CENG 789 – Digital Geometry Processing

07- Shape Generation

Prof. Dr. Yusuf Sahillioğlu

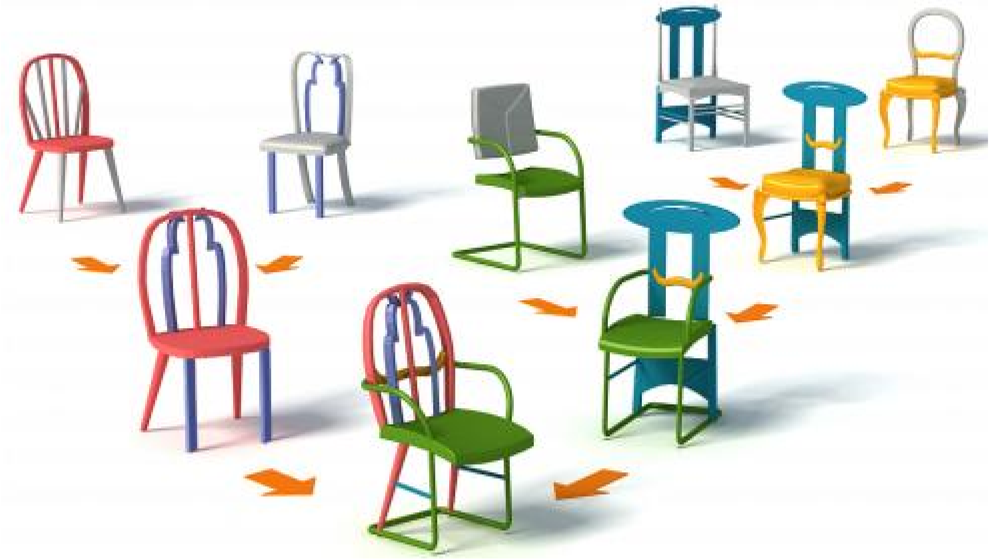
Computer Eng. Dept, MIDDLE EAST TECHNICAL UNIVERSITY , Turkey

Shape Synthesis / Mesh Generation

- ✓ Previously, we dealt with generation of meshes that represent shape surfaces from implicit scalar fields (in an effort to perform scientific visualization).
- ✓ One can also generate meshes (= synthesize shapes) by growing or learning a shape space.
- ✓ A totally different approach will also be discussed in the end (Shape from X).

Part-based Shape Synthesis

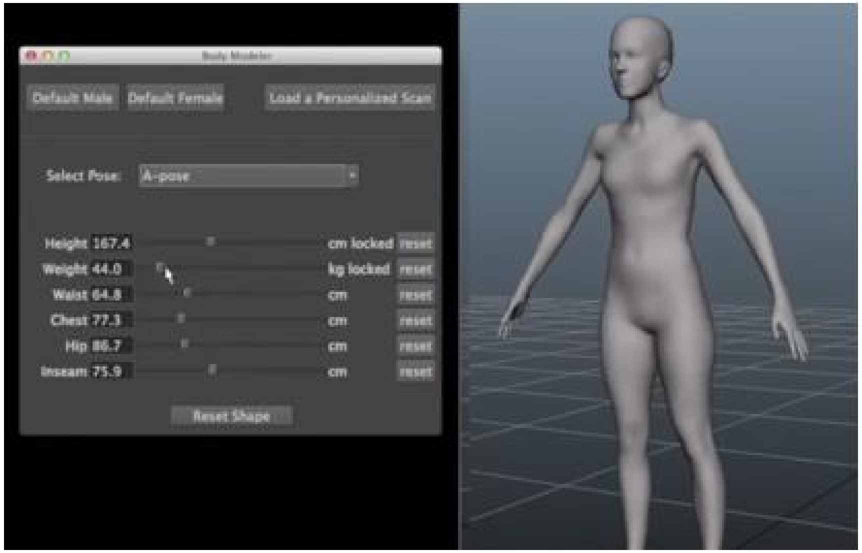
- ✓ One of the advantages of using digital models is the simplicity to generate new models based on the seed models.
- ✓ One method is shuffling and deforming existing parts in correspondence.
- ✓ Genetic algo: crossovers (shuffling) and mutations (deformations).



- ✓ Paper: Set Evolution for Inspiring 3D Shape Galleries.

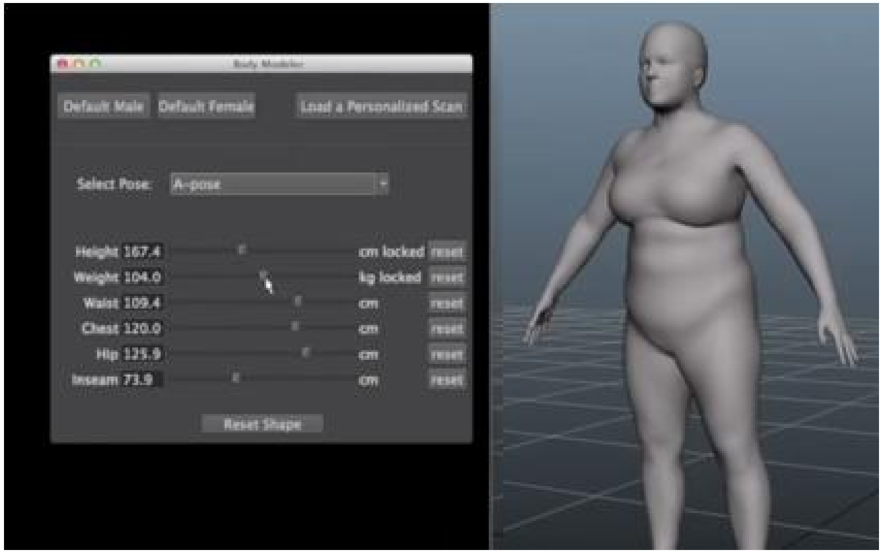
PCA-based Shape Synthesis

- ✓ One of the advantages of using digital models is the simplicity to generate new models based on the seed models.
- ✓ We will learn a different approach: learning shape space via PCA.



Select Pose: A-pose

Height 167.4 cm locked
Weight 44.0 kg locked
Waist 64.8 cm
Chest 77.3 cm
Hip 86.7 cm
Inseam 75.9 cm

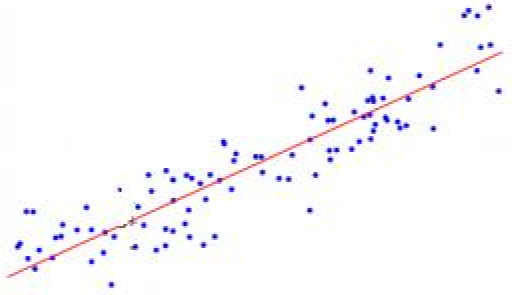


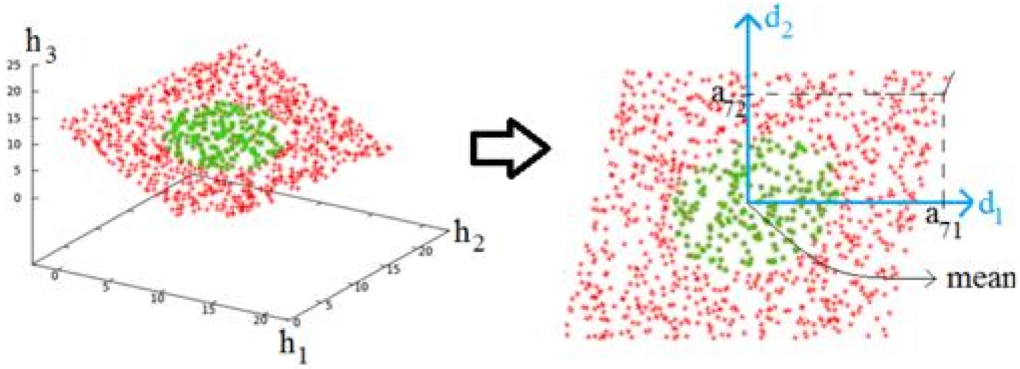
Select Pose: A-pose

Height 167.4 cm locked
Weight 104.0 kg locked
Waist 109.4 cm
Chest 120.0 cm
Hip 125.9 cm
Inseam 73.9 cm

PCA-based Shape Synthesis

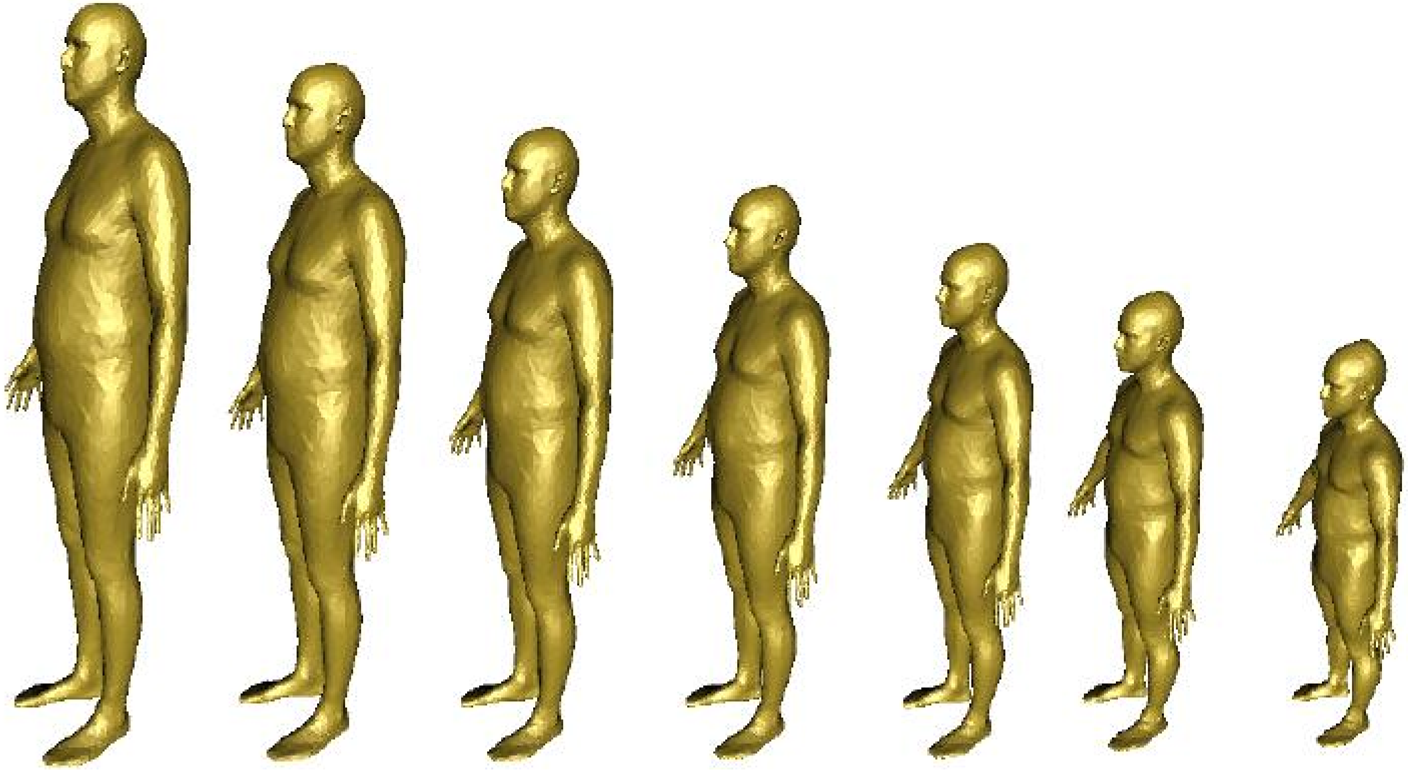
- ✓ One of the advantages of using digital models is the simplicity to generate new models based on the seed models.
- ✓ We will learn a different approach: learning shape space via PCA.





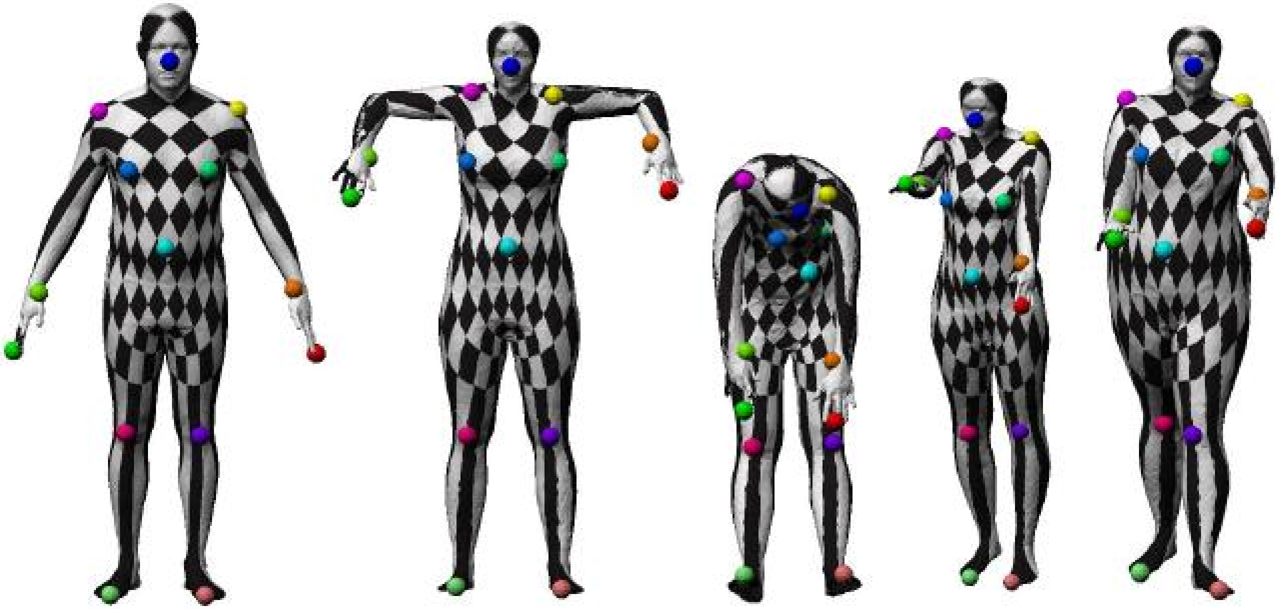
PCA-based Shape Synthesis

- ✓ One of the advantages of using digital models is the simplicity to generate new models based on the seed models.
- ✓ Having learned the variation below, synthesize the next shape.



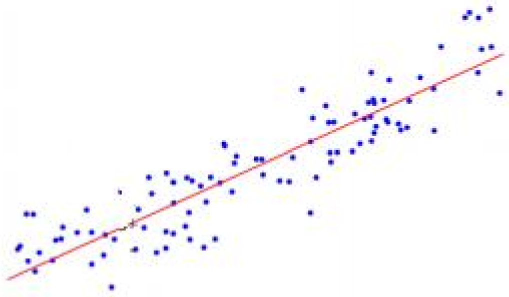
PCA-based Shape Synthesis

- ✓ One of the advantages of using digital models is the simplicity to generate new models based on the seed models.
- ✓ How about this variation?



PCA

- ✓ PCA: Principal Component Analysis.
- ✓ PCA: given a point* set, PCA finds a good orthogonal basis that represents these points well.
- ✓ PCA: given a point set, PCA finds a good orthogonal basis that is well-aligned with data.
- ✓ PCA: discovers directions among which data has big variation.

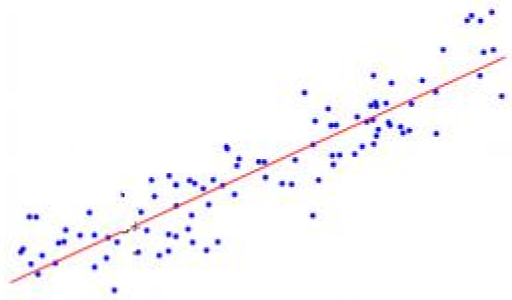


[*] Note that points may represent shapes as well, e.g., we are in R^{3000} for 1000-vertex mesh in 3D.

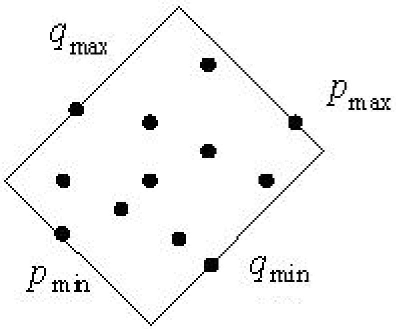
PCA Applications

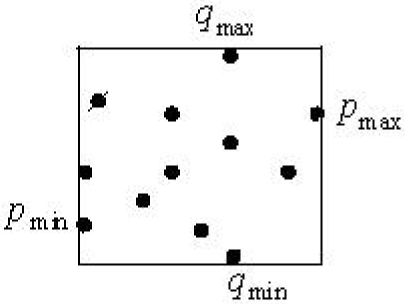
- ✓ Line fit, plane fit.

- ✓ Dimension reduction.



- ✓ Oriented bounding box (OBB) computation.

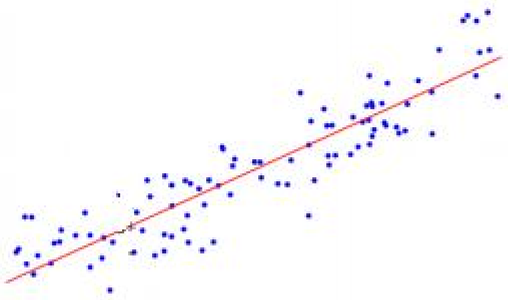




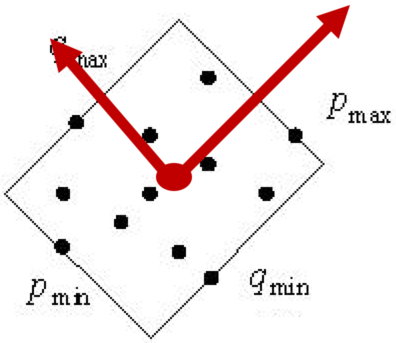
This is axis aligned bounding box (AABB).

PCA Applications

- ✓ Line fit, plane fit.

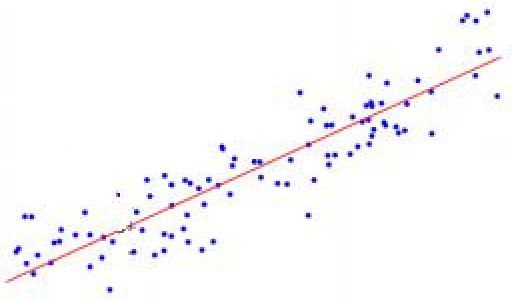


- ✓ Dimension reduction.
- ✓ Oriented bounding box (OBB) computation.
  - ✓ Dot products to get p/q min/max.



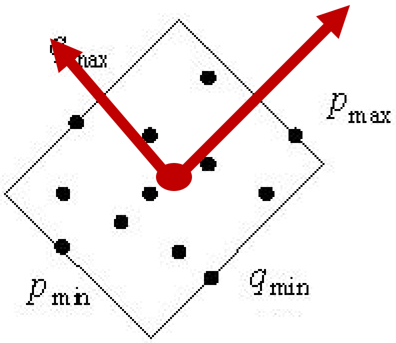
PCA Applications

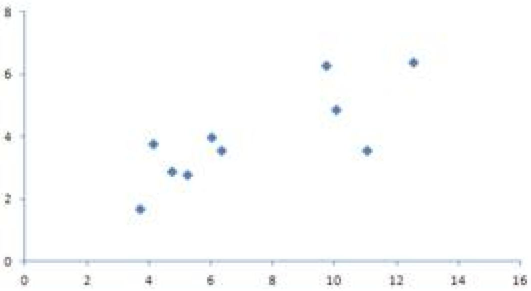
- ✓ Line fit, plane fit.

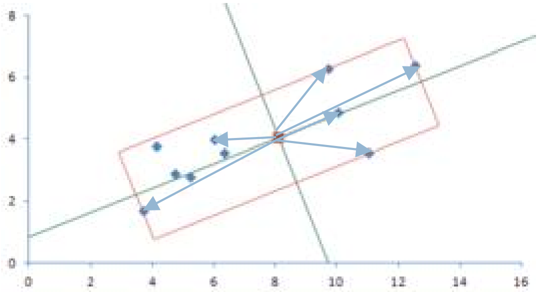


- ✓ Dimension reduction.

- ✓ Oriented bounding box (OBB) computation.
  - ✓ Dot products to get p/q min/max.
  - ✓ Projecting (blue) to principal components (green) gives the p/q min/max.

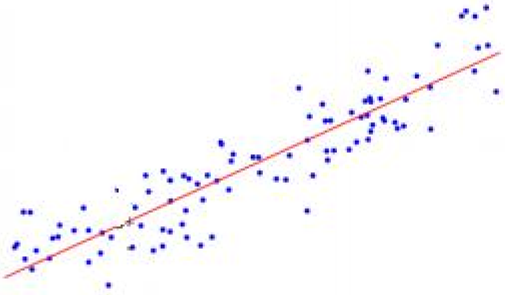




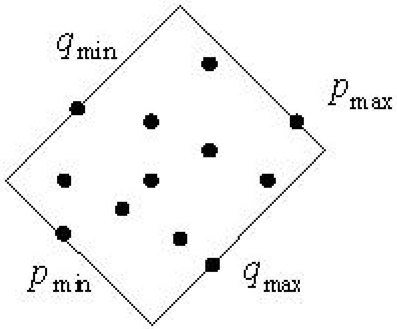


PCA Applications

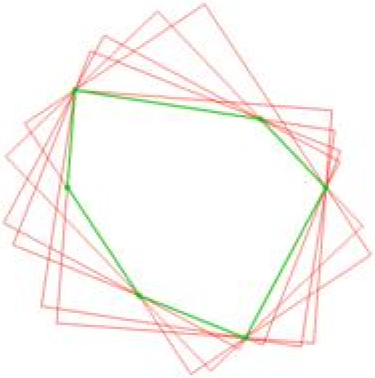
- ✓ Line fit, plane fit.



- ✓ Dimension reduction.

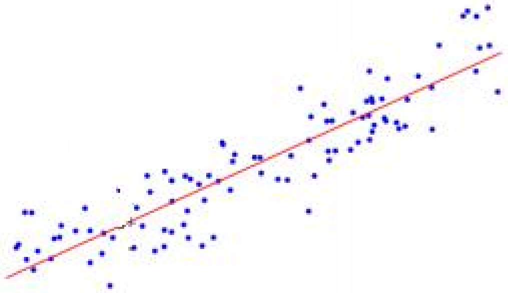


- ✓ Oriented bounding box (OBB) computation.
  - ✓ Rotating Calipers method based on the fact that OBBB has a side collinear w/ one of the edges of pnt set’s convex hull (1975 Freeman theorem.)
  - ✓ Reds are OBB candidates for the green convex hull (linear time). Answer: Min-area red is the OBBB.



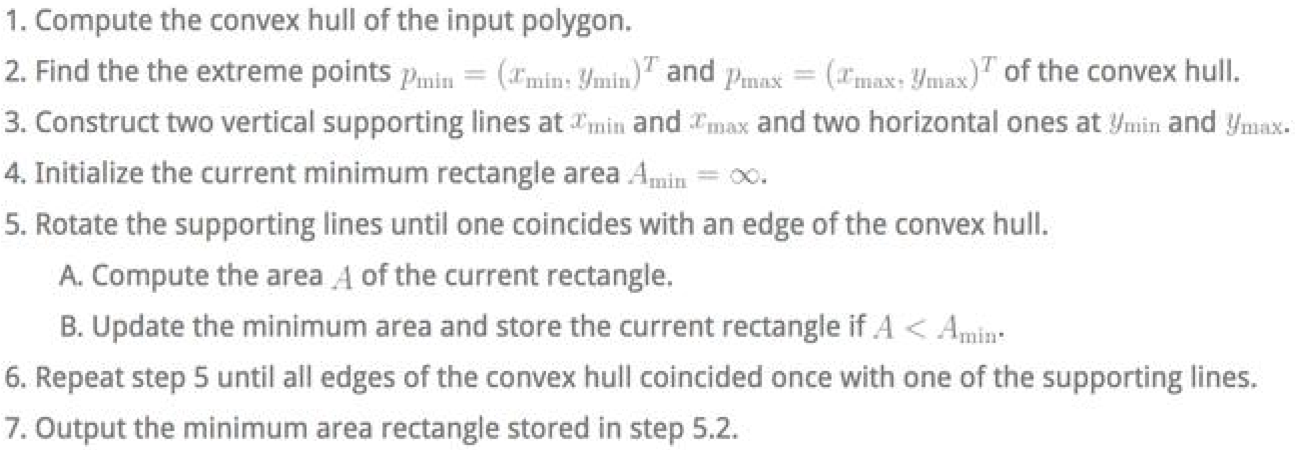
PCA Applications

- ✓ Line fit, plane fit.

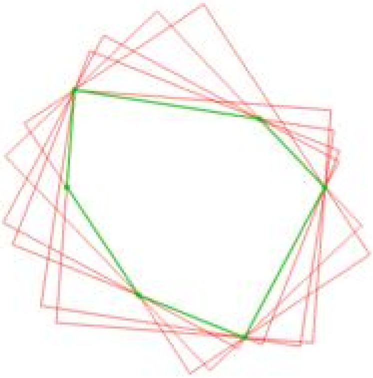


- ✓ Dimension reduction.

- ✓ Oriented bounding box (OBB) computation.
  - ✓ Rotating Calipers:

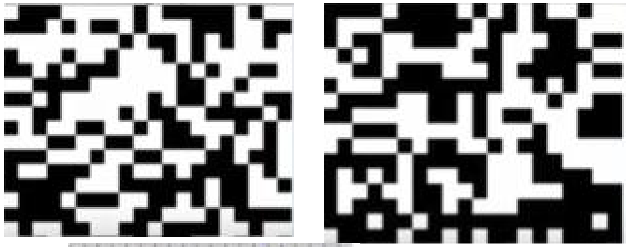


1. Compute the convex hull of the input polygon.
2. Find the the extreme points $p_{\min} = (x_{\min}, y_{\min})^T$ and $p_{\max} = (x_{\max}, y_{\max})^T$ of the convex hull.
3. Construct two vertical supporting lines at $x_{\min}$ and $x_{\max}$ and two horizontal ones at $y_{\min}$ and $y_{\max}$.
4. Initialize the current minimum rectangle area $A_{\min} = \infty$.
5. Rotate the supporting lines until one coincides with an edge of the convex hull.
   A. Compute the area A of the current rectangle.
   B. Update the minimum area and store the current rectangle if $A < A_{\min}$.
6. Repeat step 5 until all edges of the convex hull coincided once with one of the supporting lines.
7. Output the minimum area rectangle stored in step 5.2.

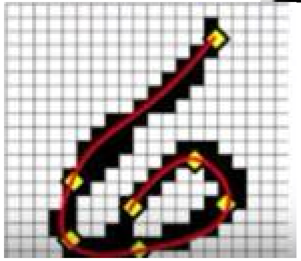


PCA Motivation

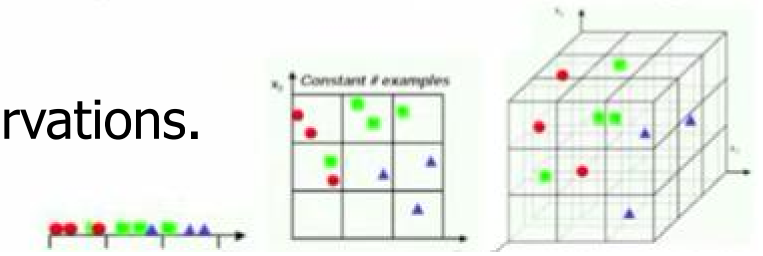
- ✓ A way to prevent Curse of Dimensionality.
- ✓ High dimensionality is a problem 'cos your space grows very quickly but the number of examples you have stays the same ➔ Redundancy.
- ✓ 20x20 bitmap to draw digits: 2^{400} dimensions but most of the configs will never be observed anyway, e.g.,



- ✓ True dimensionality is about how much variation I've when writing a digit, e.g.,



- ✓ Same # examples in 3 regions (1D), 3^2 regions, 3^3 regions, .. ➔ Sparse observations.



PCA Computation

- ✓ Compute center of mass m of n data points x1 to xn.
- ✓ Translate data points so that origin is at m: yi = xi – m for 1≤i≤n.
- ✓ Given matrix

$$Y = \begin{bmatrix} y1 & y2 & & yn \end{bmatrix}_{dxn}$$

(d=3 in general, but for shape synthesis each 3D vertex is concatenated into one big 3Vx1 column vector yi, yielding Y of size 3Vxn, where V = # of mesh vertices)

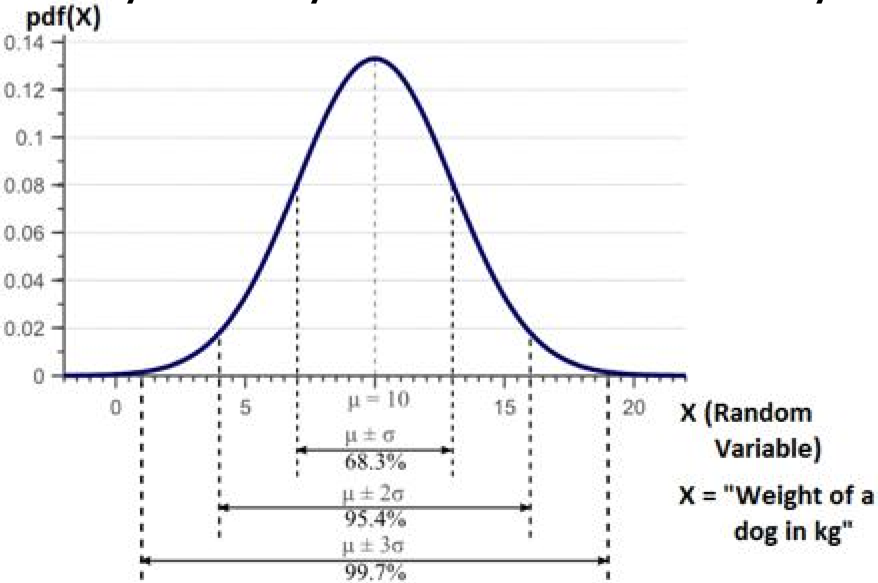
compute scatter (covariance) matrix $S = YY^T$ measuring the variance of points in different directions.

- ✓ Eigenvectors of S (in the descending order of associated eigenvalues) give the desired principal directions.

Variance vs. Covariance

- ✓ Variance represents the spread of the data points: $\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$
- ✓ Variance is the square of the standard deviation σ.

- ✓ If data is normally distributed, 68.3% of the observations/samples will have a value b/w $\mu - \sigma$ and $\mu + \sigma$, where μ is the mean of the data pts.
  - ✓ Hence, normally distributed data can be characterized by mean and variance.
- ✓ Here is a probability density function for normally distributed data:



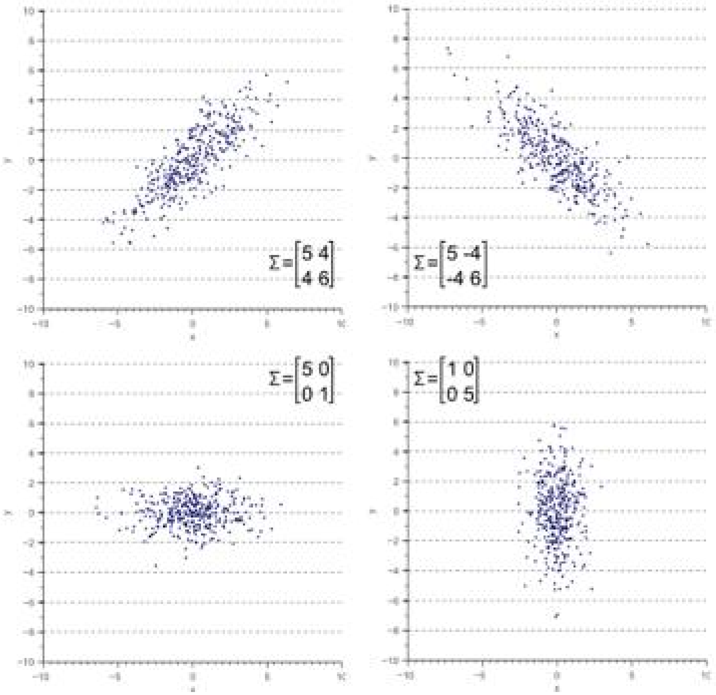
Variance vs. Covariance

- ✓ Variance represents the spread of the data points in axis-aligned directions, e.g., x-direction ($\sigma(x,x)$) or y-direction ($\sigma(y,y)$) for 2D data.

- ✓ However, horizontal and vertical spreads may not be enough, e.g., they do not explain the clear diagonal correlation below:
  - ✓ On average, if x increases, so does y; hence a positive correlation captured by $\sigma(x,y) = \mathbb{E}[(x-\mathbb{E}(x))(y-\mathbb{E}(y))]$, and summarized in this covariance matrix:

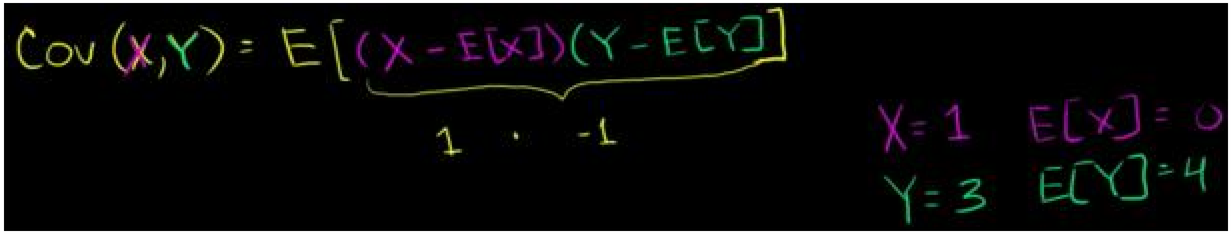
$$\Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix}$$

Symmetric matrix w/ variances on diagonal, covariances off-diagonal.



Variance vs. Covariance

- ✓ Covariance b/w 2 attributes (X and Y here) is an indication of whether they change together or in opposite directions.
- ✓ $\sigma(x,y) = \mathbb{E}[(x-\mathbb{E}(x))(y-\mathbb{E}(y))]$ $\quad \Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix}$

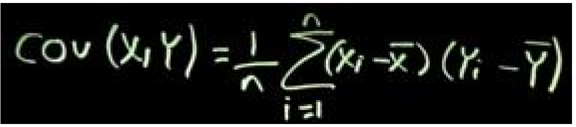


$$\mathrm{Cov}(X,Y) = E[\underbrace{(X-E[X])(Y-E[Y])}_{1 \cdot -1}]$$

$X = 1 \quad E[X] = 0$

$Y = 3 \quad E[Y] = 4$

- ✓ X and Y are my random variables. Take one point sample w/ X=1, Y=3.
- ✓ I know their means in advance: E[X]=0, E[Y]=4.
- ✓ For this sample, X is above average, but Y is below. So, it looks like as X is growing (above mean), Y is sinking. If this pattern continues for all the samples (sum), we have a negative covariance b/w X and Y.

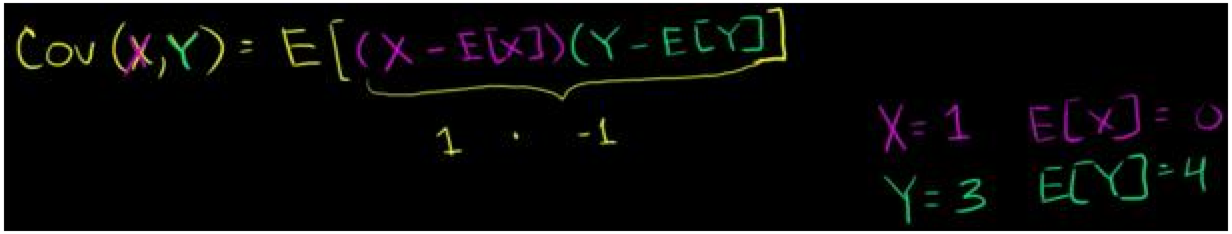


$$\mathrm{Cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})$$

Variance vs. Covariance

- ✓ Covariance b/w 2 attributes (X and Y here) is an indication of whether they change together or in opposite directions.

- ✓ $\sigma(x, y) = \mathbb{E}[(x - \mathbb{E}(x))(y - \mathbb{E}(y))]$ $\Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix}$

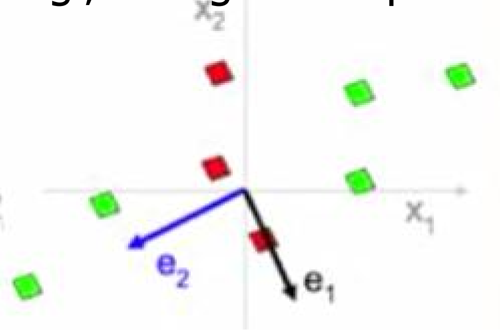


$$\mathrm{Cov}(X,Y) = E[\underbrace{(X - E[X])(Y - E[Y])}_{1 \ \cdot \ -1}]$$

$X = 1$ $E[X] = 0$

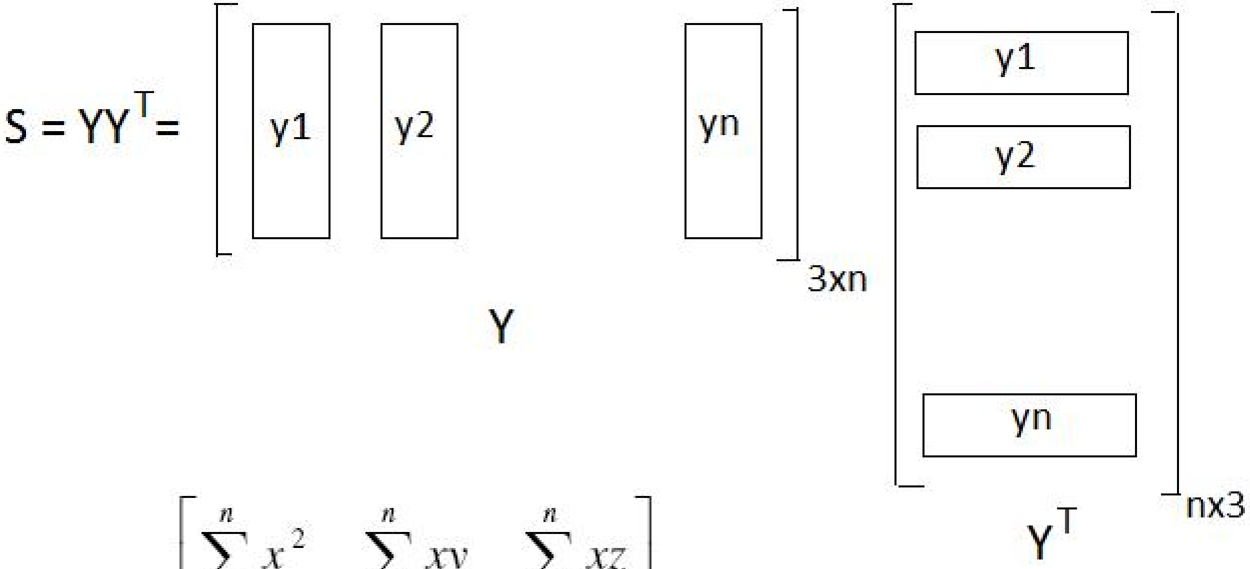
$Y = 3$ $E[Y] = 4$

- ✓ More on intuition: green and red pnts contribute +vely and –vely, resp.
  - ✓ Mean at the origin; e.g., for rightmost pnt x1 & x2 higher than the means 0, 0.



Variance vs. Covariance

- ✓ 2D data is fully explained by its mean and 2x2 covariance matrix.
- ✓ 3D data is fully explained by its mean and 3x3 covariance matrix.



✓

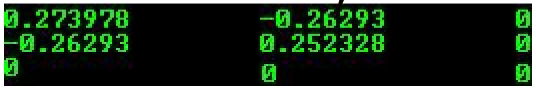
$$S = YY^T = \begin{bmatrix} \sum_{i=1}^{n} x^2 & \sum_{i=1}^{n} xy & \sum_{i=1}^{n} xz \\ \sum_{i=0}^{n} xy & \sum_{i=1}^{n} y^2 & \sum_{i=1}^{n} yz \\ \sum_{i=1}^{n} xz & \sum_{i=1}^{n} yz & \sum_{i=1}^{n} z^2 \end{bmatrix}$$

Recall that origin (0,0) is at center of mass of the data points, hence mean x and mean y are both 0.

$$\mathrm{cov}(X,Y) = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})$$

Variance vs. Covariance

- ✓ 2D data is fully explained by its mean and 2x2 covariance matrix.
- ✓ 3D data is fully explained by its mean and 3x3 covariance matrix.
- ✓ Example: variance b/w x-coordinates = .27, y-coordinates = .25, and covariance b/w x- and y-coordinates = -.26 here: (no variance between z-coordinates).



```
0.273978         -0.26293          0
-0.26293         0.252328          0
0                0                 0
```

$$S = YY^{T} = \begin{bmatrix} \sum_{i=1}^{n} x^{2} & \sum_{i=1}^{n} xy & \sum_{i=1}^{n} xz \\ \sum_{i=0}^{n} xy & \sum_{i=1}^{n} y^{2} & \sum_{i=1}^{n} yz \\ \sum_{i=1}^{n} xz & \sum_{i=1}^{n} yz & \sum_{i=1}^{n} z^{2} \end{bmatrix}$$

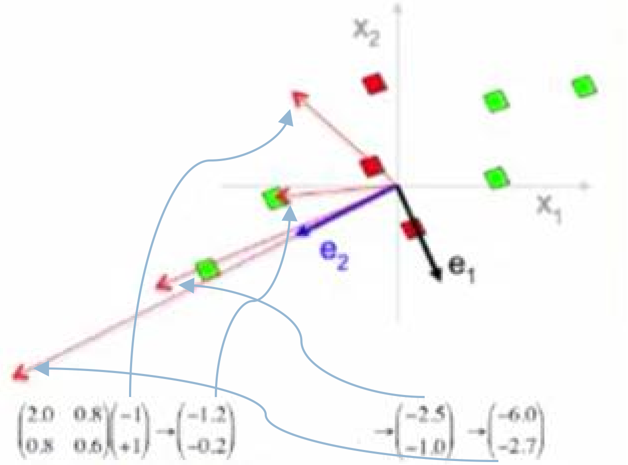
Eigendecomposition of Covariance Matrix

- ✓ Recall that eigenvectors of the covariance matrix S give the desired principal directions of variations, i.e., principal components (Slide 15).
- ✓ Eigenvalues represent the variance of data along the corresponding eigenvector directions; so pick the components w.r.t. eigenvalues.

- ✓ In 3D, all eigenvalues similar → no preferred direction, i.e., sphere.
- ✓ In 3D, one tiny eigenvalue → data sampled on a plane whose normal is that tiny eigenvalue (this is how we fit planes to 3D data).
- ✓ In 2D, all eigenvalues similar → circle.
- ✓ In 2D, one tiny eigenvalue → line whose direction is the big eigenvalue (this is how we fit lines to 2D data).

Eigendecomposition of Covariance Matrix

- ✓ Why do the eigenvectors of the covariance matrix S give the desired principal directions of variations, i.e., principal components (Slide 15)?

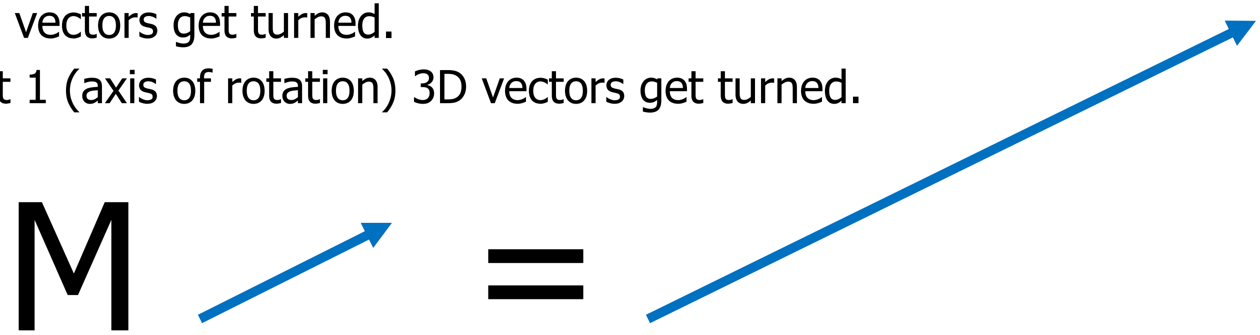


$$\begin{pmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{pmatrix}\begin{pmatrix} -1 \\ +1 \end{pmatrix} \rightarrow \begin{pmatrix} -1.2 \\ -0.2 \end{pmatrix} \rightarrow \begin{pmatrix} -2.5 \\ -1.0 \end{pmatrix} \rightarrow \begin{pmatrix} -6.0 \\ -2.7 \end{pmatrix}$$

- ✓ Take any vector, e.g., $[-1\ 1]^T$, transform it via S, get $[-1.2\ -0.2]^T$, repeat.
- ✓ Turns out, multiplying by S spins the initial vector towards direction of the greatest variance. It is also scaled along the way but it's OK.
- ✓ So, look for vectors that do not get turned when multiplied by S, hence look for eigenvectors of S.
- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue: Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ.

Eigenvector Pop Quiz

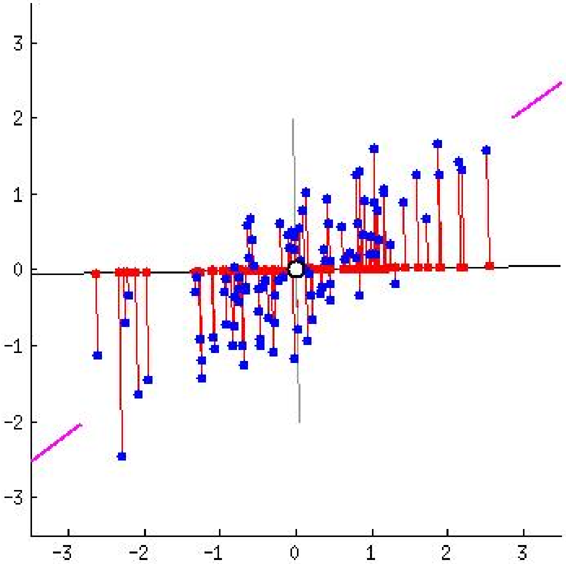
- How many eigenvectors does a 2D reflection matrix have, and geometric meaning?
  - 2: along the reflection line and perpendicular to the reflection line.
- Same for i) 3D uniform scaling mtrx? ii) 2D rotation mtr? iii) 3D rot. m?
  - Infinitely many: every vector preserves direction and changes magnitude after uniform scaling matrix is applied.
  - 0: all 2D vectors get turned.
  - 1: all but 1 (axis of rotation) 3D vectors get turned.

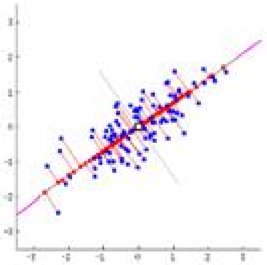


- Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue: Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ. In the example above, $\lambda = 3$.

PCA Summary

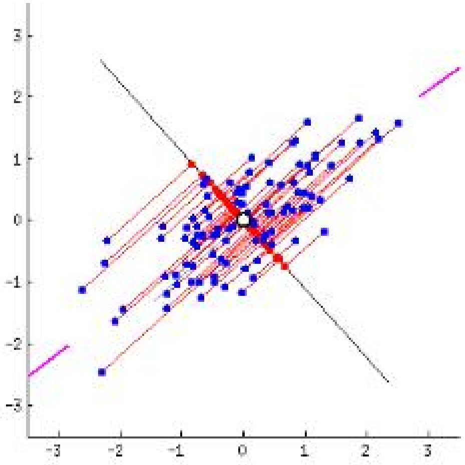
- ✓ PCA finds the best possible characteristics, best summary of data.
- ✓ Best: maximum variance/spread occurs along the principal direction.
- ✓ Best: minimum error (total red line length) along the principal directin.

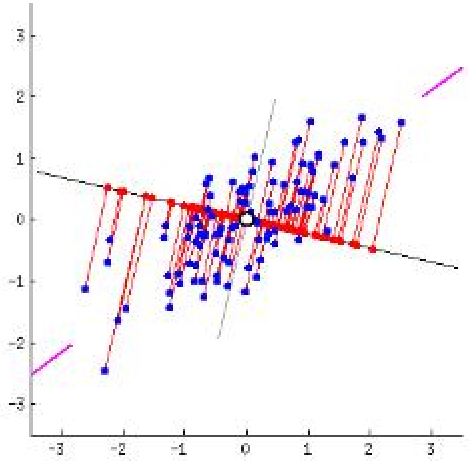


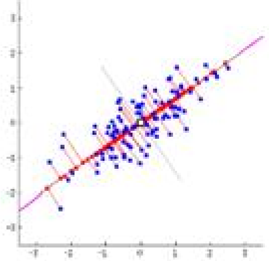


PCA Summary

- Why does max variance give min error, or vice versa?
- Variance: average squared distance from center to each red dot.
- Error: average squared red line lengths.
- Average squared distance from center to each blue dot is constant.
- Constant = Variance + Error by Pythagoras Theorem.
  - So the higher the variance the lower the error, viceversa (to keep the sum constant).

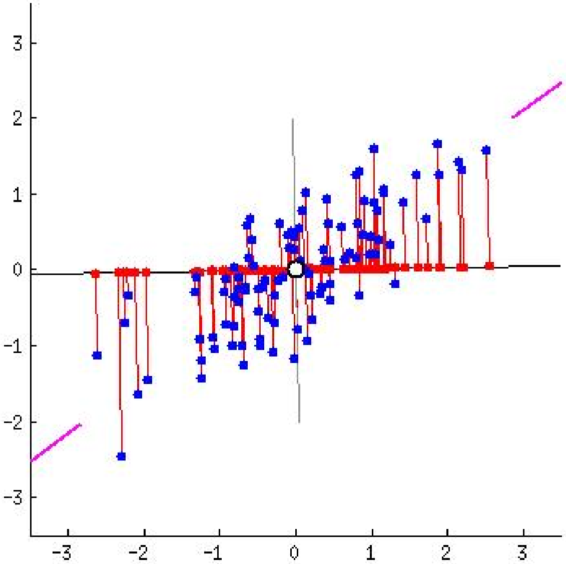


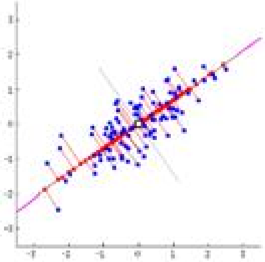




PCA Summary

- ✓ PCA constructs this cool principal direction which provides the max variance, min error (and the other orthonormal directions, which are the next principals).



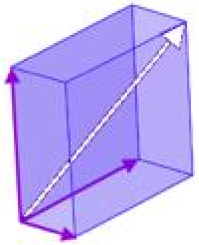


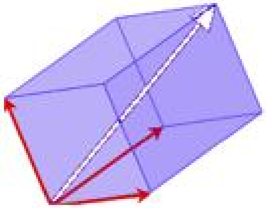
Linear Algebra w/ Geometric Perspective

- ✓ For a geometric interpretation of eigenstuff, let’s visualize what matrices do: they represent linear transformations.
  - ✓ v’ = Mv is the linearly transformed version of v.
  - ✓ Rotation, scaling, reflection, shear are common linear transforms.
  - ✓ A transformation is linear if lines remain lines & origin remains fixed

Basis

- ✓ For a geometric interpretation of eigenstuff, let's visualize what matrices do: they represent linear transformations.
  - ✓ v' = Mv is the linearly transformed version of v.
  - ✓ Rotation, scaling, reflection, shear are common linear transforms.
  - ✓ A transformation is linear if lines remain lines & origin remains fixed
- ✓ Any matrix can be analyzed by studying its operation on the basis vectors of the space, e.g., standard basis: **i**=(1,0,0), **j**=(0,1,0),**k**=0,0,1
- ✓ Basis: linearly independent set of vectors that span the vector space.



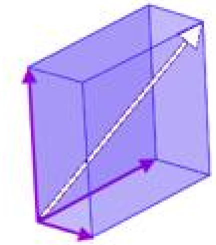


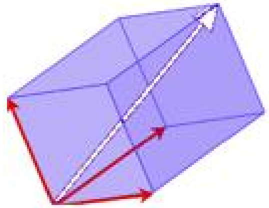
  - ✓ Any vector in vector space (white) can be represented as a linear combination of basis vectors (purple set, or red set).
  - ✓ A good basis often consists of orthogonal vectors, e.g., **i**, **j**, **k**.
  - ✓ Orthogonality implies linear independence but not vice versa.
  - ✓ Orthonormality: 2 vectors are orthogonal and unit (length=1).

$$\|\mathbf{a}\| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2} = \sqrt{\mathbf{a}\cdot\mathbf{a}}$$

Basis

- For a geometric interpretation of eigenstuff, let's visualize what matrices do: they represent linear transformations.
  - v' = Mv is the linearly transformed version of v.
  - Rotation, scaling, reflection, shear are common linear transforms.
  - A transformation is linear if lines remain lines & origin remains fixed
- Any matrix can be analyzed by studying its operation on the basis vectors of the space, e.g., standard basis: **i**=(1,0,0), **j**=(0,1,0),**k**=0,0,1
- Basis: linearly independent set of vectors that span the vector space.



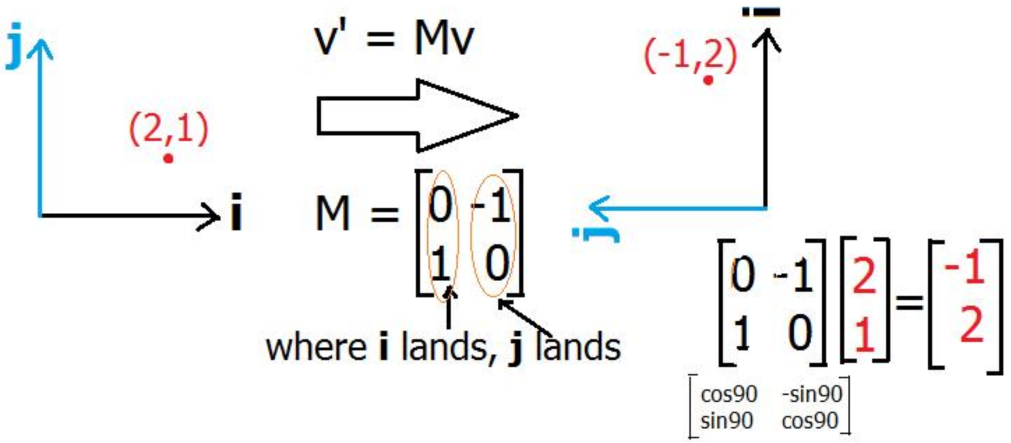


  - Any vector in vector space (white) can be represented as a linear combination of basis vectors ($v = a_1 v_1 + a_2 v_2 + \ldots + a_n v_n$).
  - A good basis often consists of orthogonal vectors, e.g., **i**, **j**, **k**.
  - Orthogonality implies linear independence but not vice versa.
  - Orthonormality: 2 vectors are orthogonal and unit (length=1).

$$\|\mathbf{a}\| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2} = \sqrt{\mathbf{a} \cdot \mathbf{a}}$$

Matrix

- ✓ For a geometric interpretation of eigenstuff, let’s visualize what matrices do: they represent linear transformations.
  - ✓ v’ = Mv is the linearly transformed version of v. M: linear operator.
  - ✓ Rotation, scaling, reflection, shear are common linear transforms.
  - ✓ A transformation is linear if lines remain lines & origin remains fixed
- ✓ Any matrix can be analyzed by studying its operation on the basis vectors of the space, e.g., standard basis: **i**=(1,0,0), **j**=(0,1,0),**k**=0,0,1
  - ✓ Columns of M are the landing points of basis vectors of the space.



v' = Mv

$$M = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

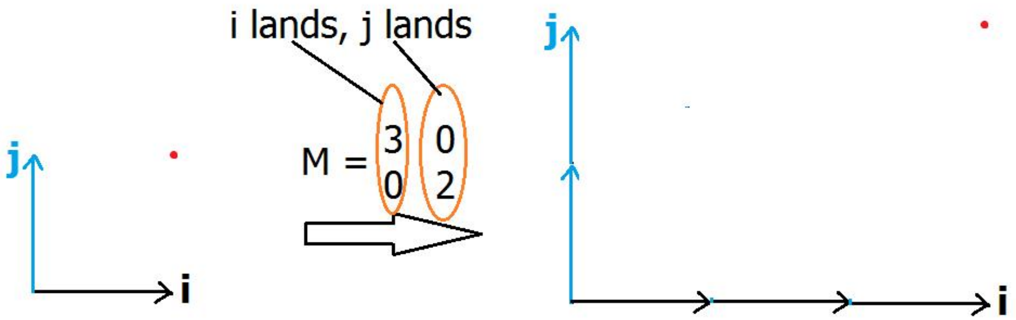
where **i** lands, **j** lands

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix}$$

Matrix

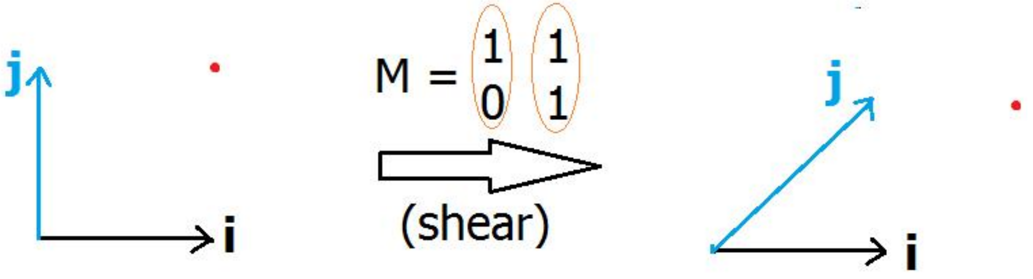
- ✓ For a geometric interpretation of eigenstuff, let’s visualize what matrices do: they represent linear transformations.
  - ✓ v’ = Mv is the linearly transformed version of v. M: linear operator.
  - ✓ Rotation, scaling, reflection, shear are common linear transforms.
  - ✓ A transformation is linear if lines remain lines & origin remains fixed
- ✓ Any matrix can be analyzed by studying its operation on the basis vectors of the space, e.g., standard basis: **i**=(1,0,0), **j**=(0,1,0),**k**=0,0,1
  - ✓ Columns of M are the landing points of basis vectors of the space.



  - ✓ det(M) = 6 is the scale factor by which areas are transformed by M.

Matrix

- For a geometric interpretation of eigenstuff, let’s visualize what matrices do: they represent linear transformations.
  - v’ = Mv is the linearly transformed version of v. M: linear operator.
  - Rotation, scaling, reflection, shear are common linear transforms.
  - A transformation is linear if lines remain lines & origin remains fixed
- Any matrix can be analyzed by studying its operation on the basis vectors of the space, e.g., standard basis: **i**=(1,0,0), **j**=(0,1,0),**k**=0,0,1
  - Columns of M are the landing points of basis vectors of the space.

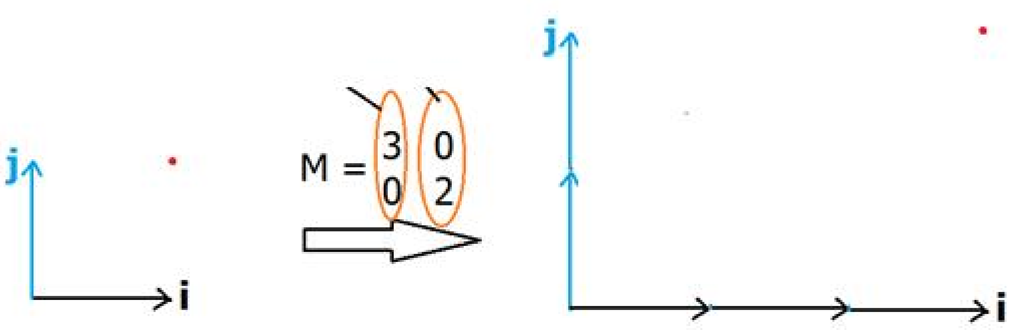


Eigenvectors and Eigenvalues

- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue.
- ✓ $Mv = \lambda v$, where v is an eigenvector of M w/ the eigenvalue λ.
- ✓ Eigendecomposition: $M = V\Lambda V^T$ (aka Spectral Decomposition).

Eigenvectors and Eigenvalues

- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue.
- ✓ Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ.
- ✓ 2 eigenvectors for this M: $[1\ 0]^T$ and $[0\ 1]^T$ w/ eigvals 3 and 2, resptvly



$$\begin{matrix}3 & 0\\ 0 & 2\end{matrix}\quad\begin{matrix}1\\ 0\end{matrix} = 3\ \begin{matrix}1\\ 0\end{matrix}\qquad\qquad\begin{matrix}3 & 0\\ 0 & 2\end{matrix}\quad\begin{matrix}0\\ 1\end{matrix} = 2\ \begin{matrix}0\\ 1\end{matrix}$$

Any vector in eigvector direction, e.g., $[9\ 0]^T$, is still an eigenvector w/ the same eigenvalue; ordinarily we consider eigenvectors as unit vectors.

Eigenvectors and Eigenvalues

- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue.
- ✓ Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ.
- ✓ Uniform scale by 2 has all vectors as eigvectors but one eigvalue of 2.
- ✓ Eigenvalue can be visualized as stretch amount of the original vector.

$$\begin{matrix}2 & 0\\ 0 & 2\end{matrix} \quad \begin{matrix}1\\ 0\end{matrix} \; = 2 \; \begin{matrix}1\\ 0\end{matrix}$$

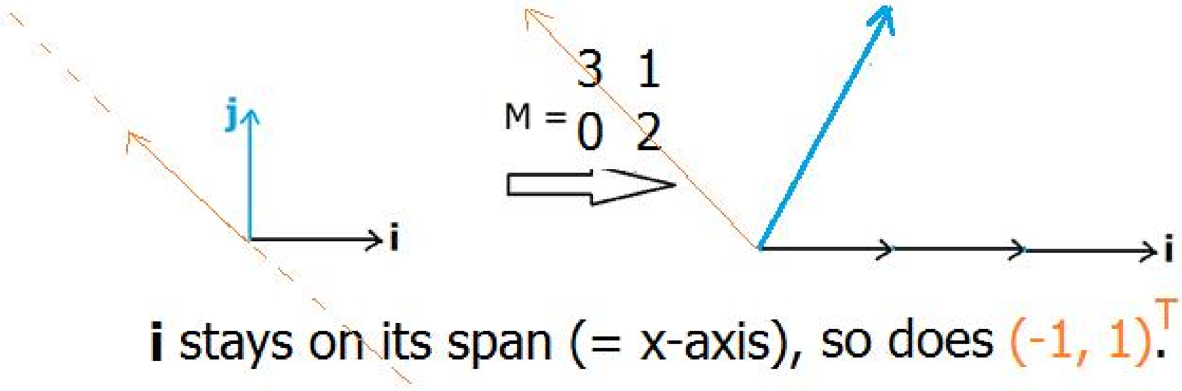
$$\begin{matrix}2 & 0\\ 0 & 2\end{matrix} \quad \begin{matrix}9\\ 5\end{matrix} \; = 2 \; \begin{matrix}9\\ 5\end{matrix}$$

Eigenvectors and Eigenvalues

✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue.

✓ Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ.

✓ 2 eigenvectors for this M: $[1\ 0]^T$ and $[-1\ 1]^T$ w/ eigvls 3 and 2, resptvly



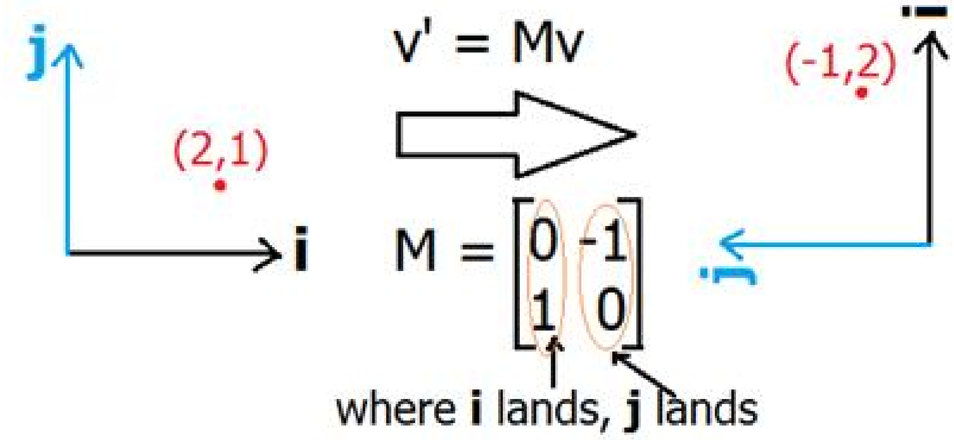
i stays on its span (= x-axis), so does $(-1, 1)^T$.

$$\begin{matrix}3 & 1\\ 0 & 2\end{matrix}\quad \begin{matrix}1\\ 0\end{matrix} = 3\ \begin{matrix}1\\ 0\end{matrix} \qquad\qquad \begin{matrix}3 & 1\\ 0 & 2\end{matrix}\quad \begin{matrix}-1\\ 1\end{matrix} = 2\ \begin{matrix}-1\\ 1\end{matrix}$$

Eigenvectors and Eigenvalues

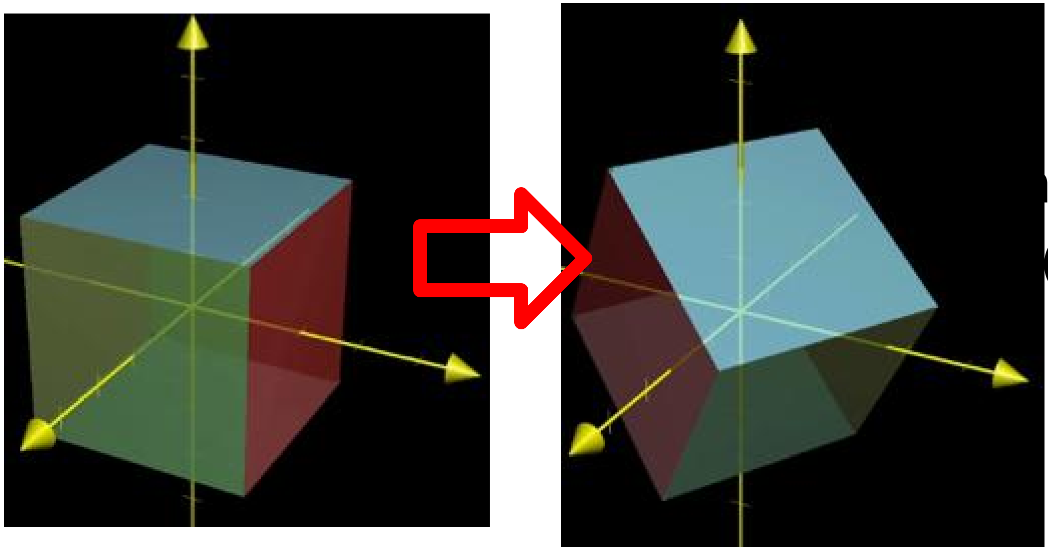
- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue.
- ✓ Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ.
- ✓ No eigenvectors for this M:

(2D rotation has no eigvecs ‘cos all vectors change direction.)



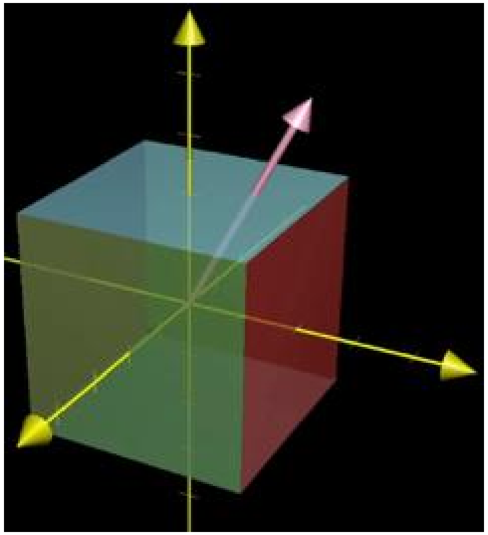
Eigenvectors and Eigenvalues

- ✓ Eigenvector of a transformation M is a vector that has not changed at all after transformation M except by a scalar known as the eigenvalue.
- ✓ Mv = λv, where v is an eigenvector of M w/ the eigenvalue λ.
- ✓ An eigenvector of 3D rotation (3x3 M) is the axis of rotation, as it stays on its span.
- ✓ Corresponding eigenvalue is 1 as rotations don’t stretch/squish objects.



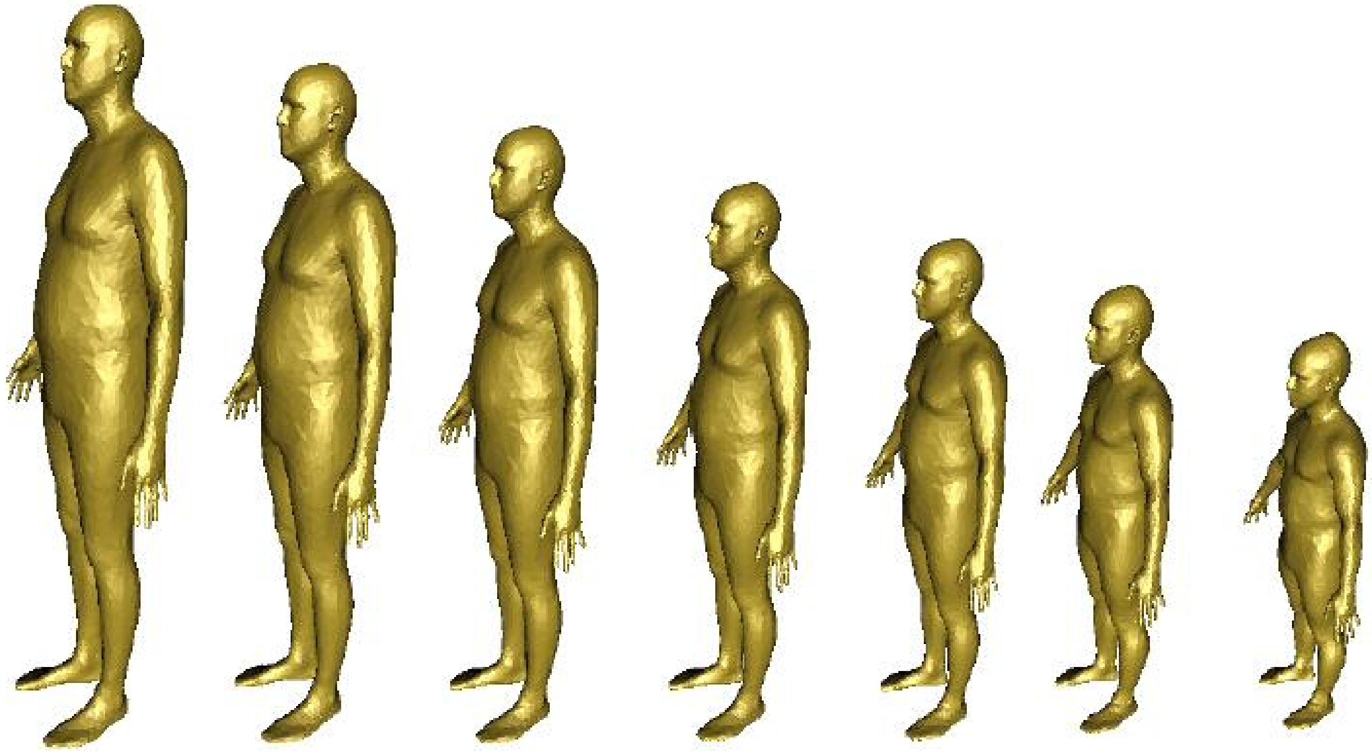
vector

(pink):



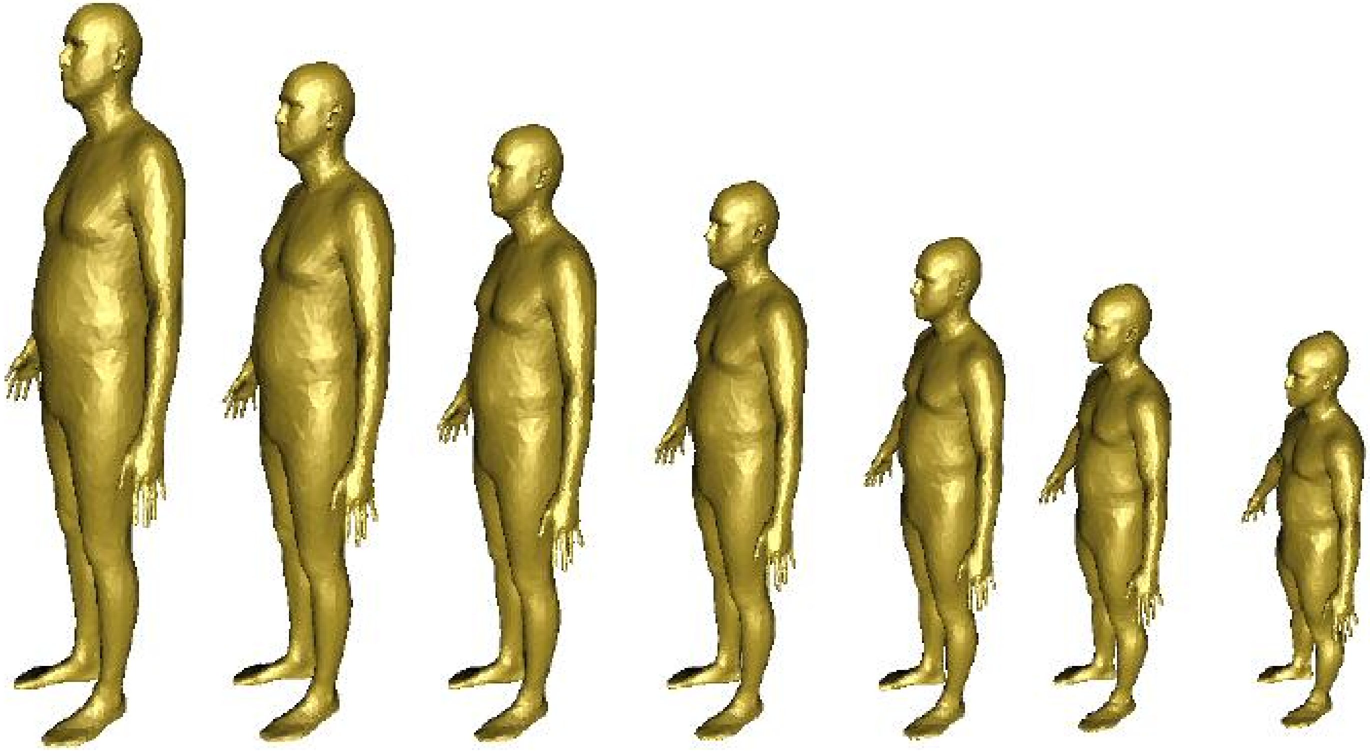
PCA-based Shape Synthesis

- ✓ Back to business after that algebraic detour.
- ✓ One of the advantages of using digital models is the simplicity to generate new models based on the seed models.
- ✓ Having learned the variation below, synthesize the next shape.



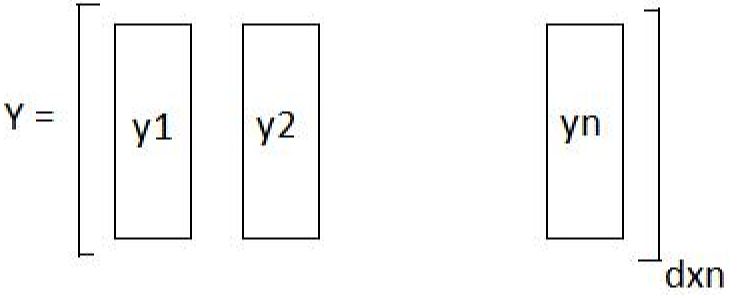
PCA-based Shape Synthesis

- ✓ Learn the shape space through PCA.
- ✓ Correlate semantic attributes and PCA coordinates.
- ✓ Create new shape based on new attributes.
  - ✓ No fancy devices/scanners, just simple keyboard inputs.



PCA Computation

- ✓ Compute center of mass m of n data points x1 to xn.
- ✓ Translate data points so that origin is at m: yi = xi – m for 1≤i≤n.
- ✓ Given matrix



(d=3 in general, but for shape synthesis each 3D vertex is concatenated into one big 3Vx1 column vector yi, yielding Y of size 3Vxn, where V = # of mesh vertices)

compute scatter (covariance) matrix $S = YY^T$ measuring the variance of points in different directions.

- ✓ Eigenvectors of S (in the descending order of associated eigenvalues) give the desired principal directions.

PCA Computation

- ✓ Compute center of mass m of n shapes x1 to xn. m: mean shape.
- ✓ Translate shapes so that origin is at m: yi = xi – m for 1≤i≤n.
- ✓ Given matrix

$$Y = \begin{bmatrix} \boxed{y1} & \boxed{y2} & & \boxed{yn} \end{bmatrix}_{dxn}$$

d=3V, typically V=12500 for a human shape:

compute scatter (covariance) matrix $S = YY^T$ measuring the variance of points in different directions.

- ✓ Eigenvectors of S (in the descending order of associated eigenvalues) give the desired principal directions.

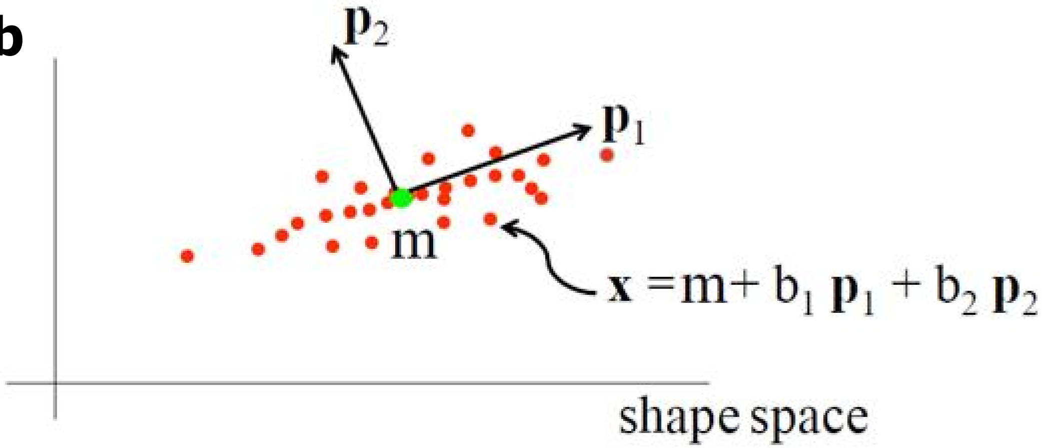
- ✓ The set of eigenvectors representing surface data in this manner is called an active shape model, or statistical shape model.

Active Shape Model

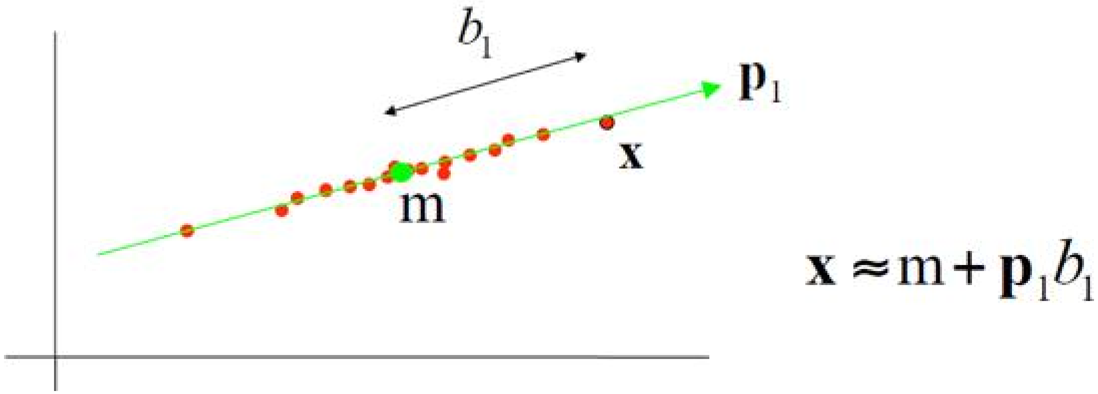
- ✓ The set of eigenvectors representing surface data in this manner is called an active shape model, or statistical shape model.
- ✓ m: mean shape.
- ✓ Linear model: x = m + **Pb**

$$\mathbf{x} = \mathrm{m} + \mathbf{Pb}$$

$$\mathbf{x} = \mathrm{m} + \mathrm{b}_1\,\mathbf{p}_1 + \mathrm{b}_2\,\mathbf{p}_2 + \ldots + \mathrm{b}_k\,\mathbf{p}_k$$

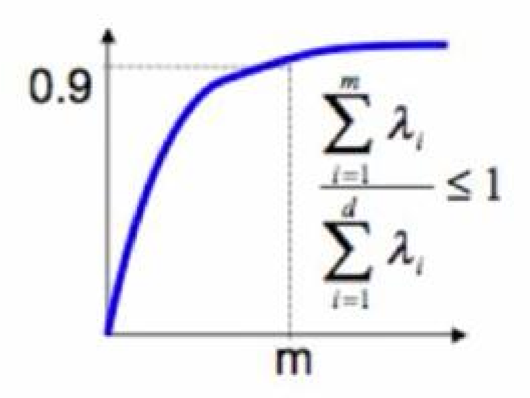

$\mathbf{p}_2$
$\mathbf{p}_1$
m
$\mathbf{x} = \mathrm{m} + \mathrm{b}_1\,\mathbf{p}_1 + \mathrm{b}_2\,\mathbf{p}_2$
shape space

- ✓ We don't lose much by approximating the new shape x using few dimensions.



PCA Computation

- ✓ How many principal components do we need?
- ✓ We want to capture as much variation as possible.
- ✓ Eigenvalue λ_i gives the amount of variation in the ith direction.



- ✓ One heuristic: pick the first m largest eigenvectors which explain 90+% of the total variance, i.e., sum of the first m eigenvalues over sum of all the eigenvalues is at least 0.9.

PCA Computation

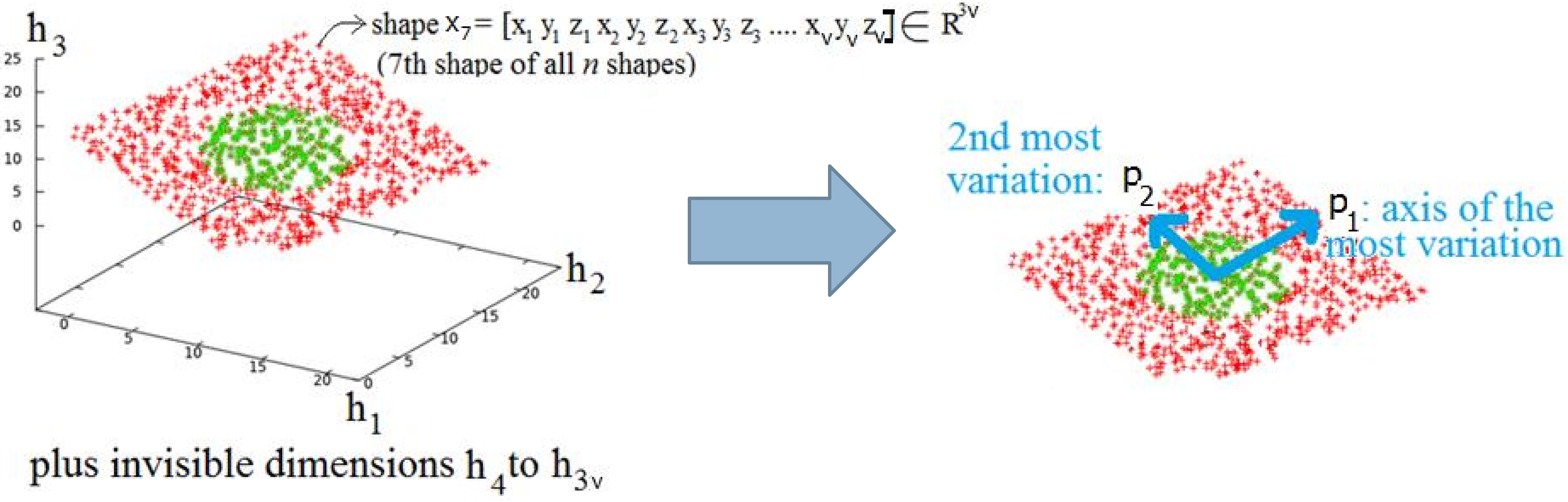
- ✓ $S = YY^T$ is huge (3V x 3V) → difficult to compute eigenvectors.
- ✓ Since there are only n samples (n << 3V) the rank of S is at most n-1 and therefore there are at most n-1 eigvectors w/ nonzero eigvalues.
- ✓ Karhunen-Loeve Transform (KLT) does the trick.

PCA Computation

- ✓ $S = YY^T$ is huge (3V x 3V) → difficult to compute eigenvectors.
- ✓ Since there are only n samples ($n << 3V$) the rank of S is at most n-1 and therefore there are at most n-1 eigvectors w/ nonzero eigvalues.
- ✓ Karhunen-Loeve Transform (KLT) does the trick:
  - ✓ Instead of computing eigvcs of YY^T, compute eigvcs of Y^TY (n x n).
  - ✓ Denote by p_i the eigenvectors of huge YY^T, and by q_i the eigvctors of small Y^TY; so we have $Y^TYq_i = \lambda_i q_i$
  - ✓ Multiply sides by Y → $YY^T(Yq_i) = (Y\lambda)_i q_i = \lambda_i(Yq)_i$
  - ✓ Thus, $p_i = Yq_i$ are the eigenvectors of YY^T
  - ✓ This means that we can compute eigenvectors of small Y^TY and multiply them by Y to get the desired eigenvectors p_i.
  - ✓ Note that each p_i is of size 3V x 1 and they are orthonormal.

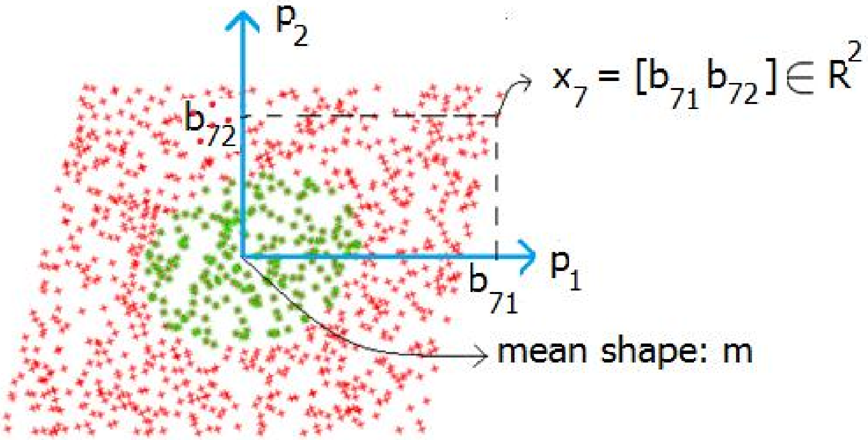
Correlation

✓ PCA constructs the principal axes of variations.

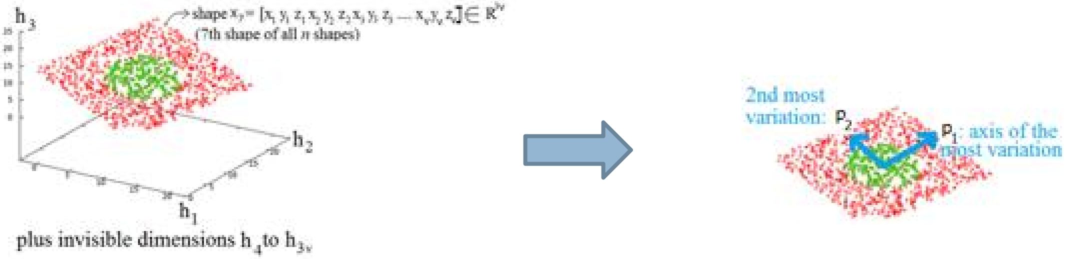

shape $x_7 = [x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3\ \ldots\ x_v y_v z_v] \in R^{3v}$
(7th shape of all *n* shapes)
h_3
25
20
15
10
5
0
h_2
0
5
10
15
20
h_1
0
5
10
15
20
plus invisible dimensions h_4 to h_{3v}
2nd most variation: p_2
p_1: axis of the most variation

Correlation

- ✓ PCA constructs the principal axes of variations, which enables a low dimensional representation to be correlated w/ semantic attributes.

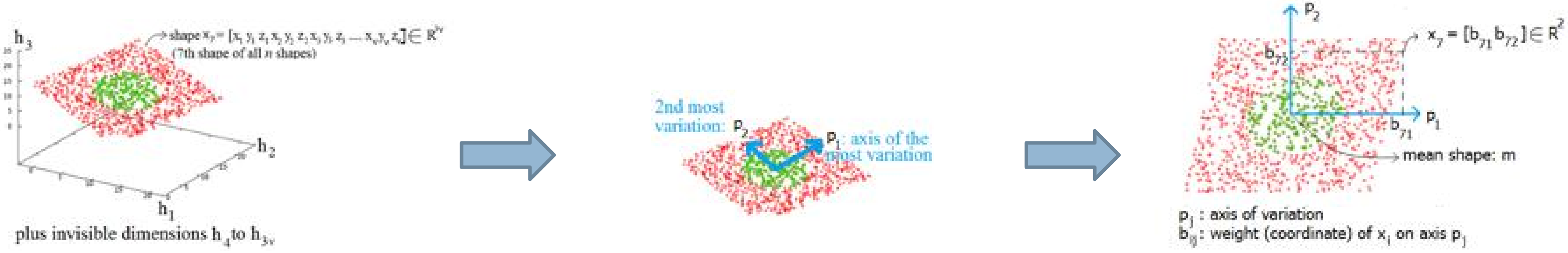


p_j : axis of variation
b_{ij} : weight (coordinate) of x_i on axis p_j


h_3
shape $x_7 = [x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3 \ldots x_v\ y_v\ z_v] \in R^{3v}$
(7th shape of all n shapes)
h_2
h_1
plus invisible dimensions h_4 to h_{3v}
2nd most variation: p_2
p_1: axis of the most variation

Correlation

- ✓ For shape x_i, correlation between semantically meaningful input data s_i = [kg, cm, ..]^T and shape's PCA coordinates b_{ij} is needed.
- ✓ Training: s_i is defined/precomputed for each shape x_i where $1 \leq i \leq n$.
- ✓ Learn correlation matrix C via linear mapping model.
  - ✓ More sophisticated learning by: Shared Gaussian Process Latent Variable Model.
- ✓ b = C s_i where b = [b_{i1} b_{i2} ..]^T for x_i (recall s_i = [kg, cm, ..]^T).
- ✓ We get n different C matrices which are then averaged into $\underline{C}$.
- ✓ Synthesize a new shape w/ the desired semantic attributes s_{new} via:
  - ✓ $b_{new} = \underline{C}\, s_{new}$


h_3
shape $x_7 = [x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3\ \dots\ x_v y_v z_v] \in R^{3v}$
(7th shape of all n shapes)
h_2
h_1
plus invisible dimensions h_4 to h_{3v}
2nd most variation: P_2
P_1: axis of the most variation
P_2
$x_7 = [b_{71}\ b_{72}] \in R^2$
b_{72}
P_1
b_{71}
mean shape: m
p_j : axis of variation
b_{ij} : weight (coordinate) of x_i on axis p_j

Correlation

✓ Synthesis based on the new PCA coordinates b_{new} is as follows:

$$\mathbf{x} = \mathrm{m} + \mathbf{Pb}$$
$$\mathbf{x} = \mathrm{m} + \mathrm{b}_1\,\mathbf{p}_1 + \mathrm{b}_2\,\mathbf{p}_2 + \ldots + \mathrm{b}_k\,\mathbf{p}_k$$

```
✓ $x_{new}$ = m;
  for each principal direction $p_j \in R^{3V}$
      for each dimension d in $b_{new}$
          x += $p_{j.d}$ * $b_{new.d}$ (where $b_{new.d}$ gives the d^th component of $b_{new}$).
```

✓ Right, assume p_1 represents variation in height (y-coord in mesh) and p_2 weight (z-coord). If $b_{new.1} = b_{new.2} = 0$, no change in mean shape. If $b_{new.1} = 0$, $b_{new.2} = 1$, then x_{new} is not stretched vertically (no change in y). Mean moves along z on some vertices (change in weight).

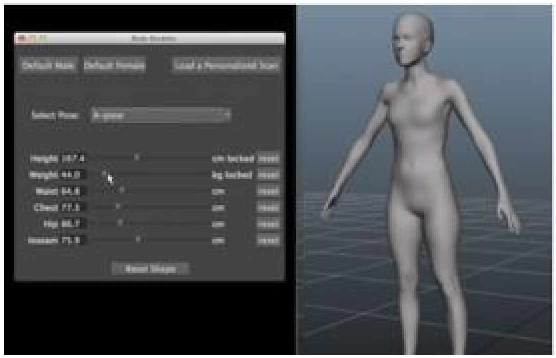
$$x_{new} = m + \underset{3V \text{ x } 2}{\begin{bmatrix} p_1 & p_2 \end{bmatrix}} \underset{2\text{x}1}{\begin{bmatrix} b_{new.1} \\ b_{new.2} \end{bmatrix}}$$

Shape to Mesh

- ✓ Once the new coordinates are computed via PCA coefficients, it is trivial to get the corresponding shape mesh: keep the connectivity of the mean mesh and change its geometry by the new coordinates.
- ✓ Note that all the seed shapes in the dataset must initially be in correspondence; this is, e.g., how we get the mean shape: by averaging the coordinates of each corresponding vertex.

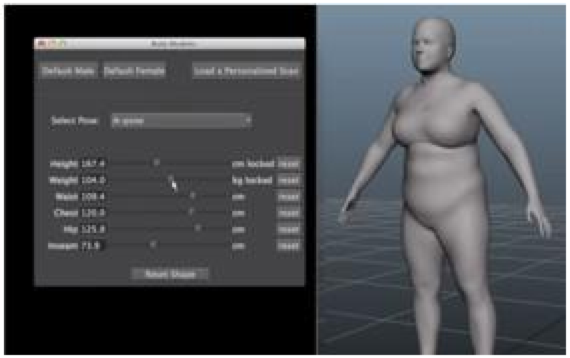
PCA is Global

- ✓ A global model may not provide sufficient flexibility/variety.



Select Pose: A-pose

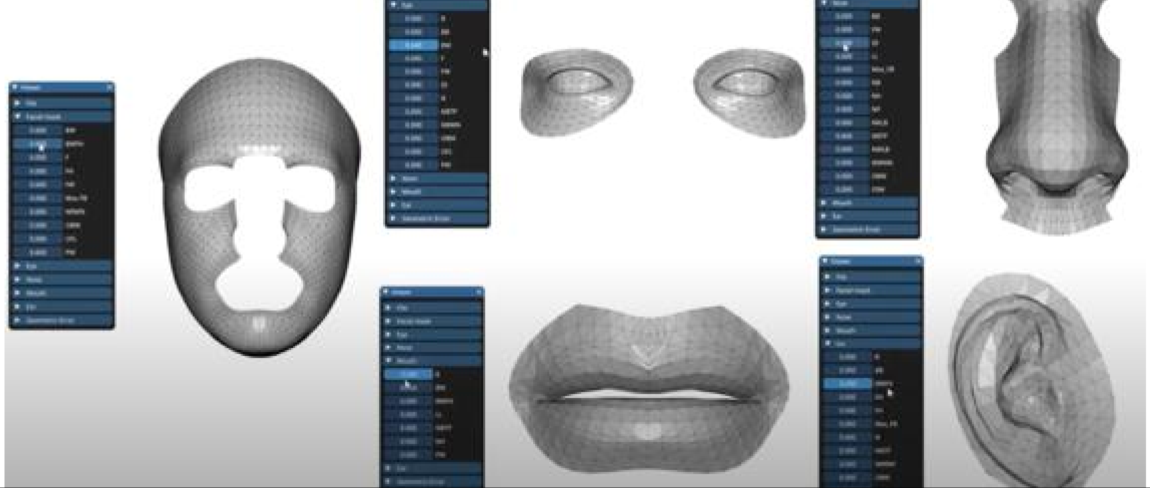
Height 167.4 cm
Weight 44.0 kg
Waist 64.6 cm
Chest 77.5 cm
Hip 85.7 cm
Inseam 71.9 cm



Select Pose: A-pose

Height 167.4 cm
Weight 104.0 kg
Waist 109.4 cm
Chest 120.0 cm
Hip 125.8 cm
Inseam 71.9 cm

- ✓ Part-based scheme for more intuitive modeling.
  - ✓ Part-Based 3D Face Morphable Model with Anthropometric Local Control, 2020.



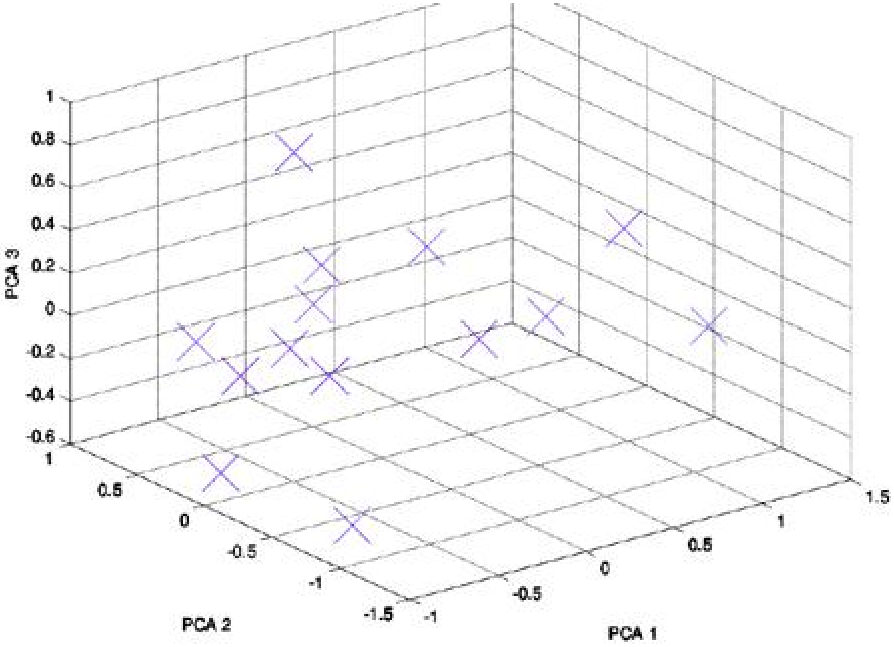
Another PCA App: Eigenfaces

- ✓ PCA for shape generation and recognition.
- ✓ Face images are highly redundant: all background pixels are the same, and each subject has the same facial features.

- ✓ Eigvcs (called eigenfaces in this context) to represent imgs compactly.
- ✓ Use this compact representatin to recognize (or create) faces efficiently.

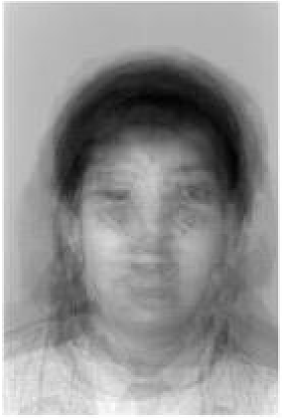
Another PCA App: Eigenfaces

- ✓ Use eigenvectors (called eigenfaces in this context) to represent images compactly, e.g., below: just 3 coefficients (weights on 3 principal axes).
- ✓ Use this compact representation, e.g., 14 images (X) below, to recognize faces efficiently.
  - ✓ Find the dimension coeffs for the query image and compare it w/ the X's below.

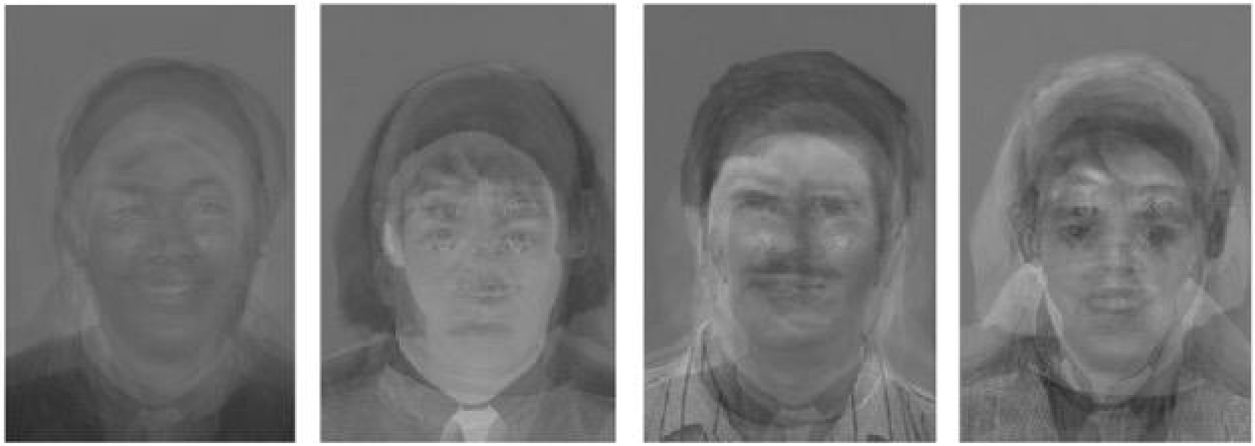


Another PCA App: Eigenfaces

✓ Mean:



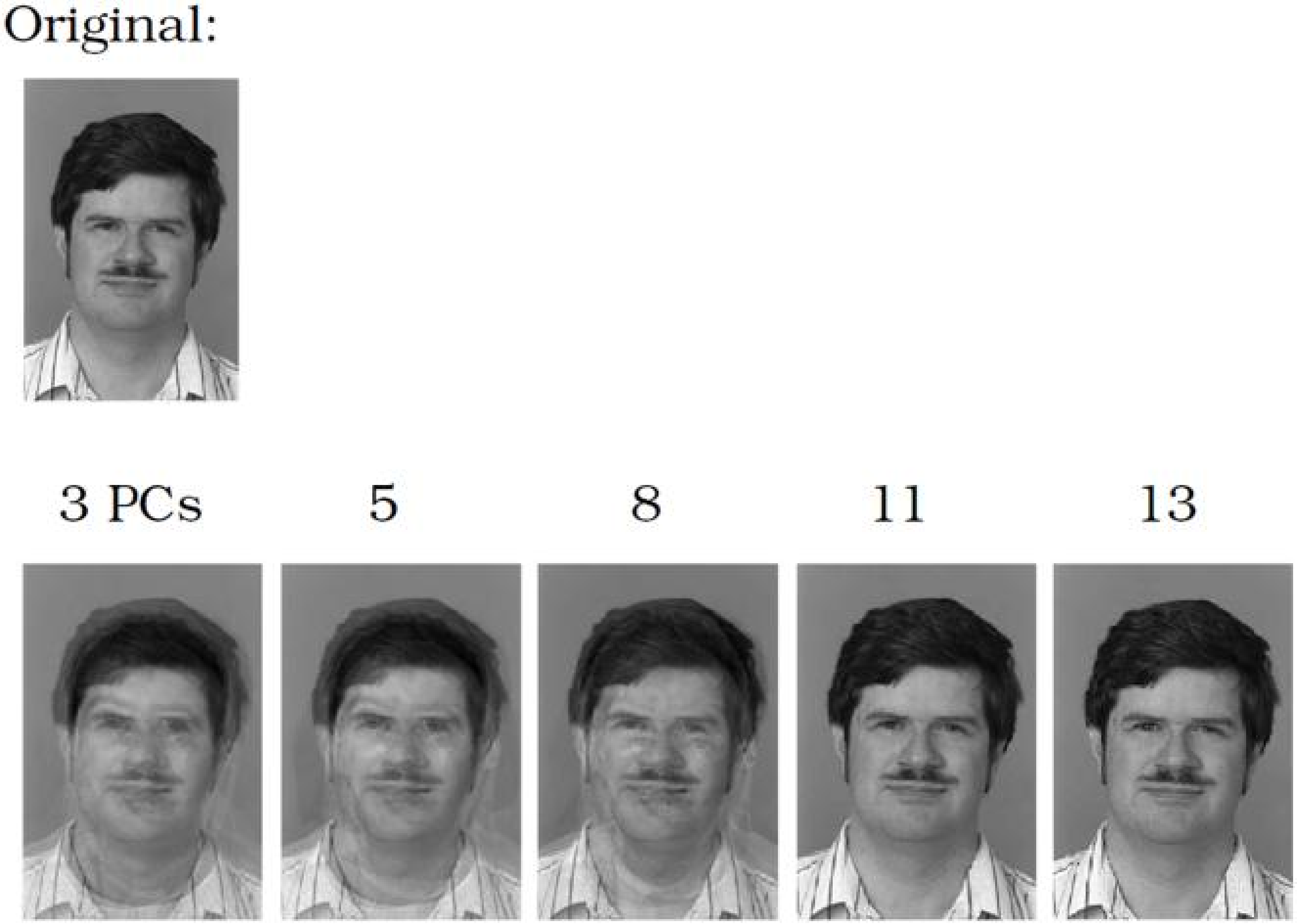
✓ Four eigenfaces with the largest eigenvalues:



✓ First one: most prominent deviations from the mean in this dataset.

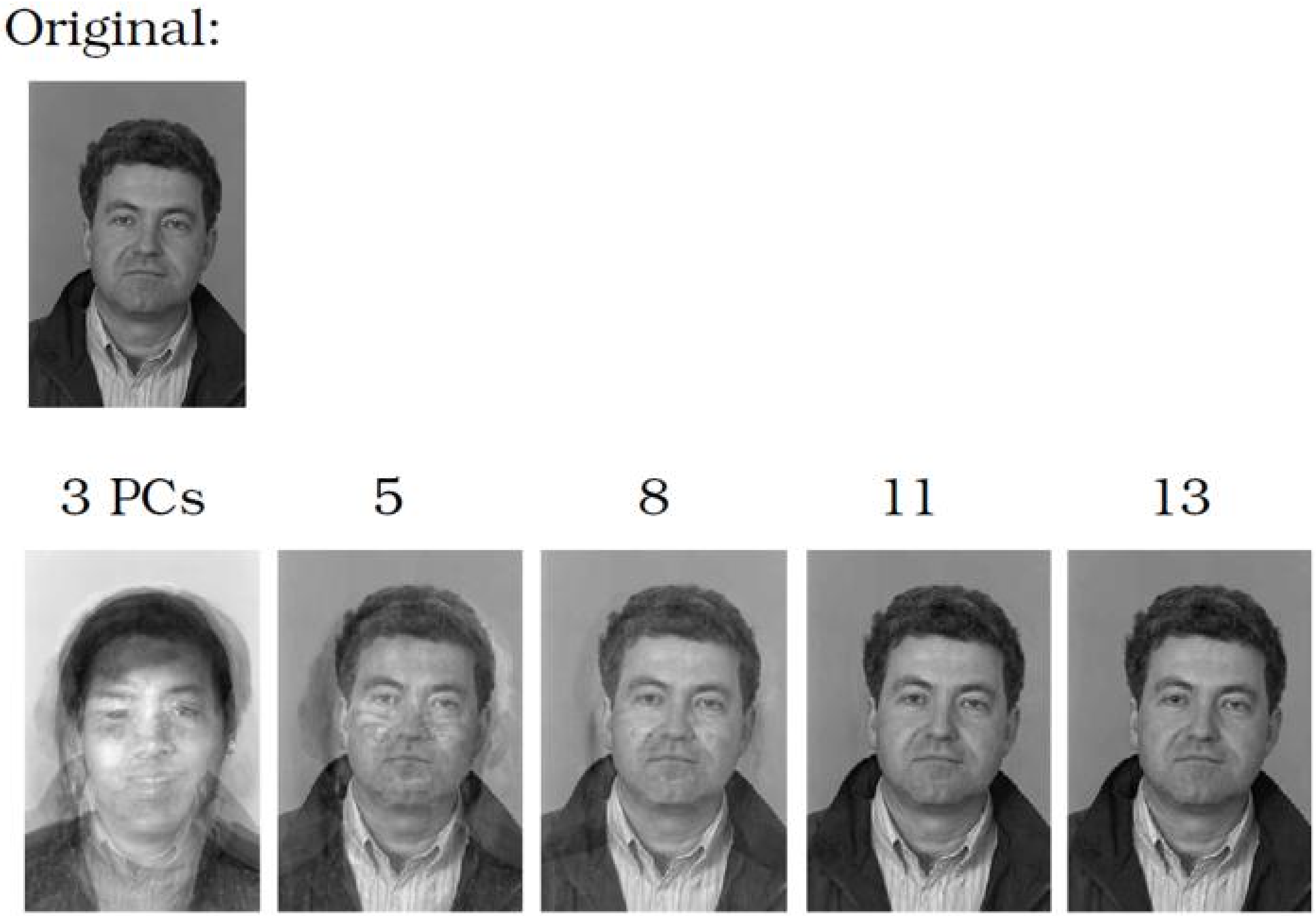
Another PCA App: Eigenfaces

- ✓ 3 principal components are still a close representation.



Another PCA App: Eigenfaces

✓ 5 principal components are still a close representation. Typically use 40.

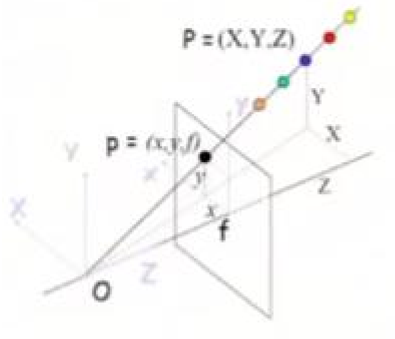


Shape from X

- ✓ Previously, we dealt with generation of meshes that represent shape surfaces from implicit scalar fields (in an effort to perform scientific visualization).
- ✓ Today, we learned shape generation by growing an existing set of seed meshes.
- ✓ A different paradigm to generate meshes, popular in Computer Vision, is based on scanning (active) and photogrammetry (passive).

Shape from Stereo

- ✓ A passive technique where we use two images of the real-world object to generate its digital mesh.
- ✓ Depth ambiguity w/ 1 img: which 3D point on ray OP is projected to p?

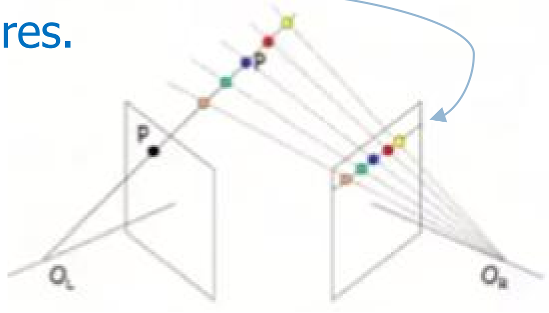


$$x = \frac{fX}{Z} = \frac{fkX}{kZ}$$

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

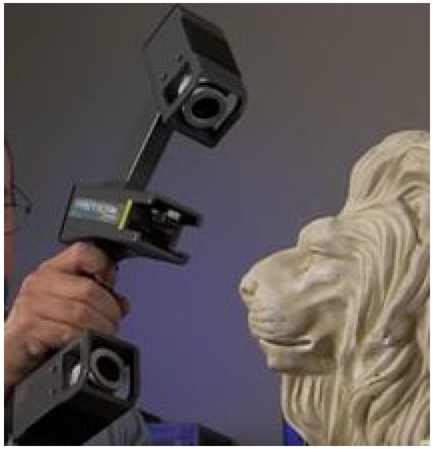
For any $k \neq 0$

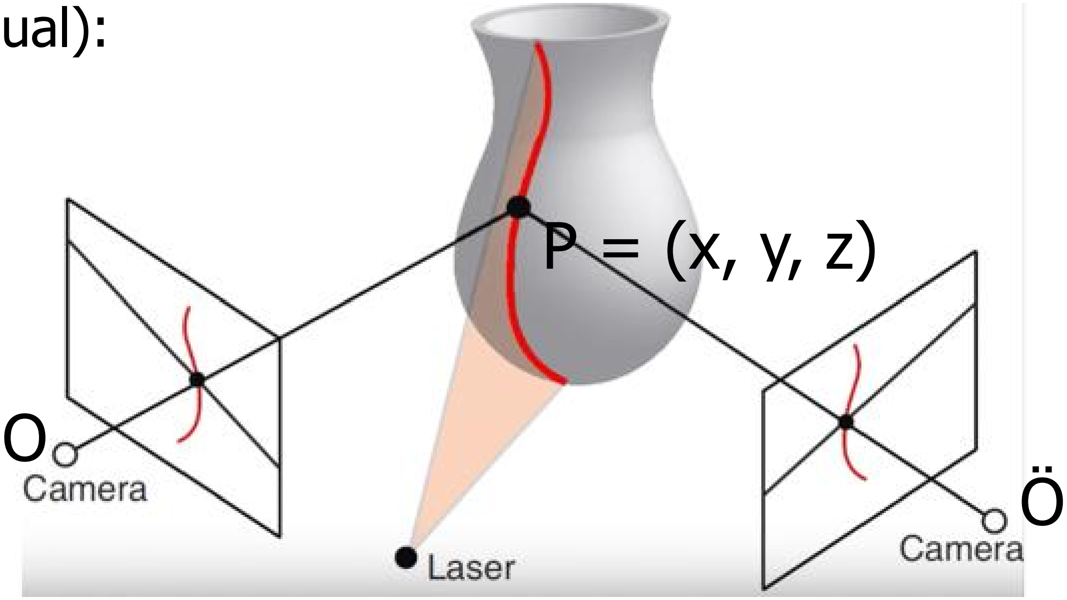
- ✓ Fixed: Find corresponding pixel at right. Intersect new 3D line w/ OP.
- ✓ Reduce your search to the epipolar line (projection of OP on right img).
  - ✓ HOG, SURF, SIFT features.



Shape from Structured Light

- ✓ An active technique where we actively send laser stripes (or other light structures/stripes) to the real-world object to generate its digital mesh.
- ✓ With two cameras (unusual):



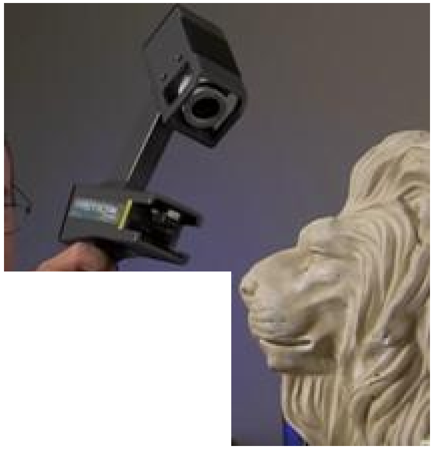


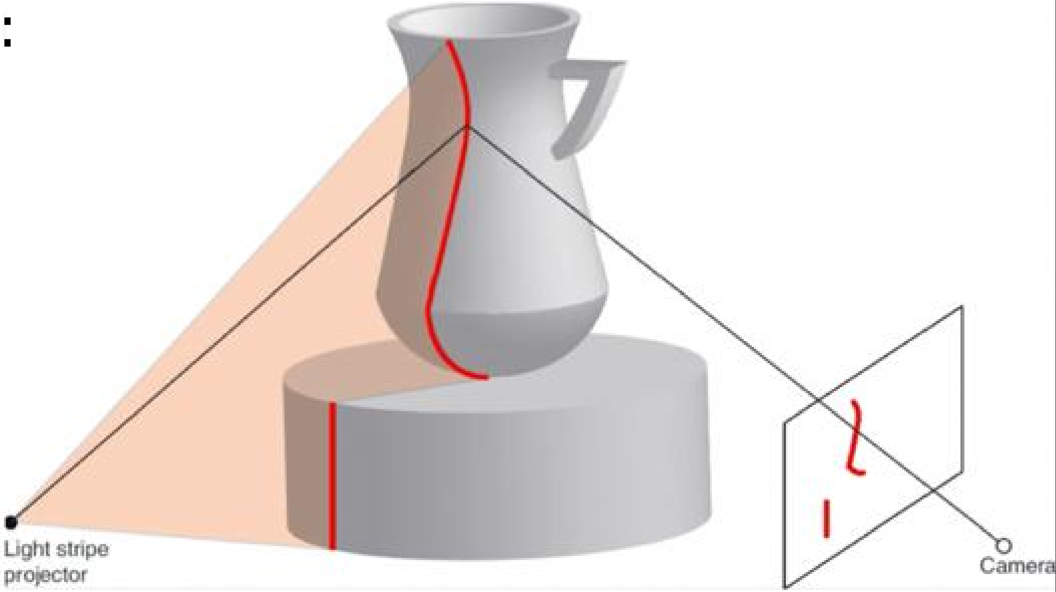
- ✓ Laser stripe is used for perfect correspondences (no HOG, .. features).
- ✓ Shine a laser stripe to object. P projects to a pixel on left. Epipolar line on right cam is intersected w/ stripe there, yielding corresponding pixel.
- ✓ Intersect OP w/ ÖP to get the desired P.

Shape from Structured Light

✓ An active technique where we actively send laser stripes (or other light structures/stripes) to the real-world object to generate its digital mesh.

✓ With one camera (common):



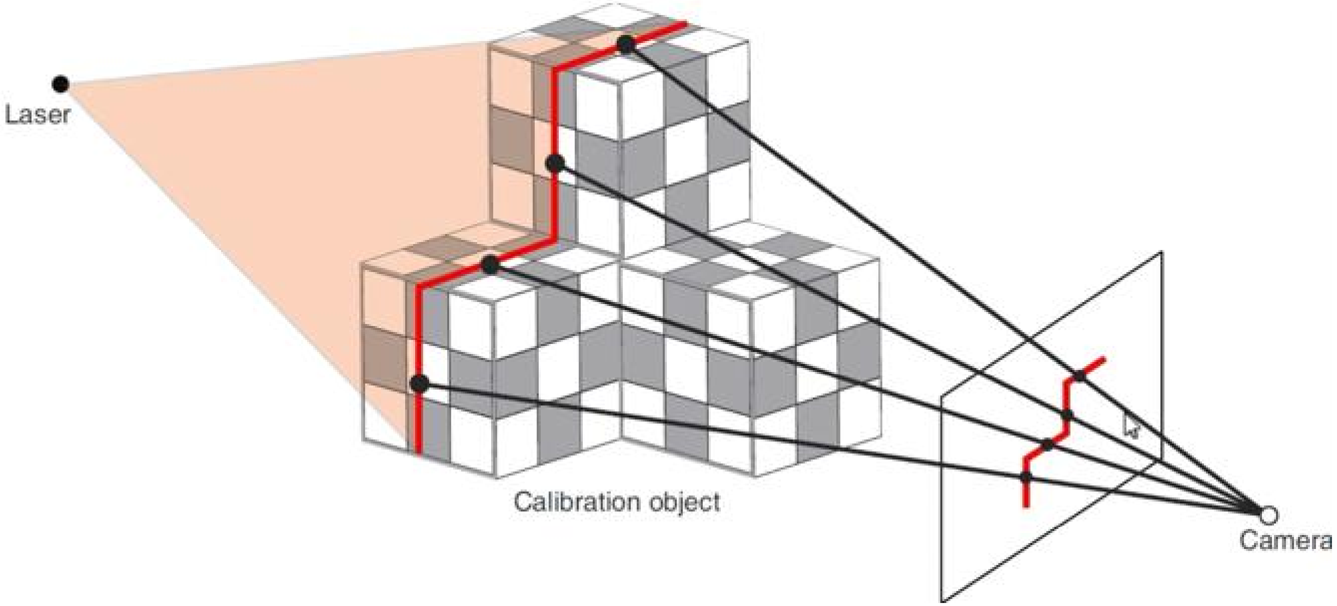


✓ Mapping from 2D image plane to 3D stripe plane (pink sheet) can be estimated via projective transformation (4 correspondences needed).

✓ When projector moves to scan a new slice, coupled camera moves together, preserving transform. params, and we get the next 3D slice.

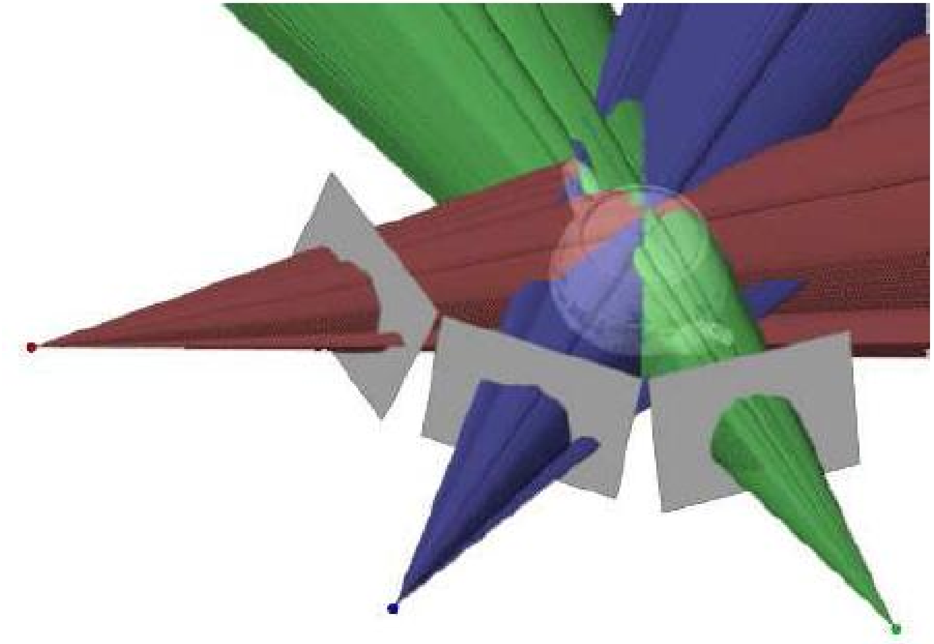
Shape from Structured Light

- ✓ An active technique where we actively send laser stripes (or other light structures/stripes) to the real-world object to generate its digital mesh.
- ✓ With one camera (common).
- ✓ Calibration that gives transformation parameters from the image plane to the laser plane requires only 4 known correspondences:



Shape from Silhouette

- ✓ A passive technique where we use multiple binary silhouette images of the real-world object to generate its digital mesh.
- ✓ Intersect the back projections of the visual hulls in 3D. One way to do is via deformation: Shape from Silhouette Using Topology-Adaptive Mesh Deformation.

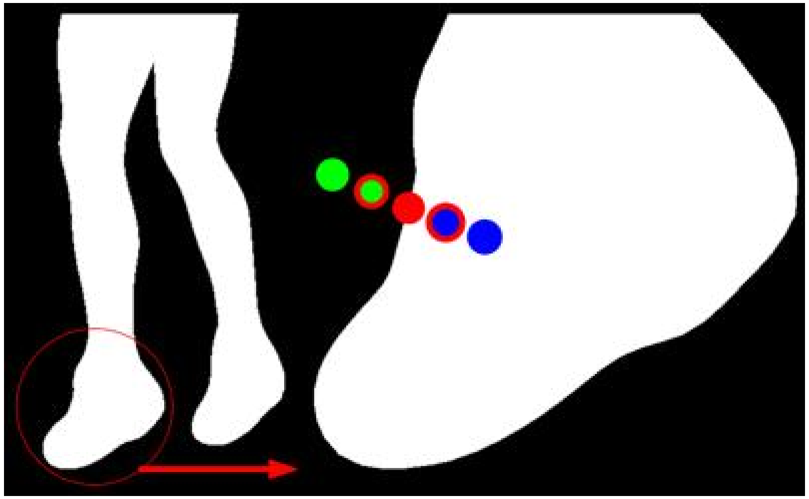


- ✓ Another passive technique is semi-automatic: https://youtu.be/Oie1ZXWceqM

Shape from Silhouette

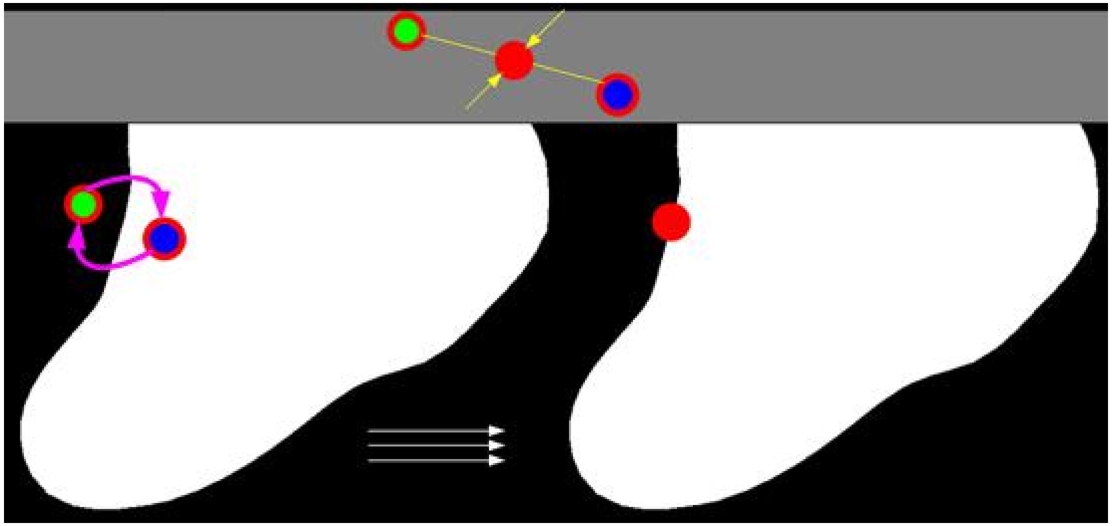
- ✓ A passive technique where we use multiple binary silhouette images of the real-world object to generate its digital mesh.
- ✓ Bounding sphere is deformed towards the target in the guidance of sils.
- ✓ A 3D point is projected to all silhouettes. If it is out (black pixel) in at least 1 image, it means it is still out so move it further in –normal dir. If it is in (white) on all images, it means it is penetrated so move it back and forth in binary search manner. Stop when at “gray” pixel on all imgs.

$$L(P) = \begin{cases} \text{VERY-OUT} & \text{if } f(P) = -0.5 \\ \text{OUT} & \text{if } -0.5 < f(P) < 0 \\ \text{ON} & \text{if } f(P) = 0 \\ \text{IN} & \text{if } 0 < f(P) < 0.5 \\ \text{VERY-IN} & \text{if } f(P) = 0.5 \end{cases}$$



Shape from Silhouette

- ✓ A passive technique where we use multiple binary silhouette images of the real-world object to generate its digital mesh.
- ✓ Bounding sphere is deformed towards the target in the guidance of sils.
- ✓ A 3D point is projected to all silhouettes. If it is out (black pixel) in at least 1 image, it means it is still out so move it further in –normal dir. If it is in (white) on all images, it means it is penetrated so move it back and forth in binary search manner. Stop when at "gray" pixel on all imgs.



Shape from Silhouette

- ✓ A passive technique where we use multiple binary silhouette images of the real-world object to generate its digital mesh.
- ✓ Bounding sphere is deformed towards the target in the guidance of sils.
- ✓ A 3D point is projected to all silhouettes. If it is out (black pixel) in at least 1 image, it means it is still out so move it further in –normal dir. If it is in (white) on all images, it means it is penetrated so move it back and forth in binary search manner. Stop when at “gray” pixel on all imgs.

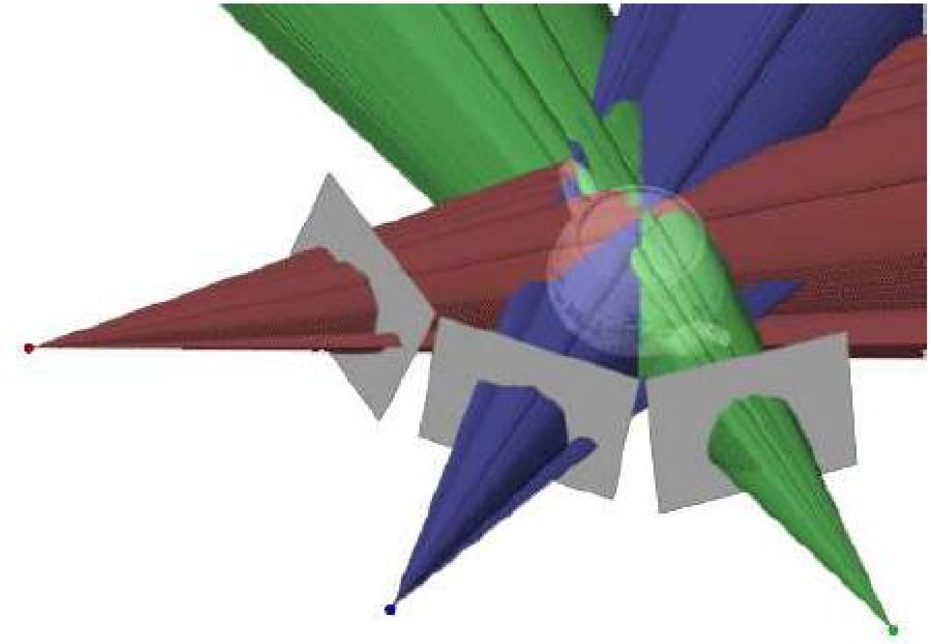
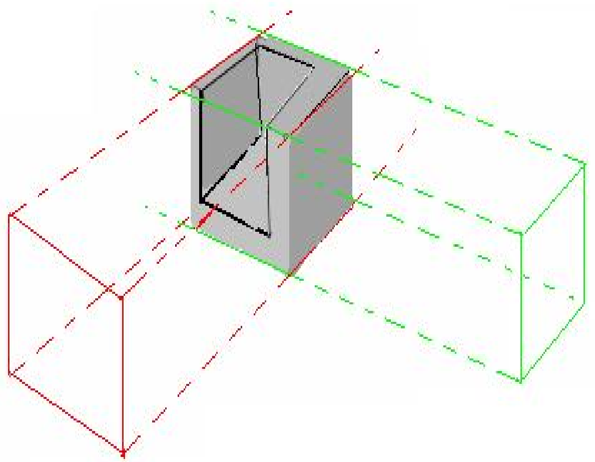
$$f(P) = \min_{n} \left\{ G\left[\mathrm{Proj}_{I_n}(P) \right] - 0.5 \right\}$$

$$\begin{aligned} G(x', y') = {} & (1-\alpha)((1-\beta)I(\lfloor x' \rfloor, \lfloor y' \rfloor) + \beta I(\lfloor x' \rfloor, \lfloor y' \rfloor + 1)) \\ & + \alpha((1-\beta)I(\lfloor x' \rfloor + 1, \lfloor y' \rfloor) + \beta I(\lfloor x' \rfloor + 1, \lfloor y' \rfloor + 1)) \end{aligned}$$

- ✓ G is the bilinear interpolation of the sub-pixelic projection (x’, y’) of the point P and takes values in [0,1]: 0 if neighborhood is all black (0), or 0.5 for combination of black and whites, hence on the boundary (gray).

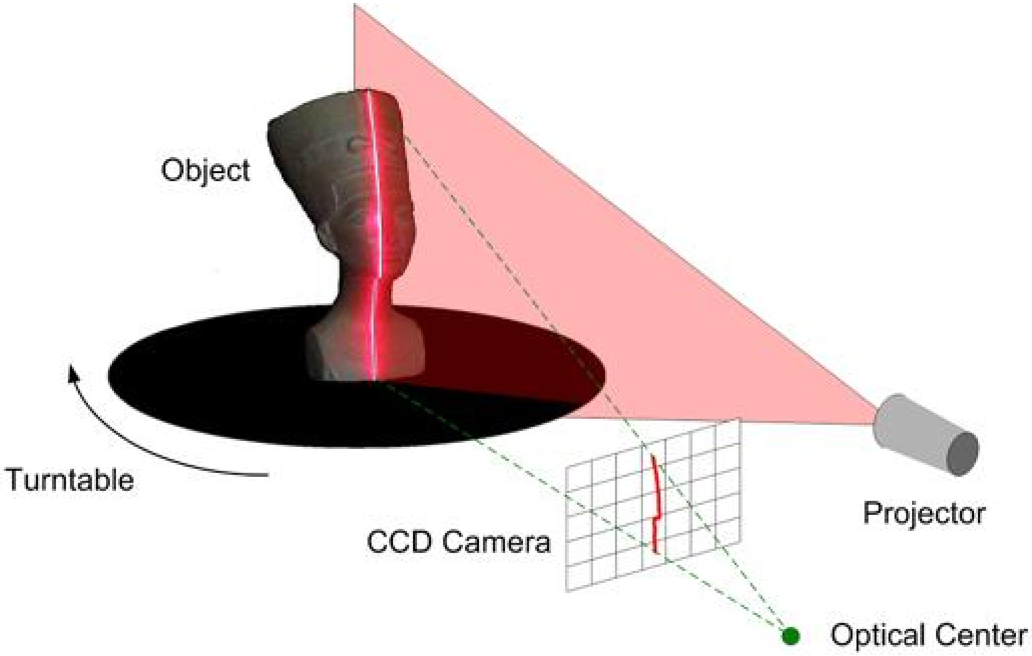
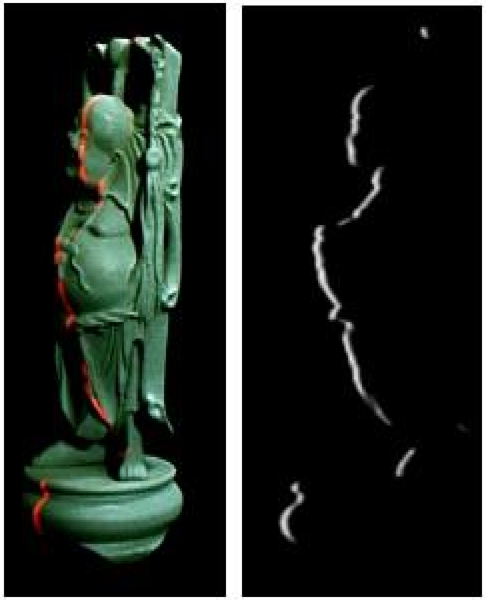
Shape from Silhouette and Structured Light

- ✓ A hybrid technique where we fuse advantages of each method.
- ✓ Shape from Silhouette is hole-free but concavity insensitive.

Idea	***Weakness***
	 Hidden Concavity

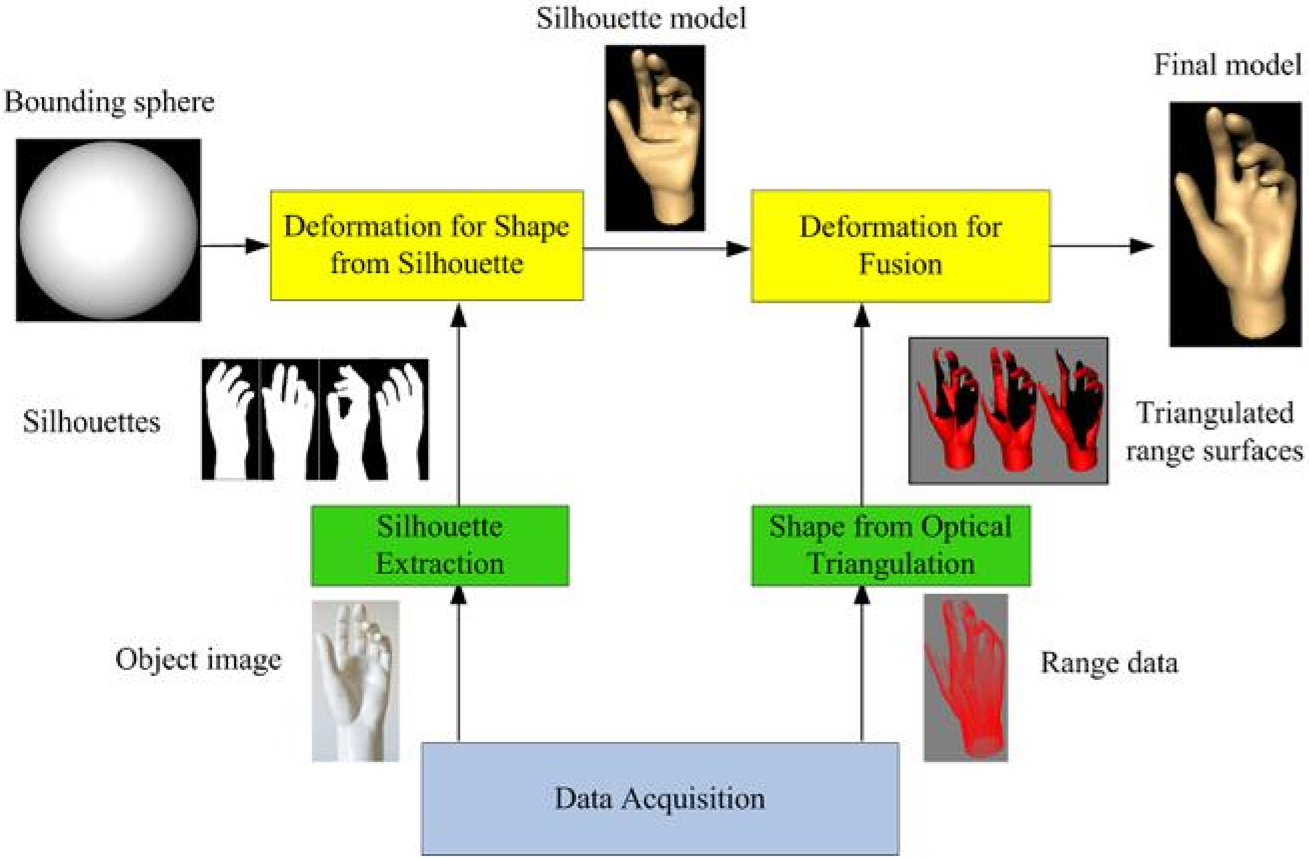
Shape from Silhouette and Structured Light

- ✓ A hybrid technique where we fuse advantages of each method.
- ✓ Shape from Structured Light is concavity sensitive but may have holes.

Idea	***Weakness***
 Object Turntable CCD Camera Projector Optical Center	 Occlusion

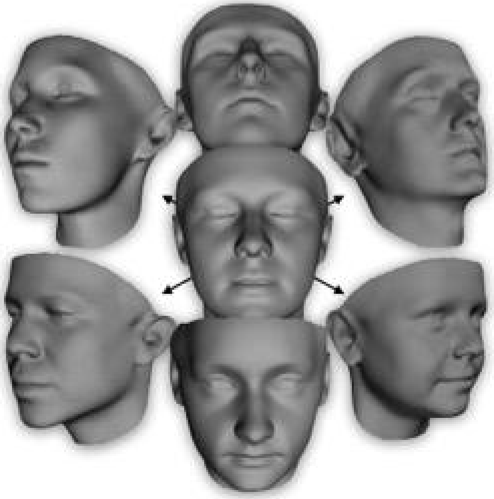
Shape from Silhouette and Structured Light

- ✓ A hybrid technique where we fuse advantages of each method.
- ✓ Combine their benefits in one framework: Coarse-to-Fine Surface Reconstruction from Silhouettes and Range Data Using Mesh Deformation.



Potential Project Topics

- ✓ Representation of all major modes of face/pose variations via PCA.



- ✓ Paper: A Morphable Model For The Synthesis Of 3D Faces. //FaceGen
  - ✓ See: https://youtu.be/nice6NYb_WA or youtu.be/KPDfMpuK2fQ blendshapes
- ✓ Paper: A 3D Morphable Model learnt from 10,000 faces. //more recent
- ✓ Paper: Efficient and Flexible Deformation Representation for Data-Driven Surface Modeling, or the one in Slide 53.
- ✓ A paper mentioned during the Shape from X part.