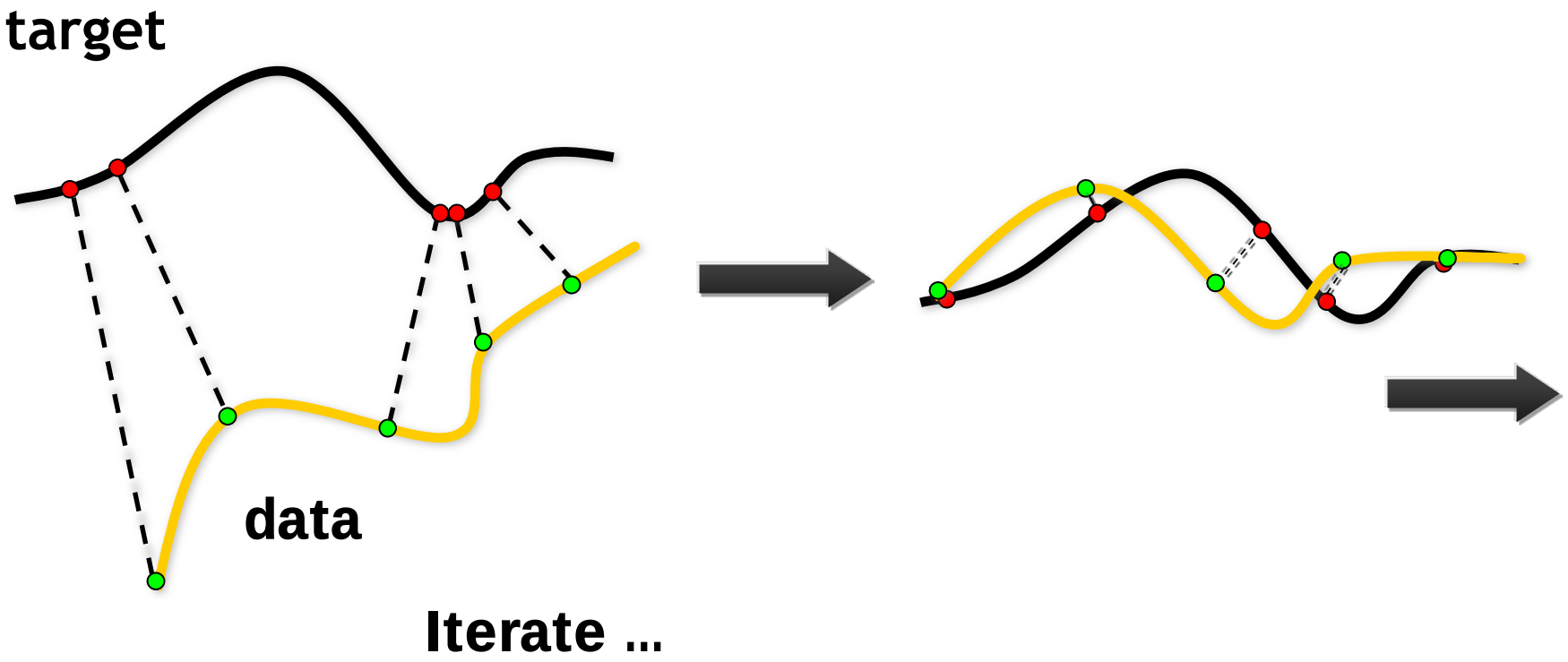
Find closest points from data to target

[Besl and Mckay 92]



Find best rigid transform to align the corresponding points

[Besl and Mckay 92]



[Besl and Mckay 92]

ICP Variants

1. Selecting source points (from one or both meshes)
2. Matching to points in the other mesh
3. Weighting the correspondences
4. Rejecting certain (outlier) point pairs
5. Assigning an error metric to the current transform
6. Minimizing the error metric w.r.t. transformation

Issues with ICP

- Many variants [Rusinkiewicz and Levoy 01]
- Greedy: bad starts lead to bad local minima
- Usually, initial data and model almost aligned
- ICP works best as a refinement scheme

Traditional approaches

- Rigid alignment
  - ICP

- Nonrigid transformation: via feature matching
  - Similarity + distance preservation

Meaningful correspondence

- Appropriate notion of “similarity” or “agreement”
- Similarity: corresponding points/regions look similar – local geometric similarity
- Agreement: close-by points should match close-by points – proximity or distortion

Problem formalization

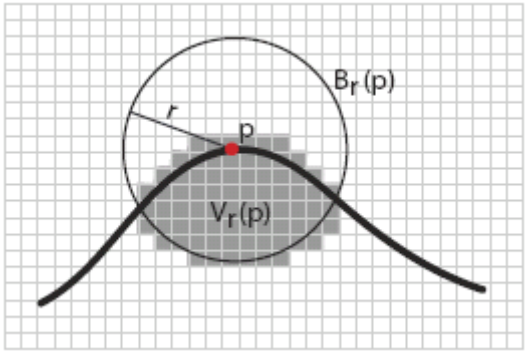
$$\min_{\mathrm{M}}[\alpha \cdot \mathrm{dist}_{\mathrm{lin}}(M) + \beta \cdot \mathrm{dist}_{\mathrm{quad}}(M \times M)]$$

- M: transformation (registration) or 1-to-1 mapping (correspondence)

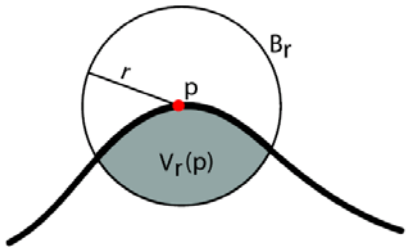
- $\mathrm{dist}_{\mathrm{lin}}$: similarity
- $\mathrm{dist}_{\mathrm{quad}}$: global consistency (compatibility, distortion) (geodesic distance, smoothness)

Local descriptor: Integral Volume Descriptor

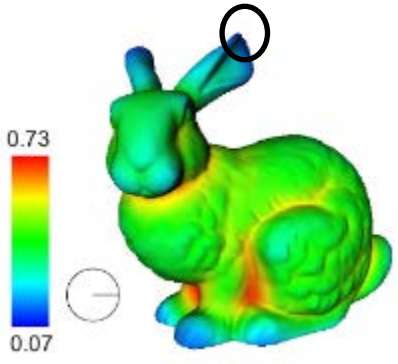
- Convolution of occupancy grid (cell) with ball



$$V(\mathbf{p}) = (G_B * G_O)(\mathbf{p})$$



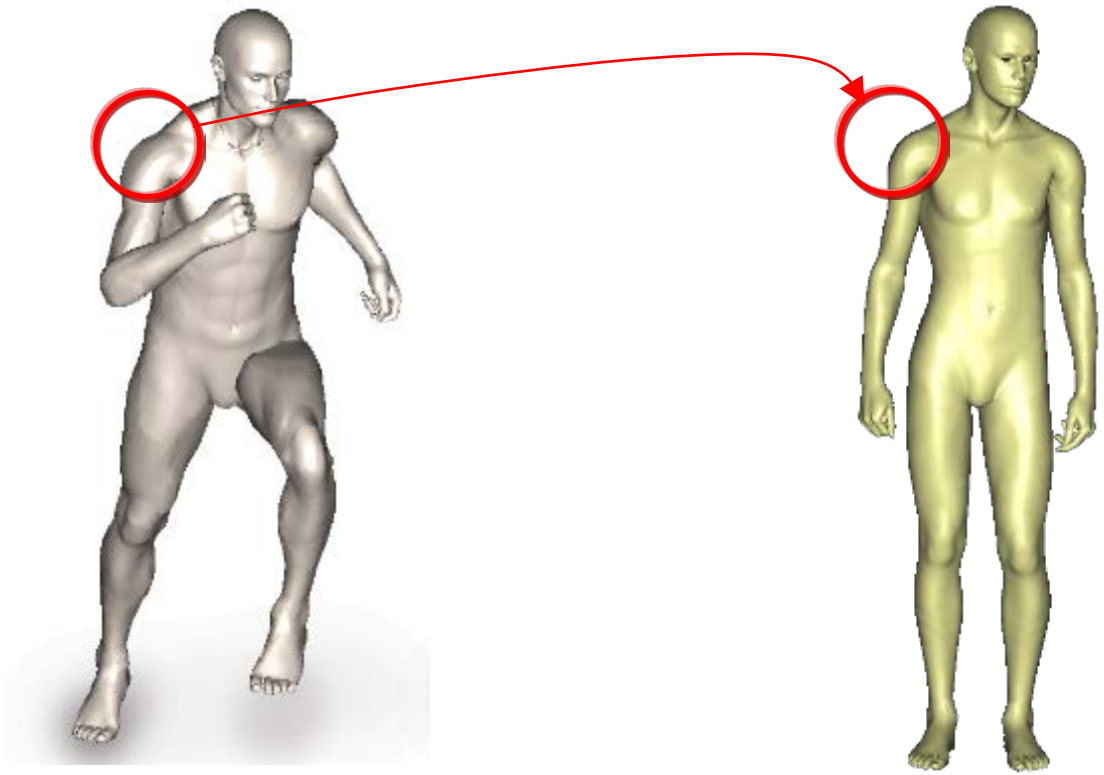
$$V_r(p) = \int_{B_r(p) \cap S} dx$$

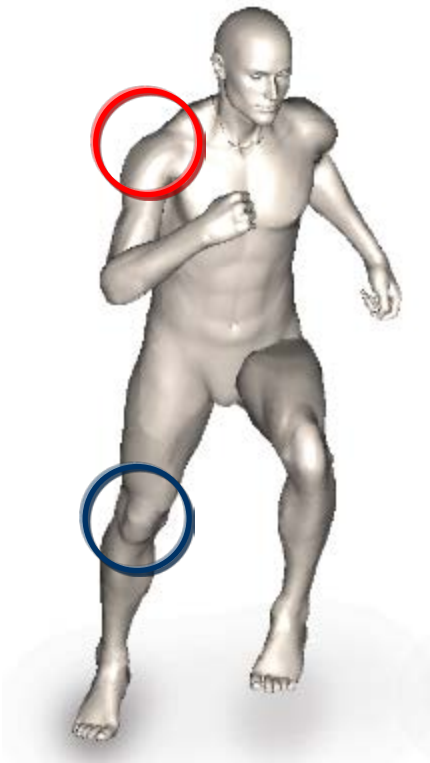


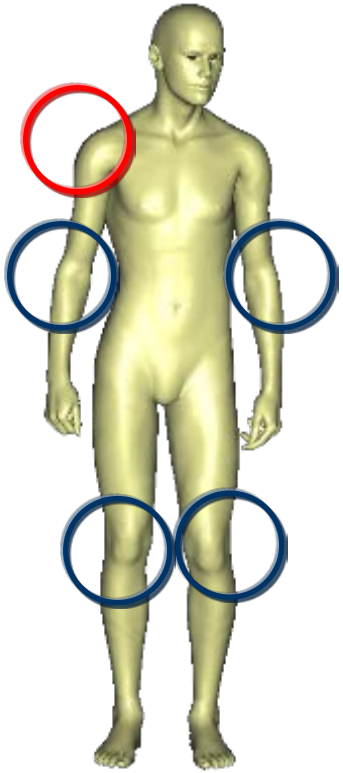
Traditional approaches

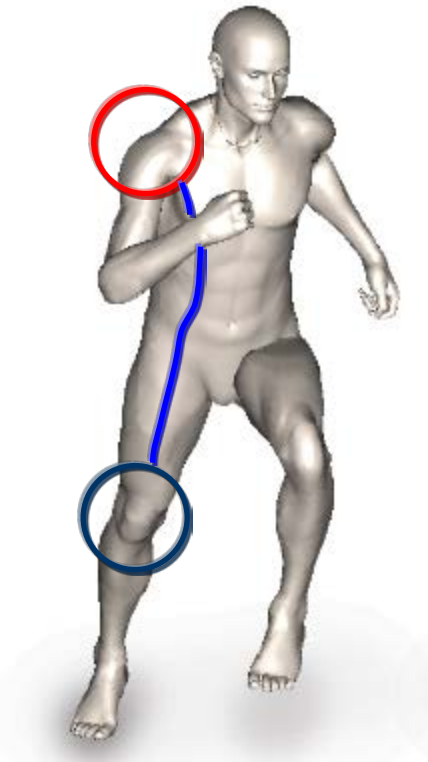
- Feature matching

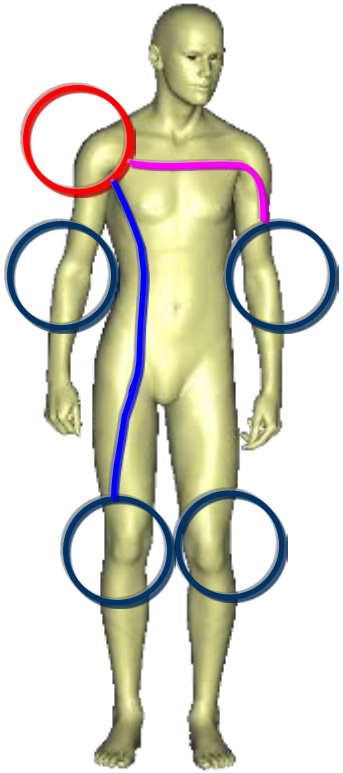
local shape similarity and distance preservation

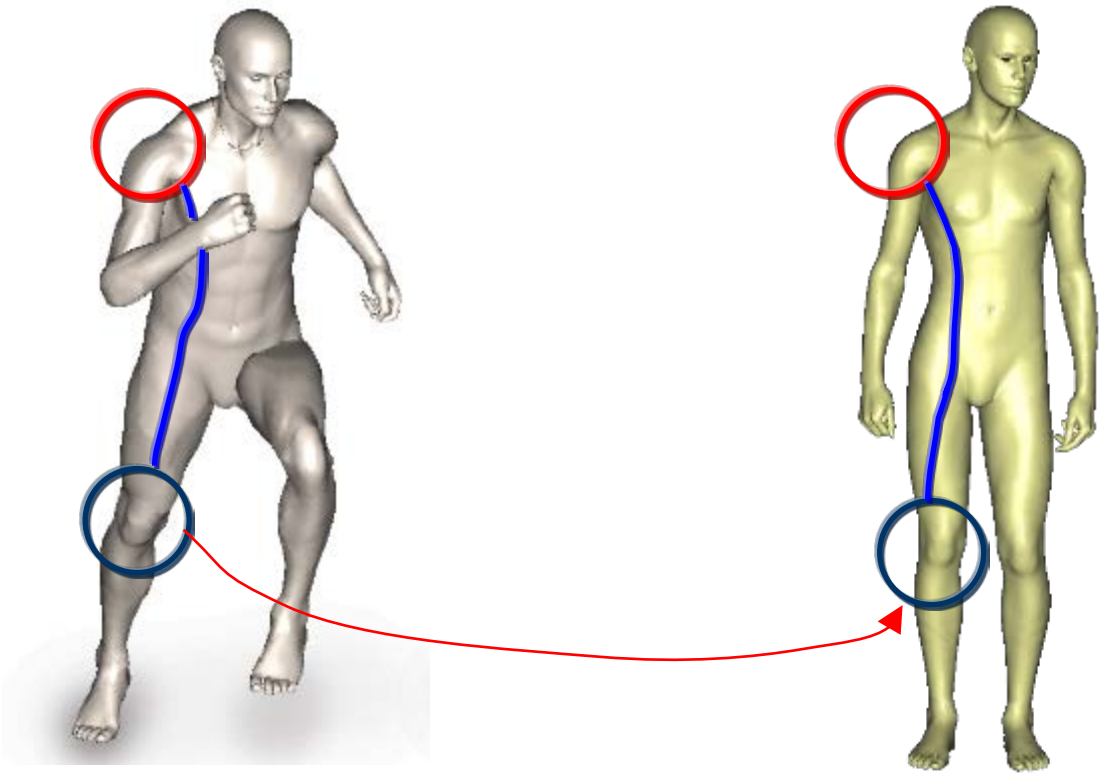












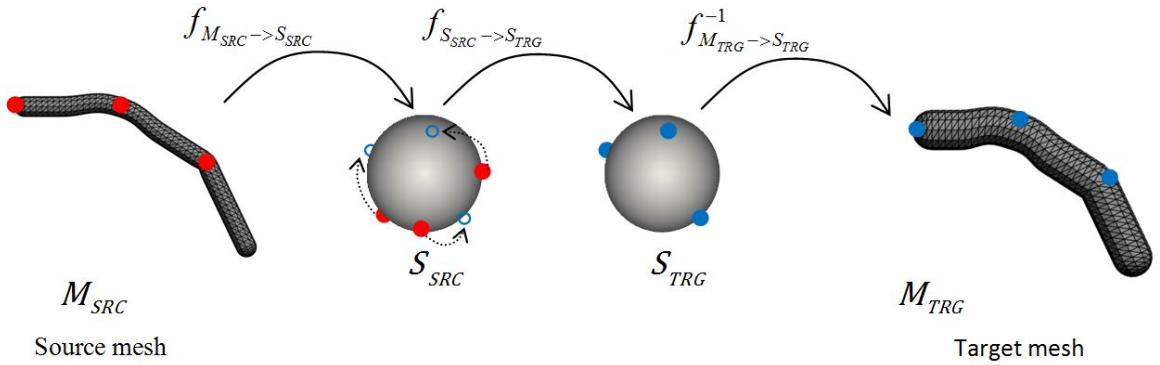
and so on…

Strategies

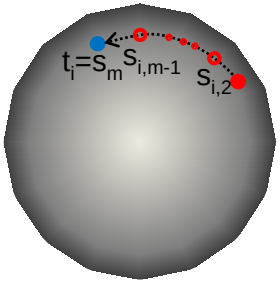
- Efficiently prune out bad solutions, or quickly approach to good solutions by using:
  - Use of prior knowledge
  - Part analysis
  - Spectral technique
  - Embedding
  - Etc.

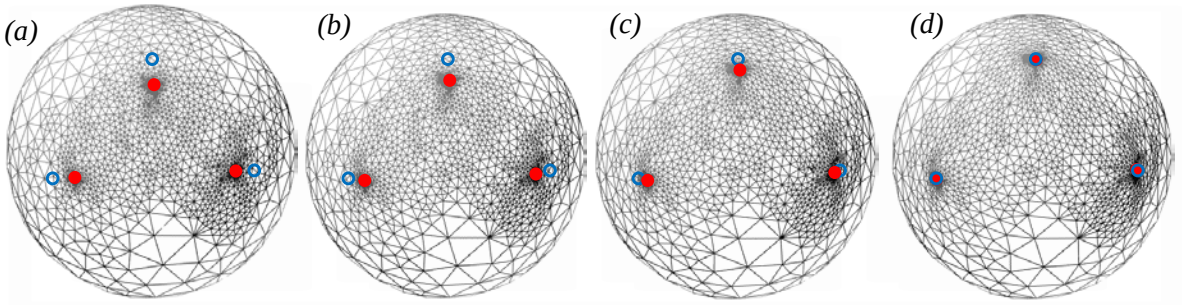
- Other important tasks:
  Design of good error function and shape descriptor

Iterative warping on spherical embedding

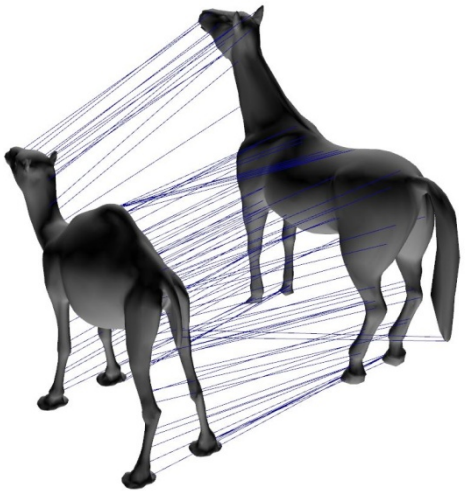
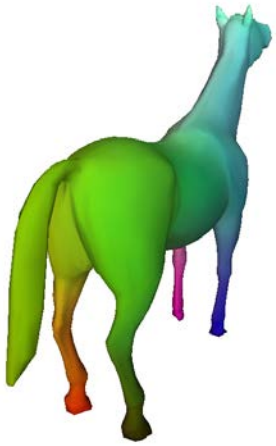
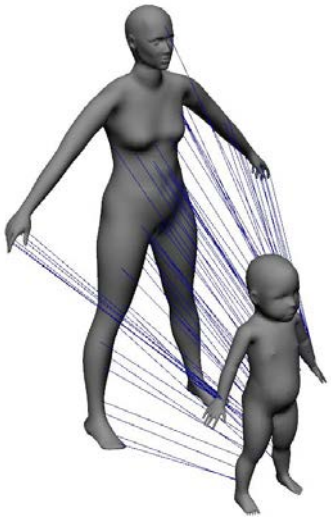
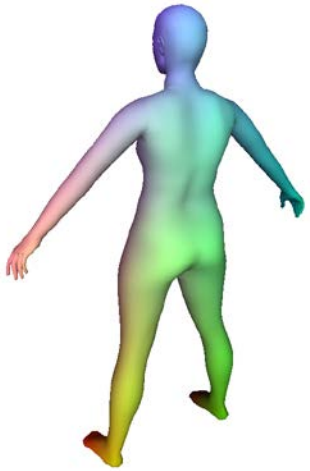
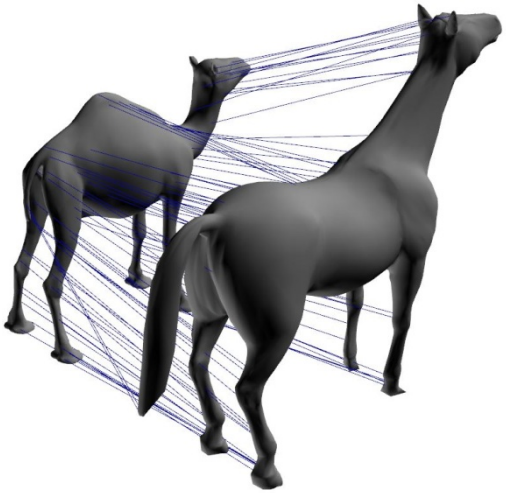


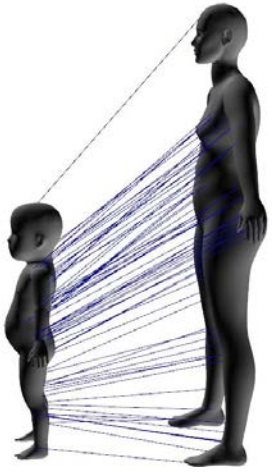
(Seo and Cordier 2010)





Results: Fine matching




Data-driven facial modeling

Data-driven facial modeling

- Database of hundreds of 3D scans of :
  - Different persons' faces
  - Facial expressions

=> A face represented as a (face) vector

Morphable model: representation (Blanz & Vetter, Siggraph99)

https://www.youtube.com/watch?v=pSRA8GpWIrA

Reference head **data i** **example i**

- Use this to define *shape* and *texture* vectors

$S_0 = (x\ y\ z\ x\ y\ z\ \ldots\ldots)^T$
$T_0 = (r\ g\ b\ r\ g\ b\ \ldots\ldots)^T$

$S_1 = (x\ y\ z\ x\ y\ z\ \ldots\ldots)^T$
$T_1 = (r\ g\ b\ r\ g\ b\ \ldots\ldots)^T$

Data-driven methods

- The main idea: Learn the model from a dataset!
  - Subspace, morphable, statistical, parametric, linear model…
  - Captures the shape and texture variations with a set of basis
- A new model is generated by a vector of blending weights

$$R_\rho\left(\begin{array}{l}\alpha_1 * \text{face} + \alpha_2 * \text{face} + \alpha_3 * \text{face} + \ldots\ldots \\ \beta_1 * \text{face} + \beta_2 * \text{face} + \beta_3 * \text{face} + \ldots\ldots\end{array}\right)$$

Morphable model

- A subspace for facial identity variation has been constructed

Shape $\mathbf{S} = (X_1, Y_1, Z_1, X_2, \ldots Z_n)$

Texture $\mathbf{T} = (R_1, G_1, B, R_2, \ldots B_n)$

PCA $\Rightarrow$ $\mathbf{S}(\vec{\alpha}) = \bar{\mathbf{S}} + \sum_{i=1} \alpha_i \cdot \mathbf{s}_i, \quad \mathbf{T}(\vec{\beta}) = \bar{\mathbf{T}} + \sum_{i=1} \beta_i \cdot \mathbf{t}_i,$

- Model fitting to a given image is formulated as an

$$E_I = \sum_{x,y} \left\| \mathbf{I}_{input}(x,y) - \boxed{\mathbf{I}_{model}(x,y)} \right\|^2 \sim \mathrm{Rend}(\mathbf{f}(\vec{\alpha}, \vec{\beta}), \vec{\rho})$$

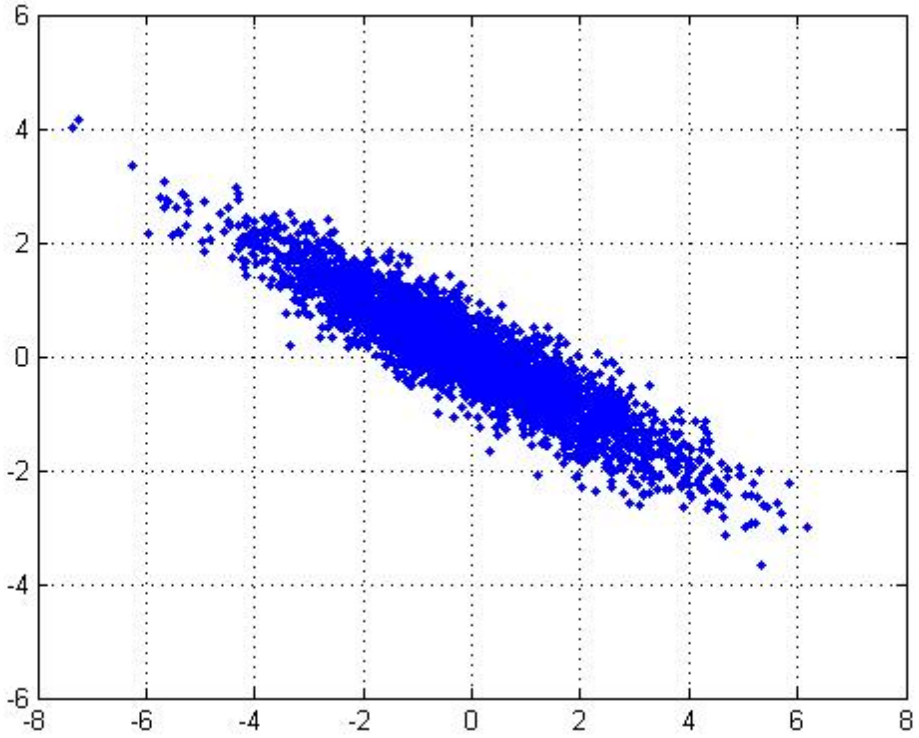
- A facial attribute is computed as weighted sums of labeled faces

$$\Delta\mathbf{S} = \sum_{i=1} \mu_i (\mathbf{S}_i - \bar{\mathbf{S}}), \quad \Delta\mathbf{T} = \sum_{i=1} \mu_i (\mathbf{T}_i - \bar{\mathbf{T}}),$$

- The solution space becomes constrained, solvable by common optimization techniques

Statistics

- Faces are points in the multi-dimensional face space



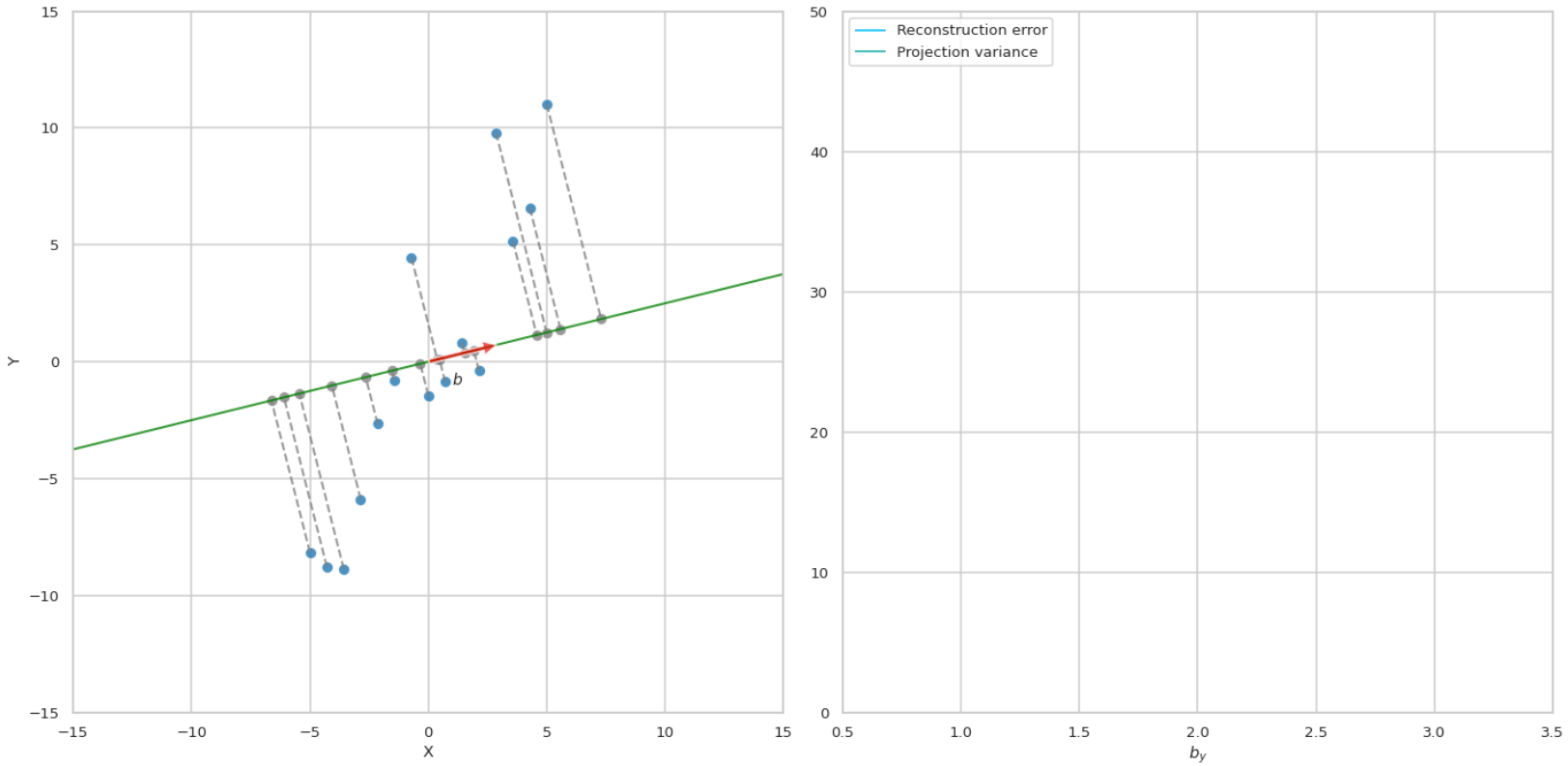
Statistics: PCA (Principal component analysis)

- Find new axis (thus space) that are suited for describing and manipulating faces and facial expressions
  - Subspace, morphable, statistical, parametric, linear model…
- Order dimensions of face space according to the variance found in data
  - *Data compression*
  - *Coarse-to-fine strategies*

PCA

- Maximum variance

https://towardsdatascience.com/dimensionality-reduction-with-pca-from-basic-ideas-to-full-derivation-37921e13cae7



- Find the new axis to maximize the variance of the projected data

Maximum variance

Sample set mean: $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$

Variance of the projected data: $\frac{1}{N}\sum_{n=1}^{N} \left\| (\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{u}_1 \right\|^2$ ← maximize

$$= \frac{1}{N}\sum_{\mathbf{x}} \mathbf{u}_1^{\mathrm{T}} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}} \mathbf{u}_1$$

$$= \frac{1}{N}\, \mathbf{u}_1^{\mathrm{T}} \left[\sum_{\mathbf{x}} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}} \right] \mathbf{u}_1 = \mathbf{u}_1^{\mathrm{T}} \mathbf{S} \mathbf{u}_1$$

where ***S*** is the data covariance matrix defined by:

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^{\mathrm{T}}$$

$\mathbf{u}_1$: axis of projection, unit vector, i.e. $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$

Maximum variance

https://fr.wikipedia.org/wiki/Multiplicateur_de_Lagrange

- Formulation of maximization using Lagrange multiplier

$$L(\mathrm{x},\lambda) = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1)$$

Constraint

(derivative w.r.t. $\mathbf{u}_1$)

https://en.wikipedia.org/wiki/Matrix_calculus

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$$

- $\mathbf{u}_1$ is eigenvector of $\mathbf{S}$, λ_1 eigenvalue
- Problem of variance maximization => that of eigenvalue maximization

Principal component anslysis

1. Form the data matrix X containing your data;
X is of size *K x N* (*K* dimension; *N* samples)

2. Calculate the covariance matrix S, based on X;

3. Solve $S\underline{e} = \lambda\underline{e}$ for the eigenvectors $\underline{e}$ and eigenvalues λ
- *K* such eigenvectors and eigenvalues.

4. Obtain new coords of each data by projecting it to first *m* (<<*N)* principal components (eigenvectors with *m* largest eigenvalues): $P = X\underline{e}$
- *N* PC coefficient vectors, each of *K* dimensions

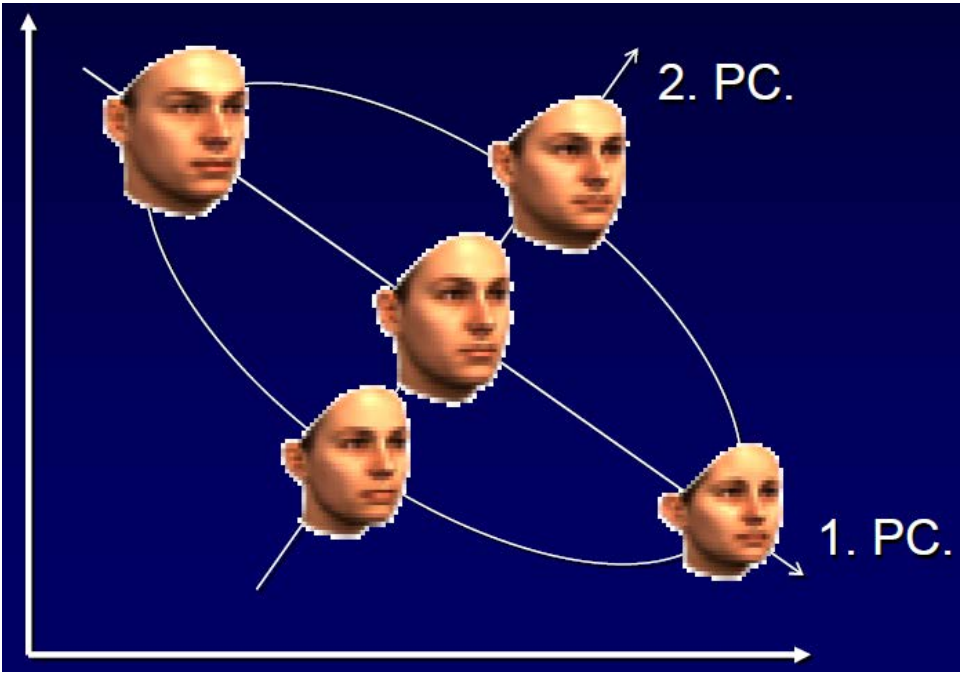
Many off-the-shelf packages, *e.g*. Mathlab, python, .. have PCA routines.

PCA – what you get

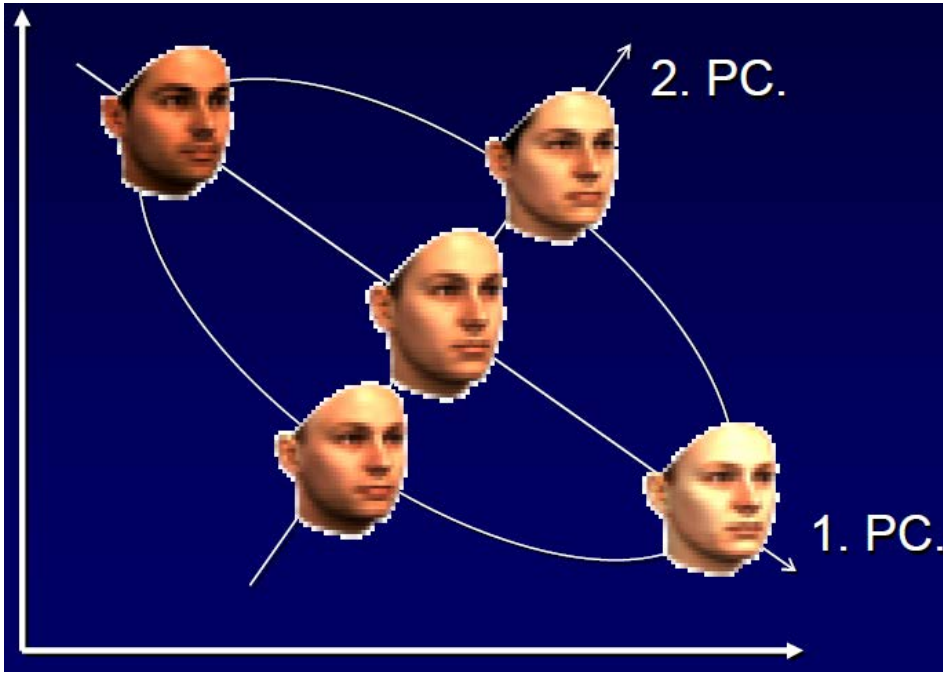
- The eigenvectors of S define a new coordinate system.
  - These eigenvectors are known as the principal components.
  - Eigenvector (1st PC) with largest eigenvalue captures direction of maximum variance among data vectors x.
  - Subsequent PCs are orthogonal to 1st PC and describe maximum residual variance.
- We can compress the data by only using the first few eigenvectors
  - A technique universally used to reduce the dimensionality of the data set.
  - Corresponds to choosing a “linear subspace”.

PCA of shapes, textures

Visualization of the shape subspace (constant texture)



Visualization of the texture subspace (constant shape?!)



Shape $\mathbf{S} = (X_1, Y_1, Z_1, X_2, \ldots Z_n)$

Texture $\mathbf{T} = (R_1, G_1, B, R_2, \ldots B_n)$

PCA

$$\mathbf{S}(\vec{\alpha}) = \bar{\mathbf{S}} + \sum_{i=1} \alpha_i \cdot \mathbf{s}_i, \qquad \mathbf{T}(\vec{\beta}) = \bar{\mathbf{T}} + \sum_{i=1} \beta_i \cdot \mathbf{t}_i,$$

Facial attribute manipulation

Goal:

- Manipulate attributes (expression, weight, gender), but leave individual characteristics unchanged, i.e. same person.

1. Linear function

- Fit a linear function to input data
  - Use Linear Regression or Discriminant Analysis or Support Vector Machine.
- Follow gradient to manipulate faces

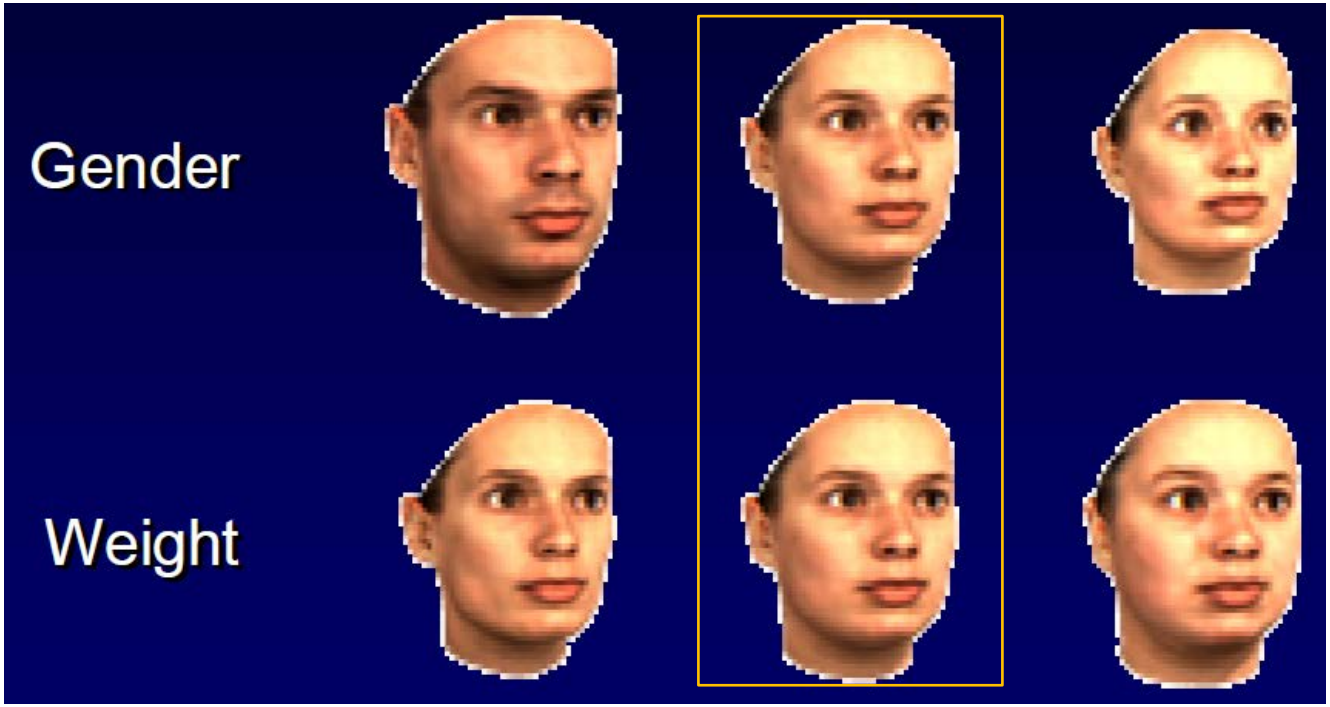
2. Directly compute the displacements from labeled examples

$$\Delta S = \sum_{i=1}^{m} \mu_i (S_i - \overline{S}), \quad \Delta T = \sum_{i=1}^{m} \mu_i (T_i - \overline{T}).$$

Facial Attributes

original

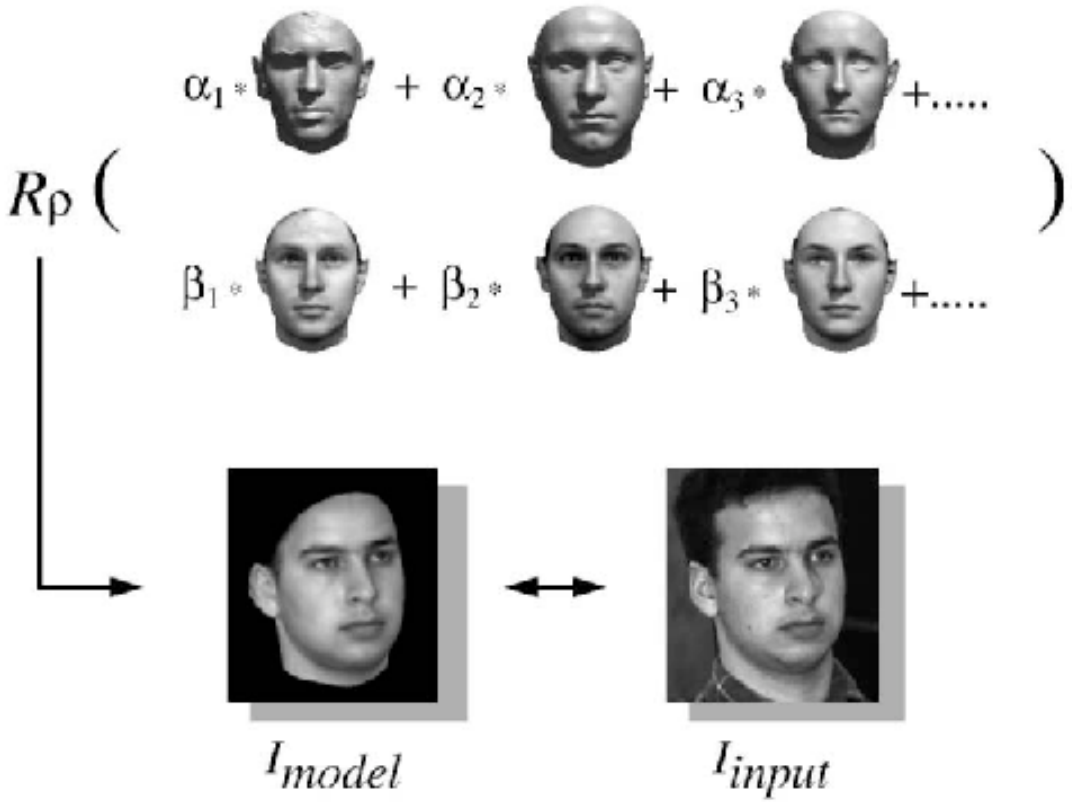
Facial Attributes



original

Facial Attributes

Fitting the model to an image



- *R* = Rendering (Perspective Projection, Phong Illumination, Cast Shadows)
- ρ = Pose, Illumination, ...

- Minimize the image difference with an optimization..

Reanimation in Images and Video

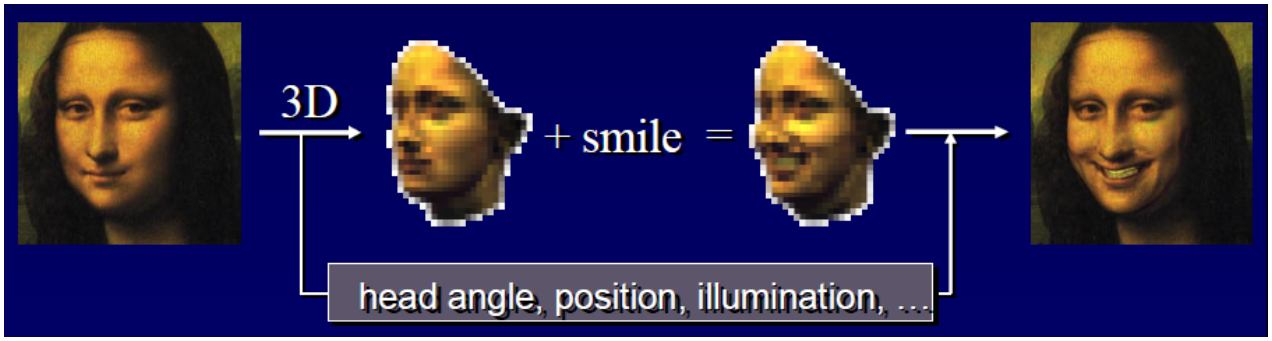
Animate
- unknown faces
- in given images or video
- at any pose and illumination

Approach

1. Reconstruct 3D shape

2. Add 3D deformation

3, Draw 3D face into the image



Automated parameter estimation

- Face parameters
  - Shape coefficients
  - Texture coefficients
- 3D geometry
  - Head position
  - Head orientation
  - Focal length
- Light and color
  - Ambient: Intensity, color
  - Directional: Intensity, color, direction

Error function

- Image difference

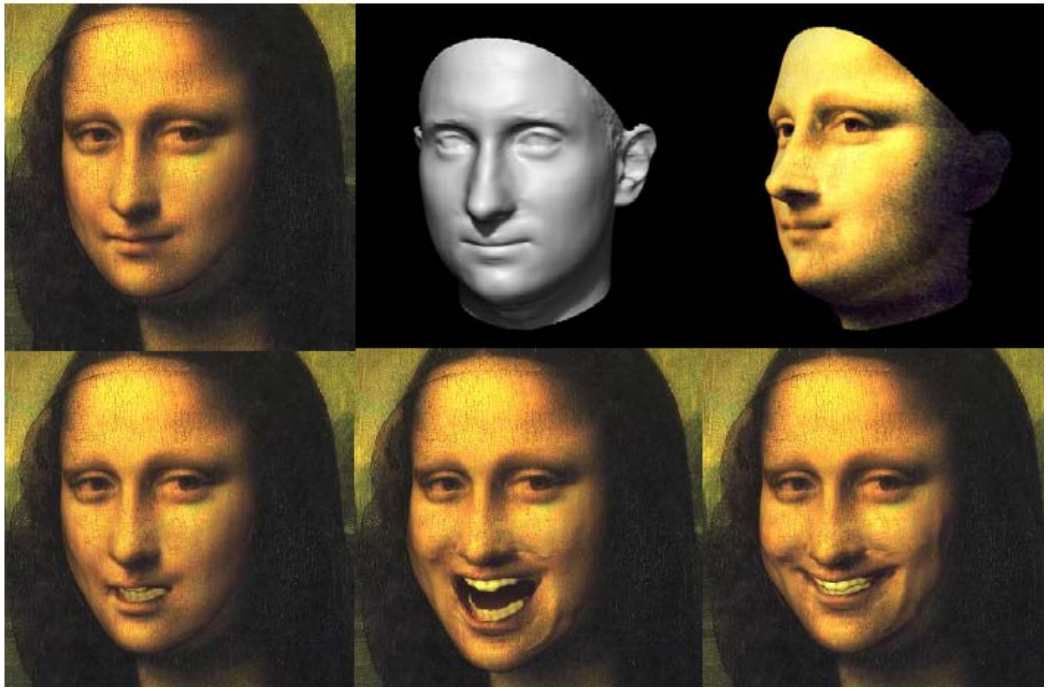
$$E_I = \sum_{x,y} \|\mathbf{I}_{input}(x,y) - \mathbf{I}_{model}(x,y)\|^2$$

- Plausibility based on PCA

$$E_{prior} = -\log(\mathrm{p}(\alpha_i, \beta_i, ..))$$

- Minimize $E = E_I + E_{prior}$

Mona Lisa



Conclusion

- Learning-based methods have a large potential for achieving photo-realistic results.
- Development of scanning technology is crucial for extensive datasets of high-quality scans.