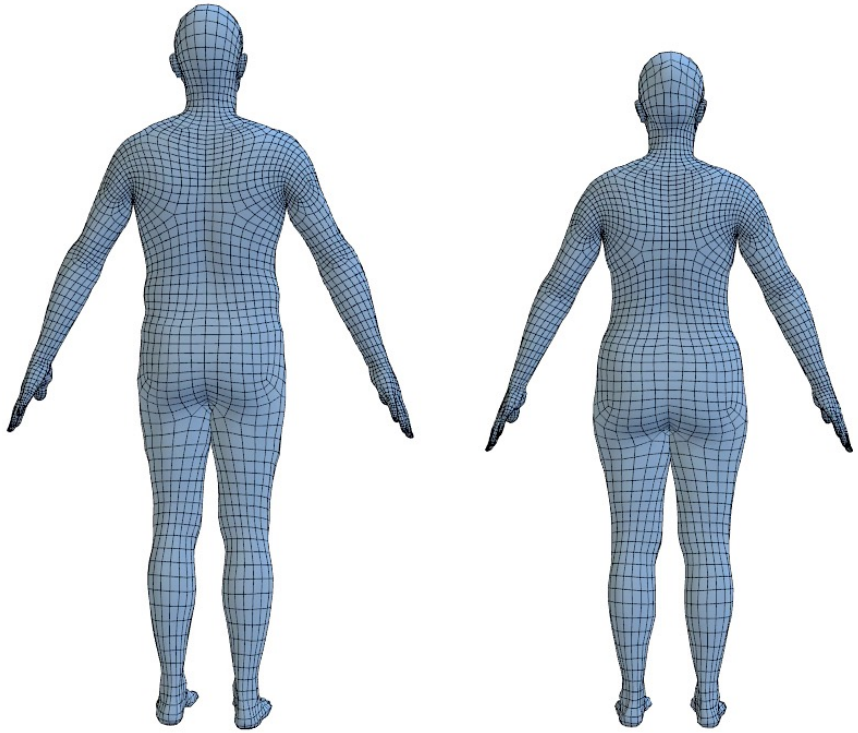
Data Driven Models

Statistical Analysis of Body Shapes

We have a data of aligned people in the exact same pose.

$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3 \times |V|}$$

Average shape of ~2000 men and women in US and Europe



Figures Courtesy of Michael Black

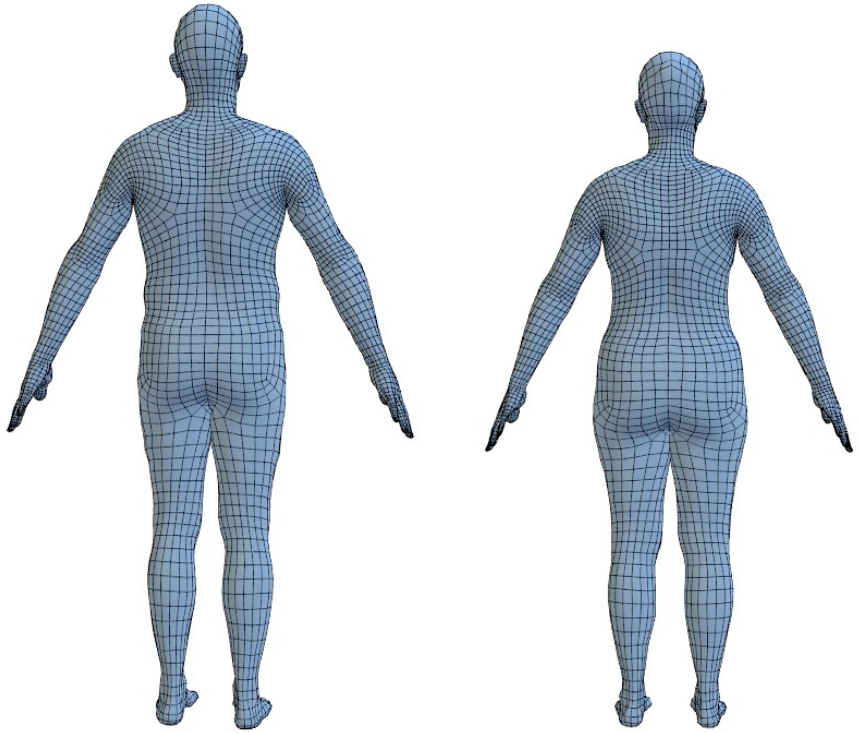
Statistical Analysis of Body Shapes

We have a data of aligned people in the exact same pose.

$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$

Now we can ask questions like: what is the average?

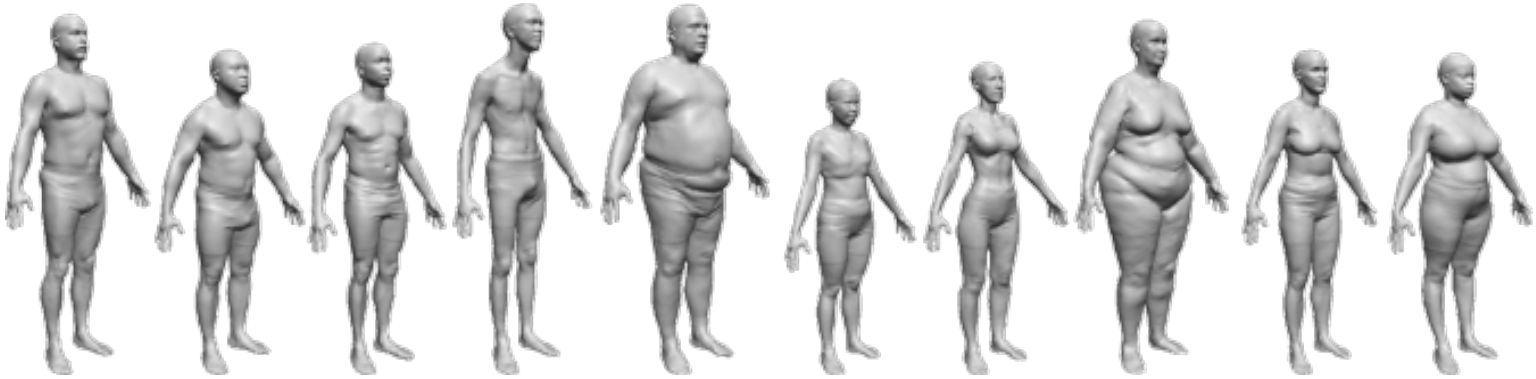
Average shape of ~2000 men and women in US and Europe



Figures Courtesy of Michael Black

Human Bodies Scans

Civilian American and European Surface Anthropometry Resource Project (CAESAR)



$$\{V_1, V_2, V_3, \ldots V_N\} \in \mathscr{R}^{3\times\#V}$$

All meshes have same topology and same vertex semantics.

e.g. the same vertex number is the the tip of nose for every mesh

Principal Component Analysis

For learning "Shape" Blend Shapes, apply PCA!

$$\left\{V_1, V_2, V_3, \ldots V_N\right\} \in \mathscr{R}^{3\times \#V}$$

Principal Component Analysis

For learning "Shape" Blend Shapes, apply PCA!

$$\{V_1, V_2, V_3, \ldots V_N\} \in \mathscr{R}^{3\times \#V}$$

$$\cong \sum_i \beta_i B_i + \bar{V}$$

Principal Component Analysis

For learning "Shape" Blend Shapes, apply PCA!

$$\{V_1, V_2, V_3, \ldots V_N\} \in \mathscr{R}^{3\times \#V}$$

$$\cong \sum_i \beta_i B_i + \bar{V}$$

Eigenvectors / Blend Shapes

Principal Component Analysis

For learning "Shape" Blend Shapes, apply PCA!

$$\{V_1, V_2, V_3, \ldots V_N\} \in \mathscr{R}^{3\times \#V}$$

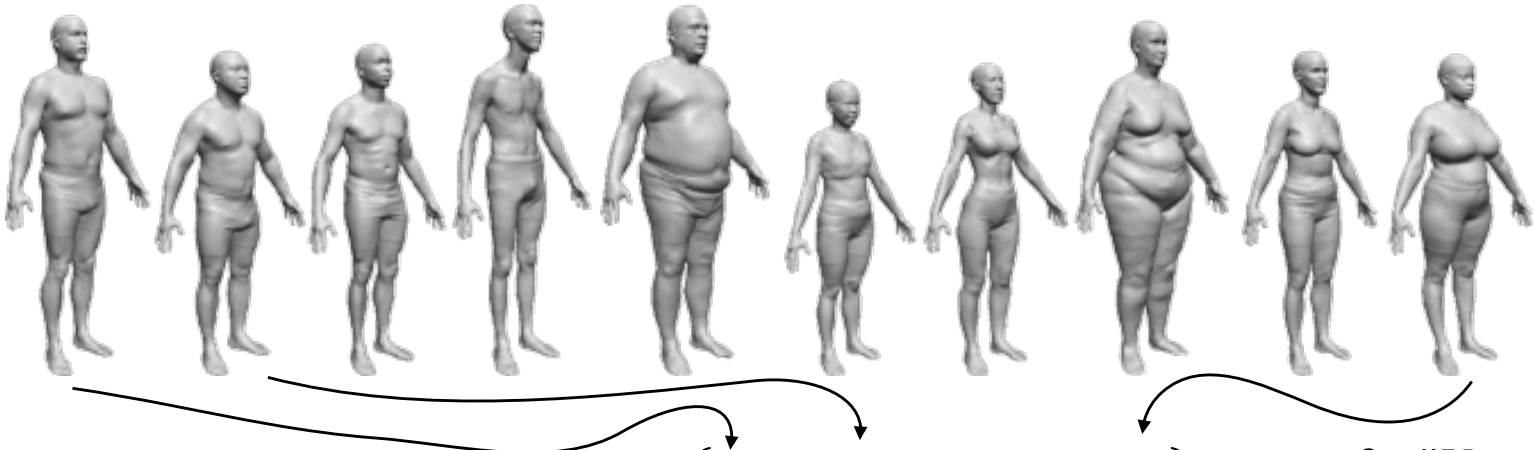
$$\cong \sum_i \beta_i B_i + \bar{V}$$

Eigenvalues

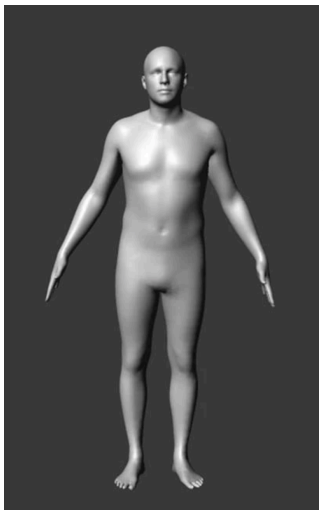
Eigenvectors / Blend Shapes

Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, V_3, \ldots V_N\} \in \mathscr{R}^{3\times \#V}$$



$$\bar{V} = \frac{\sum V_i}{N}$$

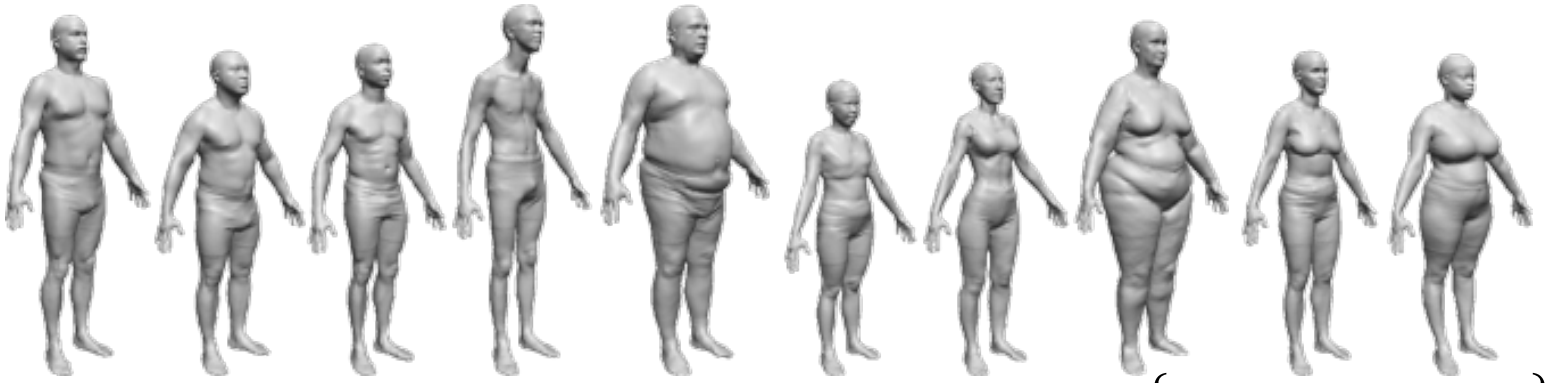
$$R_i = V_i - \bar{V}$$

Residual

- $\bar{V}$ minimizes $\sum |R_i|^2$
- What do the R_i look like?

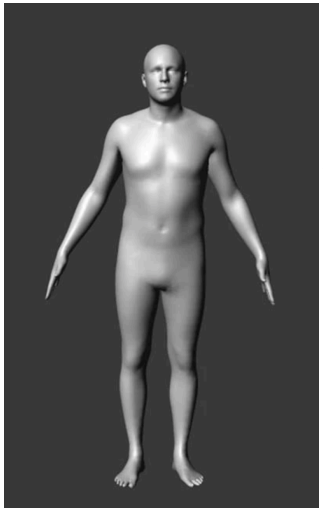
Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, V_3, \ldots V_N\} \in \mathscr{R}^{3\times \#V}$$

$$R_i = V_i - \bar{V}$$

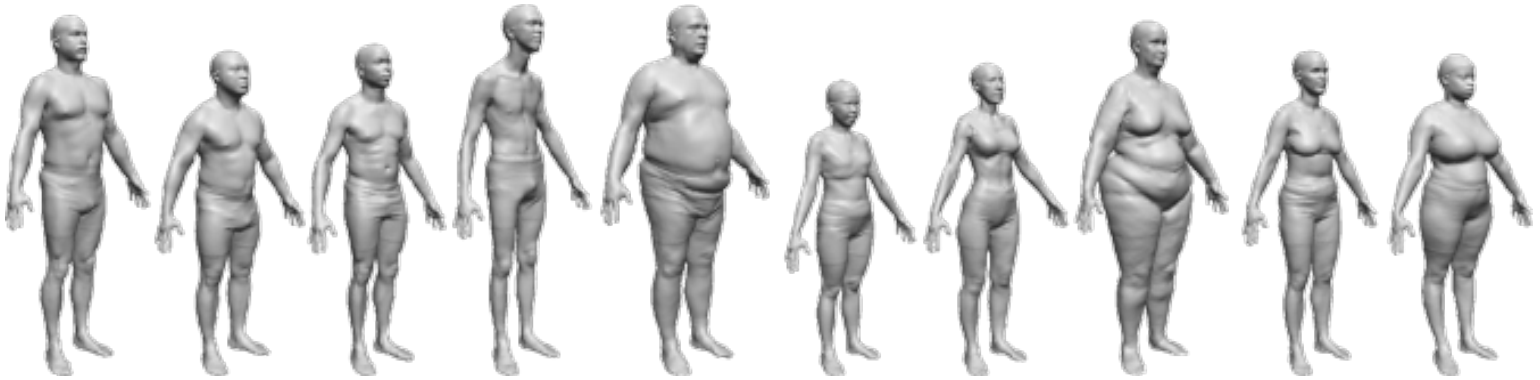


$\bar{V}$

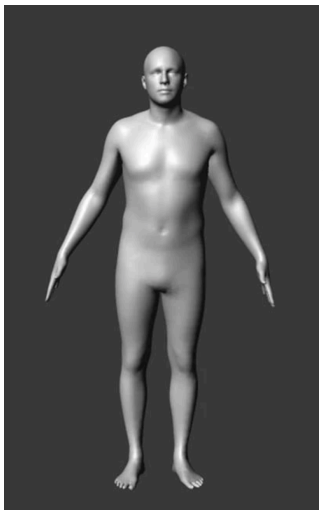
$$\hat{B}_1 = \underset{\hat{B}_1}{\operatorname{argmin}} \left(\min_{\{\alpha_i\}} \sum \left| R_i - \hat{B}_1 \alpha_i \right|^2 \right) \quad , \quad |\hat{B}_1| = 1$$

Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$



$\bar{V}$

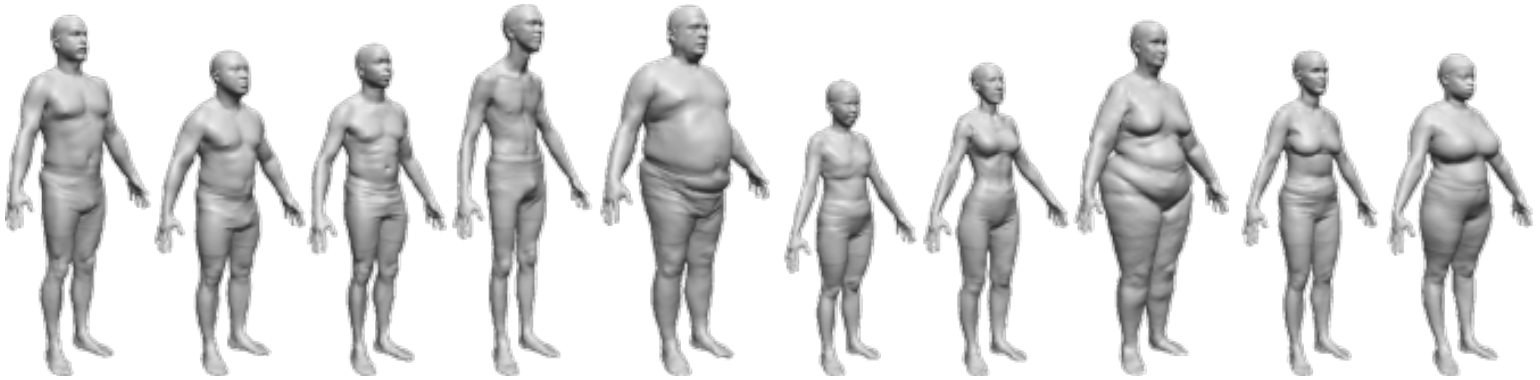
$$R_i = V_i - \bar{V}$$

$$\hat{B}_1 = \underset{\hat{B}_1}{\text{argmin}} \left(\min_{\{\alpha_i\}} \sum \left| R_i - \hat{B}_1 \alpha_i \right|^2 \right) \quad , \quad |\hat{B}_1| = 1$$

$$= \underset{\hat{B}_1}{\text{argmax}} \left(\sum \left| R_i \cdot \hat{B}_1 \right|^2 \right) \quad , \quad |\hat{B}_1| = 1$$

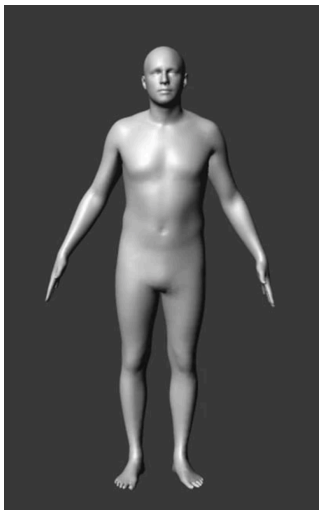
Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$

$$R_i = V_i - \bar{V}$$



$\bar{V}$

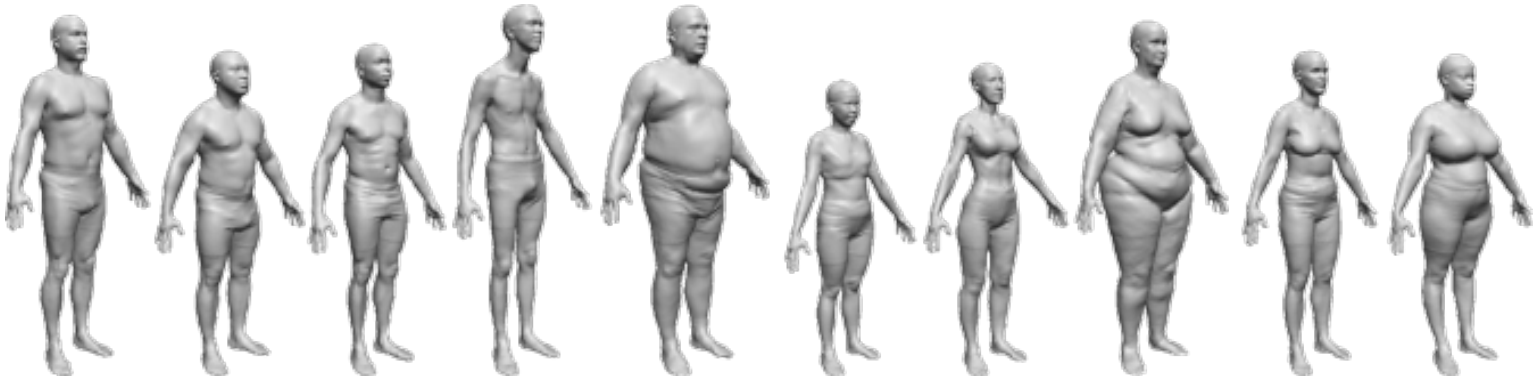
$$\hat{B}_1 = \underset{\hat{B}_1}{\text{argmin}} \left(\min_{\{\alpha_i\}} \sum \left| R_i - \hat{B}_1 \alpha_i \right|^2 \right) \quad , \quad |\hat{B}_1| = 1$$

$$= \underset{\hat{B}_1}{\text{argmax}} \left(\sum \left| R_i \cdot \hat{B}_1 \right|^2 \right) \quad , \quad |\hat{B}_1| = 1$$

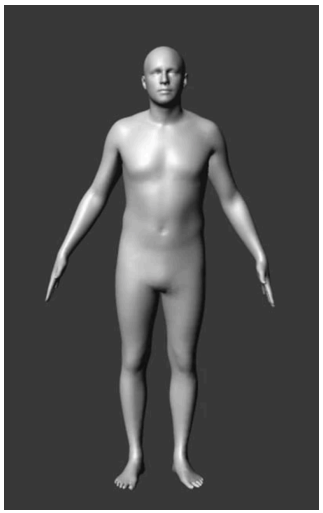
How does this turn into this?

Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$



$\bar{V}$

$$R_i = V_i - \bar{V}$$

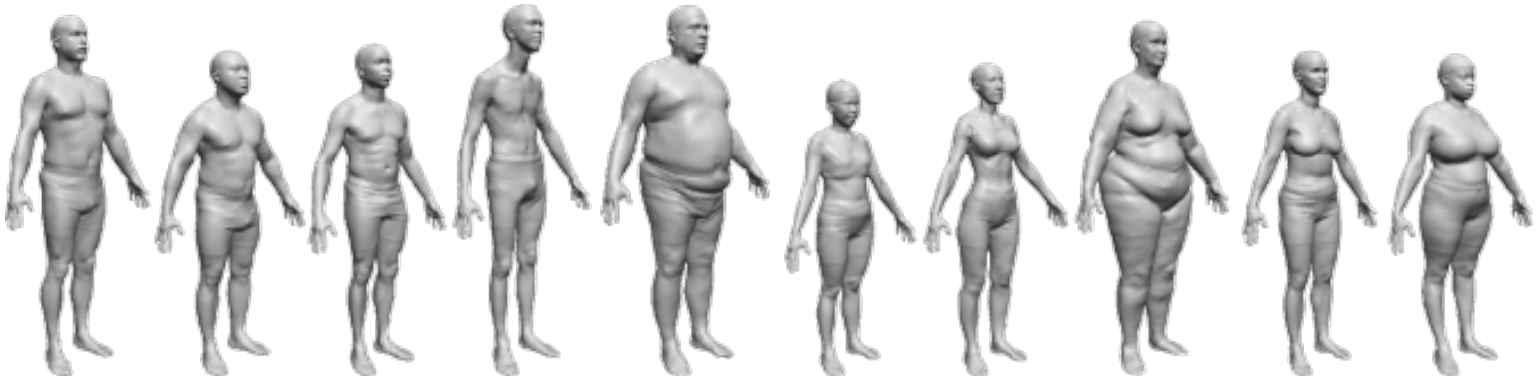
$$\hat{B}_1 = \underset{\hat{B}_1}{\text{argmin}} \left(\min_{\{\alpha_i\}} \sum \left| R_i - \hat{B}_1 \alpha_i \right|^2 \right) \quad , \ |\hat{B}_1| = 1$$

$$= \underset{\hat{B}_1}{\text{argmax}} \left(\sum \left| R_i \cdot \hat{B}_1 \right|^2 \right) \quad , \ |\hat{B}_1| = 1$$

$$= \underset{\hat{B}_1}{\text{argmax}} \left(\sum \left| \hat{B}_1^{\mathsf{T}} \cdot R_i^{\mathsf{T}} \cdot R_i \cdot \hat{B}_1 \right| \right) \quad , \ |\hat{B}_1| = 1$$

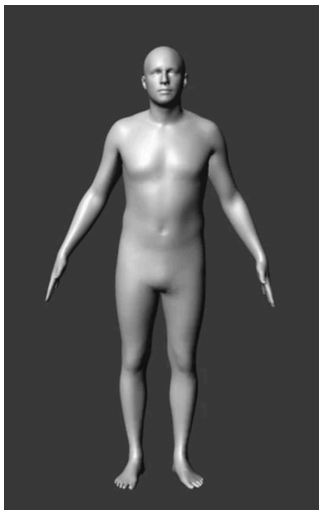
Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$

$$R_i = V_i - \bar{V}$$



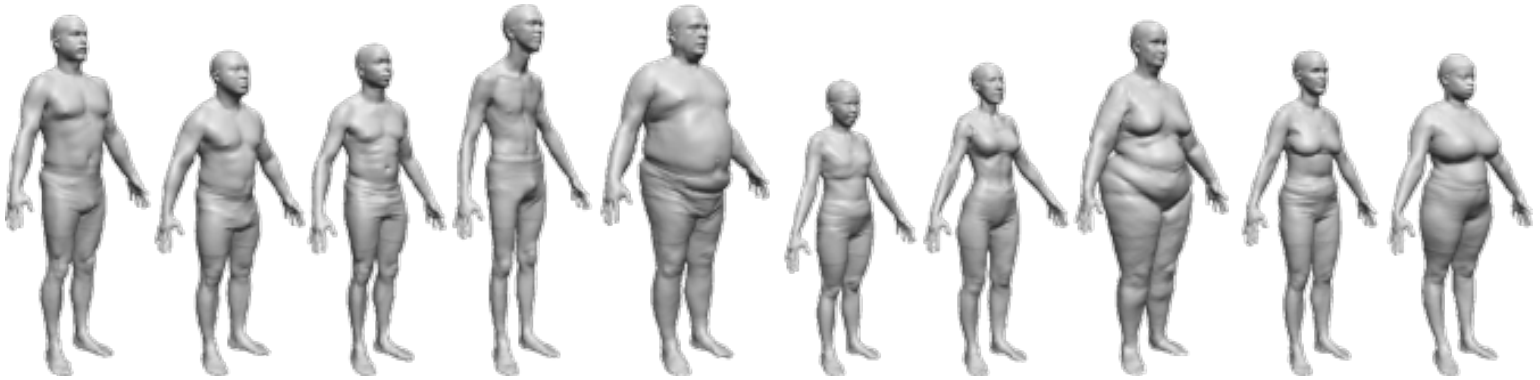
$\bar{V}$

$$\hat{B}_1 = \underset{\hat{B}_1}{\text{argmax}} \left(\sum \left| \hat{B}_1^{\mathsf{T}} \cdot R_i^{\mathsf{T}} \cdot R_i \cdot \hat{B}_1 \right| \right) \quad , \quad |\hat{B}_1| = 1$$

$$= \underset{\hat{B}_1}{\text{argmax}} \left(\hat{B}_1^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \mathbf{R} \hat{B}_1 \right) \quad , \quad |\hat{B}_1| = 1$$

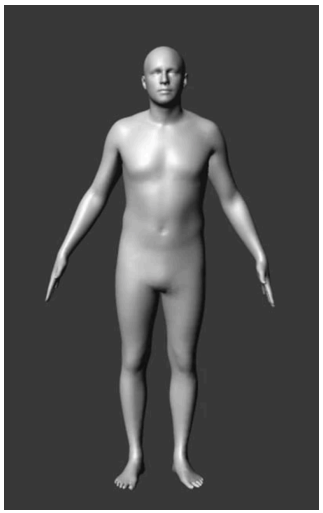
Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$

$$R_i = V_i - \bar{V}$$



$\bar{V}$

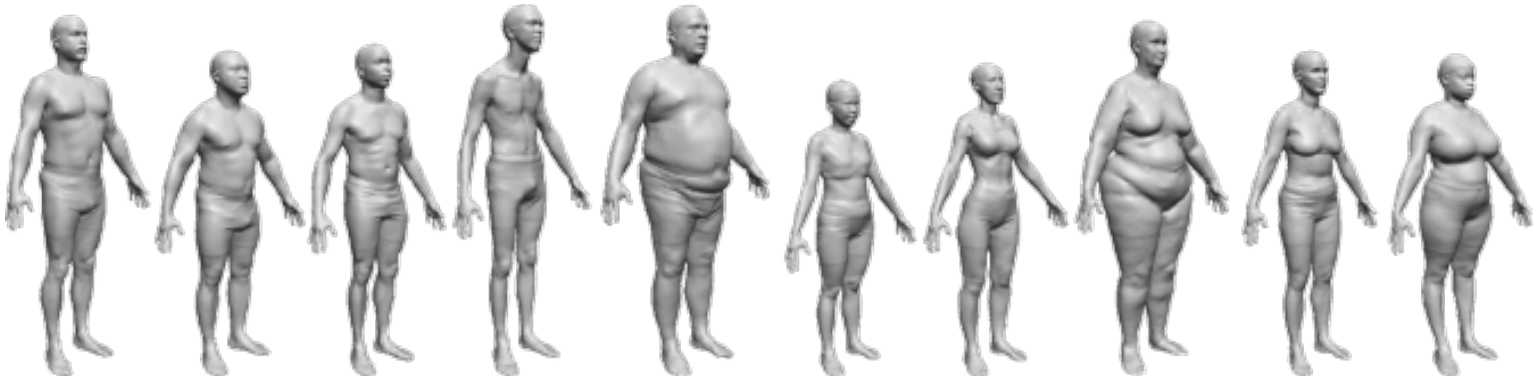
$$\hat{B}_1 = \underset{\hat{B}_1}{\text{argmax}} \left(\sum \left| \hat{B}_1^{\mathsf{T}} \cdot R_i^{\mathsf{T}} \cdot R_i \cdot \hat{B}_1 \right| \right) \quad , \quad |\hat{B}_1| = 1$$

$$= \underset{\hat{B}_1}{\text{argmax}} \left(\hat{B}_1^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \mathbf{R} \hat{B}_1 \right) \quad , \quad |\hat{B}_1| = 1$$

Thus $\hat{B}_1$ must be the eigenvector of $\mathbf{R}^{\mathsf{T}}\mathbf{R}$ with largest eigenvalue!

Principal Component Analysis

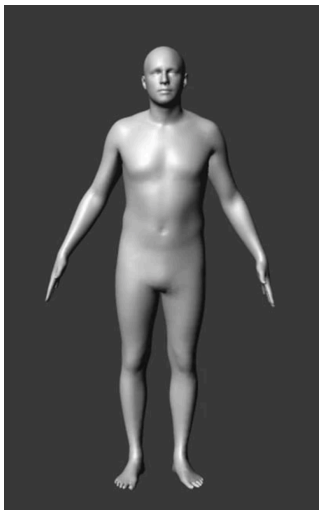
Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$

$$R_i = V_i - \bar{V}$$

$$R_i' = R_i - \hat{B}_1\alpha_i \quad = R_i - \hat{B}_1(R_i \cdot \hat{B}_1)$$

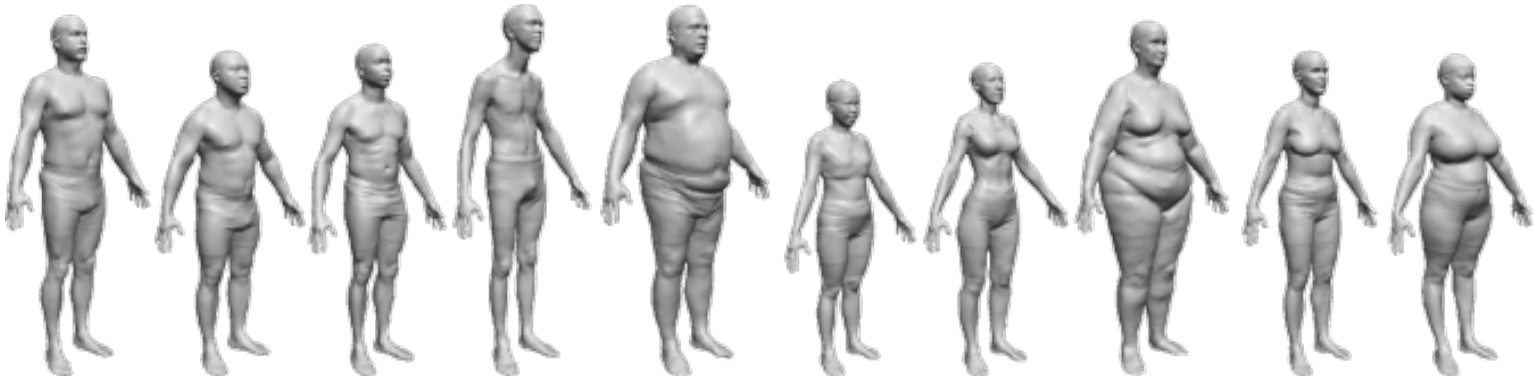


$\bar{V}$

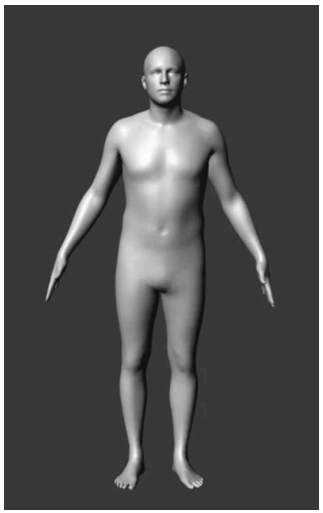
We can keep going…

Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$



$\bar{V}$

$$R_i = V_i - \bar{V}$$

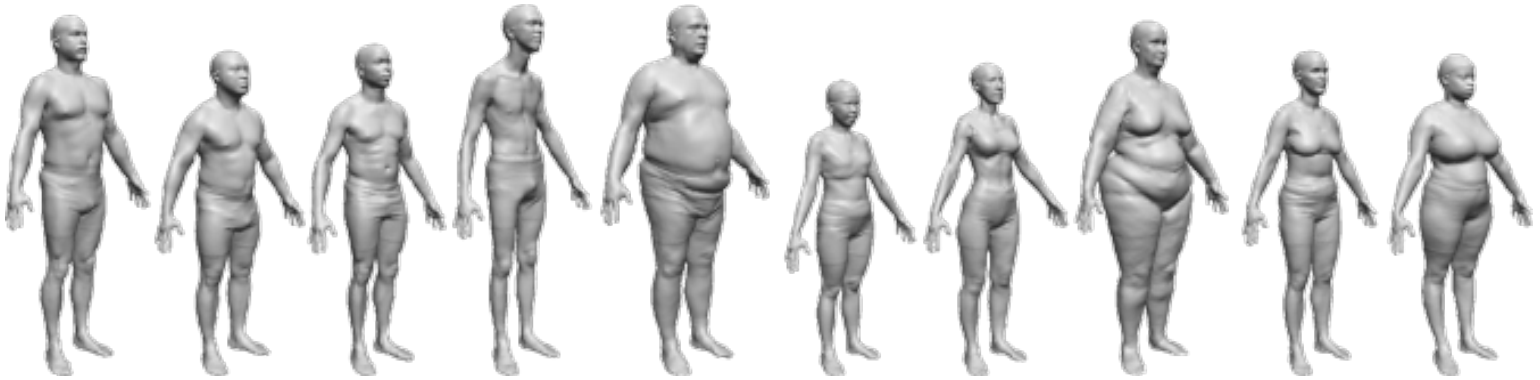
$$R_i' = R_i - \hat{B}_1\alpha_i \quad = R_i - \hat{B}_1(R_i \cdot \hat{B}_1)$$

Thus $\hat{B}_2$ must be the eigenvector of $\mathbf{R'}^{\mathsf{T}}\mathbf{R'}$ with largest eigenvalue!

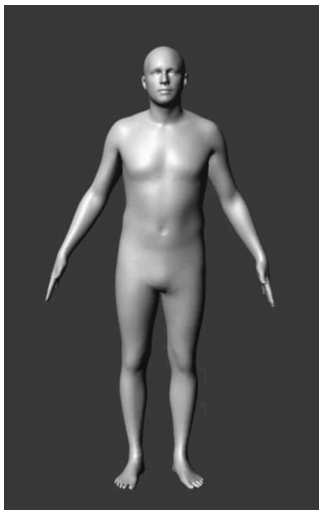
We can keep going…

Principal Component Analysis

Simplified for people who haven't seen PCA before



$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$



$\bar{V}$

$$R_i = V_i - \bar{V}$$

$$R_i' = R_i - \hat{B}_1\alpha_i \quad = R_i - \hat{B}_1(R_i \cdot \hat{B}_1)$$

Thus $\hat{B}_2$ must be the eigenvector of $\mathbf{R'}^{\mathsf{T}}\mathbf{R'}$ with largest eigenvalue!

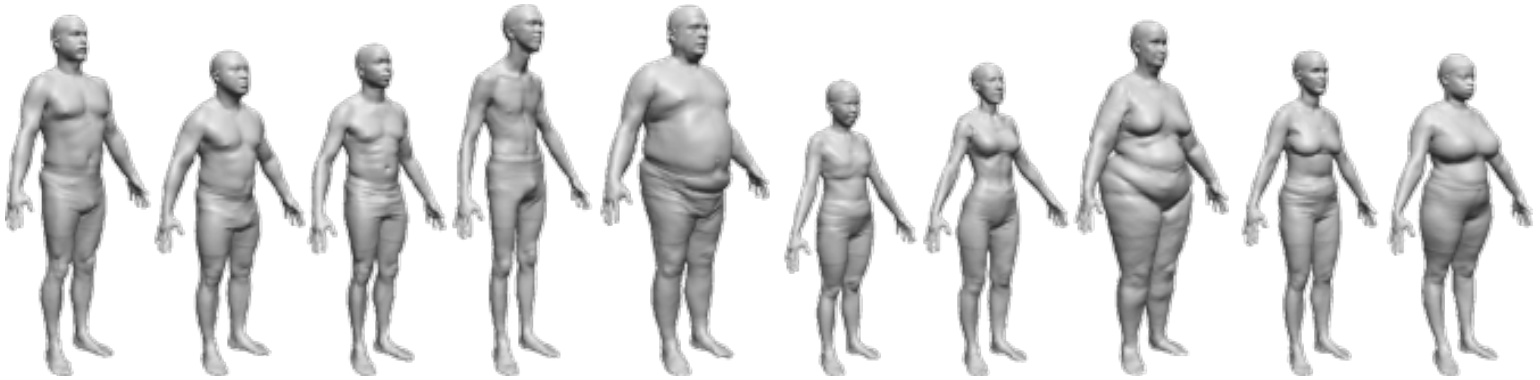
Equivalently:

$\hat{B}_2$ must be the eigenvector of $\mathbf{R}^{\mathsf{T}}\mathbf{R}$ with second largest eigenvalue!

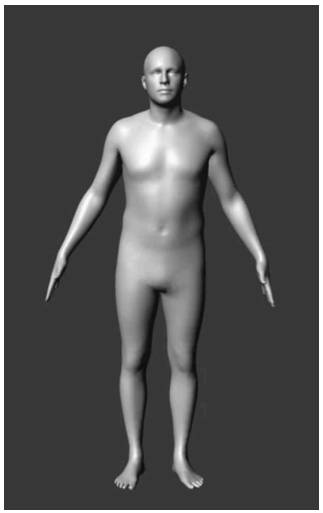
We can keep going…

Principal Component Analysis

Simplified for people who haven't seen PCA before



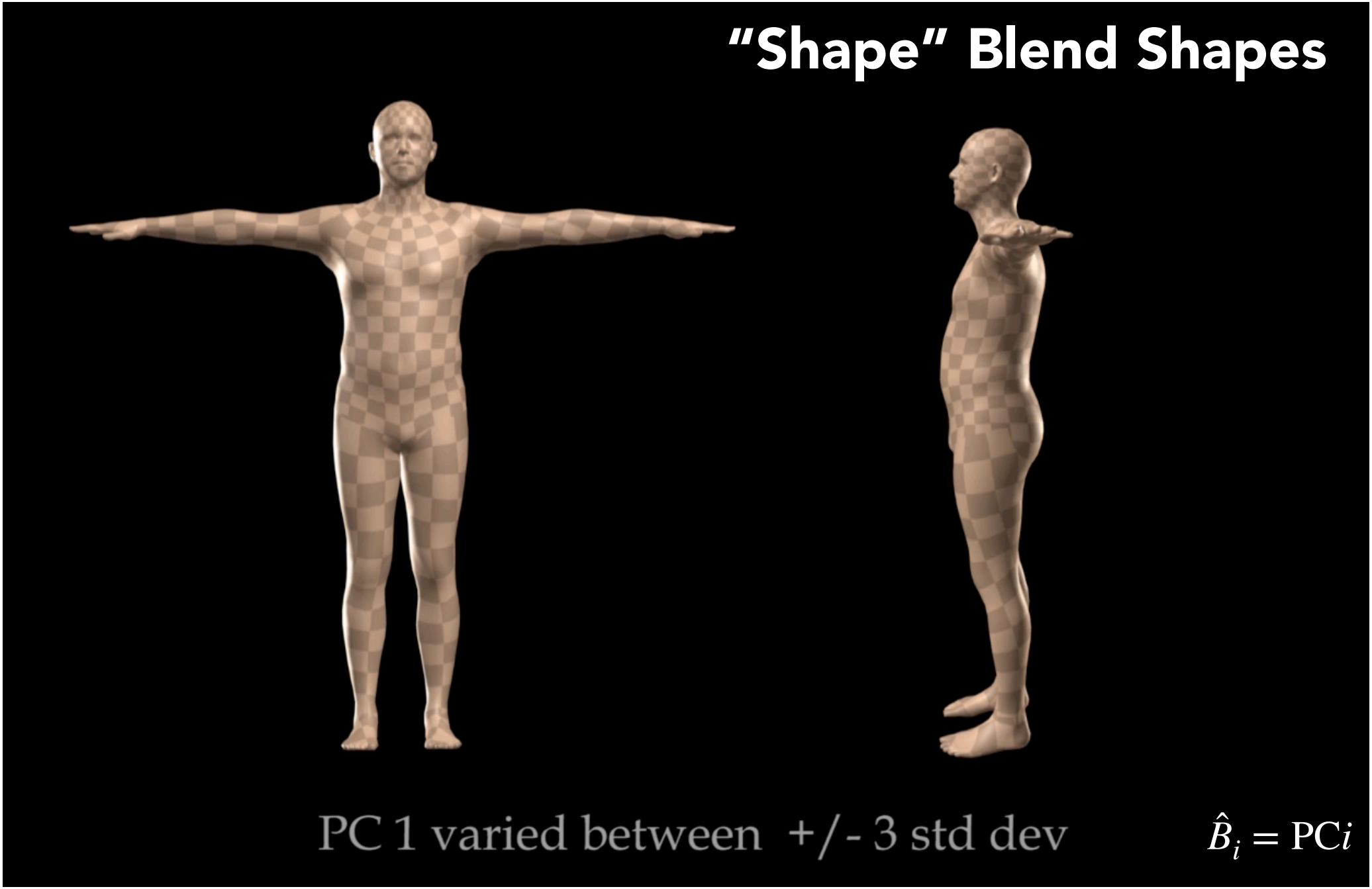
$$\{V_1, V_2, \cdots, V_n\}, V \in \mathbf{R}^{3\times|V|}$$



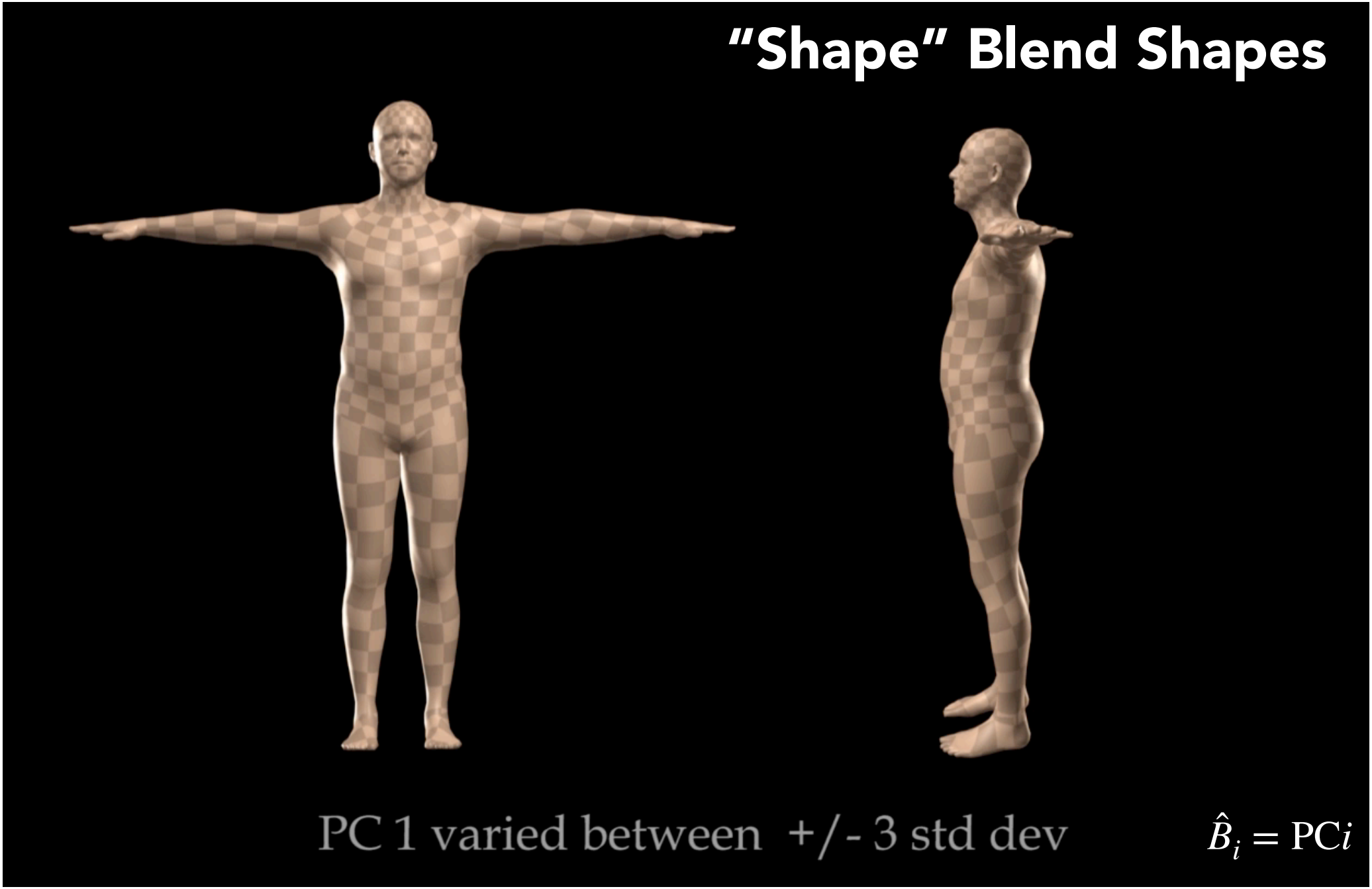
$\bar{V}$

- **The eigenvectors of $\mathbf{R}^\mathsf{T}\mathbf{R}$, $\left\{\hat{B}_i\right\}$, form a complete basis**
  - **Any $V = \sum \hat{B}_i \beta_i$ for some $\{\beta_i\}$**
- **The first several $\hat{B}_i$ capture most of the interesting variation**
  - **Only need to keep K of them where $K \ll \#V$ and $K \ll N$**

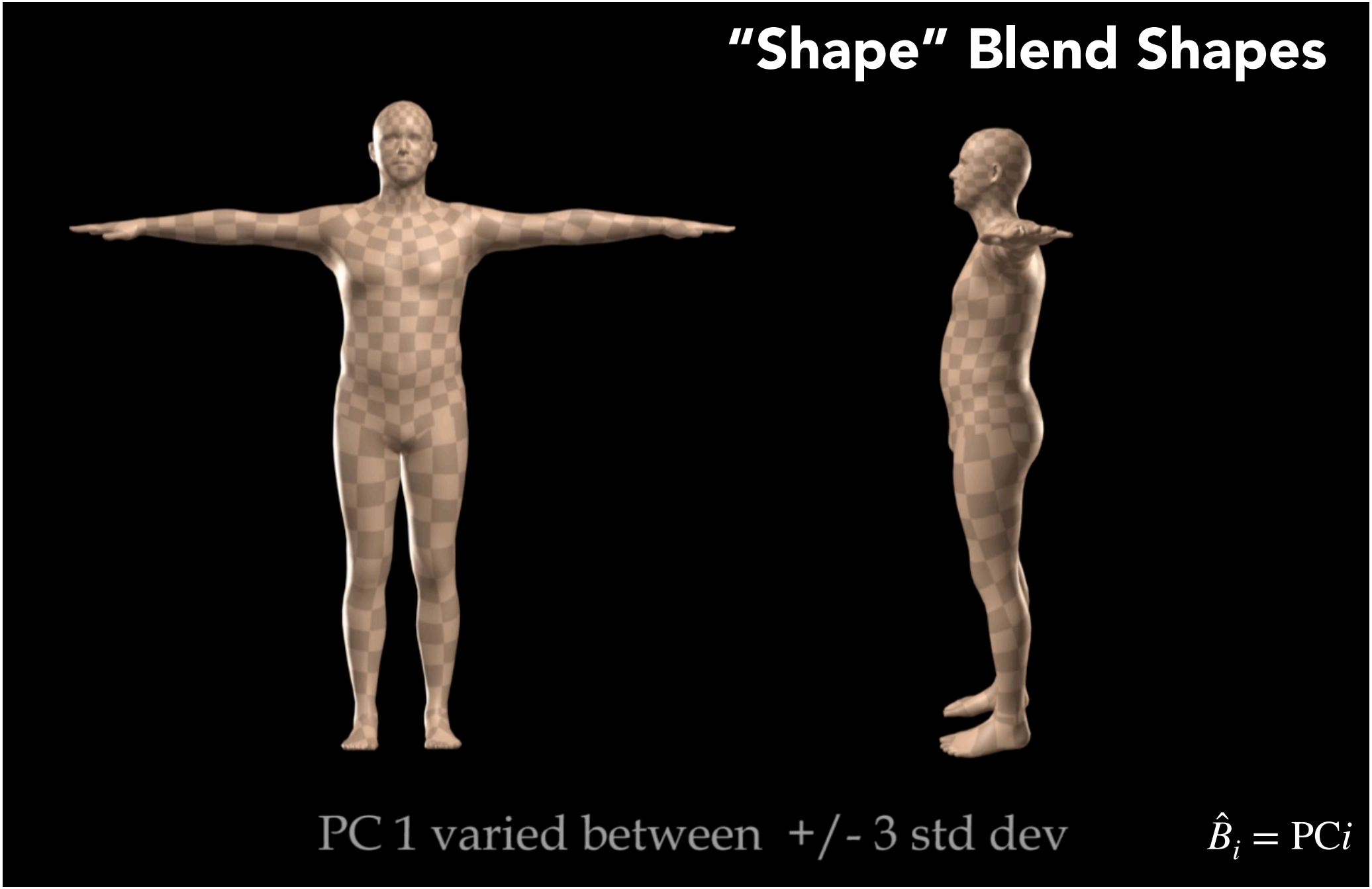
Principal components: Male



Principal components: Male



Principal components: Male



Attendance

If you are seated in class, go to this form and sign in:

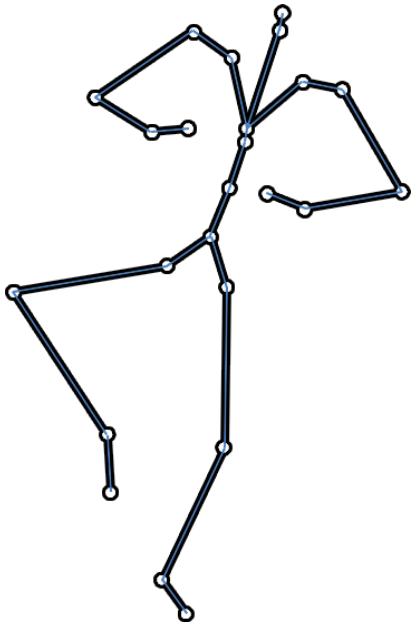
- https://bit.ly/184attendance

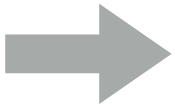
NOTE: This is a different URL than previously!

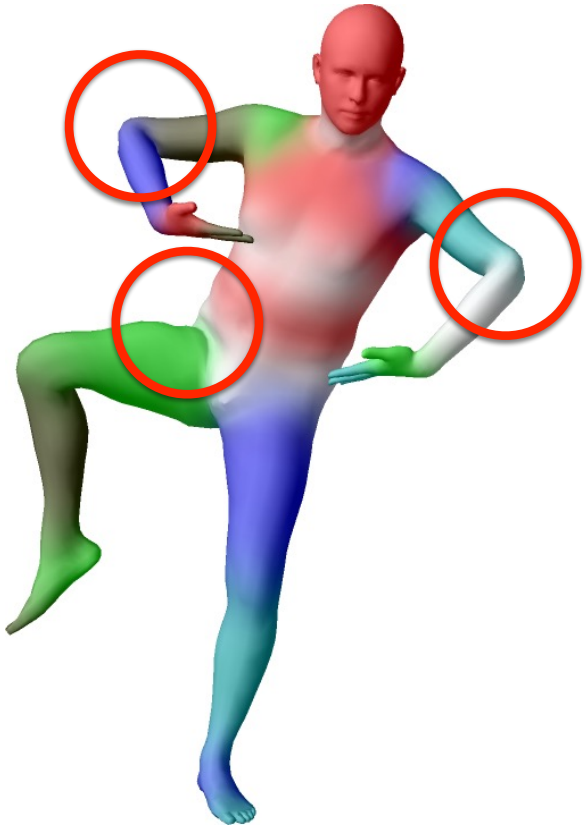
Notes:

- Time-stamp will be taken when you submit form. Do it now, won't count later.
- Don't tell friends outside class to fill it out now, because we will audit at some point in semester.
- Failing audit will have large negative consequence. You don't need to, because you have an alternative!

Pose with kinematics + Linear Blend Skinning







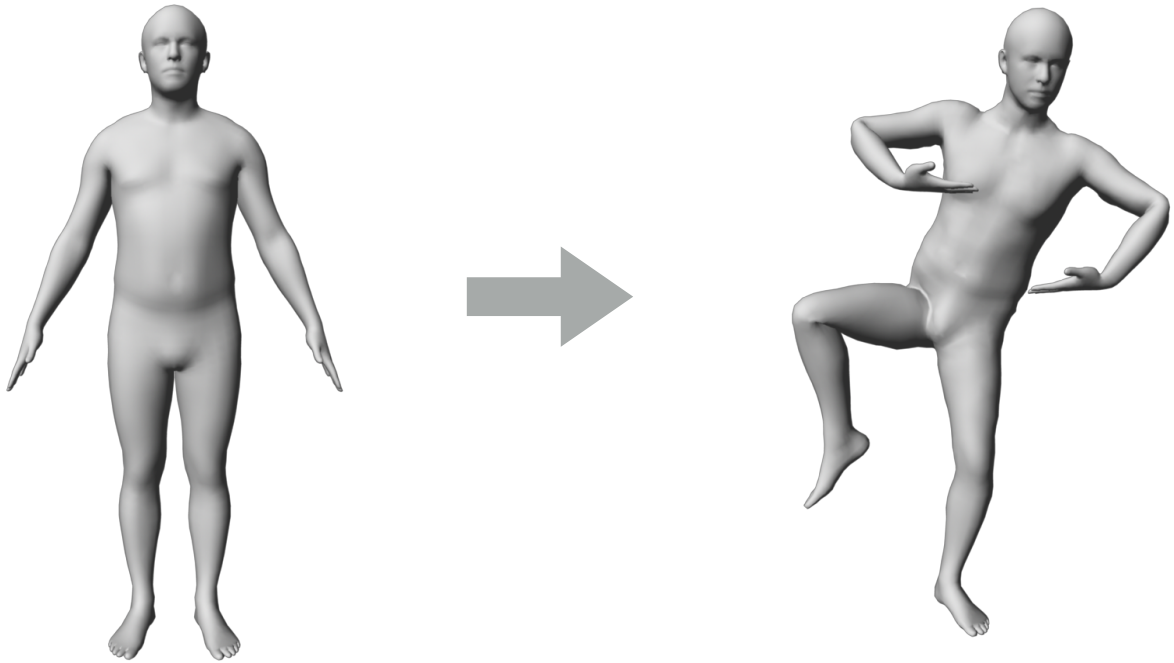
User Input:
pose as relative rotation

$$\theta \in \mathbb{R}^{3 \times |B|}$$

Corrective Blend Shapes

Blend Shape: a set of vertex displacements in rest pose.

Corrective Blend Shape: Offsets as a function of pose, so after LBS it looks good.

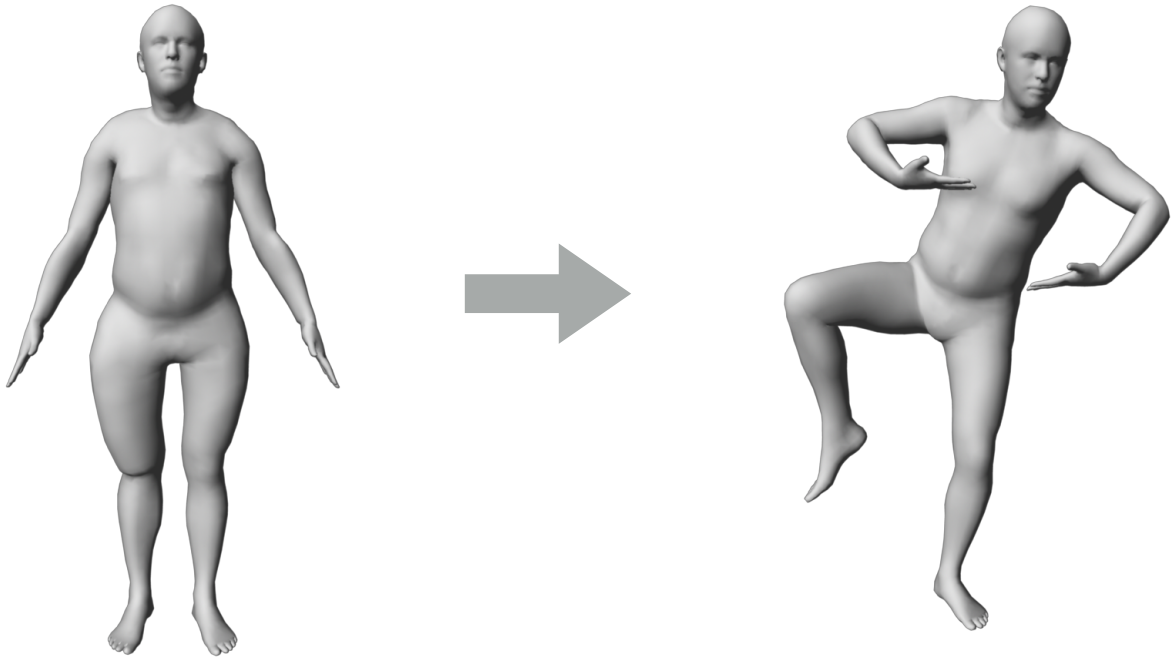


"Pose" blend shapes, bc if fixes issues with posing

Corrective Blend Shapes

Blend Shape: a set of vertex displacements in rest pose.

Corrective Blend Shape: Offsets as a function of pose, so after LBS it looks good.

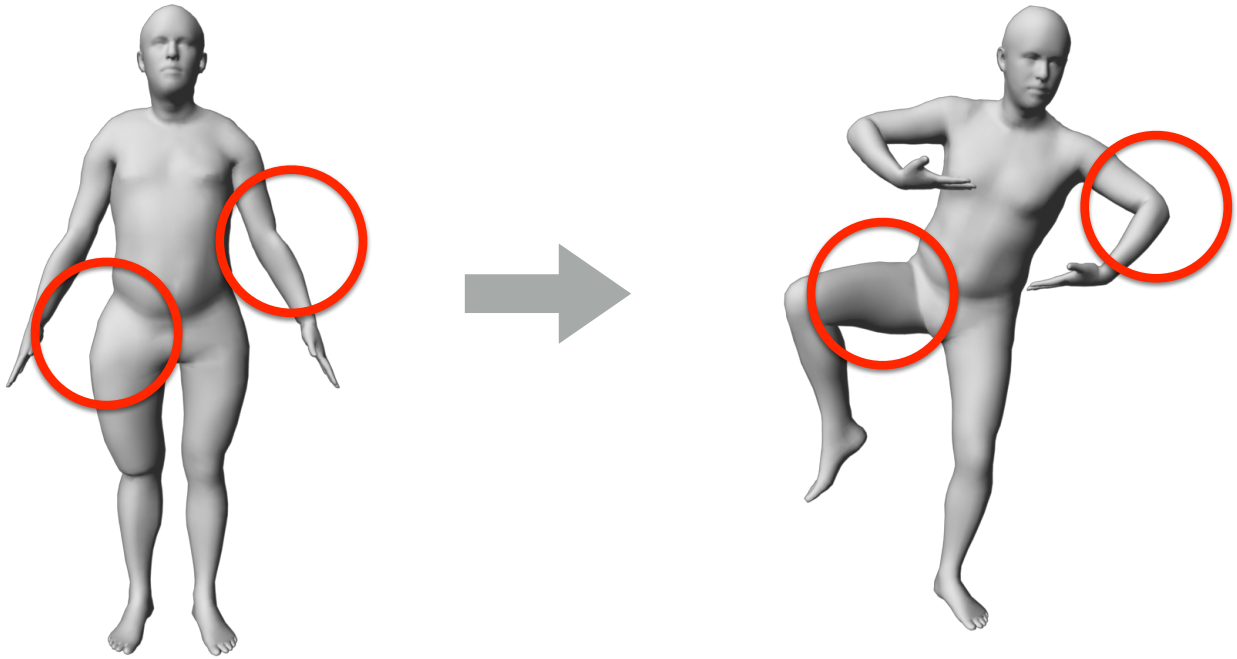


"Pose" blend shapes, bc if fixes issues with posing

Corrective Blend Shapes

Blend Shape: a set of vertex displacements in rest pose.

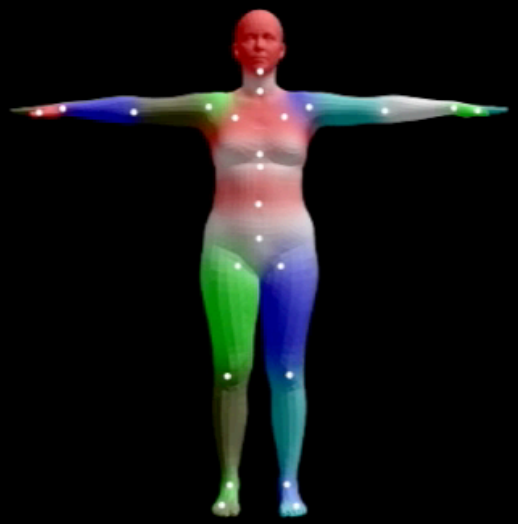
Corrective Blend Shape: Offsets as a function of pose, so after LBS it looks good.



"Pose" blend shapes, bc if fixes issues with posing

SMPL Summary

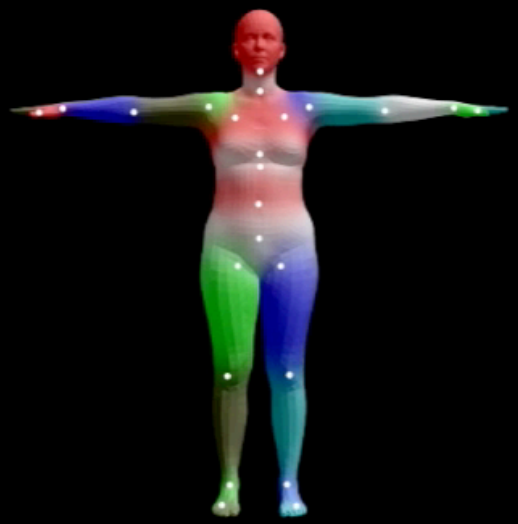
SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



Template Mesh

SMPL Summary

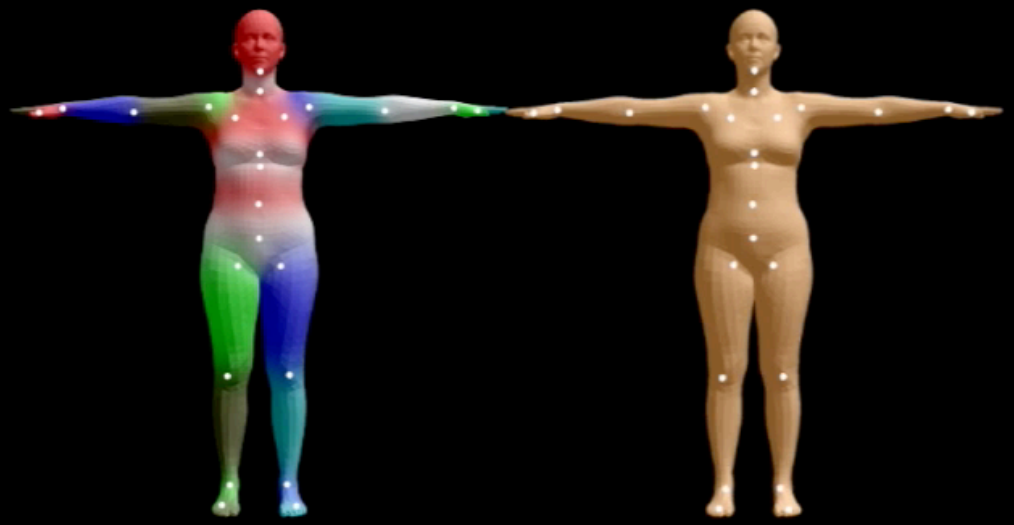
SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



Template Mesh

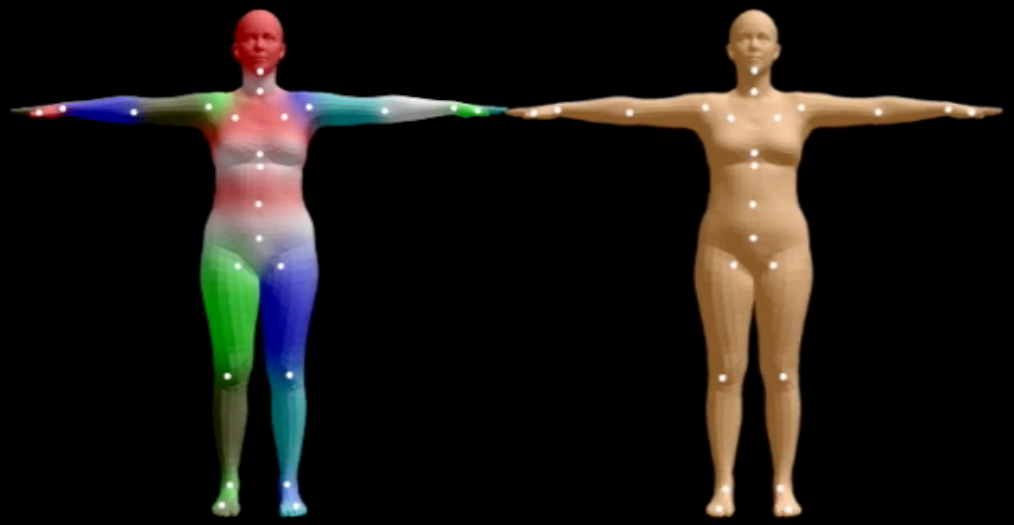
SMPL Summary

SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



SMPL Summary

SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.

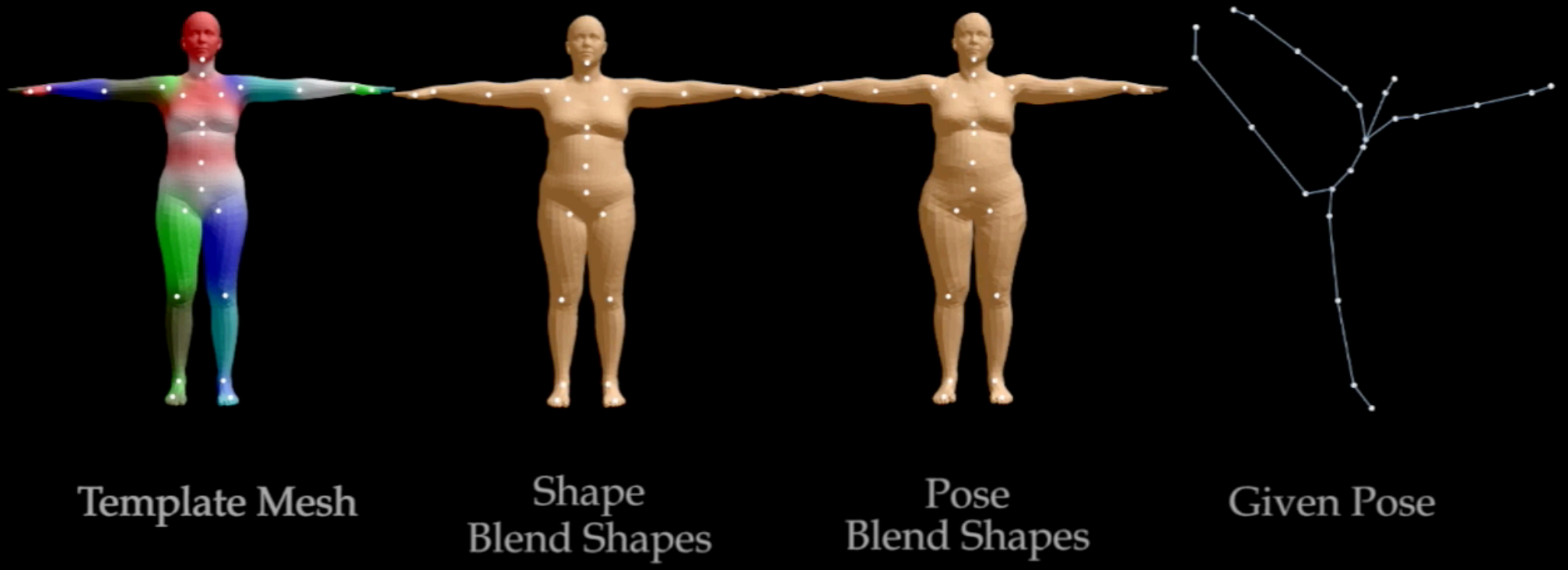


Template Mesh

Shape
Blend Shapes

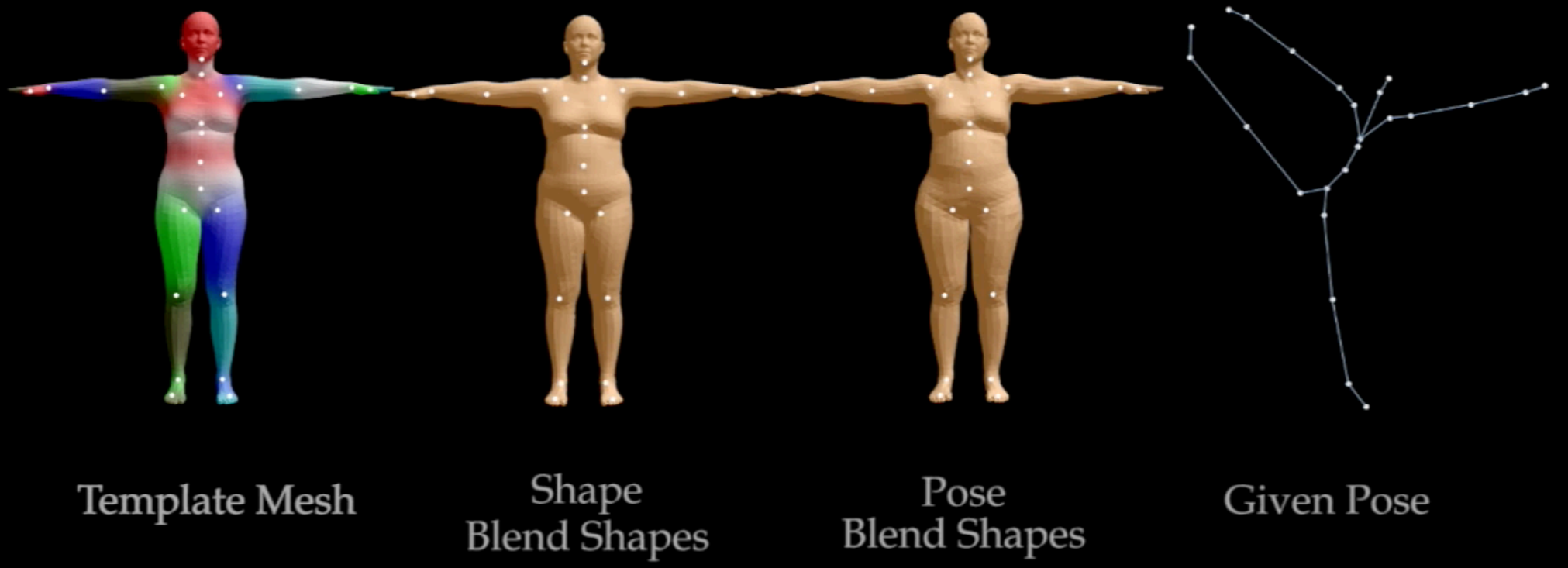
SMPL Summary

SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



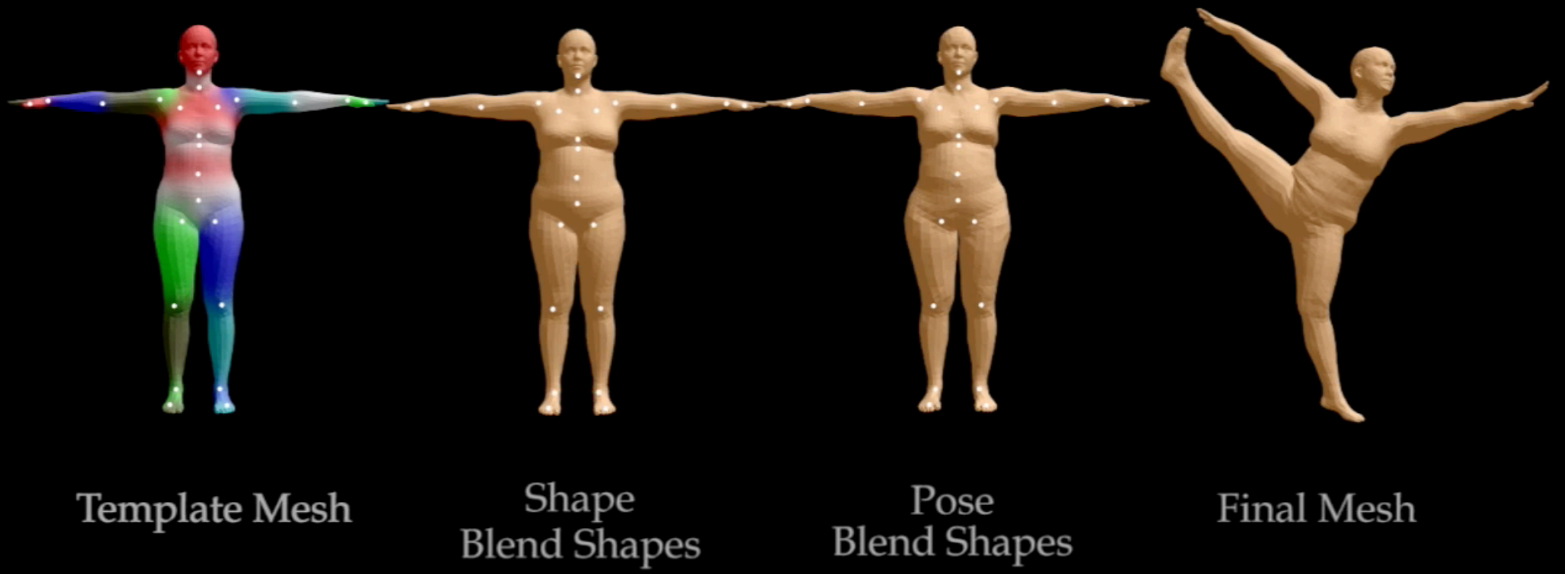
SMPL Summary

SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



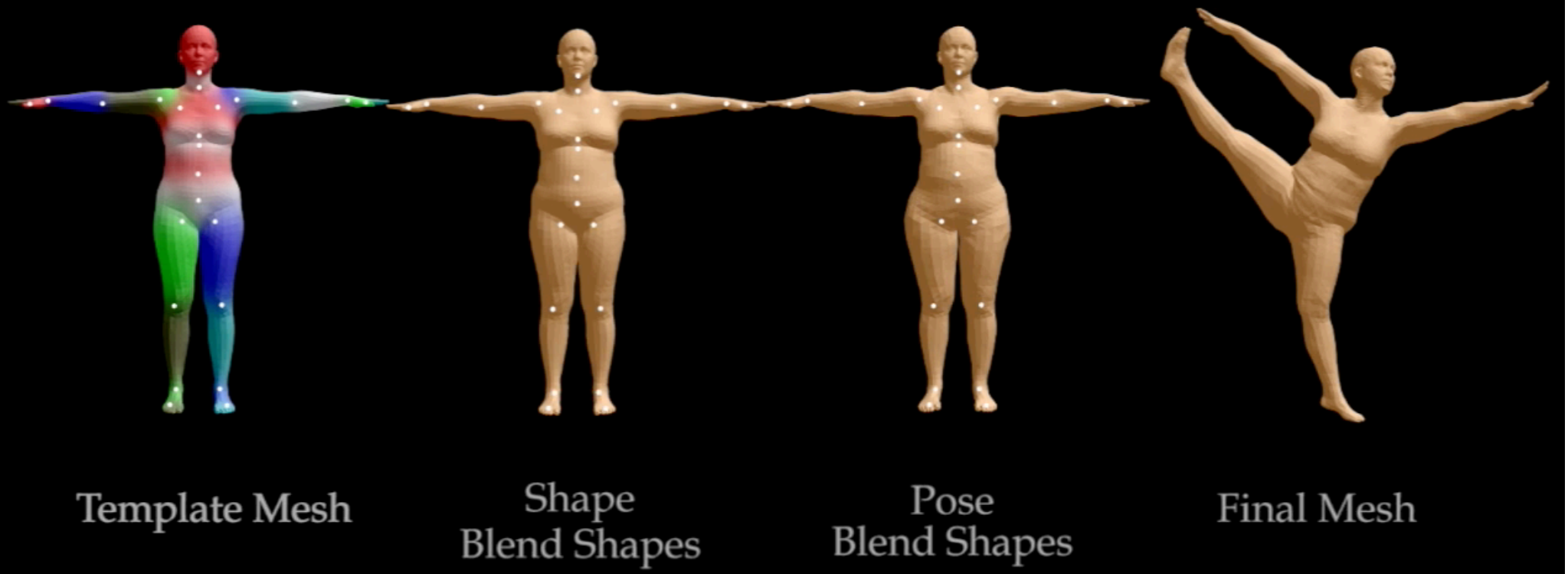
SMPL Summary

SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



SMPL Summary

SMPL is a blend skinned model with blend shapes.
Model parameters are learned from data.



Inverse Kinematics

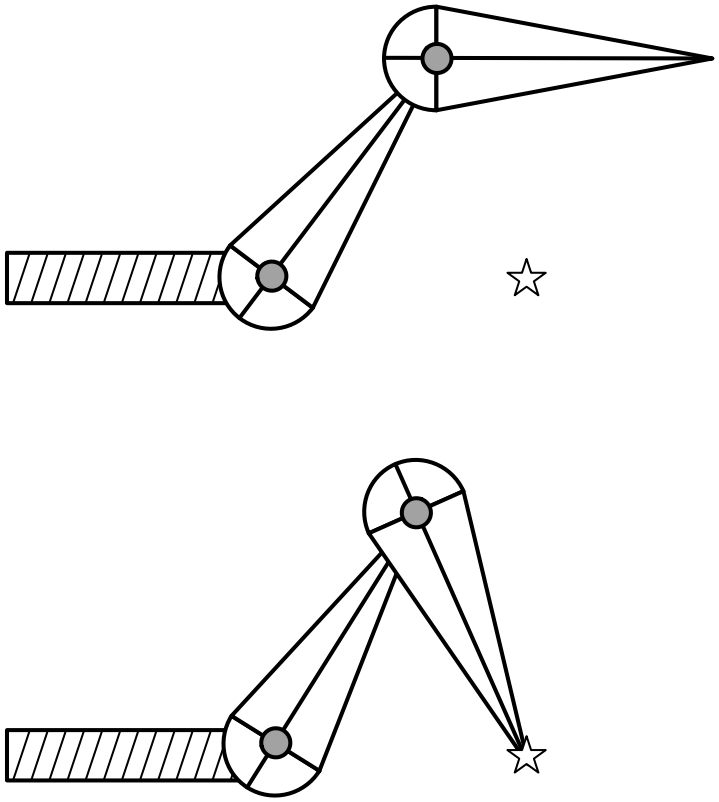
Inverse Kinematics

Given

- Root transformation
- Initial configuration
- Desired end point location

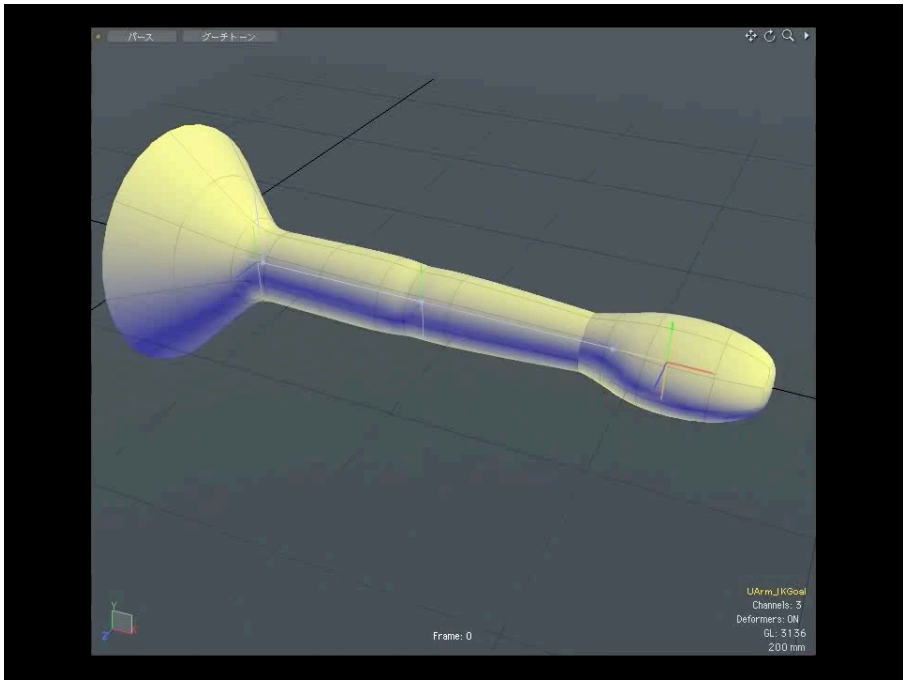
Find

- Interior parameter settings



Inverse Kinematics

Given the end effector position, find the joint angles.

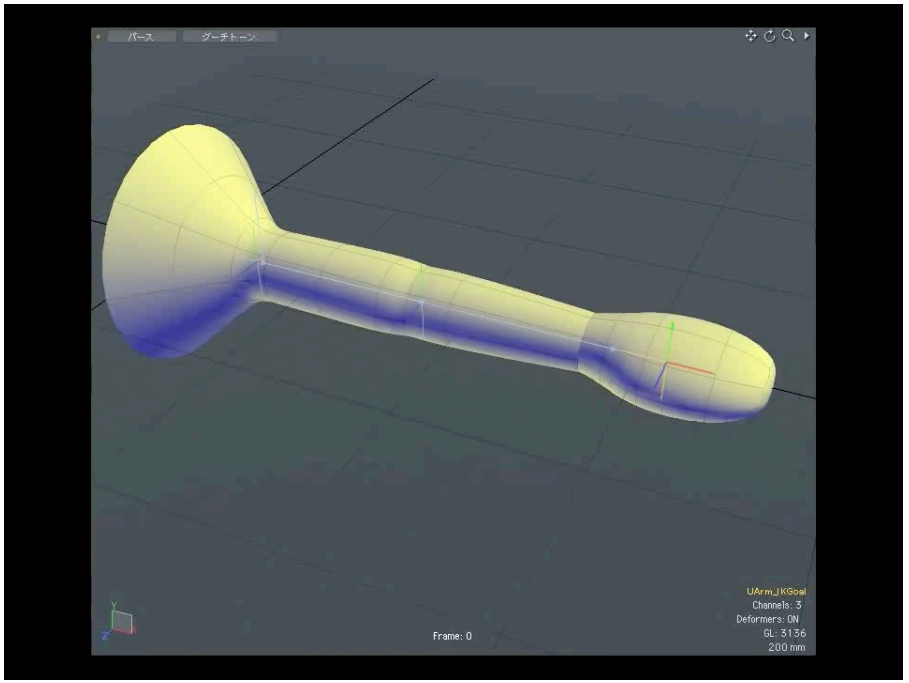


Goals

- Keep end of limb fixed while body moves
- Position end of limb by direct manipulation
- (More general: arbitrary constraints)

Inverse Kinematics

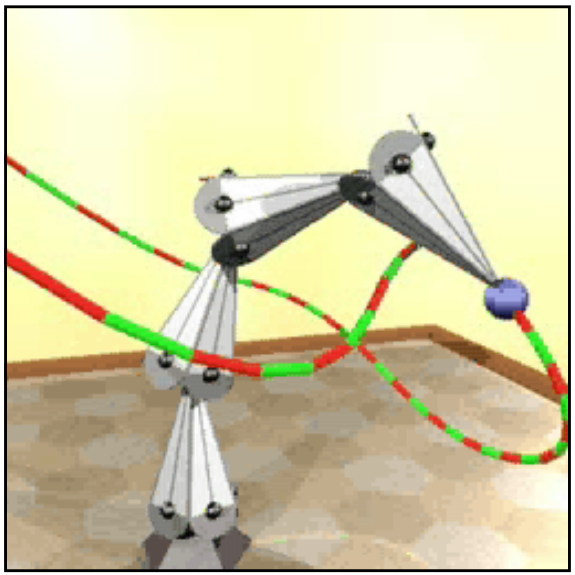
Given the end effector position, find the joint angles.

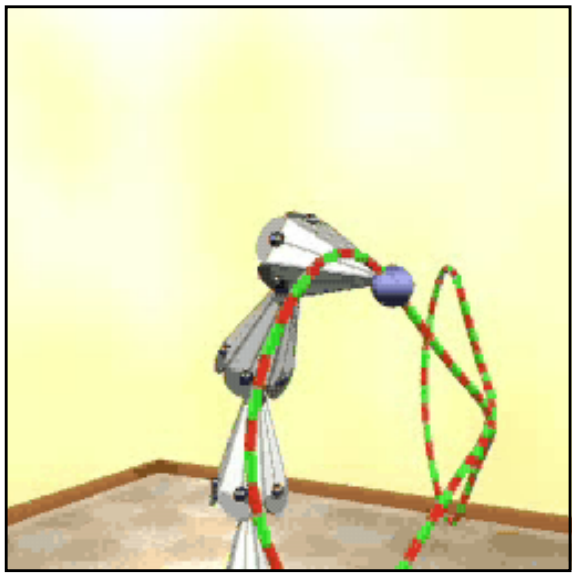


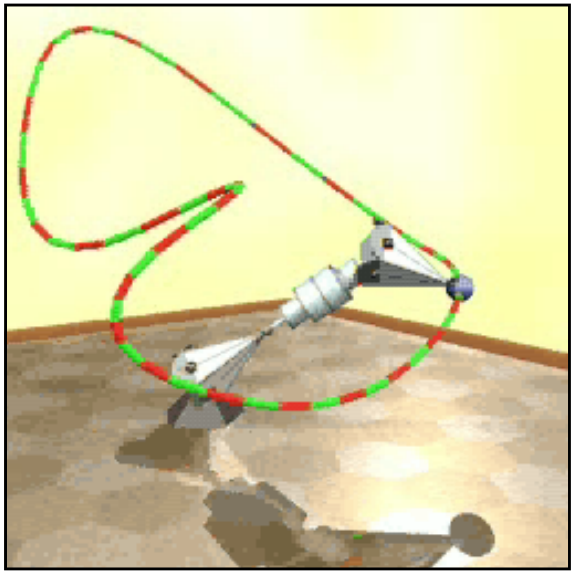
Goals

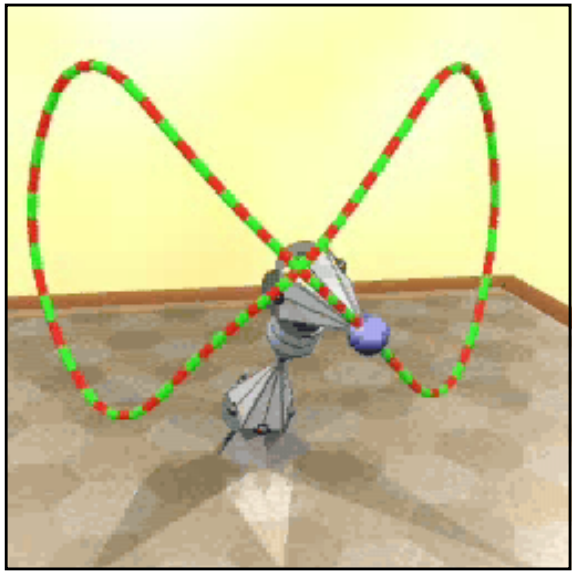
- Keep end of limb fixed while body moves
- Position end of limb by direct manipulation
- (More general: arbitrary constraints)

Inverse Kinematics



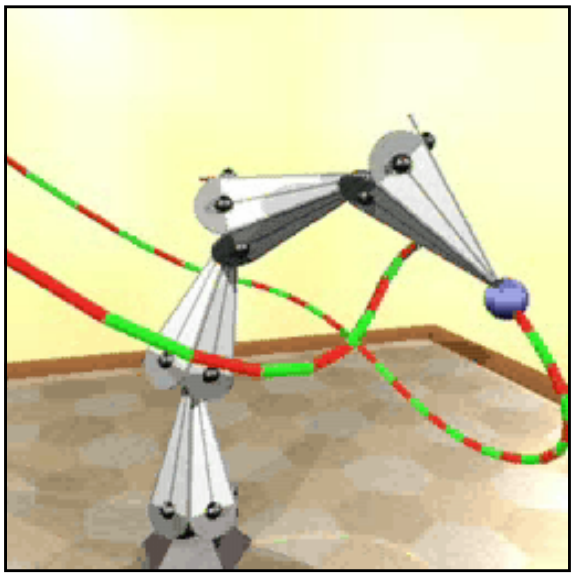


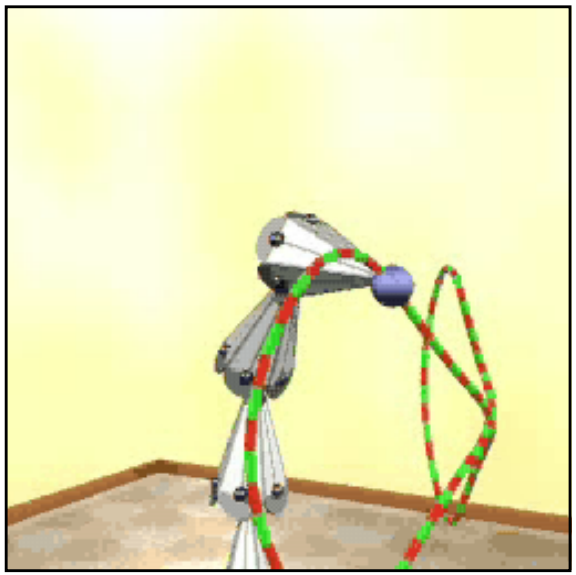


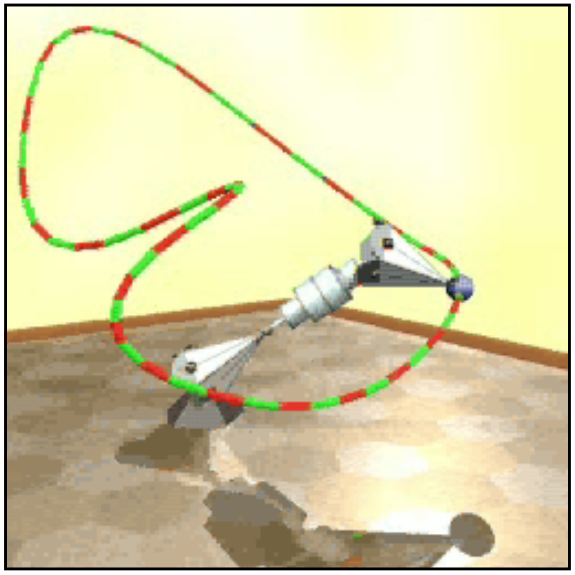


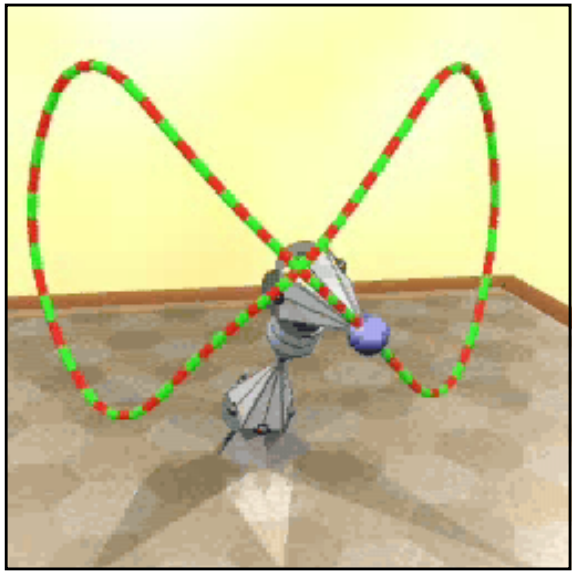
Egon Pasztor

Inverse Kinematics





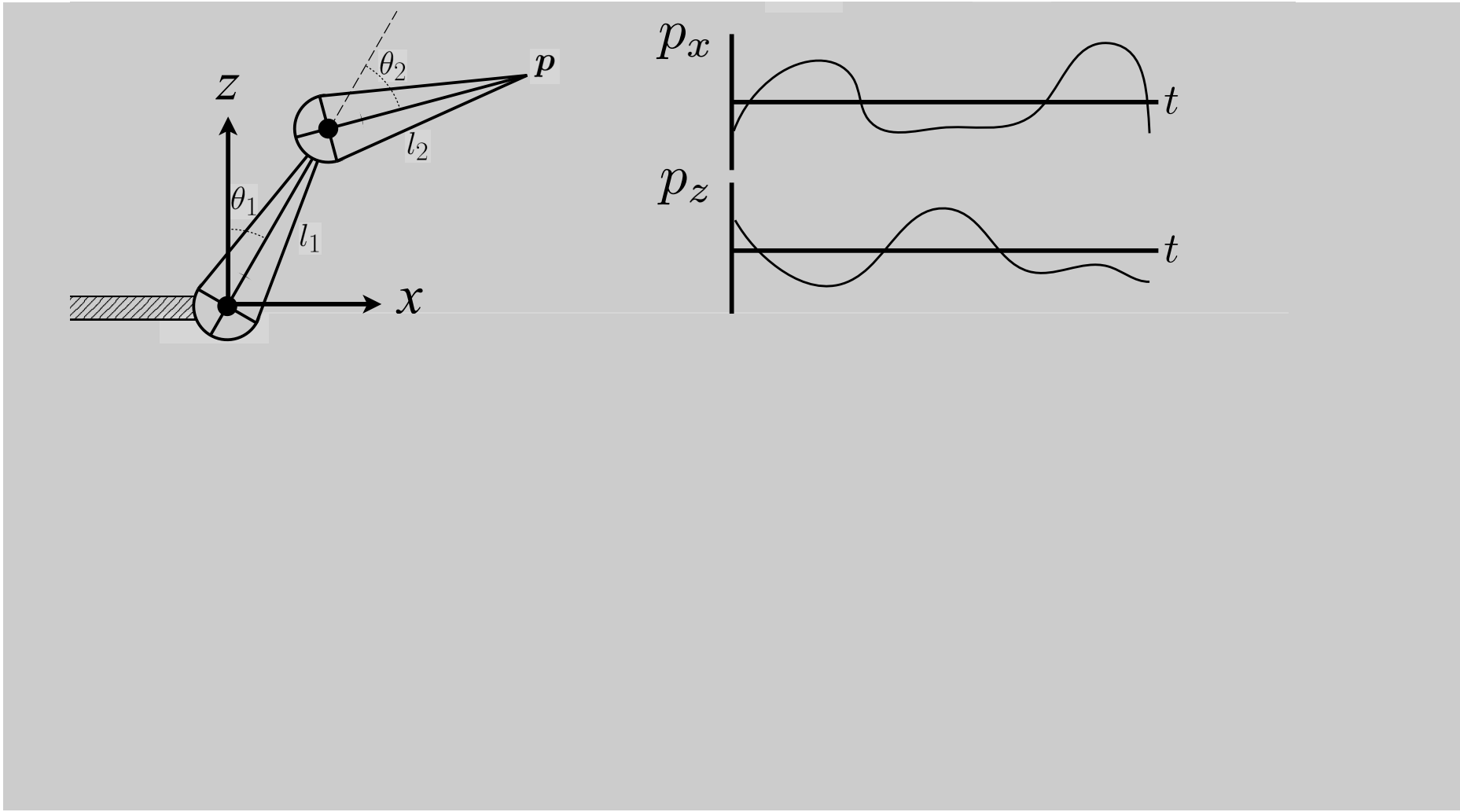




Egon Pasztor

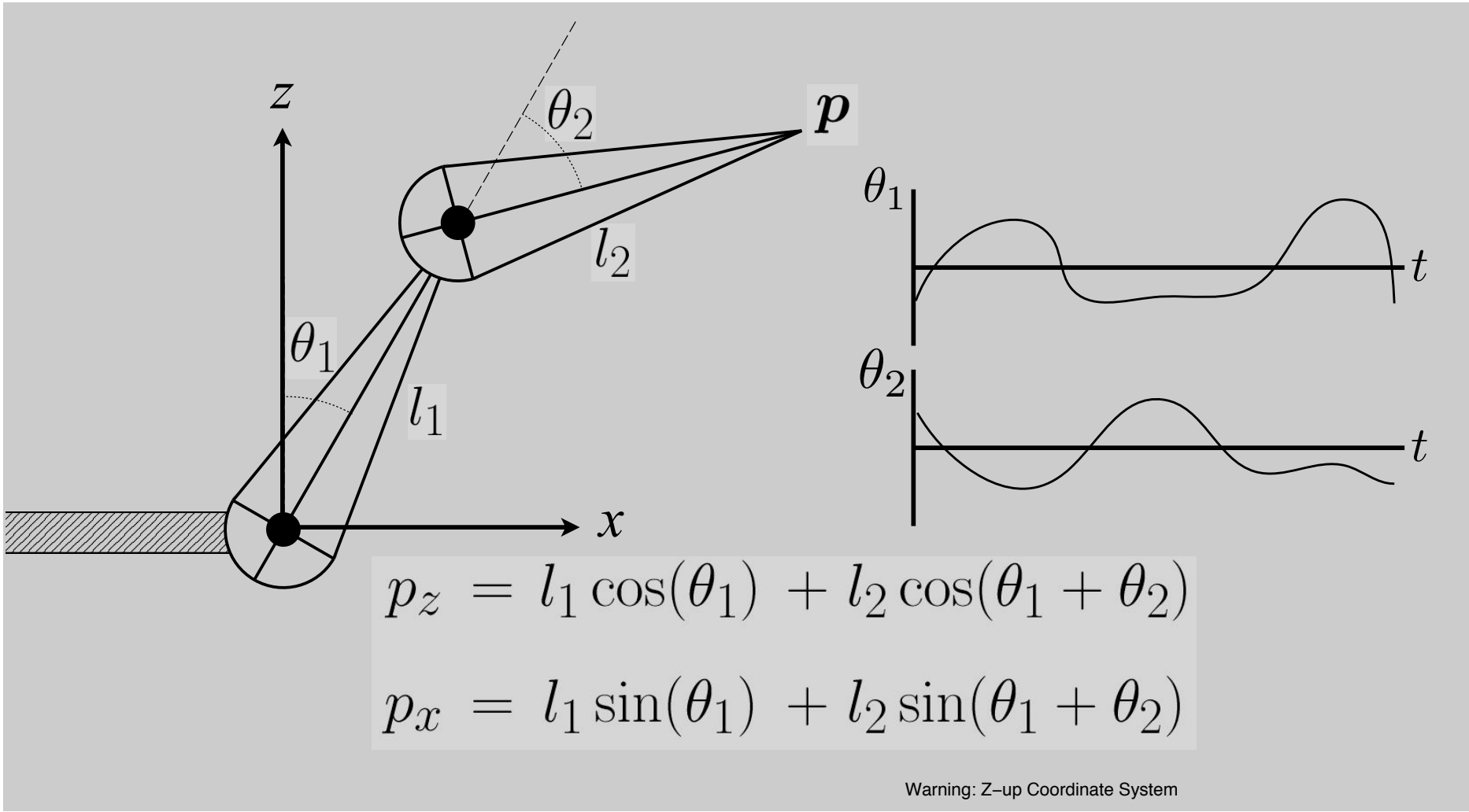
Inverse Kinematics

Animator provides position of end-effector, and computer must determine joint angles that satisfy constraints



Inverse Kinematics

Recall simple example forward kinematic function:



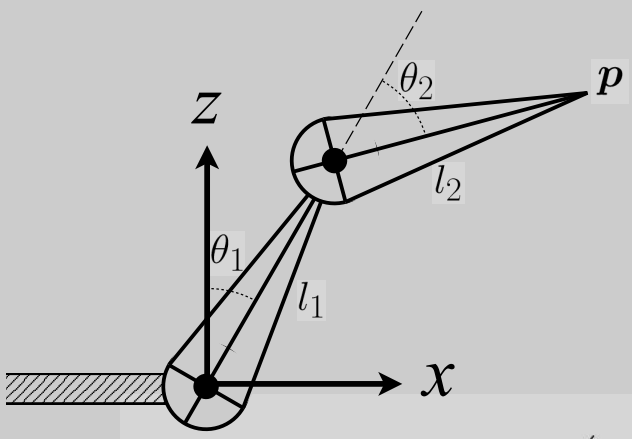
$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

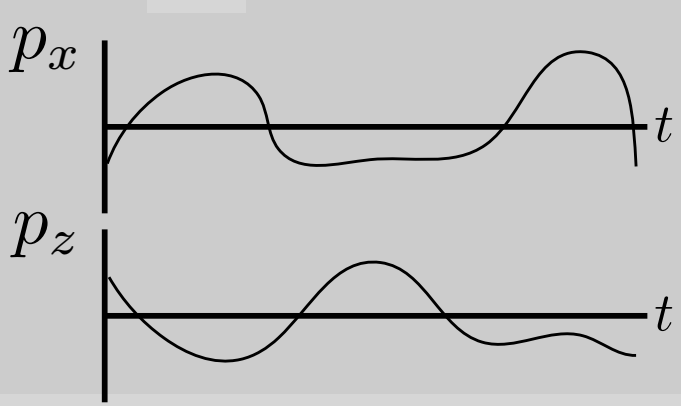
$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Warning: Z–up Coordinate System

Inverse Kinematics

Direct inverse kinematics: for two-segment arm, can solve for parameters analytically





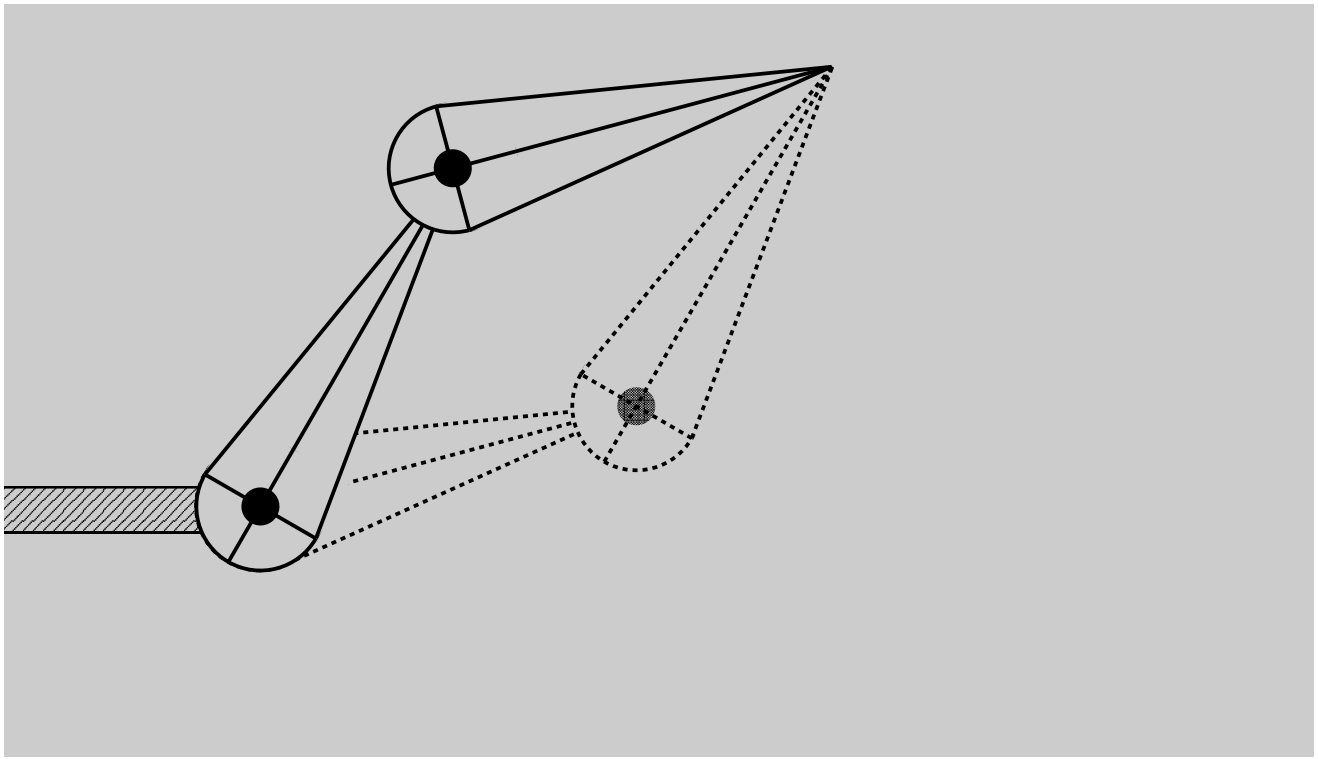
$$\theta_2 = \cos^{-1}\left(\frac{p_z^2 + p_x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\theta_1 = \frac{-p_z l_2 \sin(\theta_2) + p_x(l_1 + l_2\cos(\theta_2))}{p_x l_2 \sin(\theta_2) + p_z(l_1 + l_2\cos(\theta_2))}$$

Inverse Kinematics

Why is the problem hard?

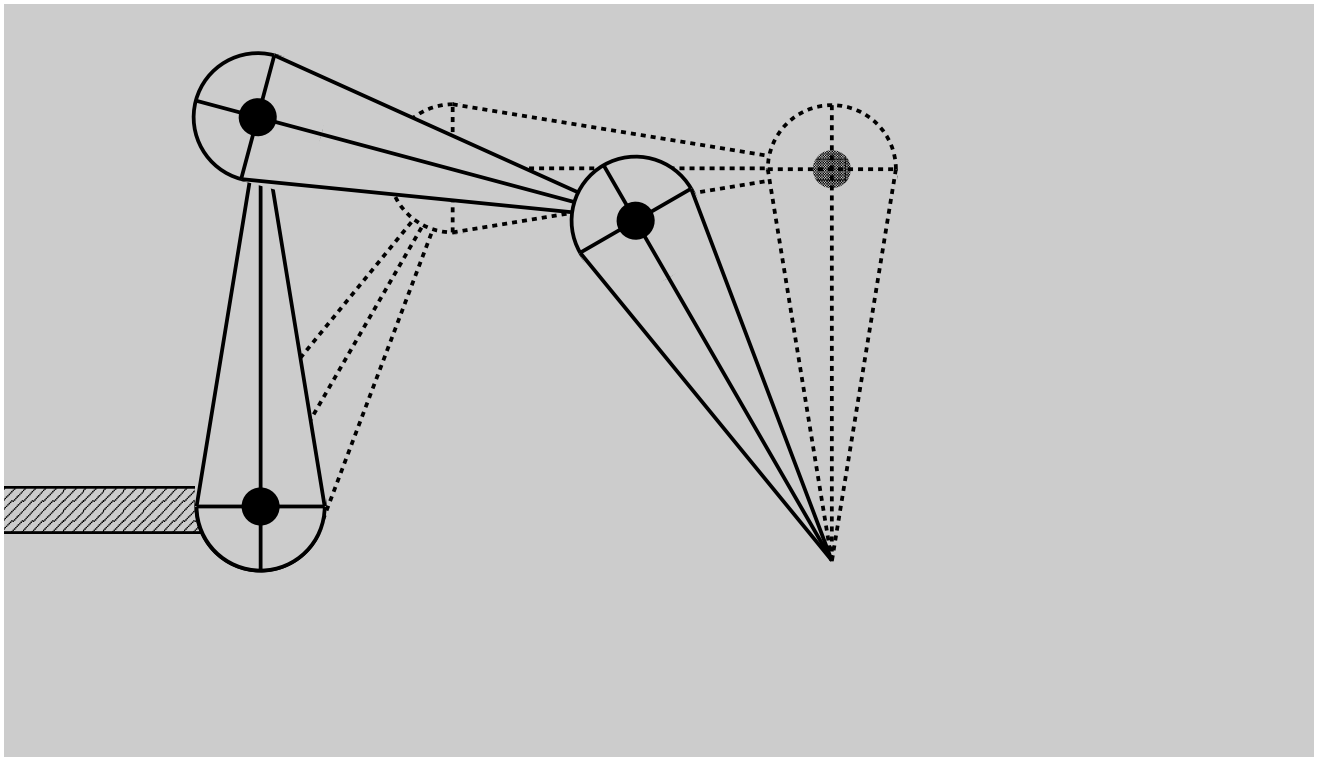
- Multiple solutions separated in configuration space



Inverse Kinematics

Why is the problem hard?

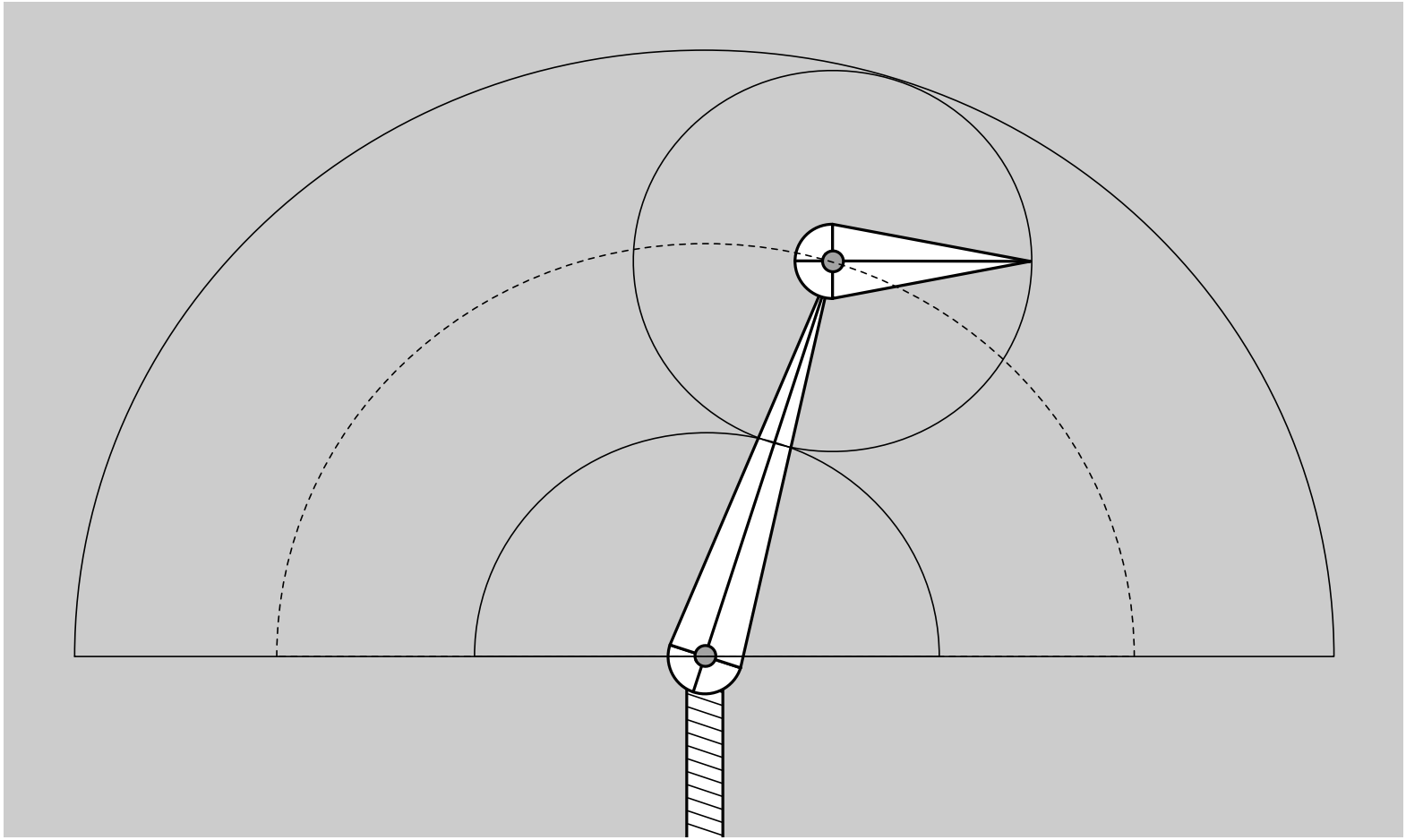
- Multiple solutions connected in configuration space



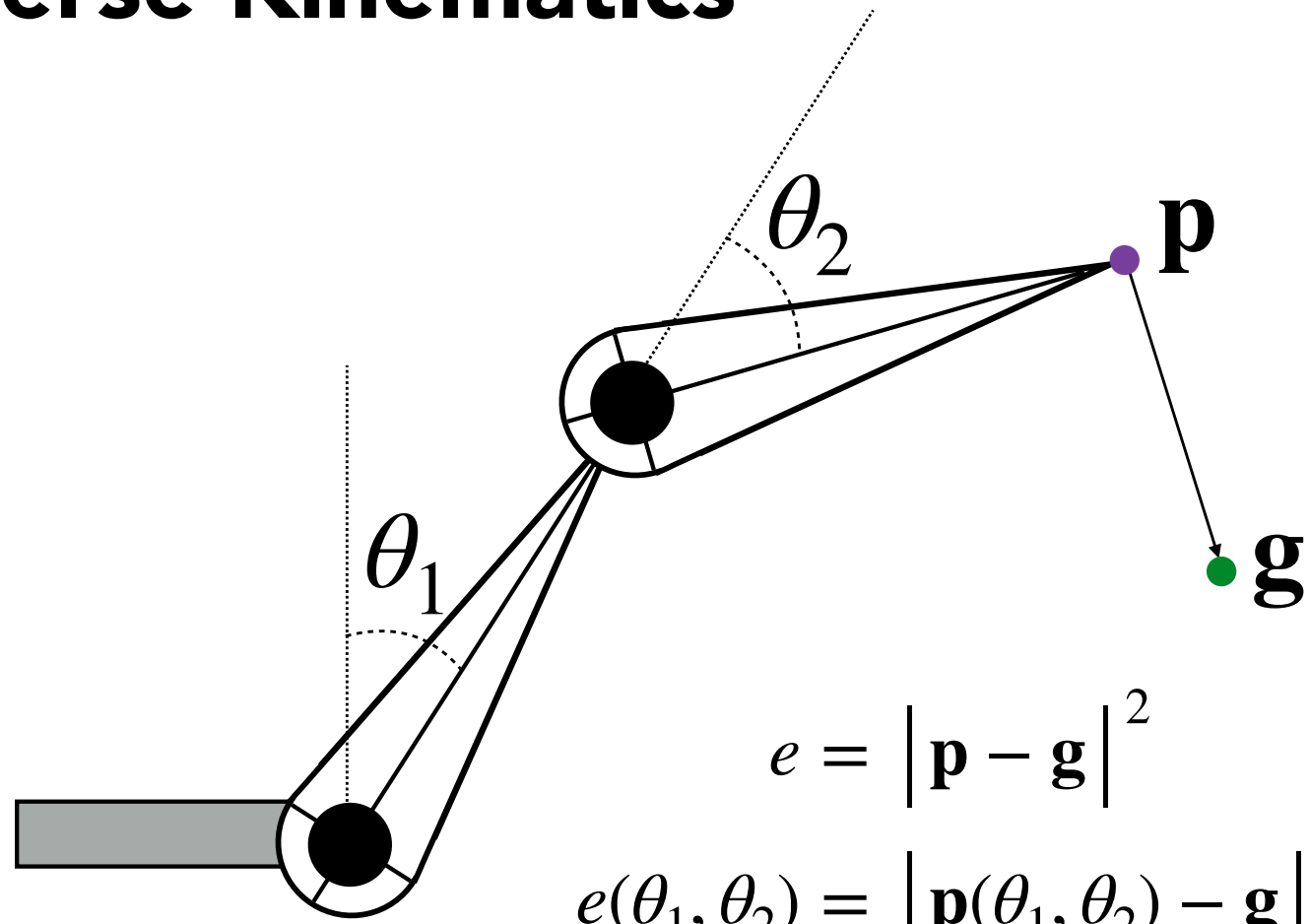
Inverse Kinematics

Why is the problem hard?

- Solutions may not always exist



Inverse Kinematics



$$e = \left|\mathbf{p} - \mathbf{g}\right|^2$$

$$e(\theta_1, \theta_2) = \left|\mathbf{p}(\theta_1, \theta_2) - \mathbf{g}\right|^2$$

Minimize e to get solution.

Inverse Kinematics

Numerical solution to general N-link IK problem

- Choose an initial configuration
- Define an error metric (e.g. square of distance between goal and current position)
- Compute gradient of error as function of configuration
- Apply gradient descent (or Newton's method, or other optimization procedure)

Inverse Kinematics

Cyclic Coordinate Decent (easy, not great)

- For each joint, check if error decreases by going either + or - one degree, take whichever is better
- Keep cycling through the joints (degrees of freedom) until no progress is made.

Stabilized Newton Iterations (harder, but great)

- Use Newton's method with line search.

Gradient Decent (easy with ML tools!)

- Go in gradient direction with line search

Issues:

- Speed
- Quality of result
- Starting point
- Single or varied solutions for same goal

Inverse Kinematics

Generalize to other constraints

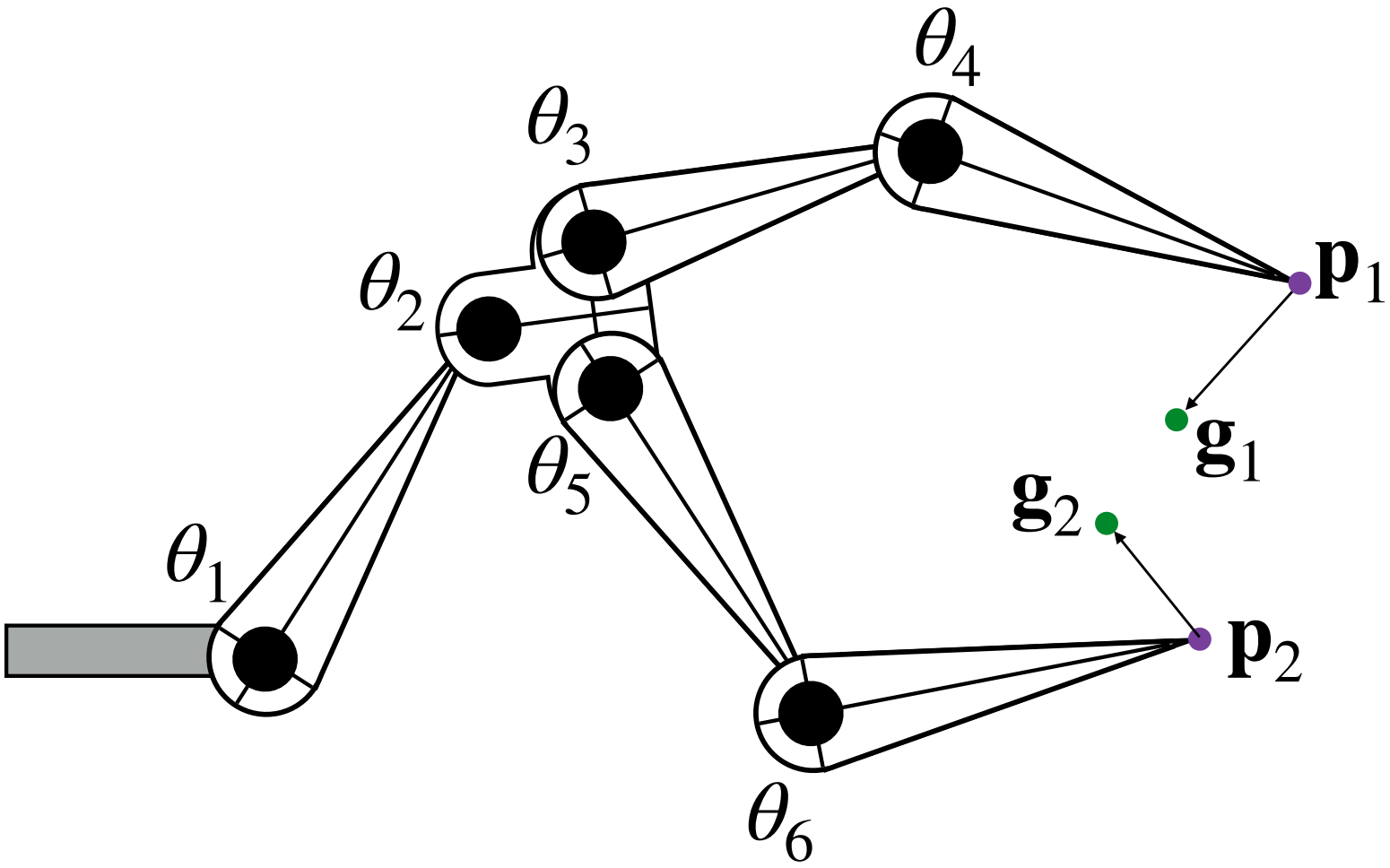
- Balance
- Gaze direction
- Foot plant (no sliding)
- Animation (combine with curves)

Combine multiple Constraints

Same general approach

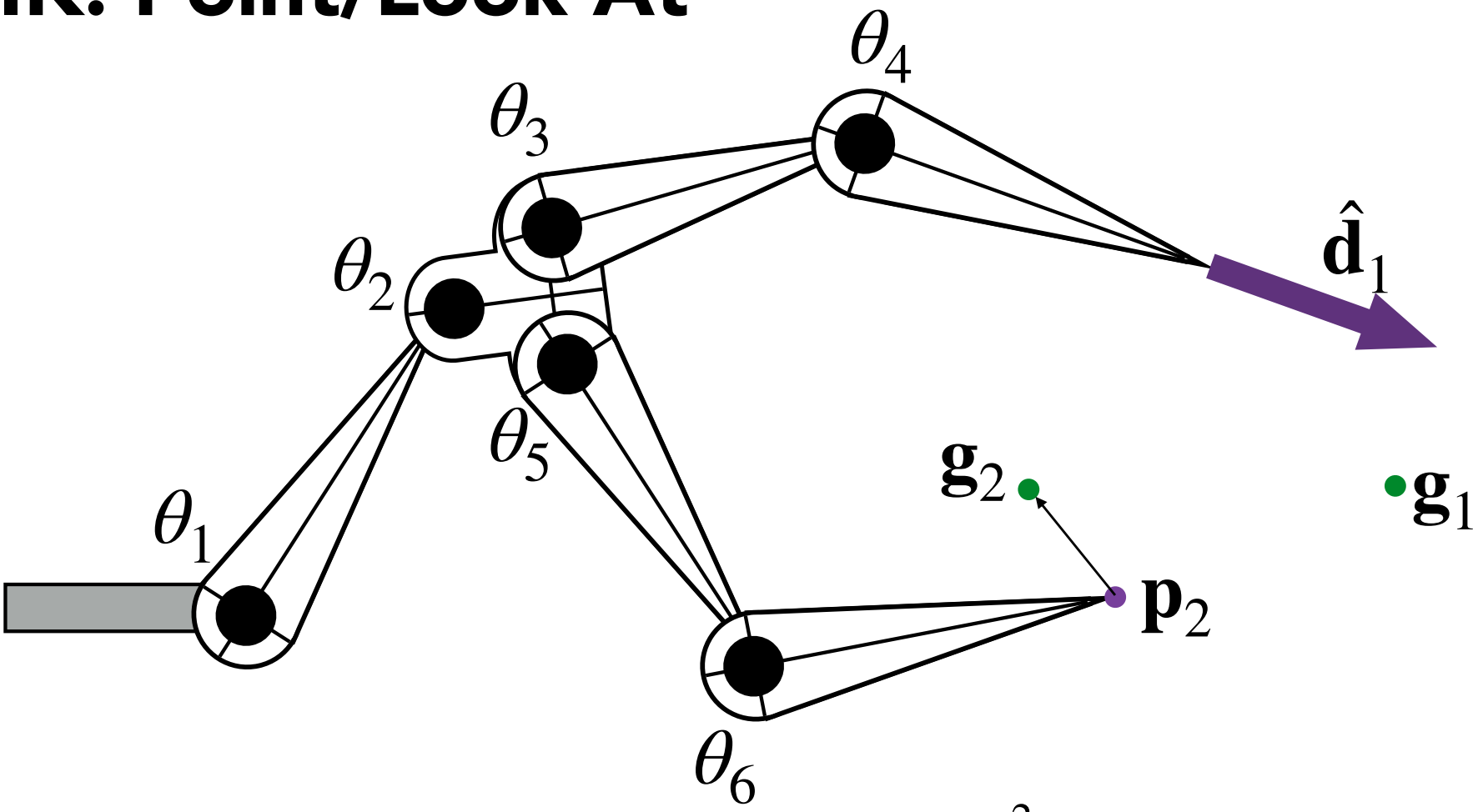
- Define error terms
- Do something to minimize the error terms
  - Cyclic Coordinate Decent, Gradient Decent, Newton, etc.

IK: Multiple Constraints



$$e(\boldsymbol{\theta}) = w_1 \left| \mathbf{p}_1(\boldsymbol{\theta}) - \mathbf{g}_1 \right|^2 + w_2 \left| \mathbf{p}_2(\boldsymbol{\theta}) - \mathbf{g}_2 \right|^2$$

IK: Point/Look At

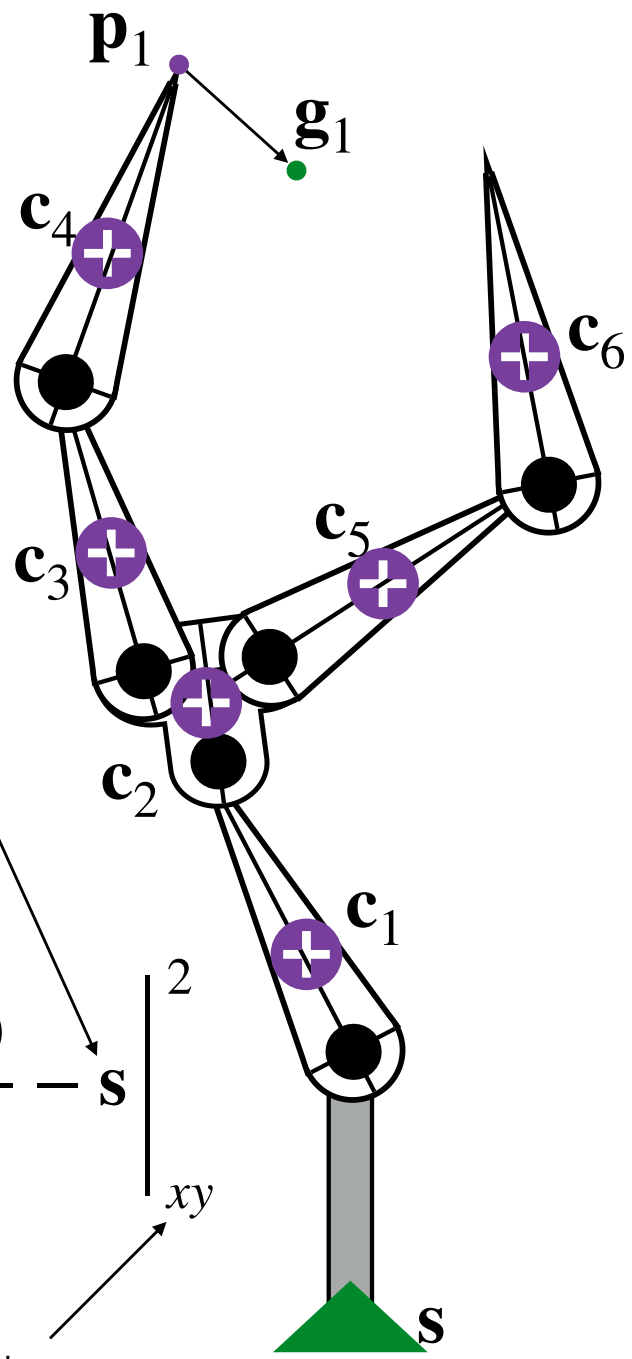


$$e(\boldsymbol{\theta}) = w_1 \left| 1 - \left(\frac{\mathbf{p}_1(\boldsymbol{\theta}) - \mathbf{g}_1}{\left| \mathbf{p}_1(\boldsymbol{\theta}) - \mathbf{g}_1 \right|} \right) \cdot \hat{\mathbf{d}} \right|^2 + w_2 \left| \mathbf{p}_2(\boldsymbol{\theta}) - \mathbf{g}_2 \right|^2$$

IK: Balance

Keep COM over support polygon

Support polygon is convex hull of ground contacts



Either center of support polygon or nearest point

body COM

$$e(\boldsymbol{\theta}) = w_1 \left| \mathbf{p}_1(\boldsymbol{\theta}) - \mathbf{g}_1 \right|^2 + w_2 \left| \frac{\sum m_i \mathbf{c}_i(\boldsymbol{\theta})}{\sum m_i} - \mathbf{s} \right|^2_{xy}$$

mass

projected in to XY plane

IK: Keyframing

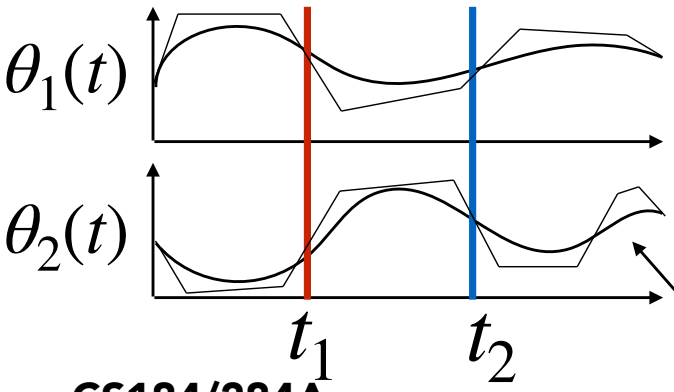
- **With key-framed animation, the θ are not independant variables:**

$$\theta_i(t) = \sum_j c_{ij} B_j(t)$$

Control points for θ_i

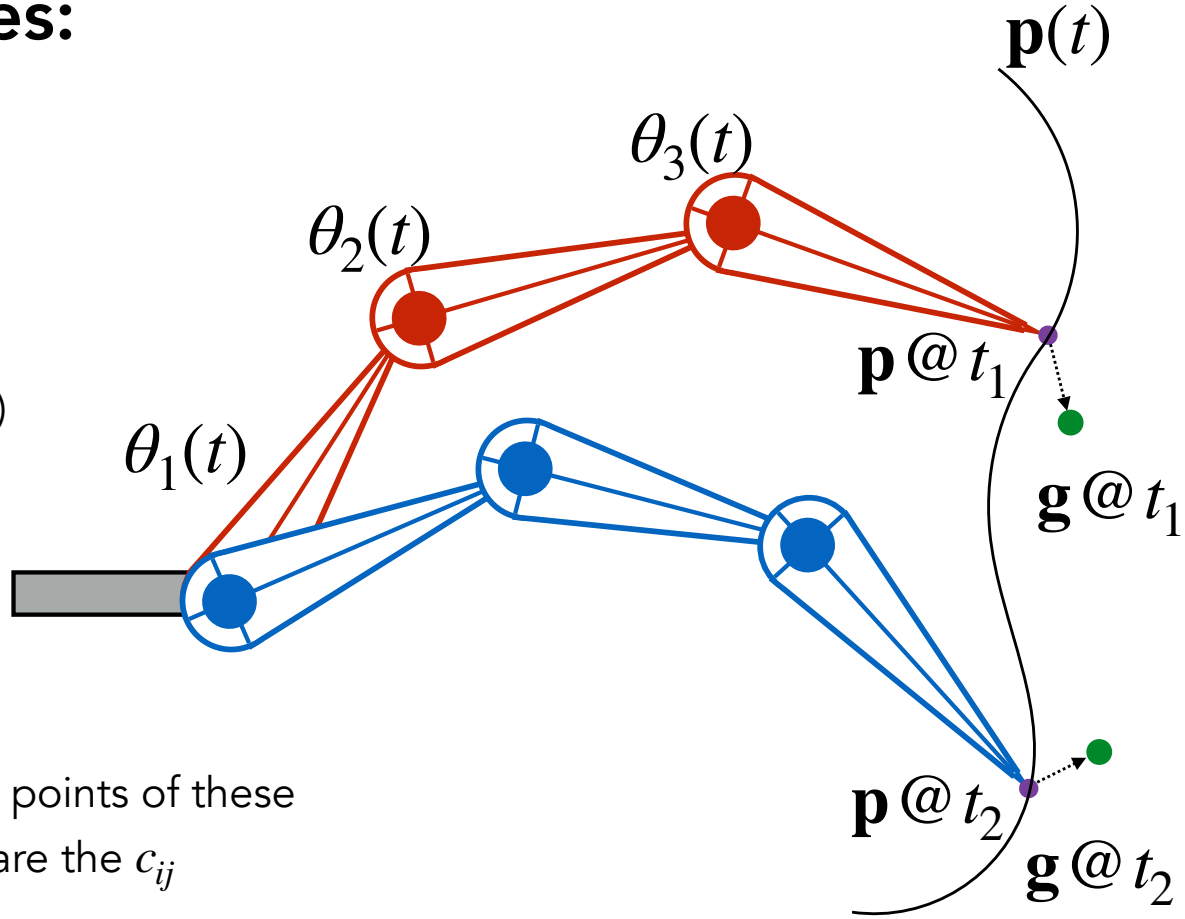
Basis functions (*e.g.* Bézier)

Joint angle is function of time (and control points)



Control points of these curves are the c_{ij}

Adjust c_{ij} to hit constraints.



Inverse Kinematics

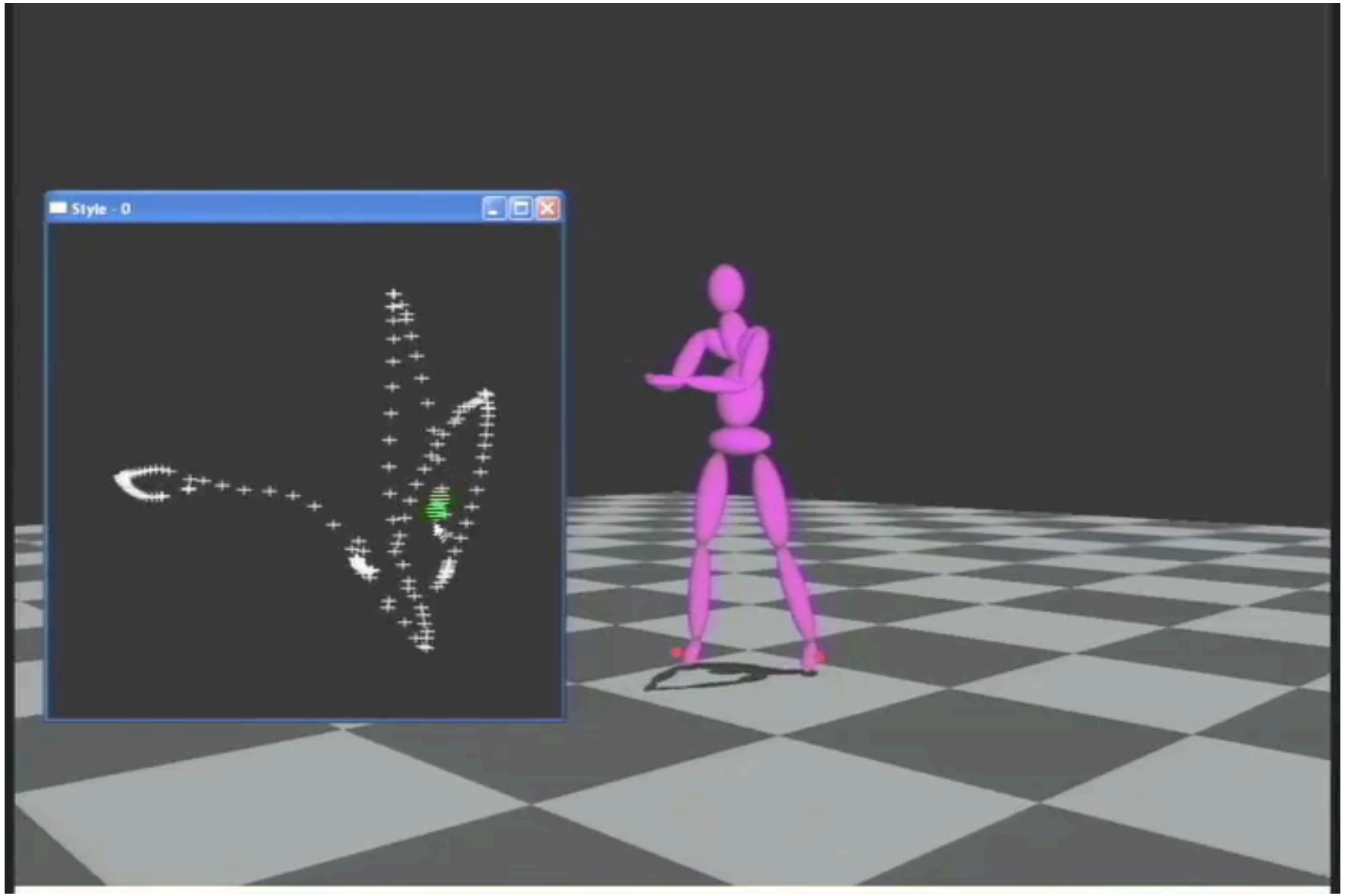
Extra material for 284A

Stabilized Newton

CS184/284A

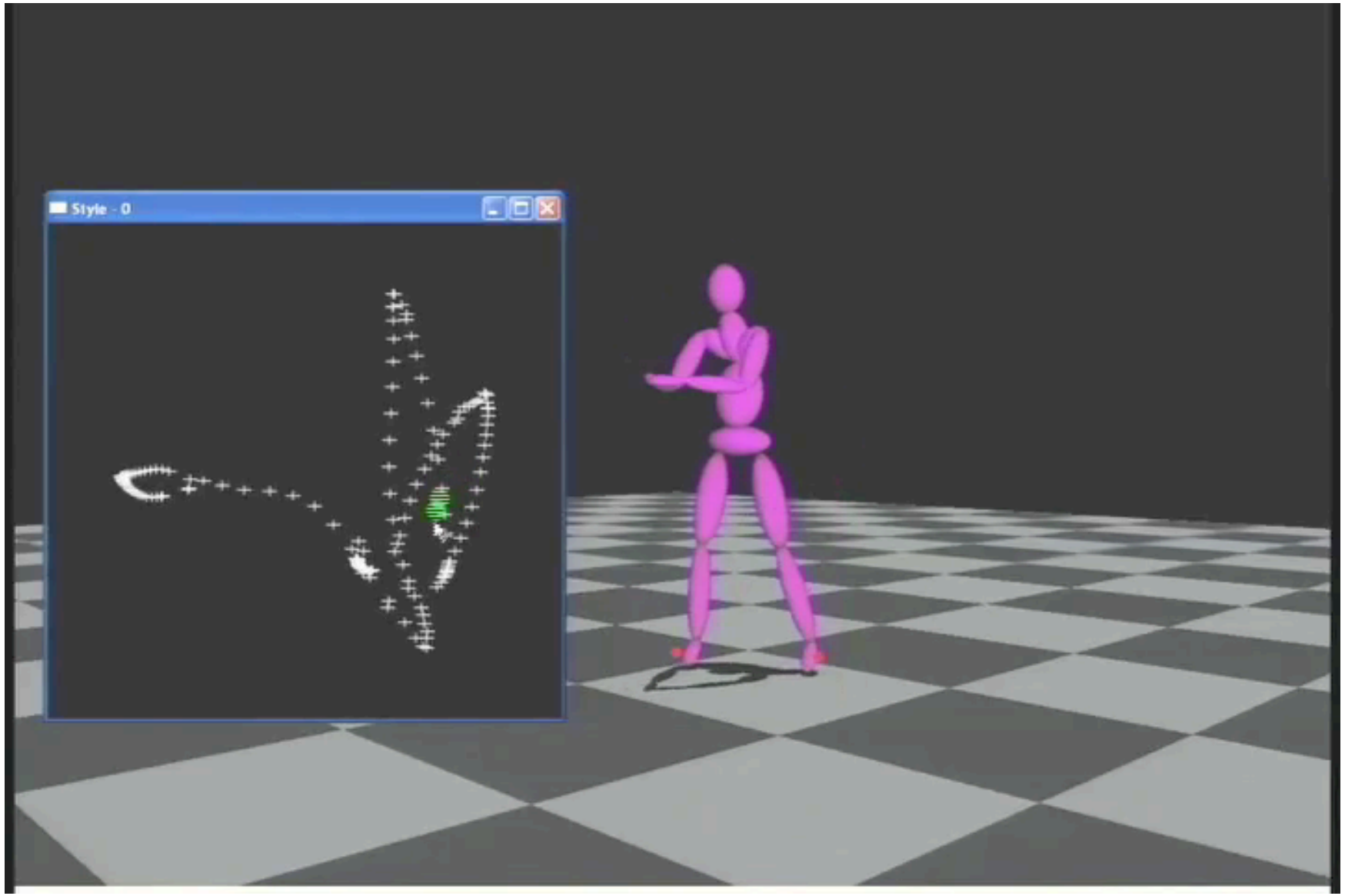
Ren Ng, James O'Brien

Style-Based IK



Grochow et al., Style Based Inverse Kinematics

Style-Based IK



Grochow et al., Style Based Inverse Kinematics

Kinematic Animation Pros and Cons

Strengths

- Direct control is convenient
- Implementation is straightforward

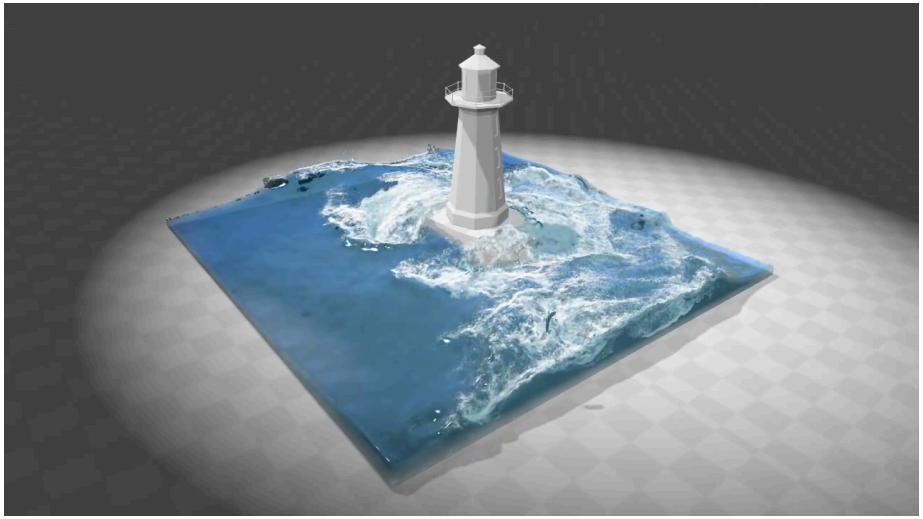
Weaknesses

- Animation may be inconsistent with physics
- Time consuming for artists

IK is often used inside of other methods

- *e.g.* planner computes desired foot placement and uses IK to compute appropriate configuration

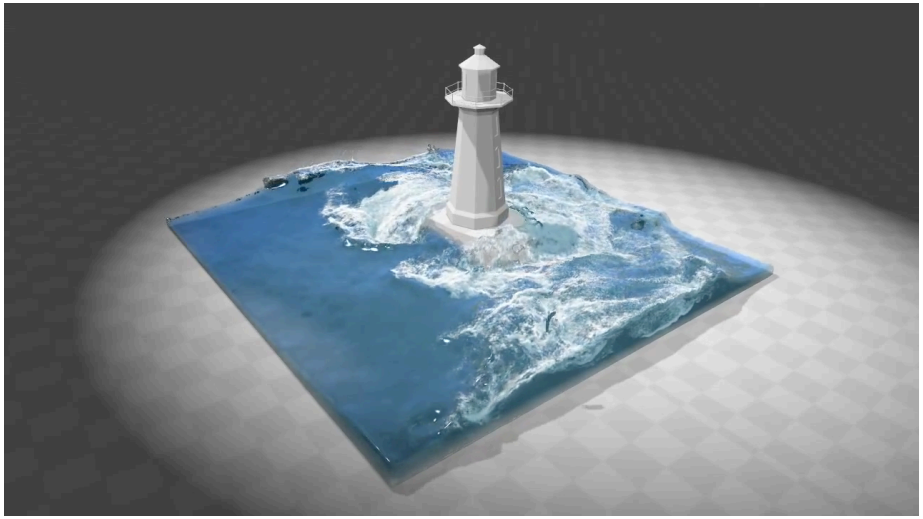
Next Time: Physical Simulation



CS184/284A

Ren Ng, James O'Brien

Next Time: Physical Simulation



CS184/284A **Ren Ng, James O'Brien**

Acknowledgments

This slide set contain contributions from:

- Kayvon Fatahalian
- David Forsyth
- Pat Hanrahan
- Angjoo Kanazawa
- Steve Marschner
- Ren Ng
- James O'Brien
- Mark Pauly