Lecture 07

Skinning

Libin Liu

School of Intelligence Science and Technology
Peking University

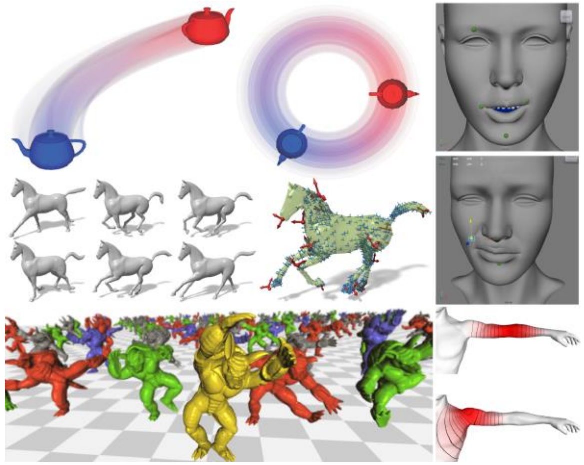
GAMES105 课程交流

VCL @ PKU

Outline

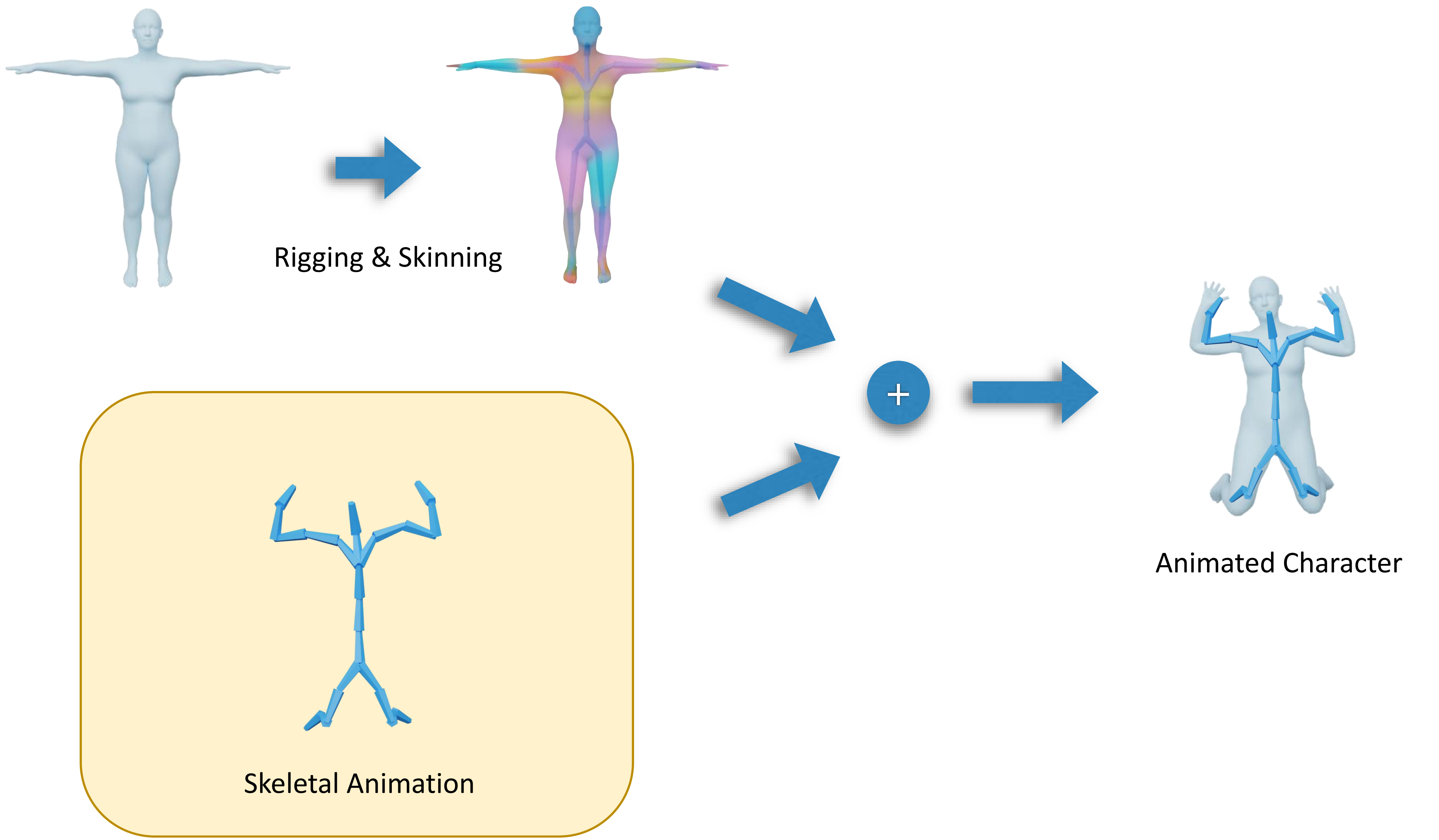
- Skinning
  - Linear Blend Skinning (LBS)
  - Dual Quaternion Skinning (DQS)
  - Blendshapes

- Examples:
  - The SMPL model
  - Facial Animation

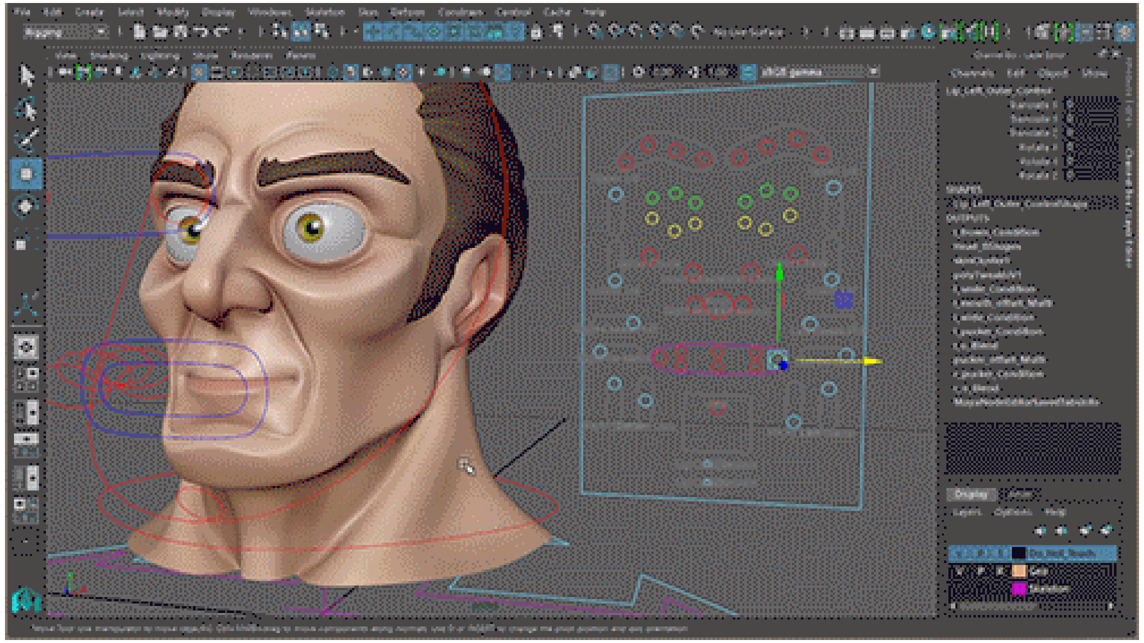


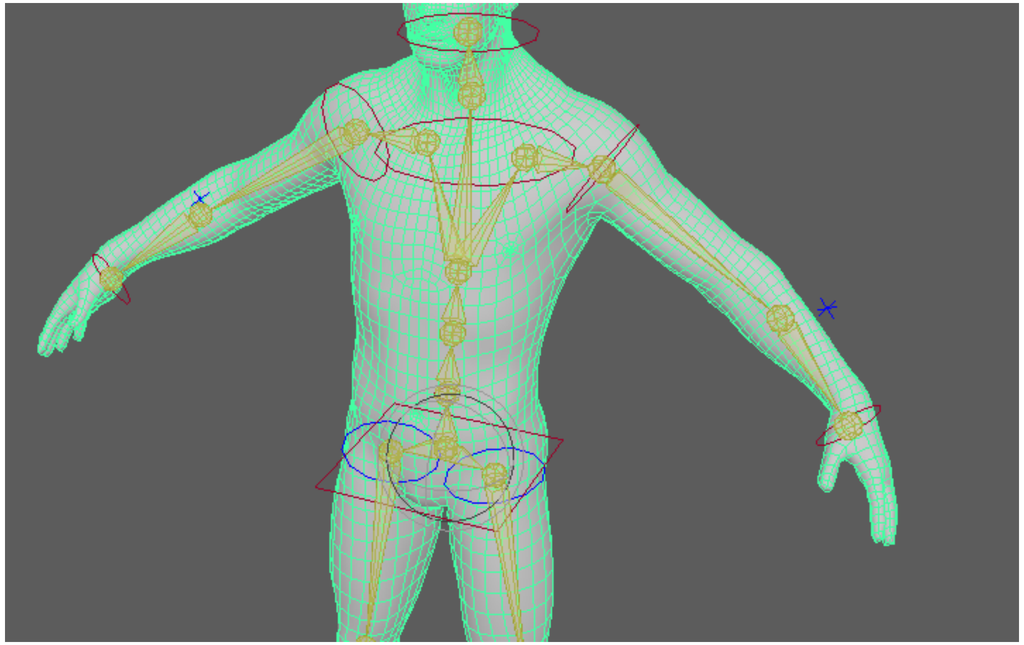
Many images are from: https://skinning.org/
Alec Jacobson, Zhigang Deng, Ladislav Kavan, and J. P. Lewis. 2014.
Skinning: real-time shape deformation*.*
In ACM SIGGRAPH 2014 Courses (SIGGRAPH '14)

Character Animation Pipeline



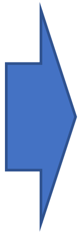
Rigging & Skinning

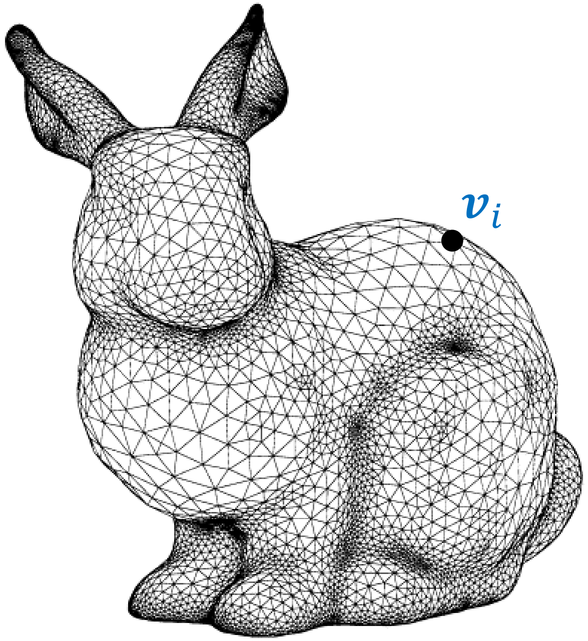




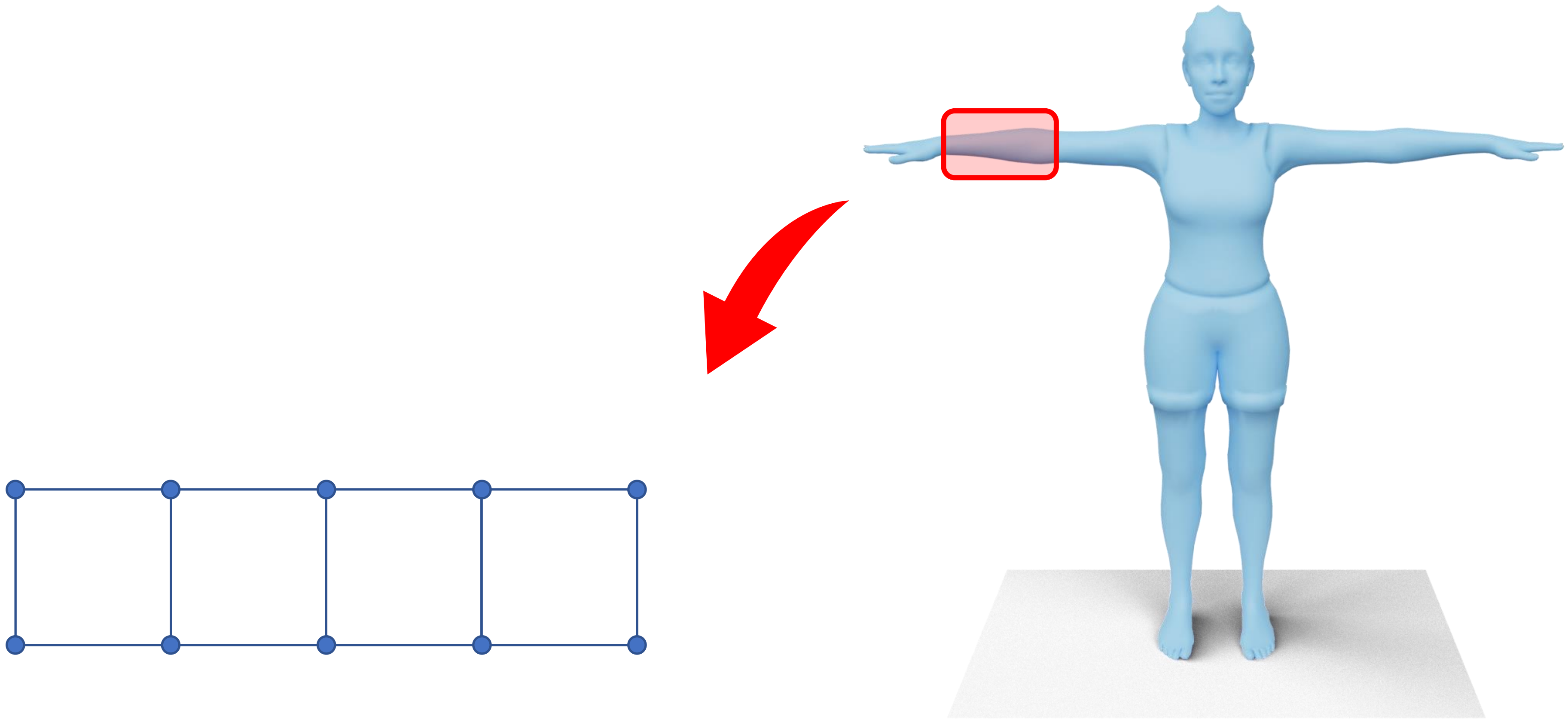
Mesh Representation

The "Stanford Bunny"

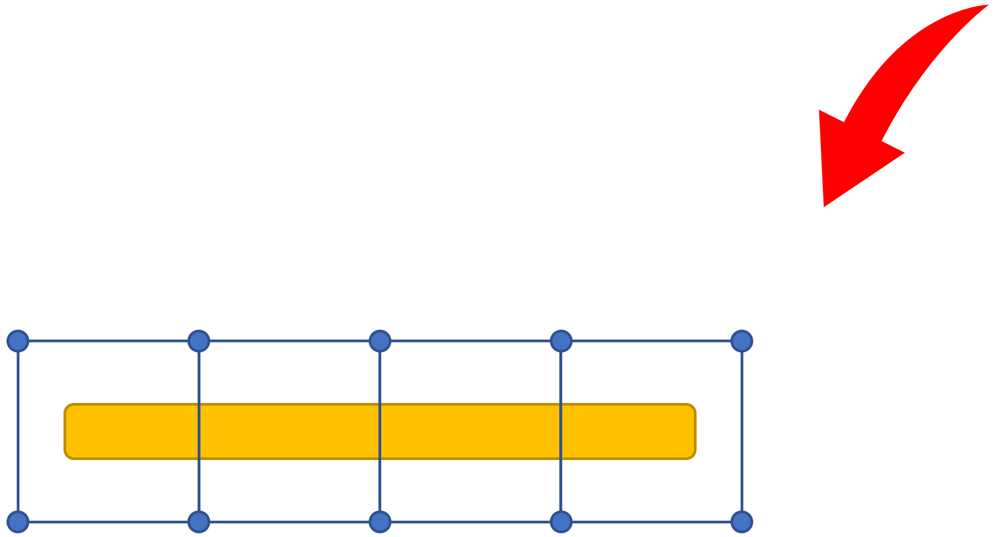


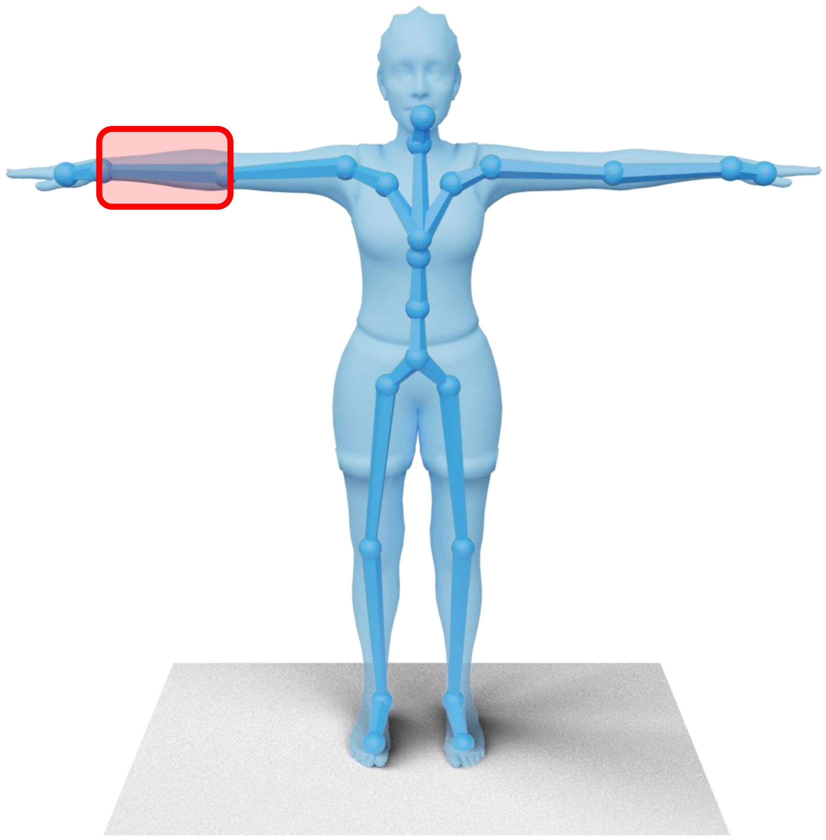


Skinning Deformation

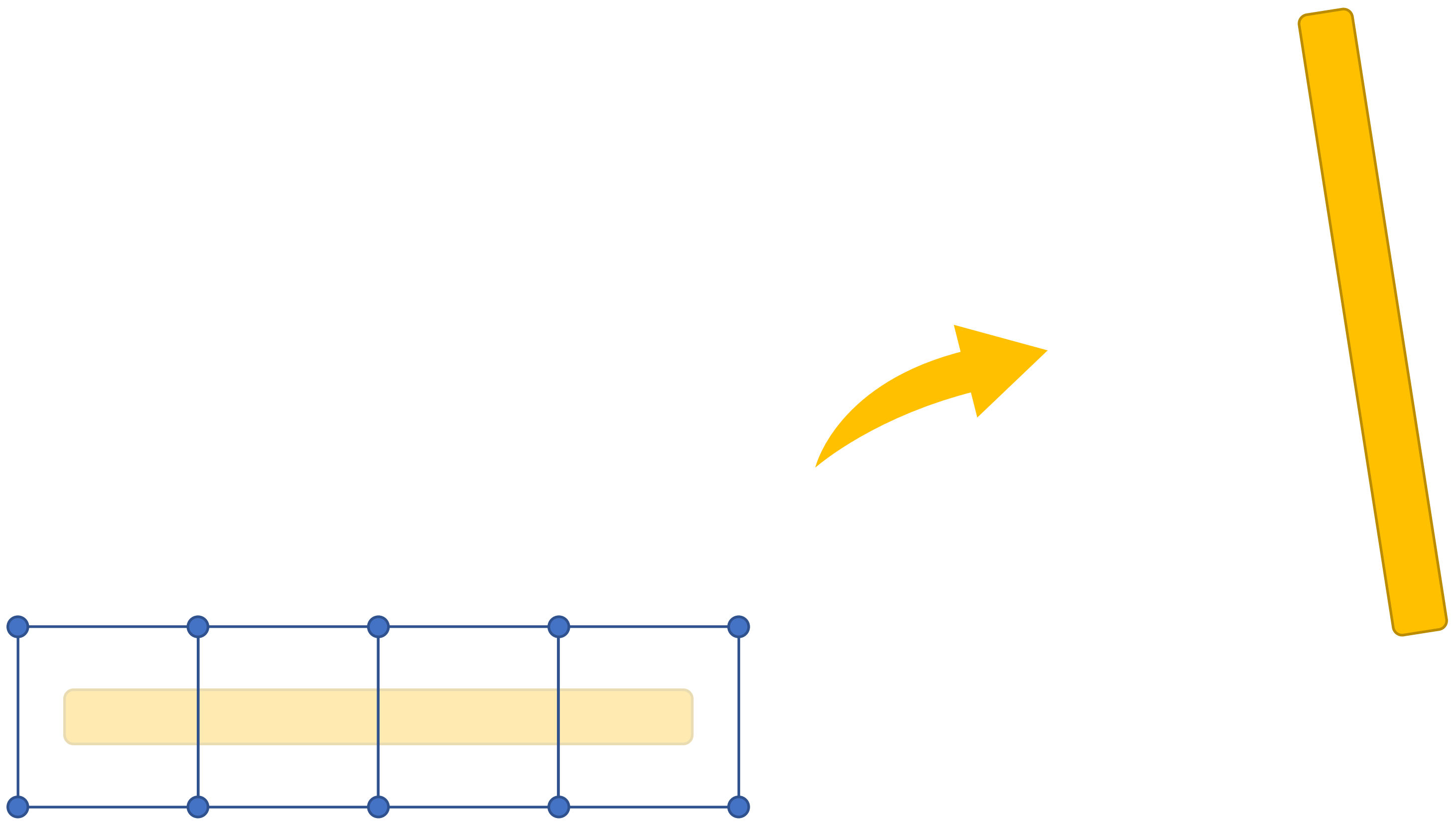


Skinning Deformation

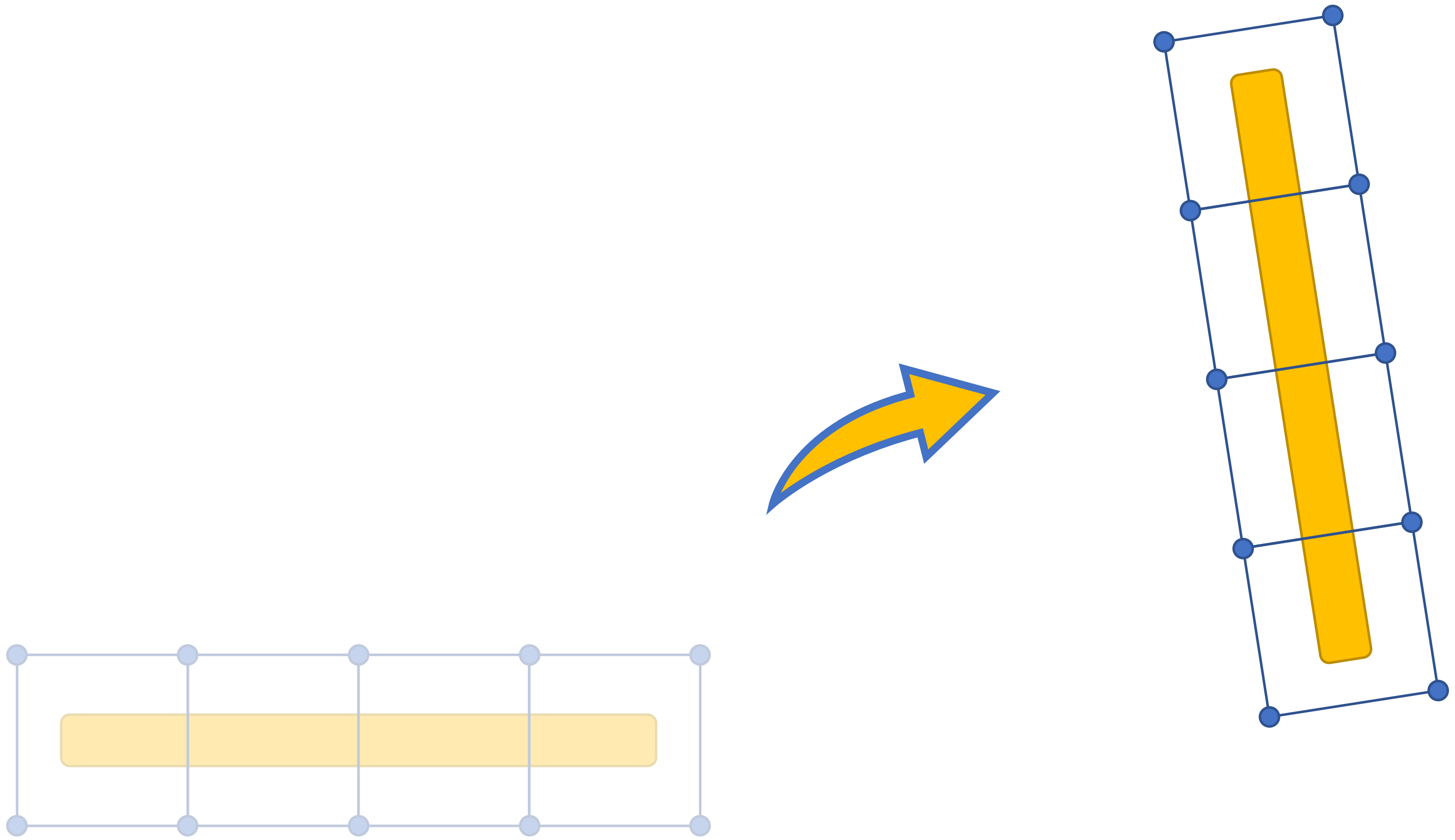




Skinning Deformation

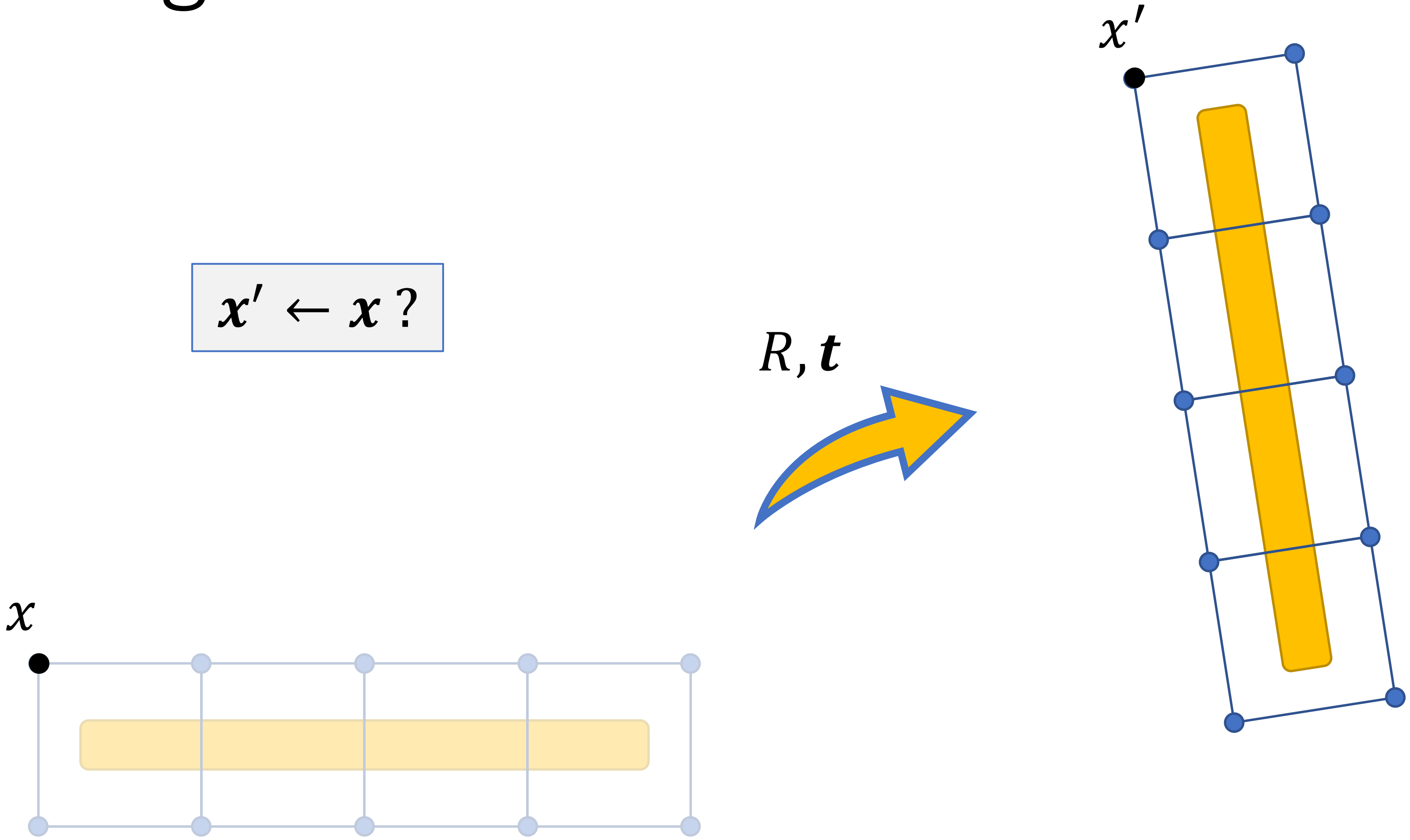


Skinning Deformation



Skinning Deformation

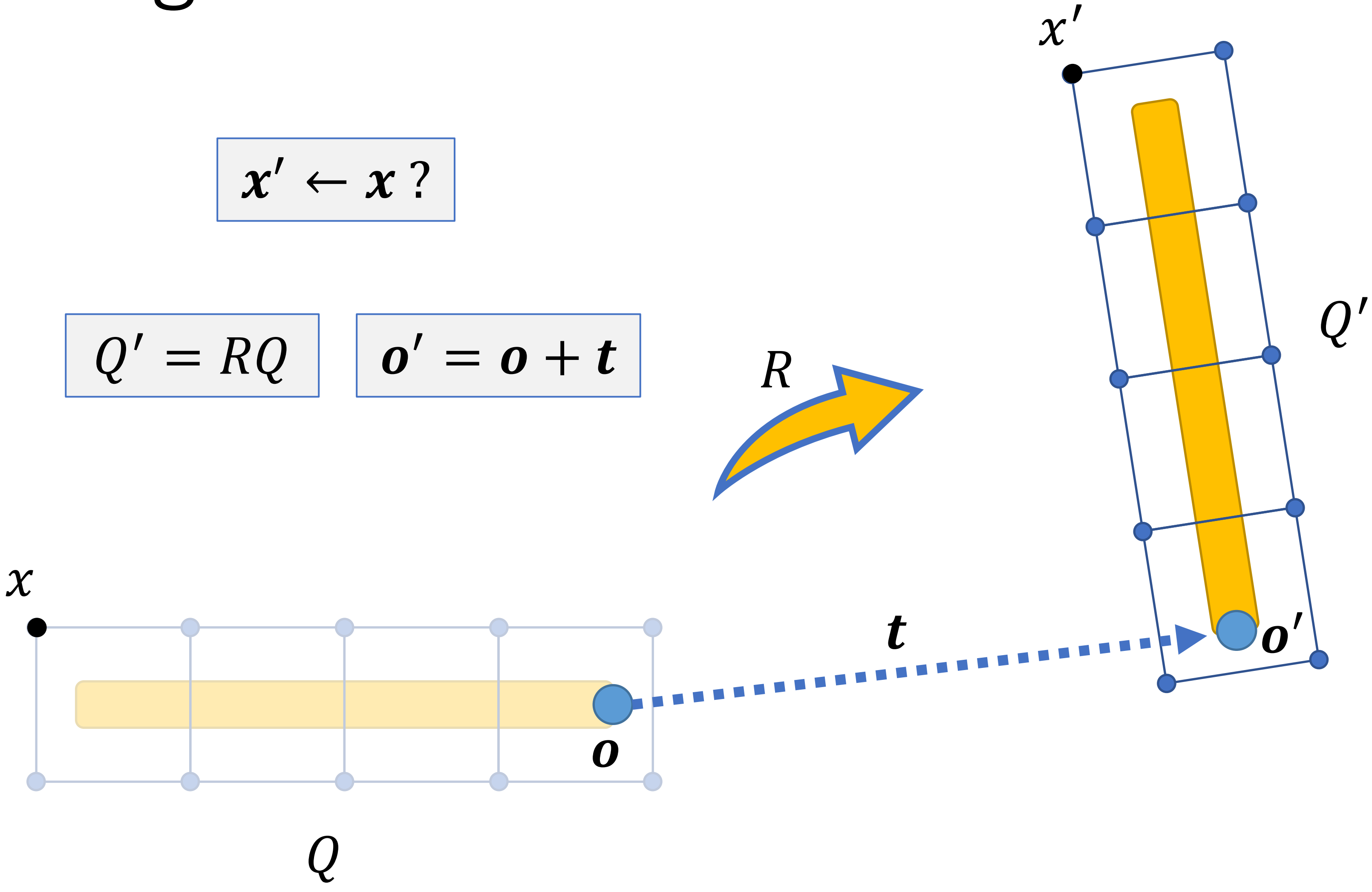
$\boldsymbol{x}' \leftarrow \boldsymbol{x}$?



Skinning Deformation

$$\boldsymbol{x}' \leftarrow \boldsymbol{x}\,?$$

$$Q' = RQ \qquad \boldsymbol{o}' = \boldsymbol{o} + \boldsymbol{t}$$

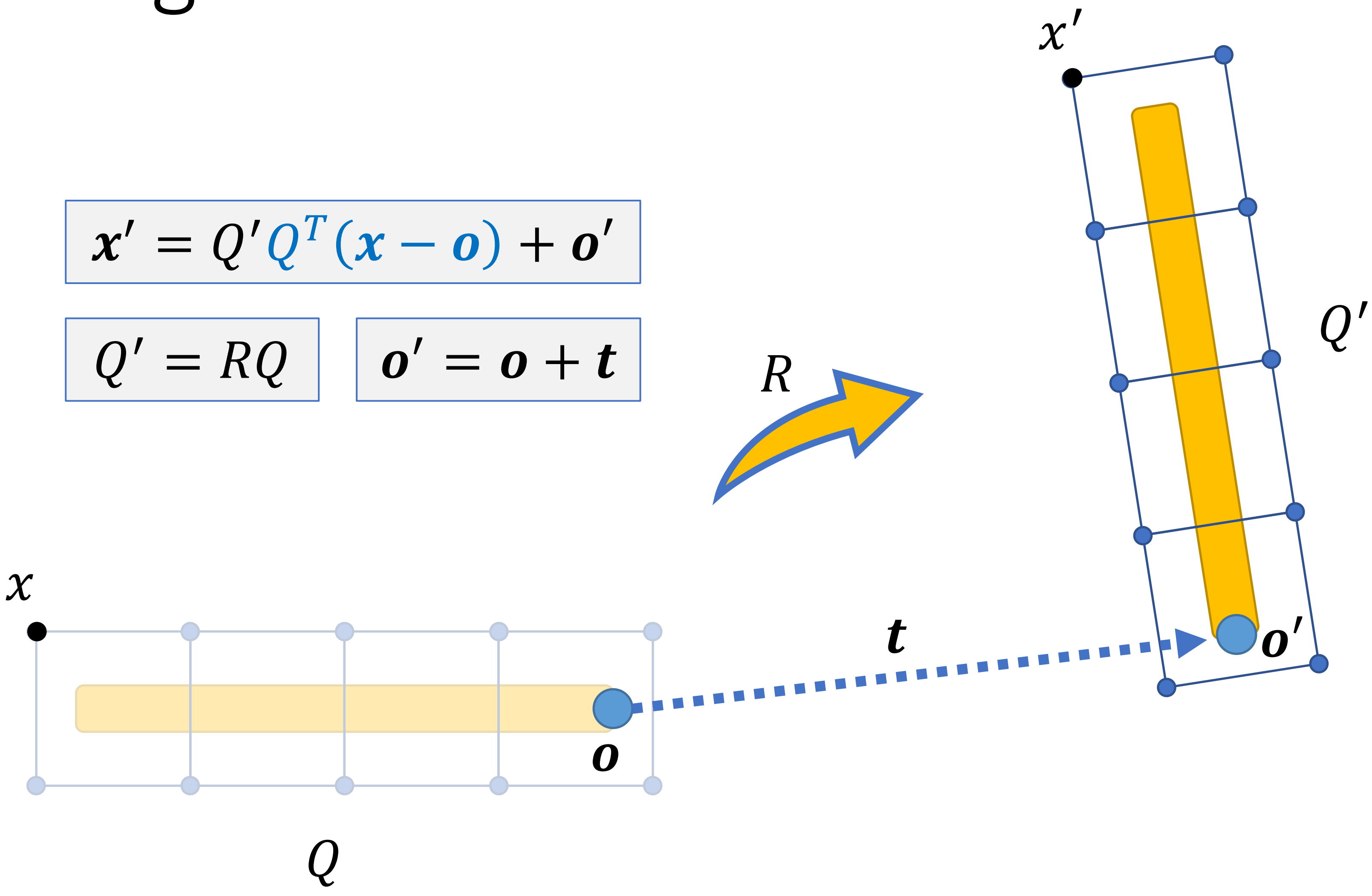


Skinning Deformation

$$\boldsymbol{x}' = Q'Q^T(\boldsymbol{x} - \boldsymbol{o}) + \boldsymbol{o}'$$

$$Q' = RQ$$

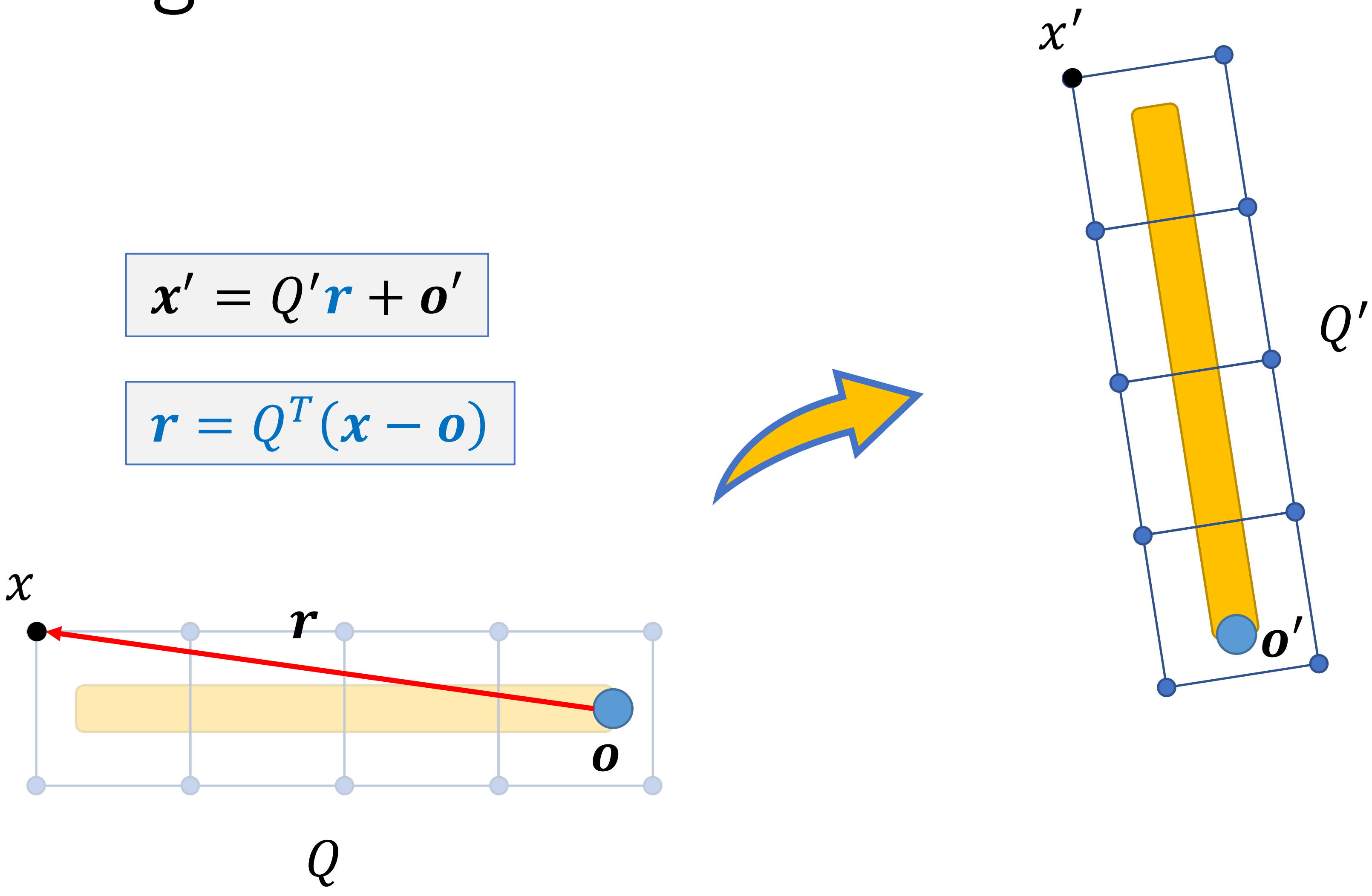
$$\boldsymbol{o}' = \boldsymbol{o} + \boldsymbol{t}$$



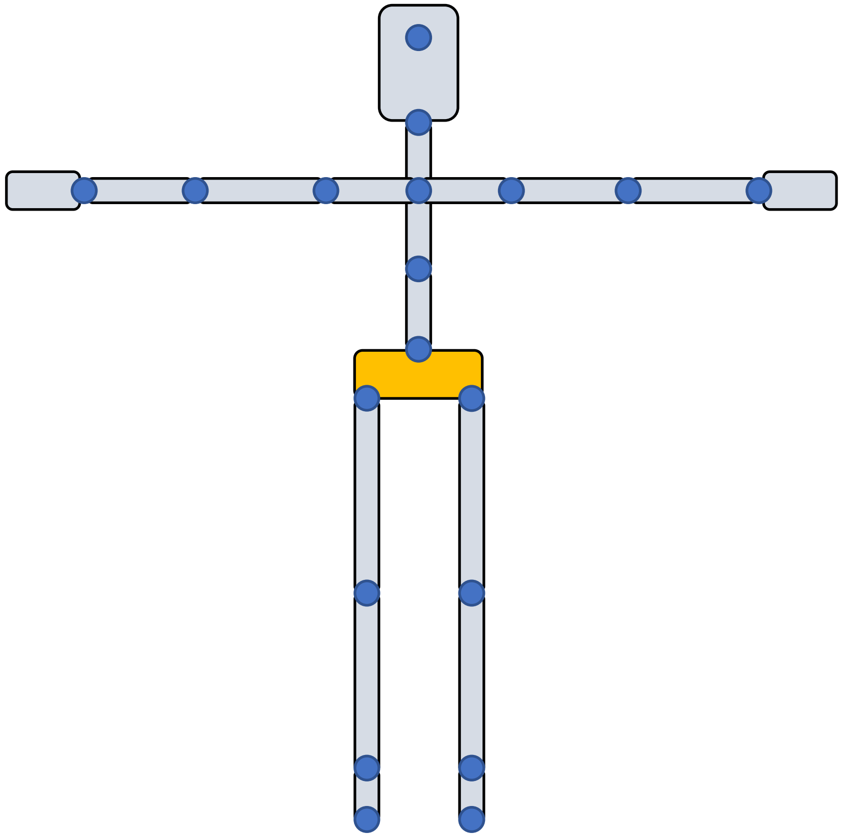
Skinning Deformation

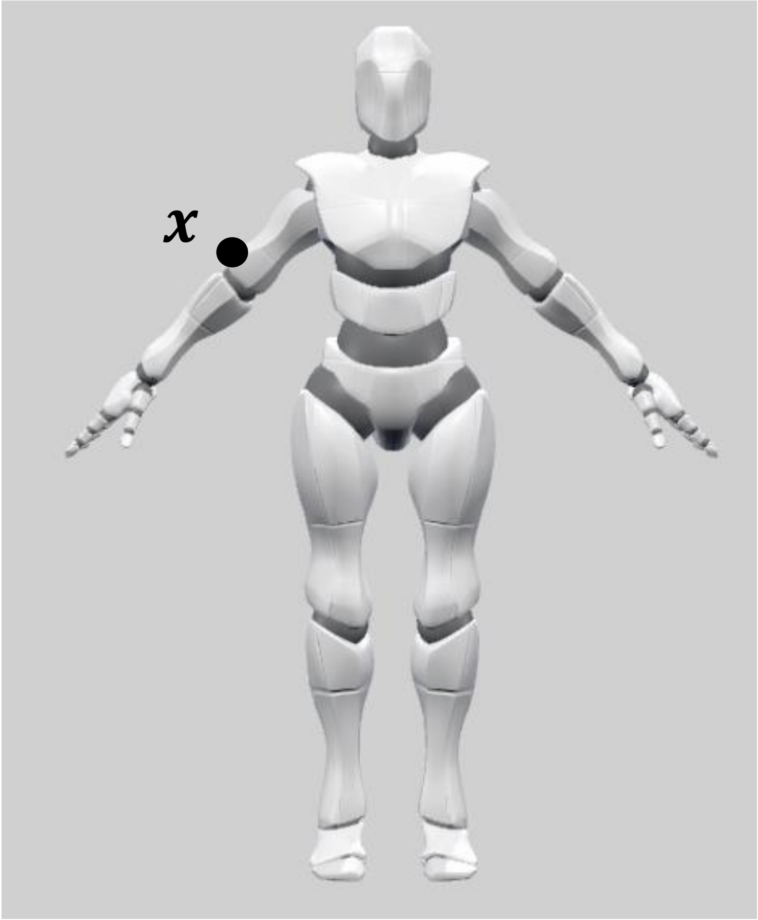
$$\boldsymbol{x}' = Q'\boldsymbol{r} + \boldsymbol{o}'$$

$$\boldsymbol{r} = Q^T(\boldsymbol{x} - \boldsymbol{o})$$

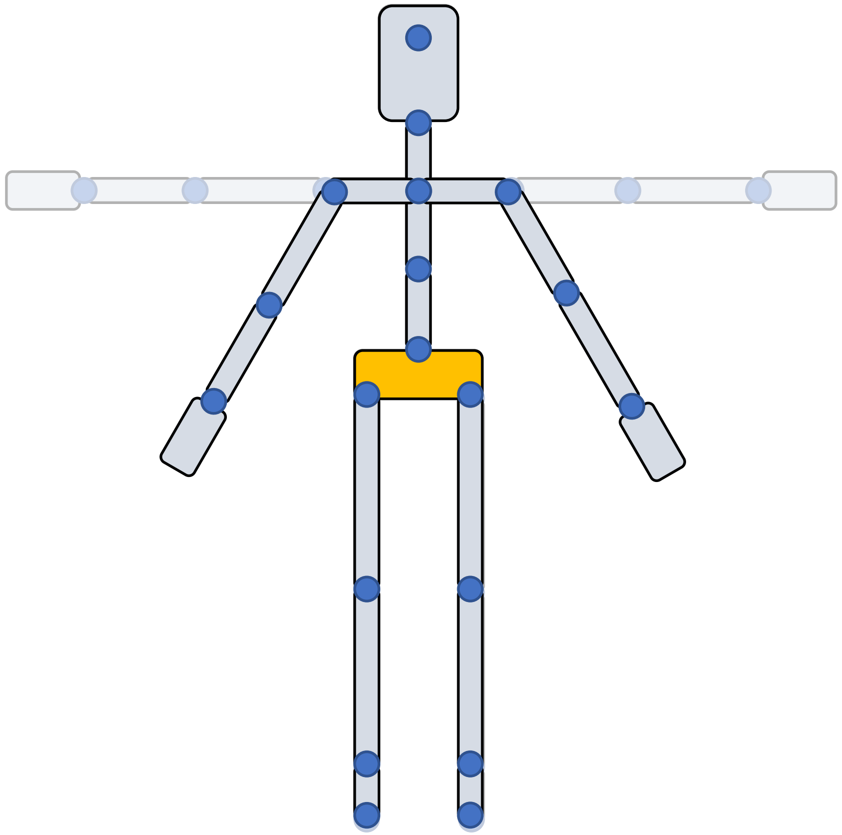


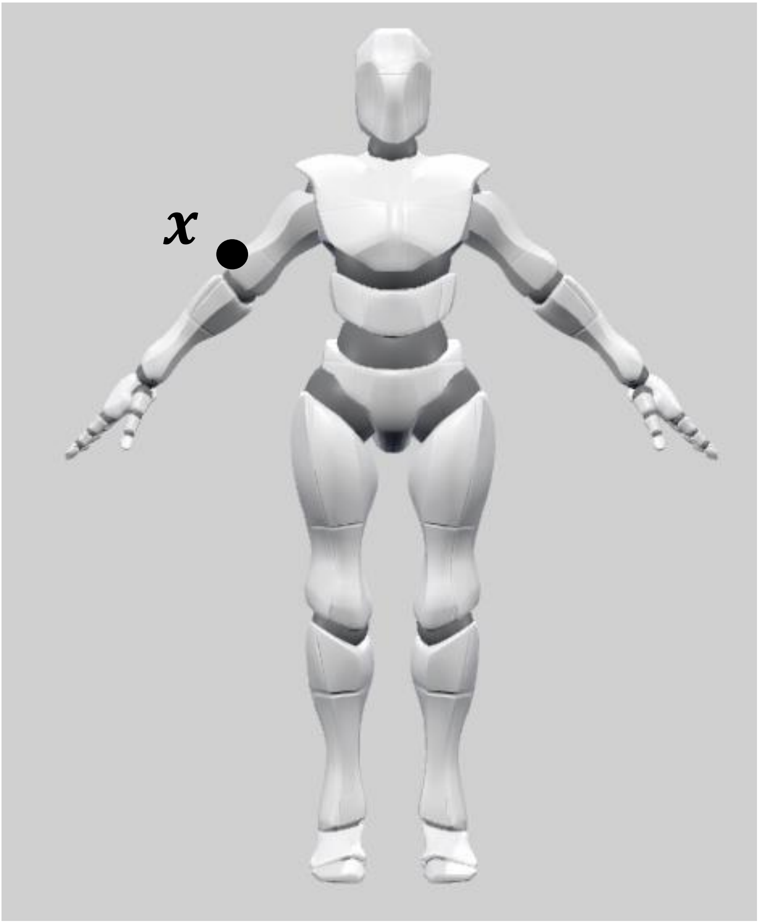
Bind Pose



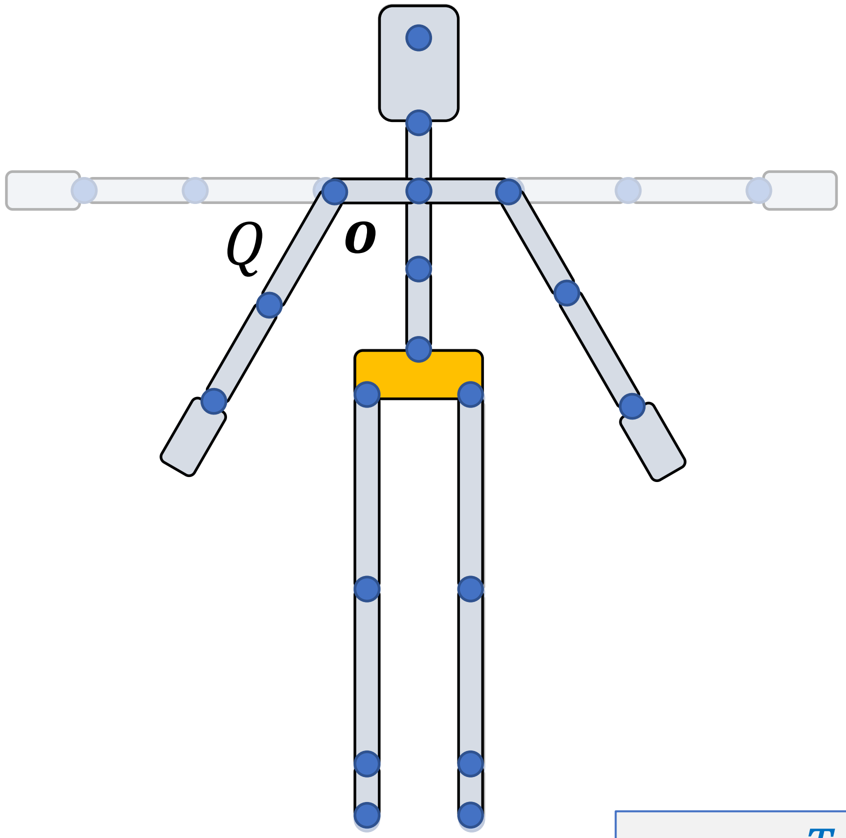


Bind Pose

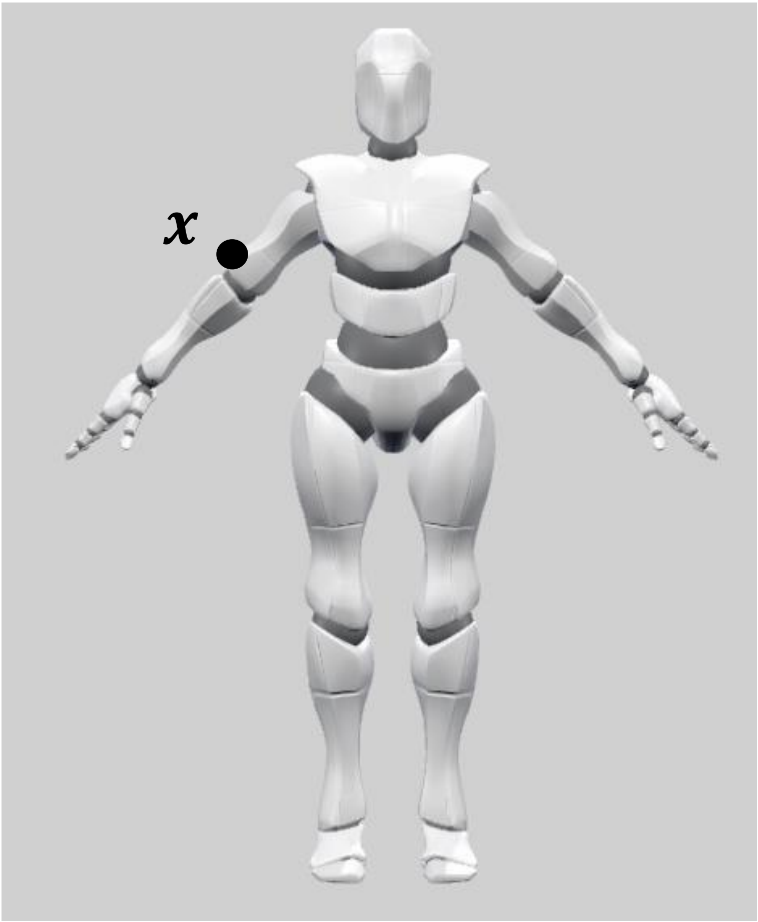




Bind Pose



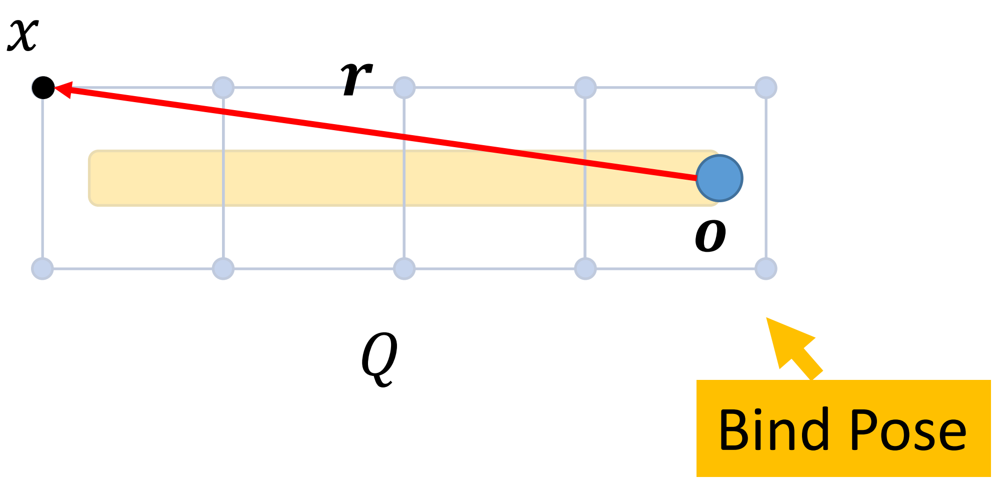
$$\boldsymbol{r} = Q^T(\boldsymbol{x} - \boldsymbol{o})$$

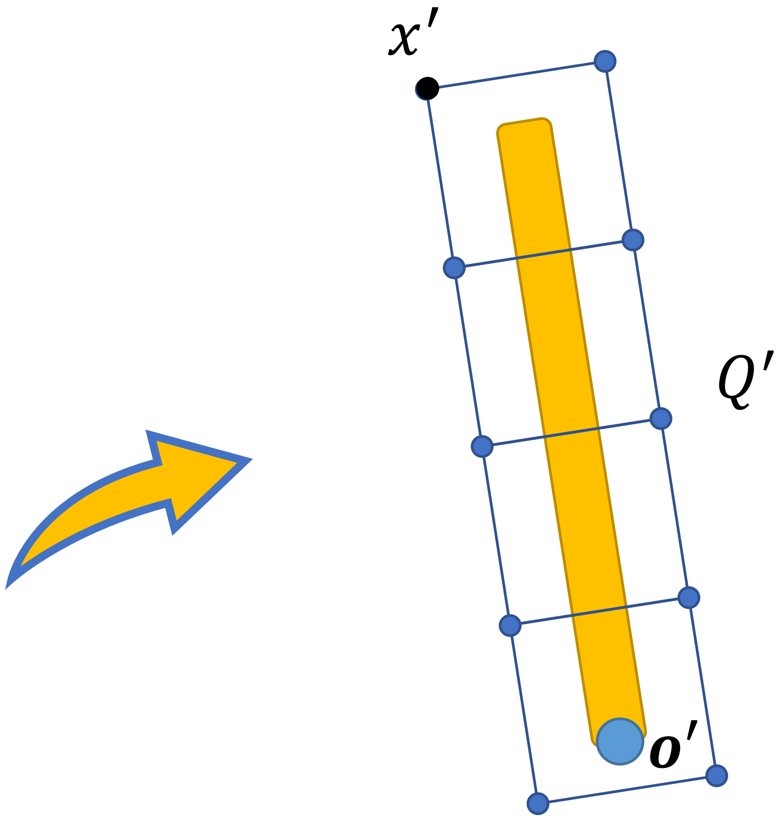


Skinning Deformation

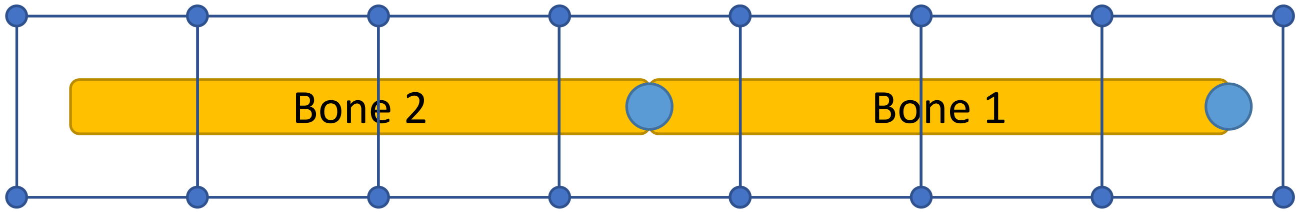
$$\boldsymbol{x}' = Q'\boldsymbol{r} + \boldsymbol{o}'$$

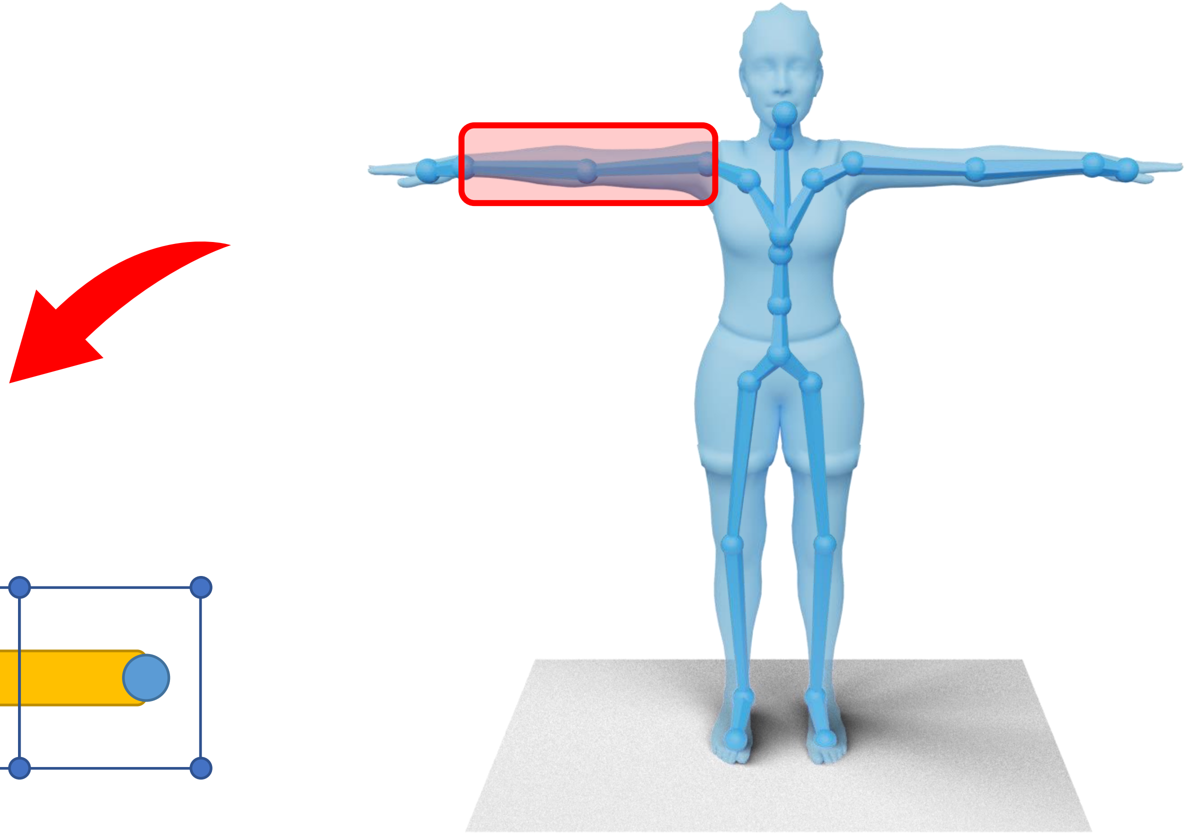
$$\boldsymbol{r} = Q^T(\boldsymbol{x} - \boldsymbol{o})$$



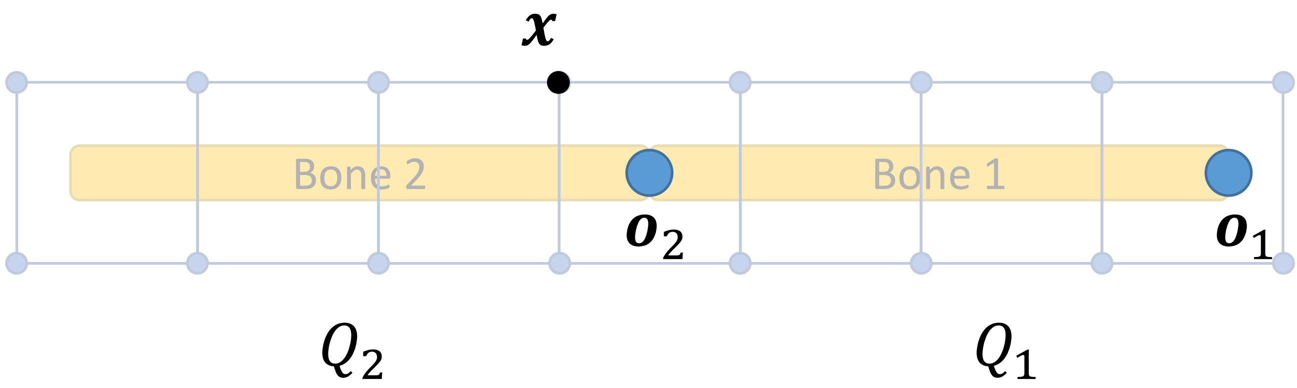


Skinning Deformation

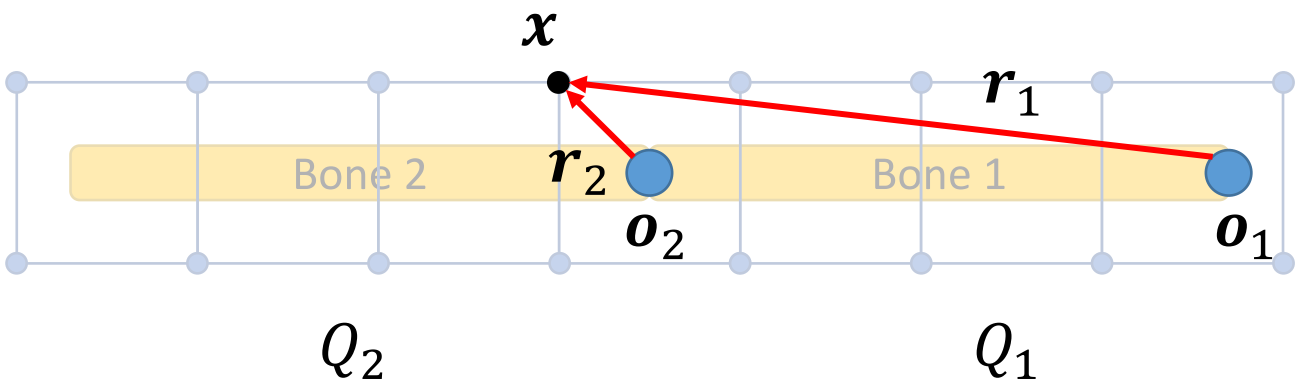




Skinning Deformation

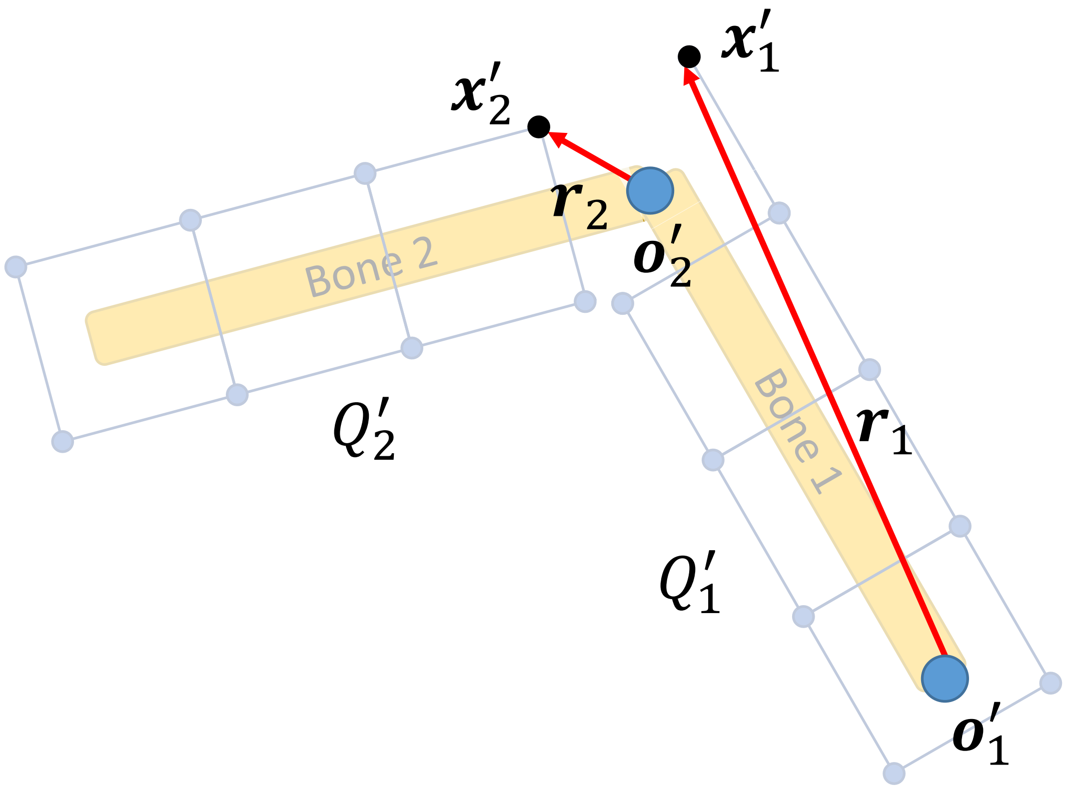


Skinning Deformation



$$\boldsymbol{r}_2 = Q_2^T(\boldsymbol{x} - \boldsymbol{o}_2) \qquad \boldsymbol{r}_1 = Q_1^T(\boldsymbol{x} - \boldsymbol{o}_1)$$

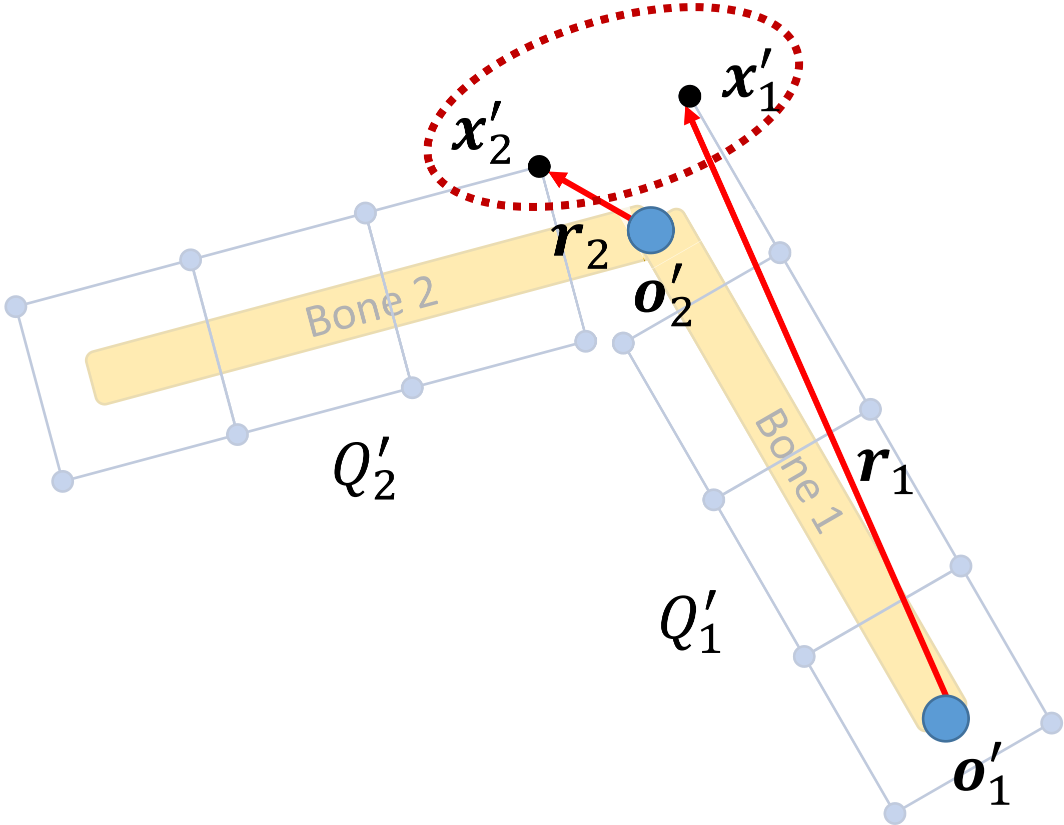
Skinning Deformation



$$x'_1 = Q'_1 r_1 + o'_1$$

$$x'_2 = Q'_2 r_2 + o'_2$$

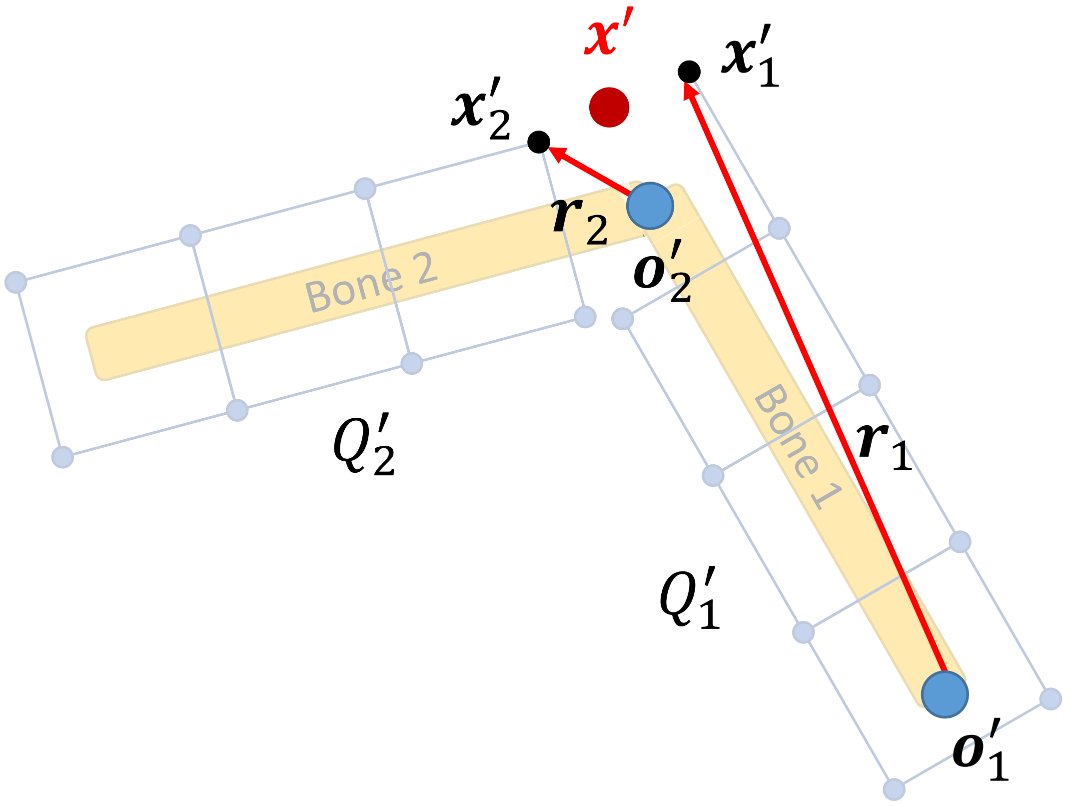
Skinning Deformation



$$x_1' = Q_1' r_1 + o_1'$$

$$x_2' = Q_2' r_2 + o_2'$$

Skinning Deformation

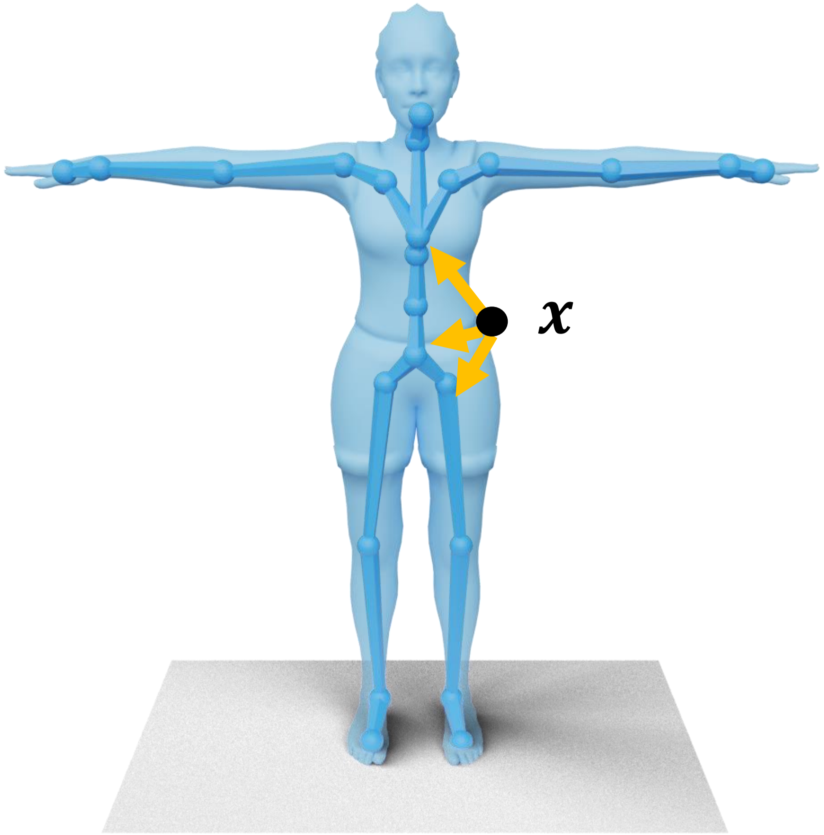


$$\boldsymbol{x}_1' = Q_1'\boldsymbol{r}_1 + \boldsymbol{o}_1'$$

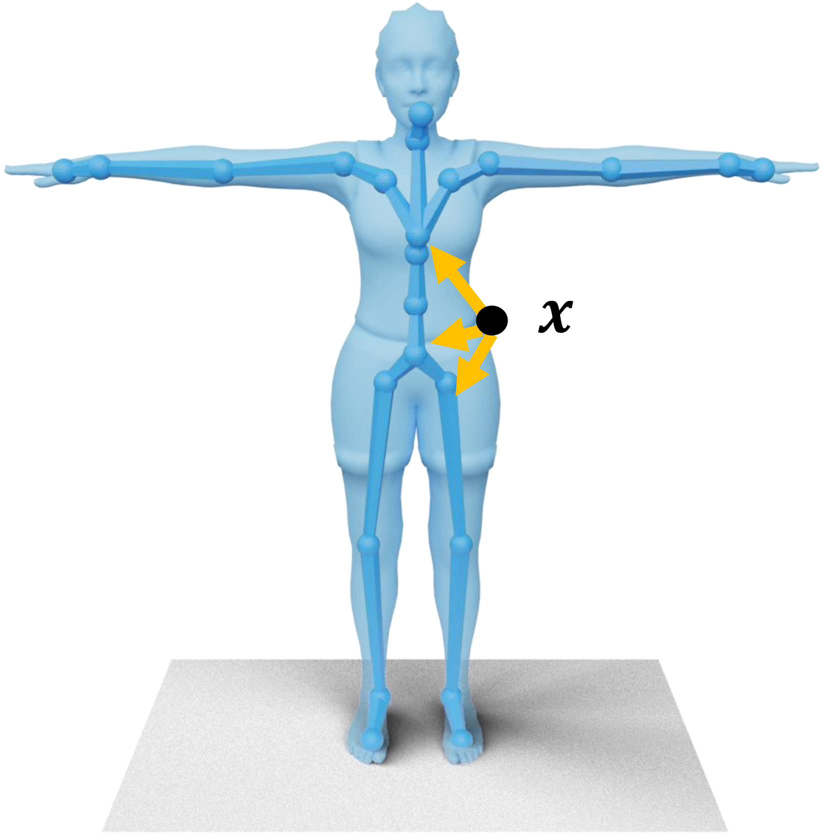
$$\boldsymbol{x}_2' = Q_2'\boldsymbol{r}_2 + \boldsymbol{o}_2'$$

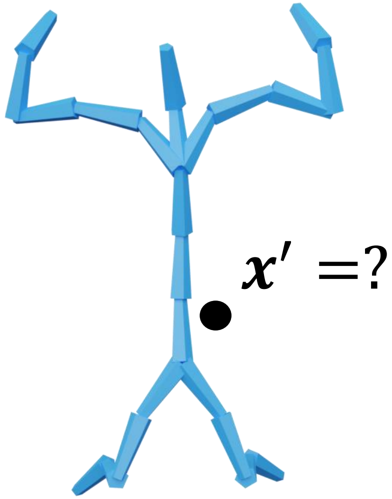
$$\boldsymbol{x}' = w_1\boldsymbol{x}_1' + w_2\boldsymbol{x}_2'$$

Skinning

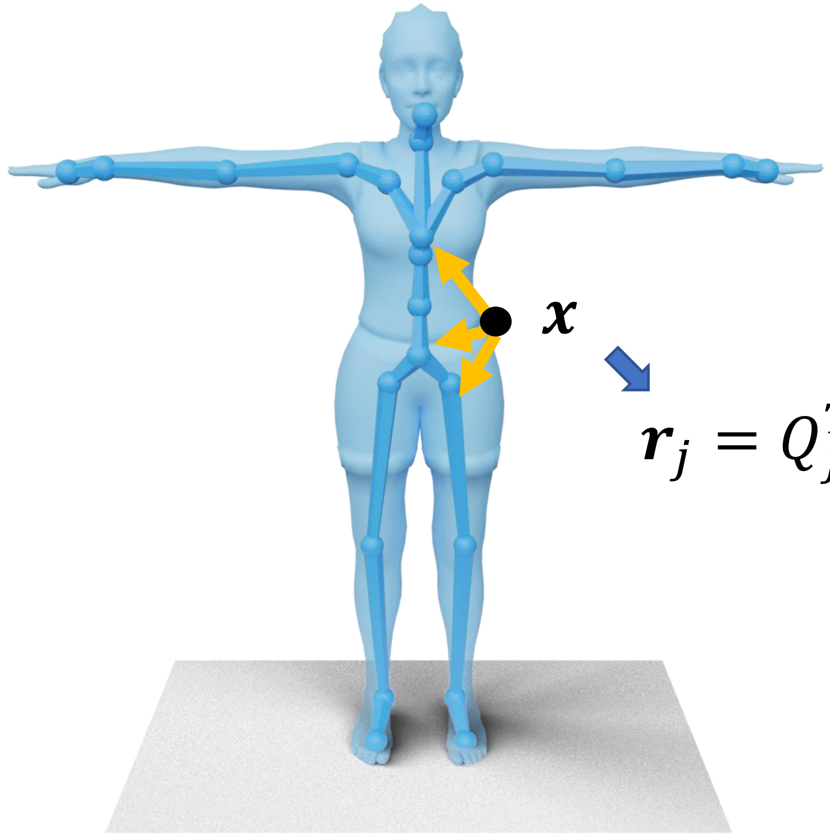


Skinning



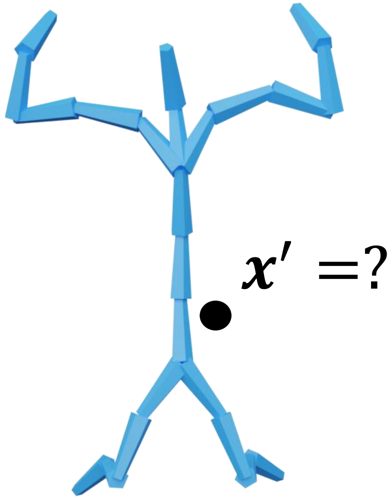


Skinning

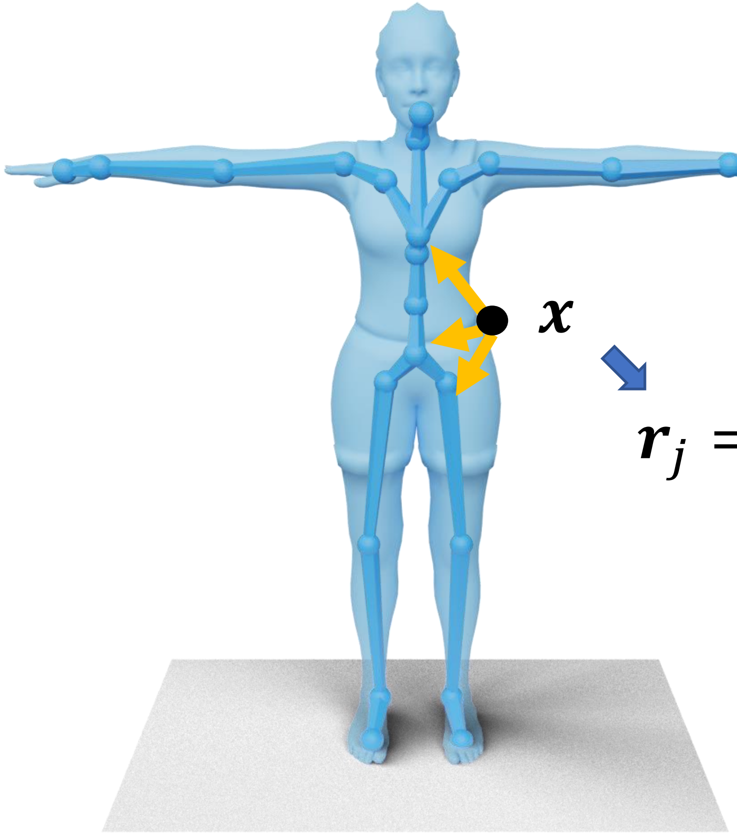


Bind Pose

$$\boldsymbol{r}_j = Q_j^T(\boldsymbol{x} - \boldsymbol{o}_j)$$



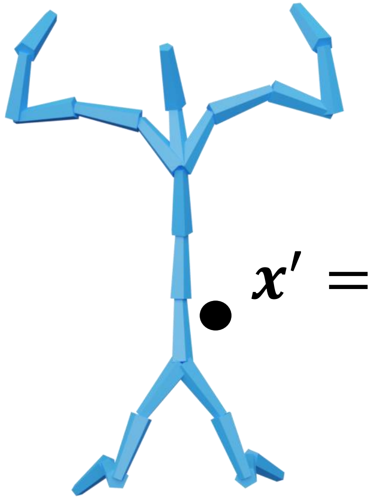
Skinning



$\boldsymbol{x}$

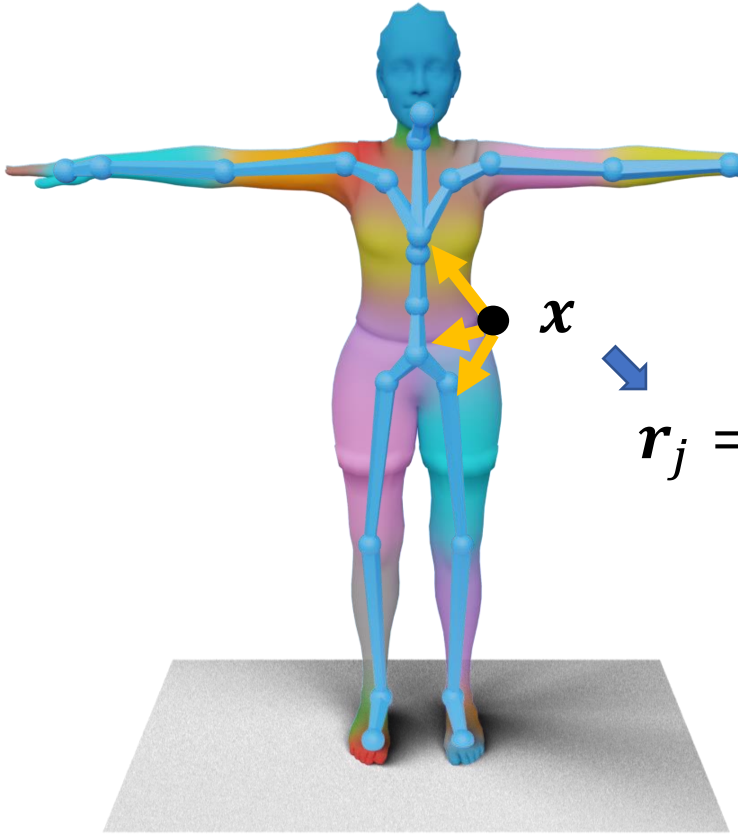
$$\boldsymbol{r}_j = Q_j^T(\boldsymbol{x} - \boldsymbol{o}_j)$$

Bind Pose

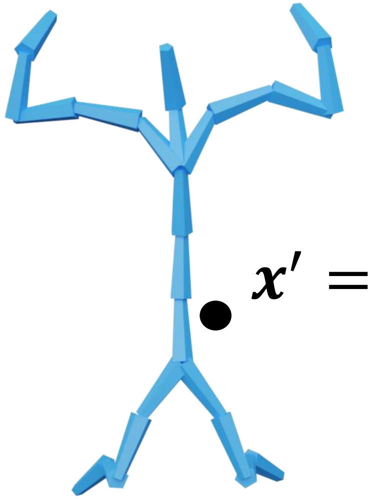


$$\boldsymbol{x}' = \sum_j w_j(Q_j'\boldsymbol{r}_j + \boldsymbol{o}_j')$$

Skinning



$$\boldsymbol{r}_j = Q_j^T(\boldsymbol{x} - \boldsymbol{o}_j)$$

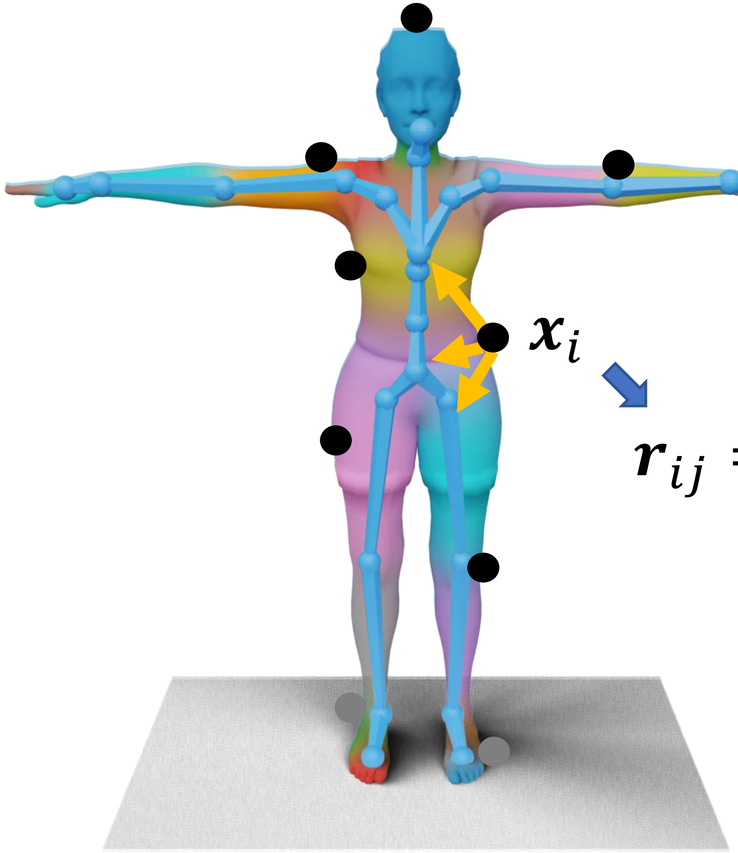


$$\boldsymbol{x}' = \sum_j w_j(Q'_j \boldsymbol{r}_j + \boldsymbol{o}'_j)$$

Bind Pose

w_j: skinning weights

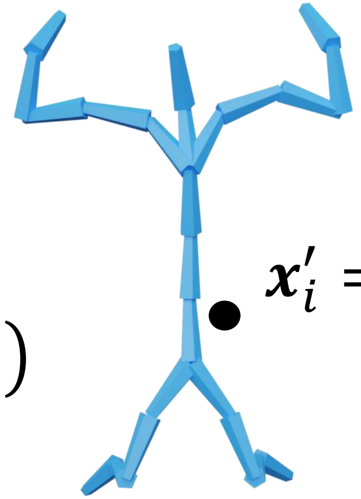
Skinning



$\boldsymbol{x}_i$

$$\boldsymbol{r}_{ij} = Q_j^T(\boldsymbol{x}_i - \boldsymbol{o}_j)$$

Bind Pose



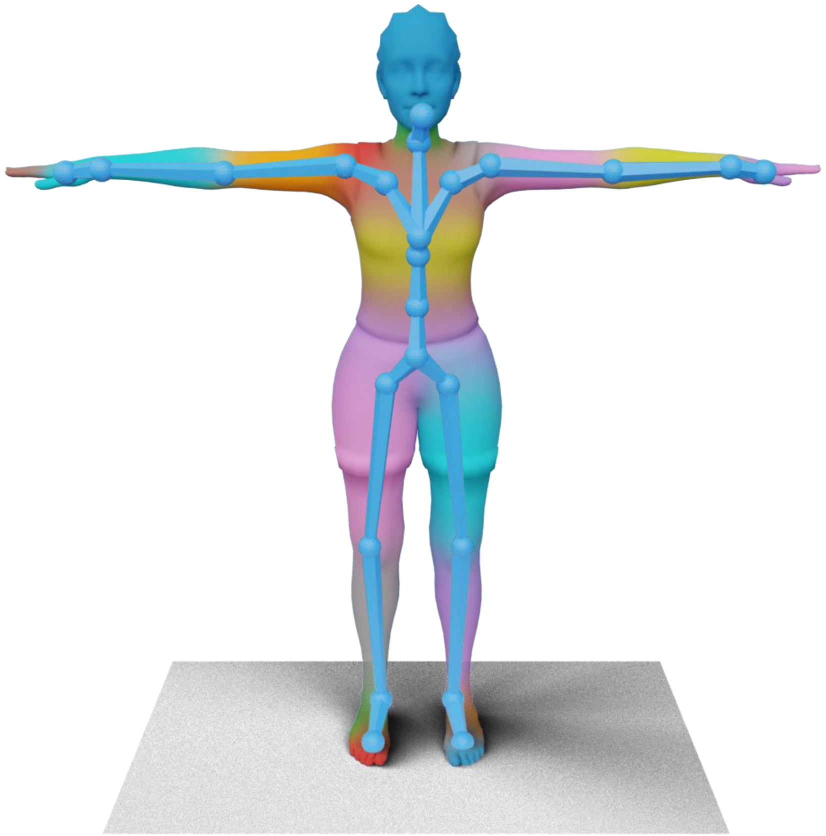
$$\boldsymbol{x}_i' = \sum_j w_{ij}(Q_j'\boldsymbol{r}_{ij} + \boldsymbol{o}_j')$$

w_{ij}: skinning weights

Linear Blend Skinning (LBS)

- Bind pose / rest pose
- Skinning weights
- Skinning transformation

$$\boldsymbol{x}_i' = \sum_{j=1}^{m} w_{ij}(Q_j' \boldsymbol{r}_{ij} + \boldsymbol{o}_j')$$

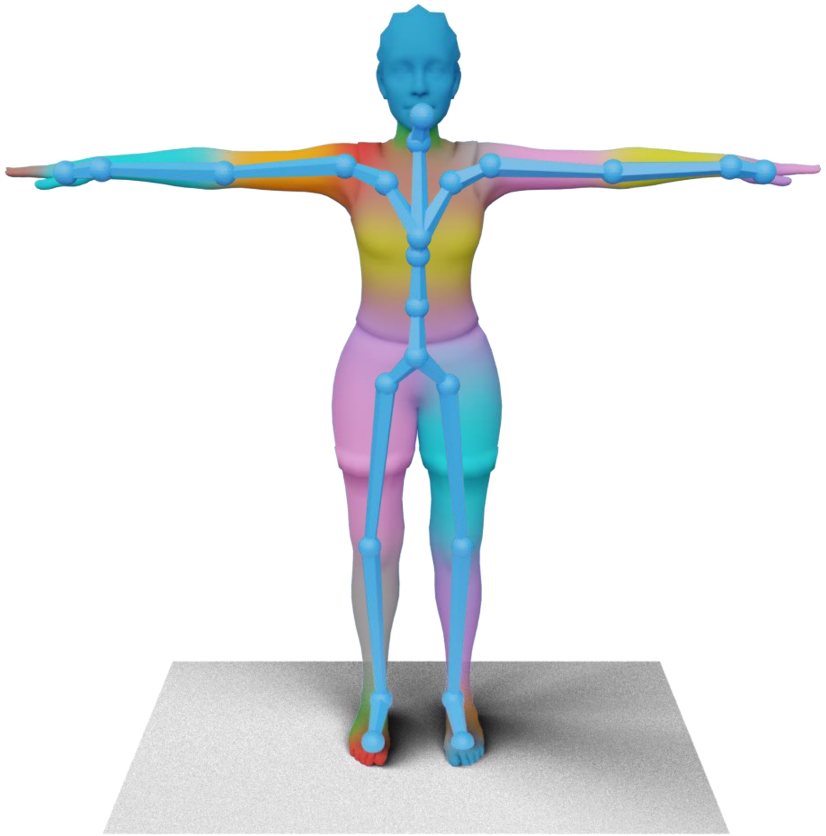


Linear Blend Skinning (LBS)

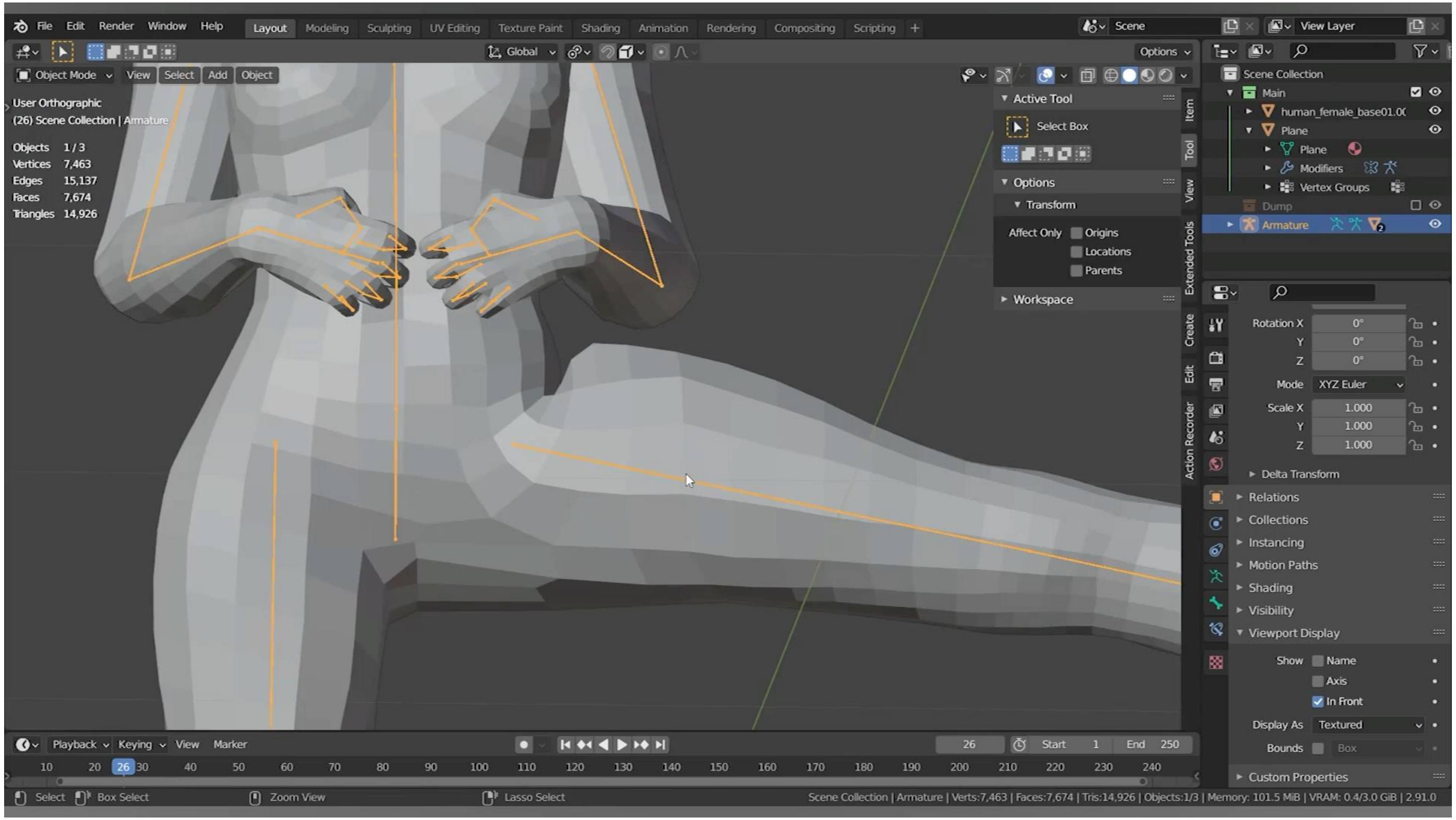
- Bind pose / rest pose
- Skinning weights
- Skinning transformation

$$\boldsymbol{x}_i' = \sum_{j=1}^{m} w_{ij}\left(Q_j' \boldsymbol{r}_{ij} + \boldsymbol{o}_j'\right)$$

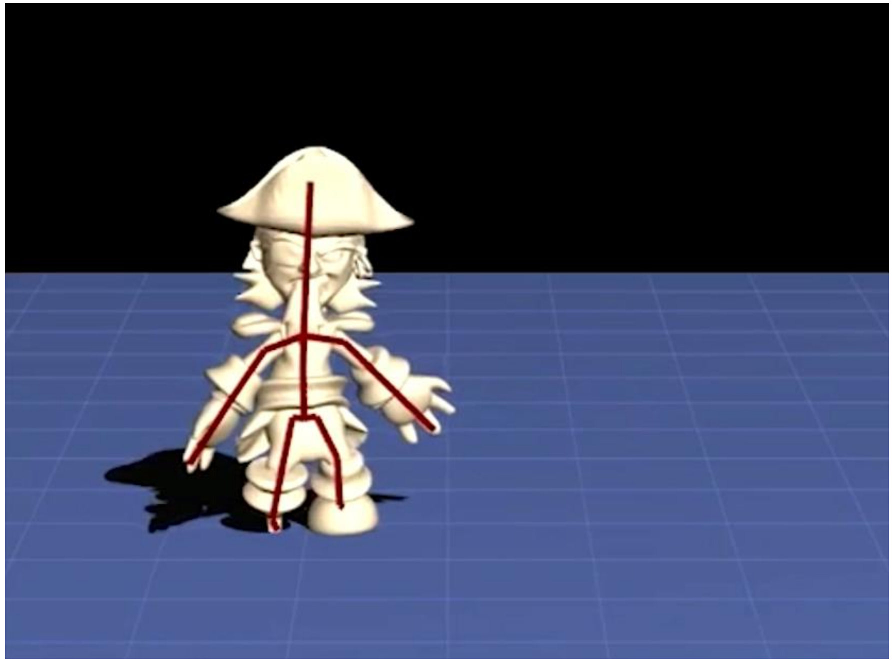
- Used widely in industry
- Efficient and GPU-friendly
  - Games like it



Skinning Weights



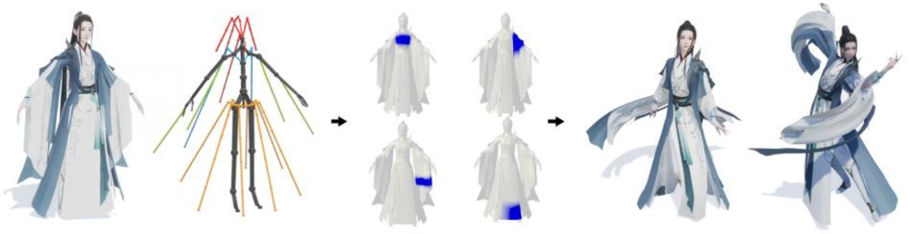
Automatic Skinning?



Pinocchio [Baran et al., 2007]

NeuroSkinning: Automatic Skin Binding for Production Characters with Deep Graph Networks

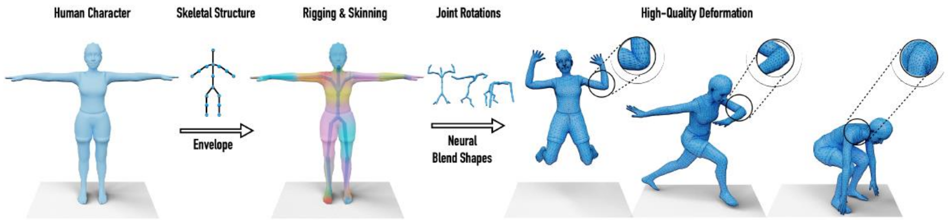
LIJUAN LIU, NetEase Fuxi AI Lab
YOUYI ZHENG, State Key Lab of CAD&CG, Zhejiang University
DI TANG, NetEase Fuxi AI Lab
YI YUAN, NetEase Fuxi AI Lab
CHANGJIE FAN, NetEase Fuxi AI Lab
KUN ZHOU, State Key Lab of CAD&CG, Zhejiang University



SIGGRAPH 2019

Learning Skeletal Articulations with Neural Blend Shapes

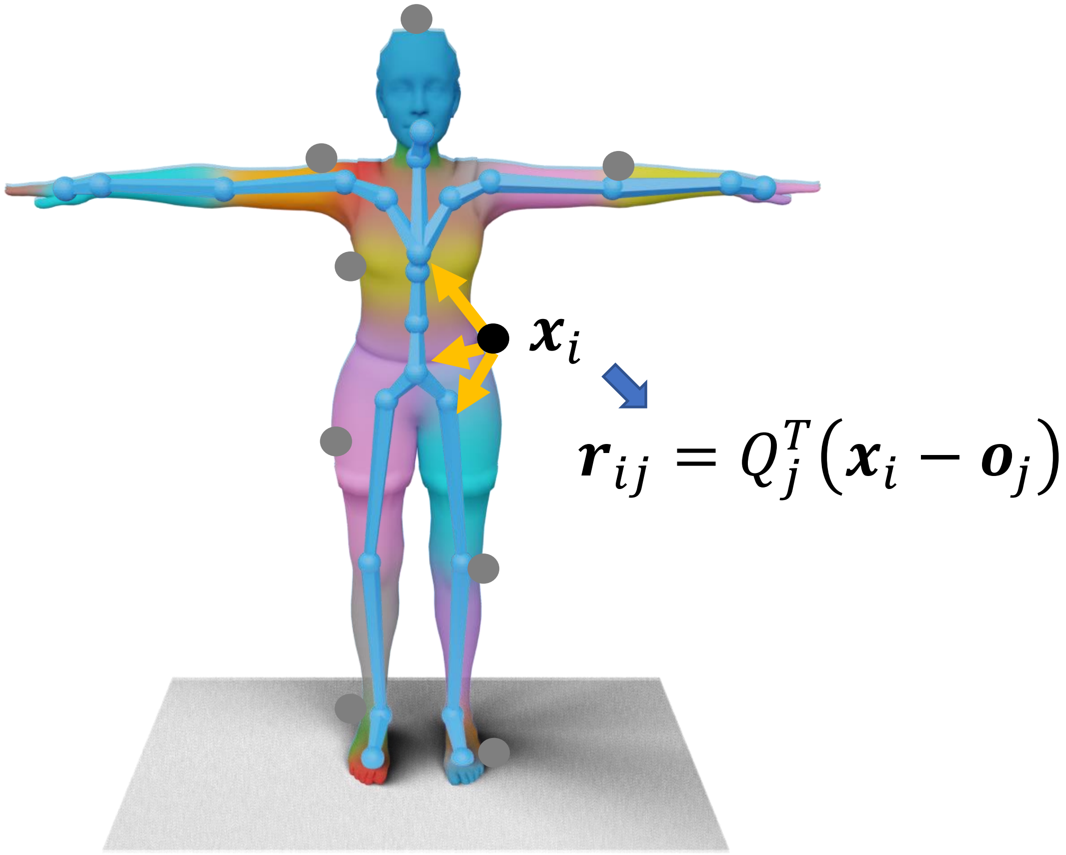
PEIZHUO LI, CFCS, Peking University & AICFVE, Beijing Film Academy
KFIR ABERMAN, Google Research
RANA HANOCKA, Tel-Aviv University
LIBIN LIU, CFCS, Peking University
OLGA SORKINE-HORNUNG, ETH Zurich & AICFVE, Beijing Film Academy
BAOQUAN CHEN*, CFCS, Peking University & AICFVE, Beijing Film Academy



SIGGRAPH 2021

Skinning Transformation

$$\boldsymbol{x}_i' = \sum_{j=1}^{m} w_{ij}(Q_j' \boldsymbol{r}_{ij} + \boldsymbol{o}_j')$$



Bind Pose

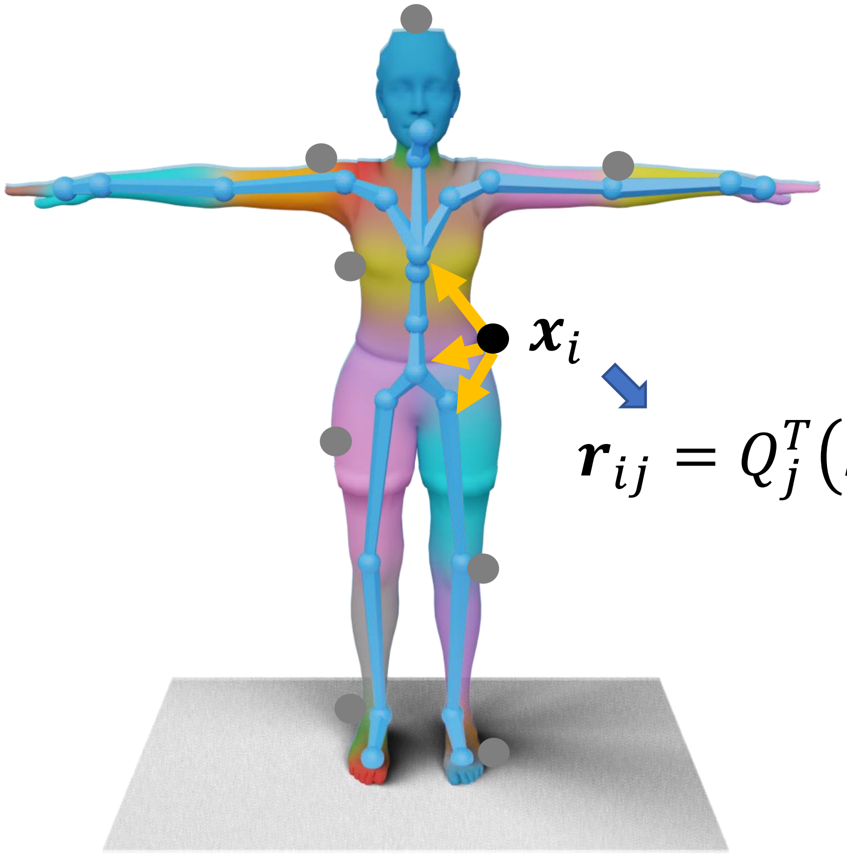
Skinning Transformation

$$\boldsymbol{x}'_i = \sum_{j=1}^{m} w_{ij}(Q'_j \boldsymbol{r}_{ij} + \boldsymbol{o}'_j)$$

$$= \sum_{j=1}^{m} w_{ij}(Q'_j Q_j^T(\boldsymbol{x}_i - \boldsymbol{o}_j) + \boldsymbol{o}'_j)$$

$$= \sum_{j=1}^{m} w_{ij}\left(Q'_j Q_j^T \boldsymbol{x}_i + (\boldsymbol{o}'_j - Q'_j Q_j^T \boldsymbol{o}_j)\right)$$

$$= \sum_{j=1}^{m} w_{ij}(R_j \boldsymbol{x}_i + \boldsymbol{t}_j)$$



$$\boldsymbol{r}_{ij} = Q_j^T(\boldsymbol{x}_i - \boldsymbol{o}_j)$$

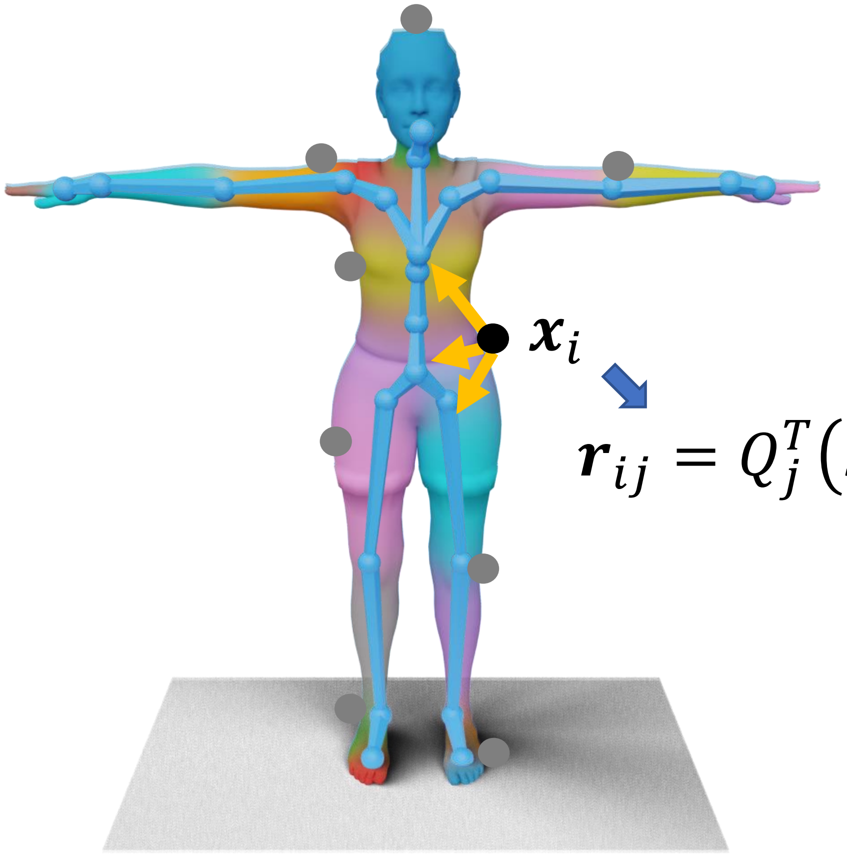
Bind Pose

Skinning Transformation

$$\boldsymbol{x}_i' = \sum_{j=1}^{m} w_{ij}\left(Q_j'\boldsymbol{r}_{ij} + \boldsymbol{o}_j'\right)$$

$$= \sum_{j=1}^{m} w_{ij}R_j\boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij}\boldsymbol{t}_j$$

$$= \left(\sum_{j=1}^{m} w_{ij}R_j\right)\boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij}\boldsymbol{t}_j$$



$$\boldsymbol{r}_{ij} = Q_j^T\left(\boldsymbol{x}_i - \boldsymbol{o}_j\right)$$

Bind Pose

Linear Blend Skinning (LBS)

transformation w.r.t. bend pose

$$\boldsymbol{x}_i' = \left(\sum_{j=1}^{m} w_{ij} R_j\right) \boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij} \boldsymbol{t}_j$$

global position in bind pose

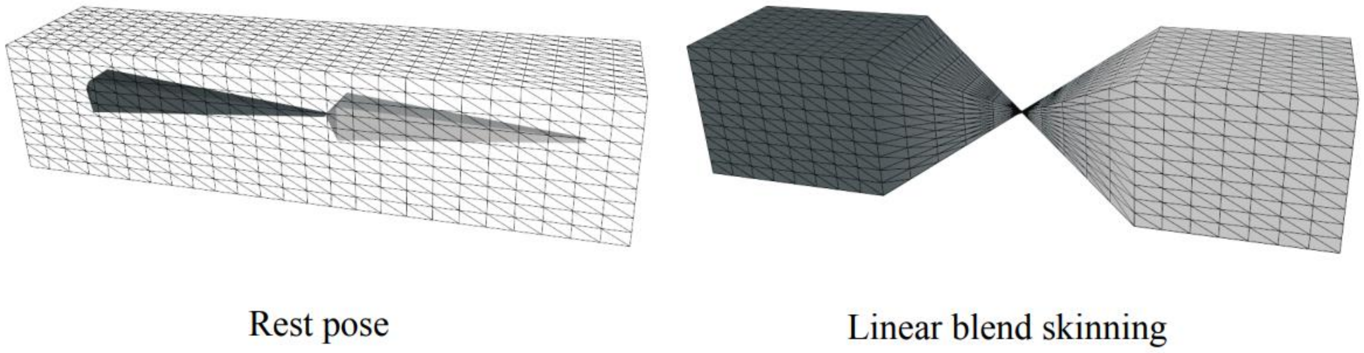
Candy-Wrapper Artifact

$$\boldsymbol{x}_i' = \left(\sum_{j=1}^{m} w_{ij} R_j\right) \boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij} \boldsymbol{t}_j$$

Consider

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

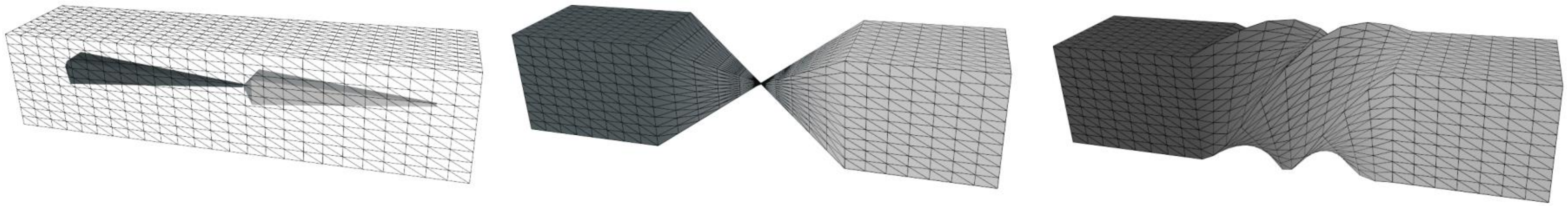
Candy-Wrapper Artifact



Advanced Skinning Methods

- Multi-linear Skinning (we will not cover this)
  - Multi-weight enveloping [Wang and Phillips 2002]
  - Animation Space [Merry et al. 2006]
  - ……

- Nonlinear Skinning
  - Dual-quaternion Skinning (DQS)



Rest pose　　Linear blend skinning　　Dual quaternion skinning

Non-linear Skinning

$$\boldsymbol{x}_i' = \left(\sum_{j=1}^{m} w_{ij} R_j\right) \boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij} \boldsymbol{t}_j$$

Linear blending causes problems…

Can we use quaternions and SLERP?

Non-linear Skinning

blending rotations with
spherical interpolations

LBS

[https://skinning.org/]

Non-linear Skinning

$$\boldsymbol{x}_i' = \left(\sum_{j=1}^{m} w_{ij} R_j\right) \boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij} \boldsymbol{t}_j$$

$$R \in SO(3)$$

$$T_j = \begin{bmatrix} R_j & \boldsymbol{t}_j \\ 0 & 1 \end{bmatrix} \in SE(3)$$

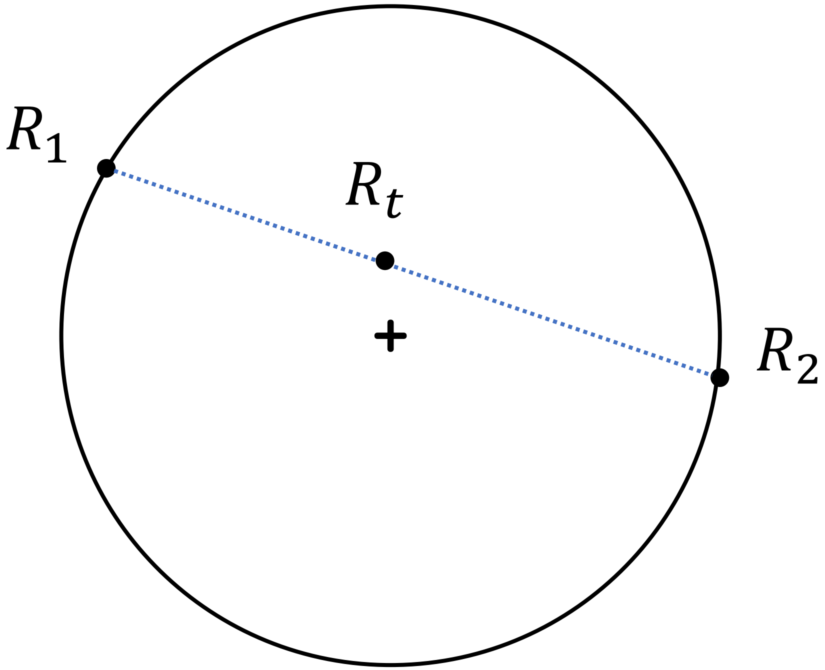
Non-linear Skinning

$$\boldsymbol{x}_i' = \left(\sum_{j=1}^{m} w_{ij} R_j\right) \boldsymbol{x}_i + \sum_{j=1}^{m} w_{ij} \boldsymbol{t}_j$$

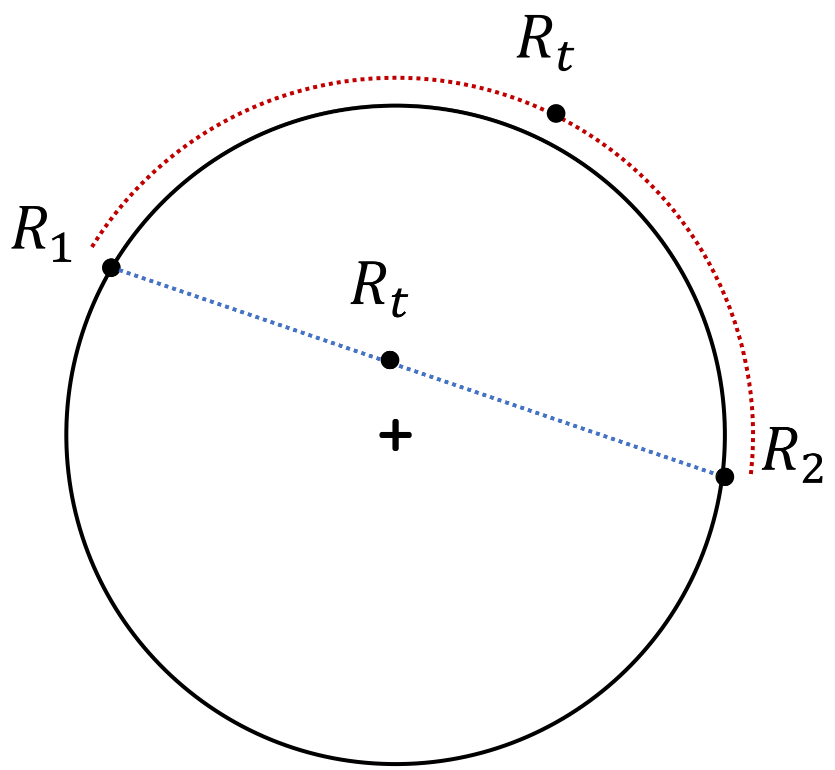
$$R \in SO(3)$$

$$T_j = \left[R_j \mid \boldsymbol{t}_j\right] \in SE(3)$$

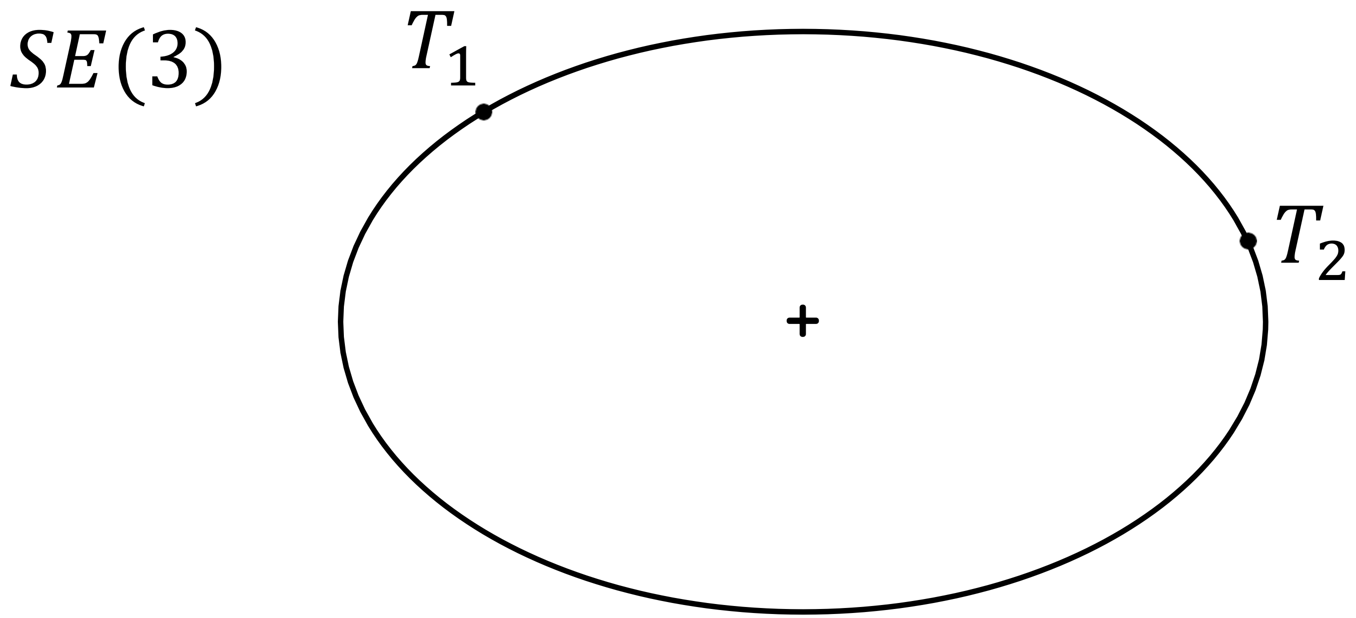
Interpolation in $SO(3)$



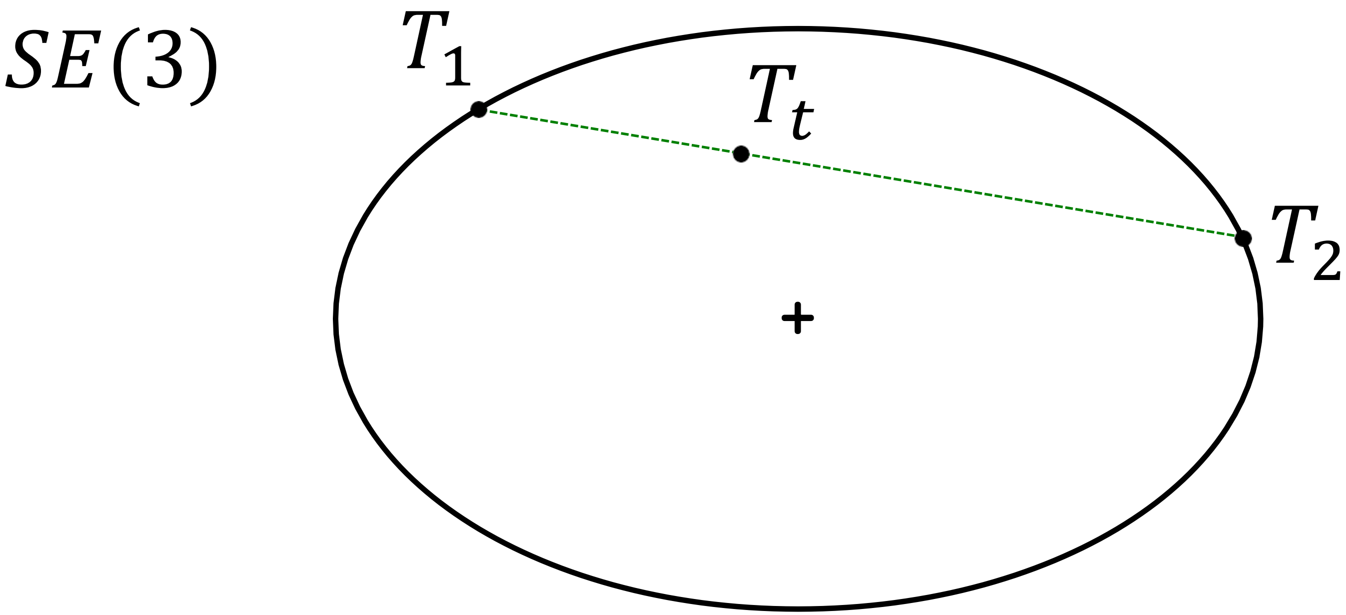
Interpolation in $SO(3)$



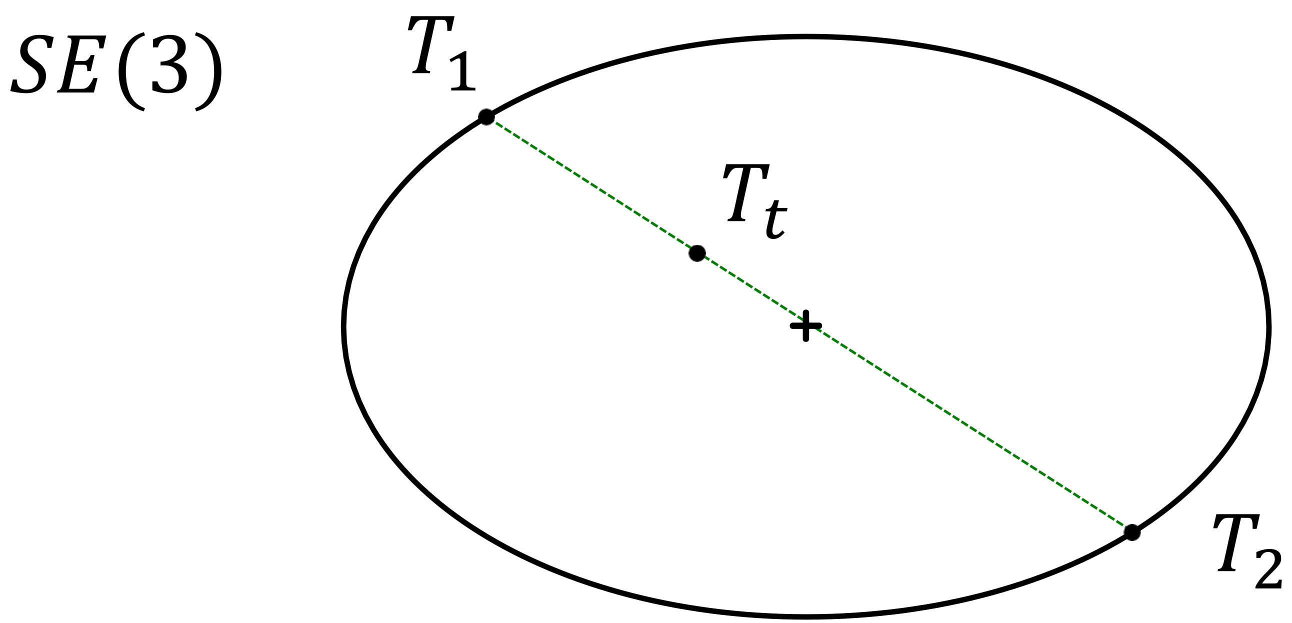
Interpolation in $SE(3)$



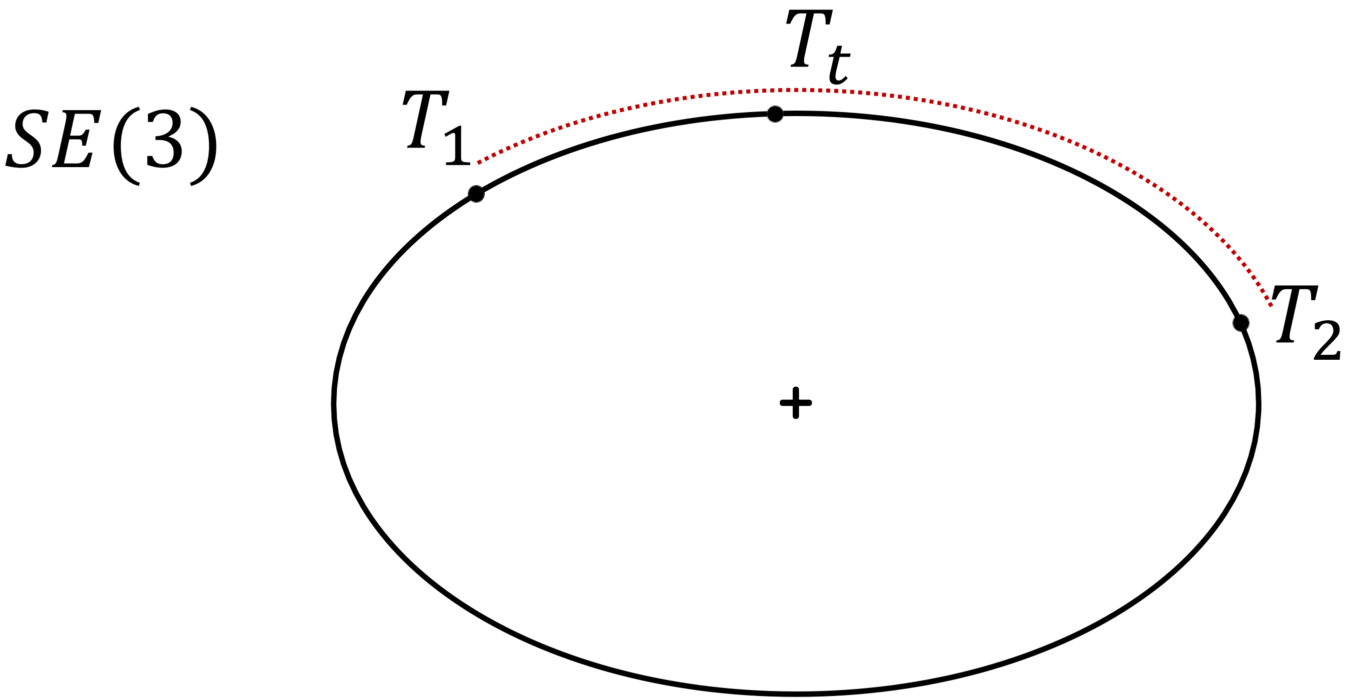
Interpolation in $SE(3)$



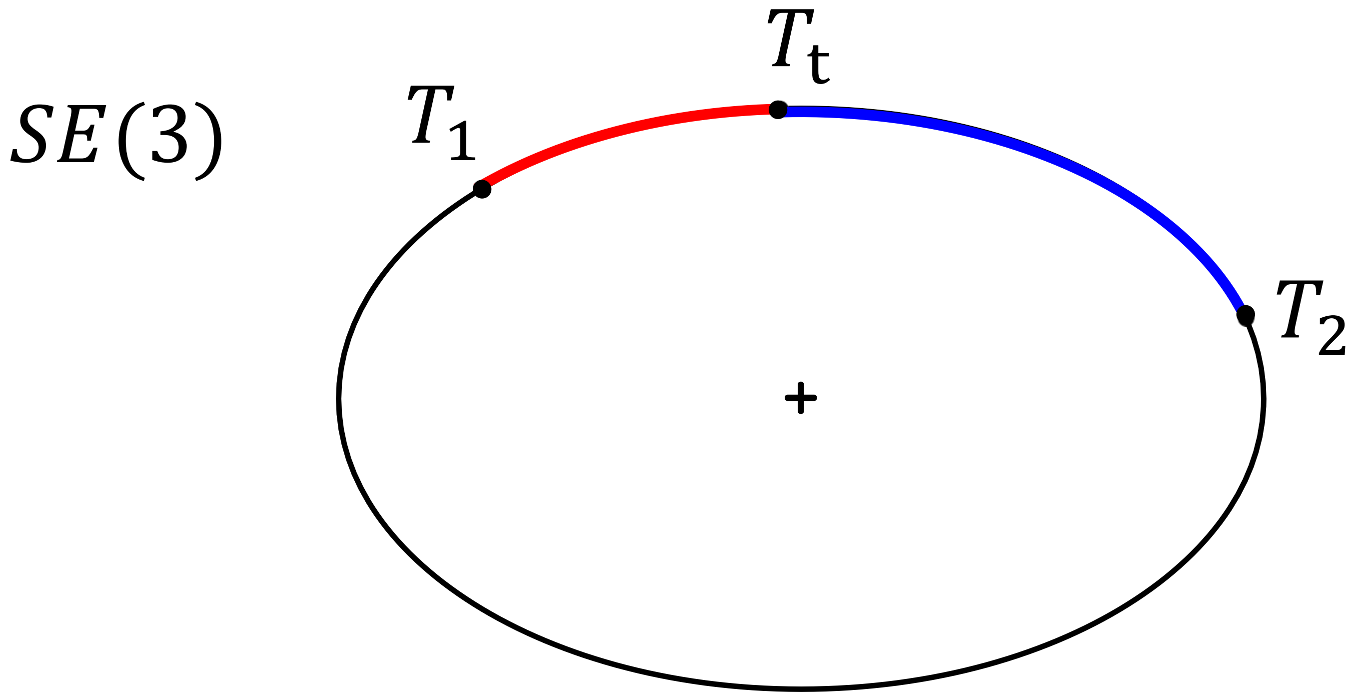
Interpolation in $SE(3)$



Intrinsic Blending

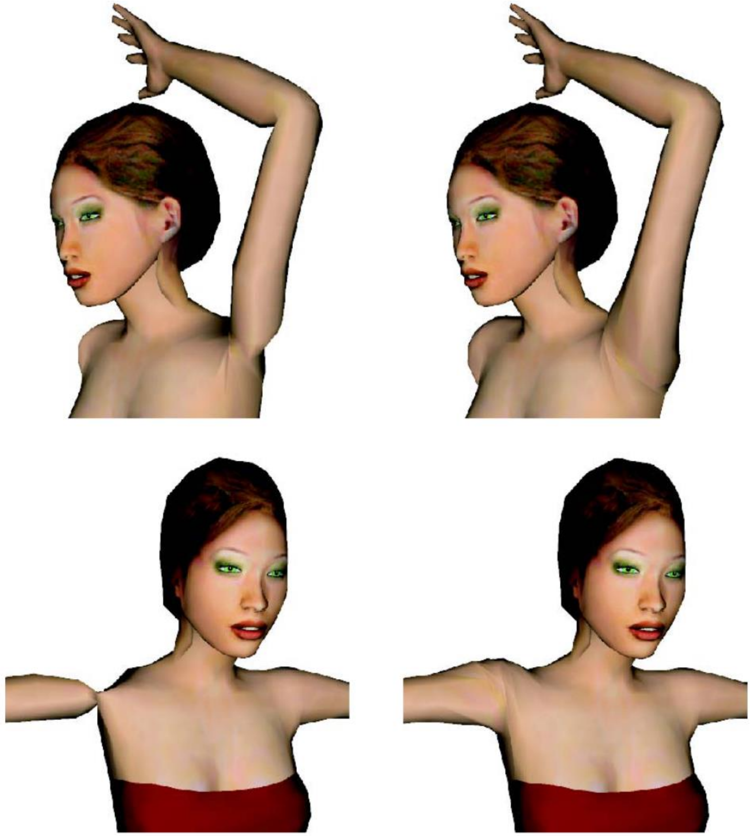


Intrinsic Blending



Dual-Quaternion Skinning (DQS)

- Approximation of intrinsic averages in SE(3)



Ladislav Kavan, Steven Collins, Jiri Zara, Carol O'Sullivan. ***Geometric Skinning with Approximate Dual Quaternion Blending***, ACM Transaction on Graphics, 27(4), 2008.

Dual Numbers

- Dual number

$$x = a + b\varepsilon$$

where $\varepsilon^2 = 0$

Recall: complex number: $x = a + bi$

$i^2 = -1$

Dual Numbers

- Dual number

$$x = a + b\varepsilon$$

where $\varepsilon^2 = 0$

- Conjugate

$$\bar{x} = \overline{a + b\varepsilon} = a - b\varepsilon$$

- Multiplication

$$(a + b\varepsilon)(c + d\varepsilon) = ac + (ad + bc)\varepsilon$$

- Complex number

$$x = a + bi$$

where $i^2 = -1$

- Conjugate

$$\bar{x} = \overline{a + bi} = a - bi$$

- Multiplication

$$(a + bi)(c + di) \\ = (ac - bd) + (ad + bc)i$$

Dual Quaternion

- Dual quaternion

$$\hat{\boldsymbol{q}} = \boldsymbol{q}_0 + \varepsilon \boldsymbol{q}_\varepsilon$$

where $\varepsilon^2 = 0$

A good note of dual-quaternion:
https://faculty.sites.iastate.edu/jia/files/inline-files/dual-quaternion.pdf

Dual Quaternion

- Scalar Multiplication

$$s\hat{\boldsymbol{q}} = s\boldsymbol{q}_r + s\boldsymbol{q}_\varepsilon\varepsilon$$

- Addition

$$\hat{\boldsymbol{q}}_1 + \hat{\boldsymbol{q}}_2 = \boldsymbol{q}_{r1} + \boldsymbol{q}_{r2} + \varepsilon(\boldsymbol{q}_{\varepsilon 1} + \boldsymbol{q}_{\varepsilon 2})$$

- Multiplication

$$\hat{\boldsymbol{q}}_1\hat{\boldsymbol{q}}_2 = \boldsymbol{q}_{r1}\boldsymbol{q}_{r2} + \varepsilon(\boldsymbol{q}_{r1}\boldsymbol{q}_{\varepsilon 2} + \boldsymbol{q}_{r2}\boldsymbol{q}_{\varepsilon 1})$$

Dual Quaternion

- Dual quaternion

$$\hat{\boldsymbol{q}} = \boldsymbol{q}_0 + \varepsilon \boldsymbol{q}_\varepsilon$$

- Conjugation

$$\text{I:} \quad \hat{\boldsymbol{q}}^* = \boldsymbol{q}_0^* + \varepsilon \boldsymbol{q}_\varepsilon^*$$

$$\text{II:} \quad \hat{\boldsymbol{q}}^\circ = \boldsymbol{q}_0 - \varepsilon \boldsymbol{q}_\varepsilon$$

$$\text{III:} \quad \hat{\boldsymbol{q}}^\star = \boldsymbol{q}_0^* - \varepsilon \boldsymbol{q}_\varepsilon^* = (\hat{\boldsymbol{q}}^*)^\circ = (\hat{\boldsymbol{q}}^\circ)^*$$

$$(\hat{\boldsymbol{q}}_1 \hat{\boldsymbol{q}}_2)^\times = \hat{\boldsymbol{q}}_2^\times \hat{\boldsymbol{q}}_1^\times$$

- Norm

$$\|\hat{\boldsymbol{q}}\| = \sqrt{\hat{\boldsymbol{q}}^* \hat{\boldsymbol{q}}} = \|\boldsymbol{q}_0\| + \frac{\varepsilon (\boldsymbol{q}_0 \cdot \boldsymbol{q}_\varepsilon)}{\|\boldsymbol{q}_0\|}$$

Dual Quaternion

- Norm

$$\|\hat{\boldsymbol{q}}\| = \sqrt{\hat{\boldsymbol{q}}^*\hat{\boldsymbol{q}}} = \|\boldsymbol{q}_0\| + \frac{\varepsilon(\boldsymbol{q}_0 \cdot \boldsymbol{q}_\varepsilon)}{\|\boldsymbol{q}_0\|}$$

- Unit dual quaternion: $\|\hat{\boldsymbol{q}}\| = 1$, which requires:

$$\|\boldsymbol{q}_{\mathbf{0}}\| = 1$$

$$\boldsymbol{q}_{\mathbf{0}} \cdot \boldsymbol{q}_\varepsilon = 0$$

Dual Quaternion ⇔ Rigid Transformation

- Like quaternion, any rigid transformation $T \in SE(3)$ can be converted into a unit dual quaternion

$$T\boldsymbol{x} = R\boldsymbol{x} + \boldsymbol{t}$$

$$T = [R \mid \boldsymbol{t}] \rightarrow \widehat{\boldsymbol{q}} = \boldsymbol{q}_0 + \varepsilon \boldsymbol{q}_\varepsilon$$

$\boldsymbol{q}_0 = \boldsymbol{r}$ quaternion of R

$\boldsymbol{q}_\varepsilon = \frac{1}{2}\boldsymbol{t}\boldsymbol{r}$ pure quaternion $\boldsymbol{t} = (0, t)$

Dual Quaternion ⇔ Rigid Transformation

- Transform a vector $\boldsymbol{v}$ using unit dual quaternion

$$\widehat{\boldsymbol{v}}' = \widehat{\boldsymbol{q}}\widehat{\boldsymbol{v}}\widehat{\boldsymbol{q}}^{\star}$$

where

$$\begin{aligned}\widehat{\boldsymbol{v}} &= 1 + \varepsilon(0, \boldsymbol{v}) \\ &= (1,0,0,0) + \varepsilon\left(0, v_x, v_y, v_z\right)\end{aligned}$$

$$\text{III:} \quad \begin{aligned}\widehat{\boldsymbol{q}}^{\star} &= \boldsymbol{q}_0^* - \varepsilon \boldsymbol{q}_\varepsilon^* \\ &= (\widehat{\boldsymbol{q}}^*)^\circ = (\widehat{\boldsymbol{q}}^\circ)^*\end{aligned}$$

Dual Quaternion ⇔ Rigid Transformation

- Like quaternion, any rigid transformation $T \in SE(3)$ can be converted into a unit dual quaternion

$$T = [R \mid t] \rightarrow \hat{\boldsymbol{q}} = \boldsymbol{q}_0 + \varepsilon \boldsymbol{q}_\varepsilon$$

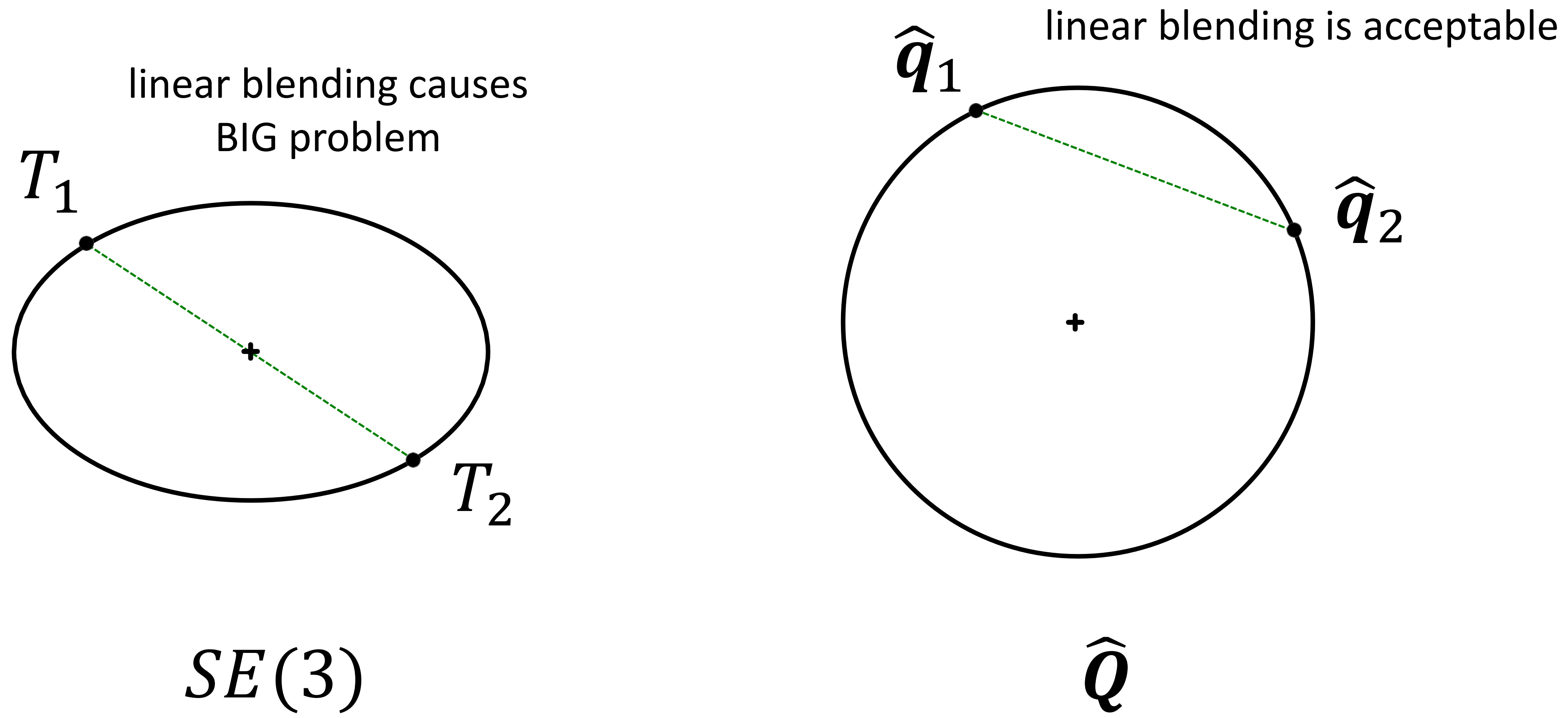
$$\boldsymbol{q}_0 = \boldsymbol{r}$$

$$\boldsymbol{q}_\varepsilon = \frac{1}{2} \boldsymbol{t} \boldsymbol{r}$$

$\hat{\boldsymbol{q}}$ and $-\hat{\boldsymbol{q}}$ represent the same transformation

$\hat{\boldsymbol{Q}}$ is a double cover of $SE(3)$

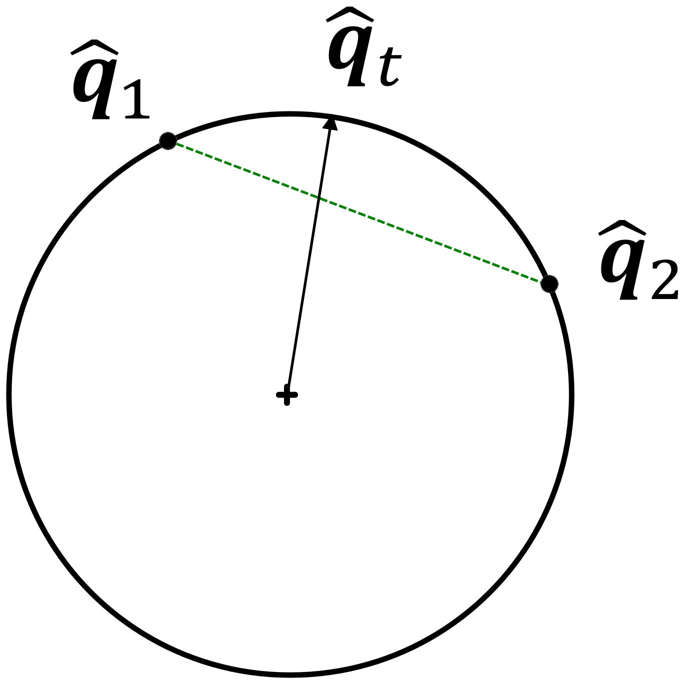
Double Cover Visualized



Interpolating Dual-Quaternion

$$\hat{\boldsymbol{q}}_t = (1-t)\hat{\boldsymbol{q}}_{\boldsymbol{0}} + t\hat{\boldsymbol{q}}_{\boldsymbol{1}}$$

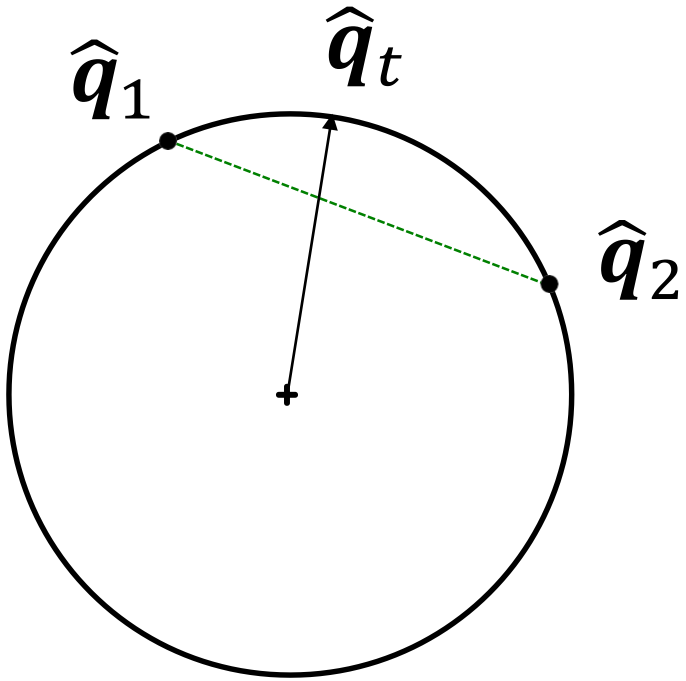
$$\hat{\boldsymbol{q}}_t = \frac{(1-t)\hat{\boldsymbol{q}}_{\boldsymbol{0}} + t\hat{\boldsymbol{q}}_{\boldsymbol{1}}}{\|(1-t)\hat{\boldsymbol{q}}_{\boldsymbol{0}} + t\hat{\boldsymbol{q}}_{\boldsymbol{1}}\|}$$



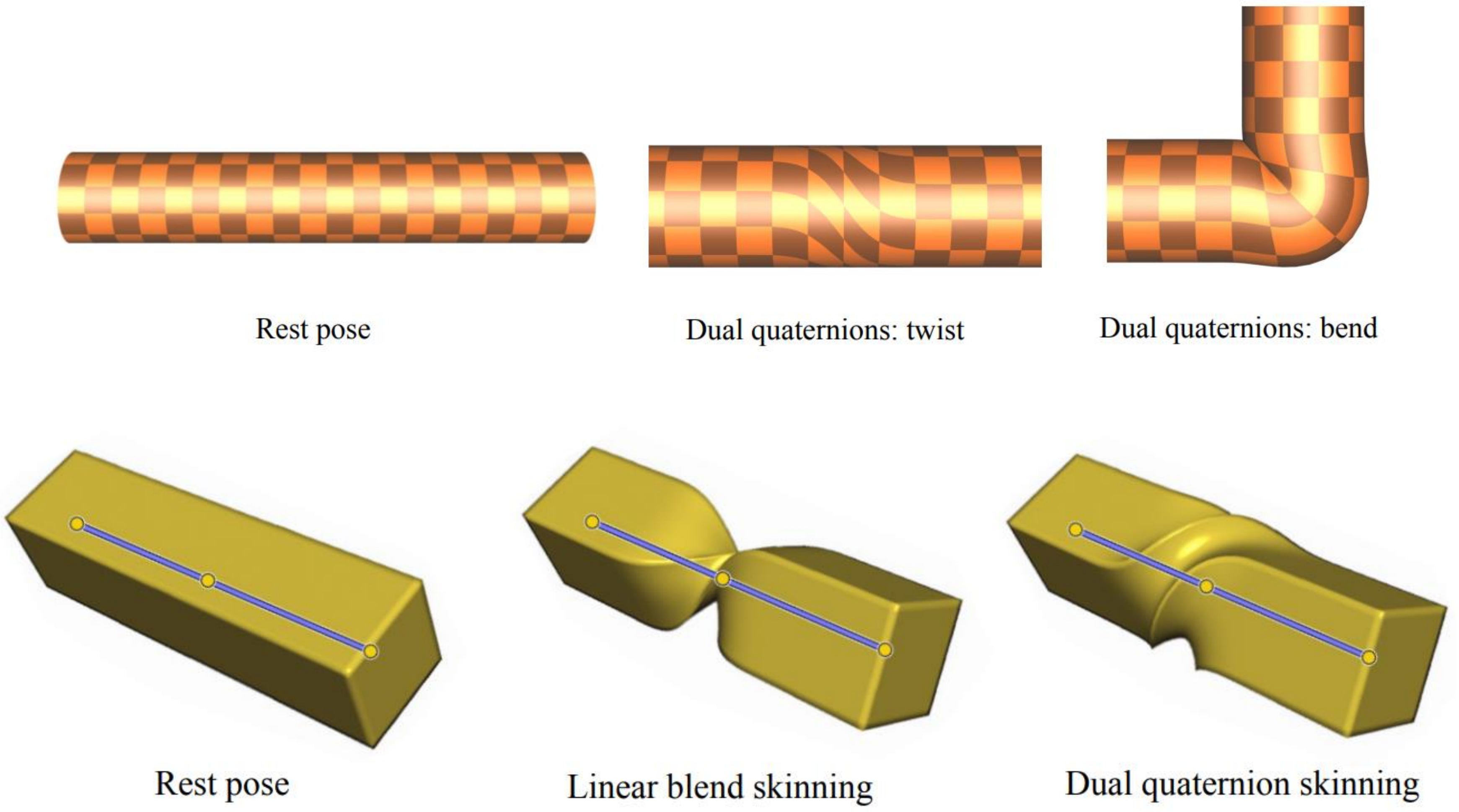
Dual-Quaternion Linear Blending (DLB)

$$\hat{\boldsymbol{q}}_t = (1-t)\hat{\boldsymbol{q}}_{\mathbf{0}} + t\hat{\boldsymbol{q}}_{\mathbf{1}}$$

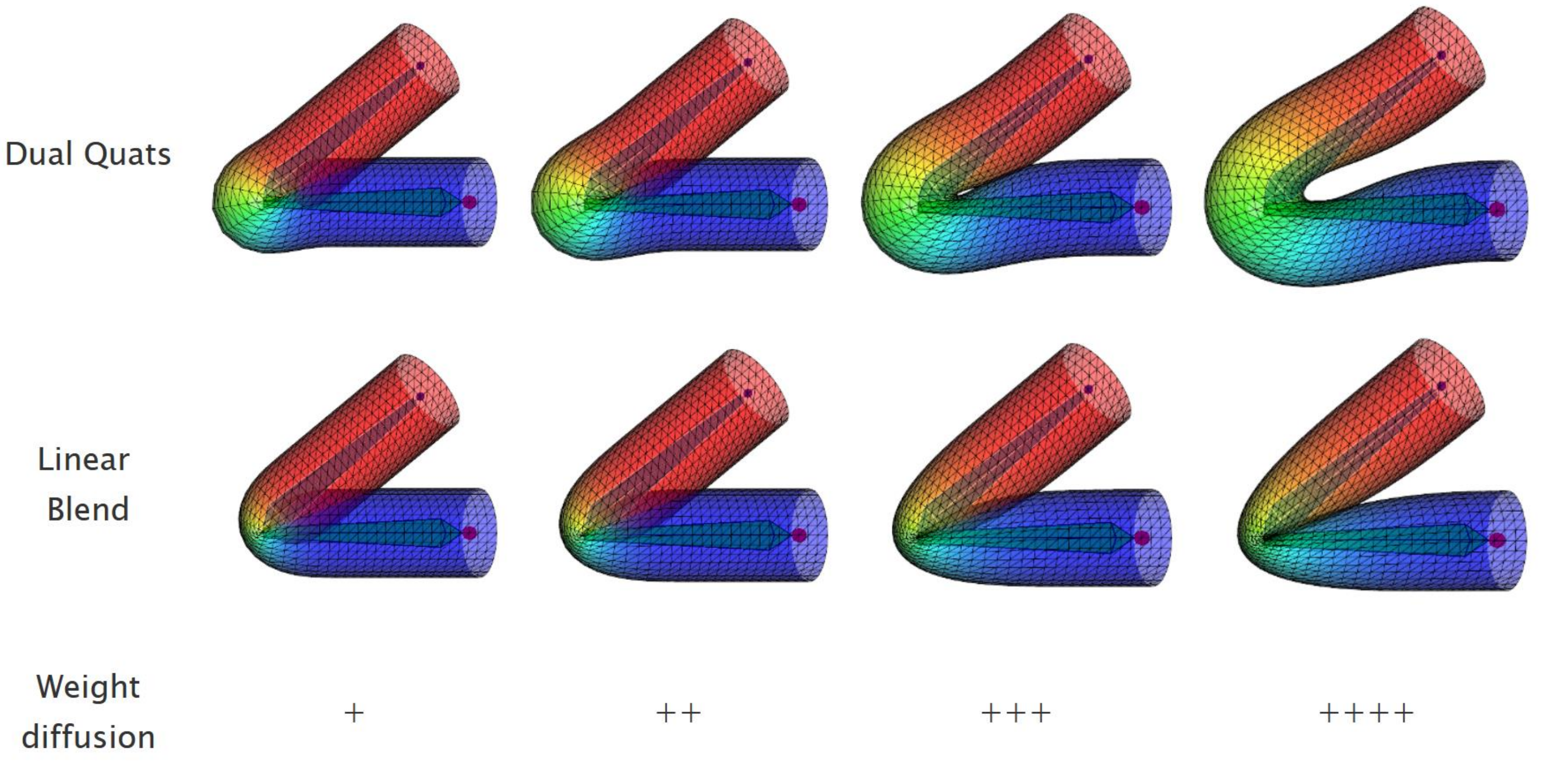
$$\hat{\boldsymbol{q}}_t = \frac{(1-t)\hat{\boldsymbol{q}}_{\mathbf{0}} + t\hat{\boldsymbol{q}}_{\mathbf{1}}}{\|(1-t)\hat{\boldsymbol{q}}_{\mathbf{0}} + t\hat{\boldsymbol{q}}_{\mathbf{1}}\|}$$



Dual-Quaternion Skinning (DQS)



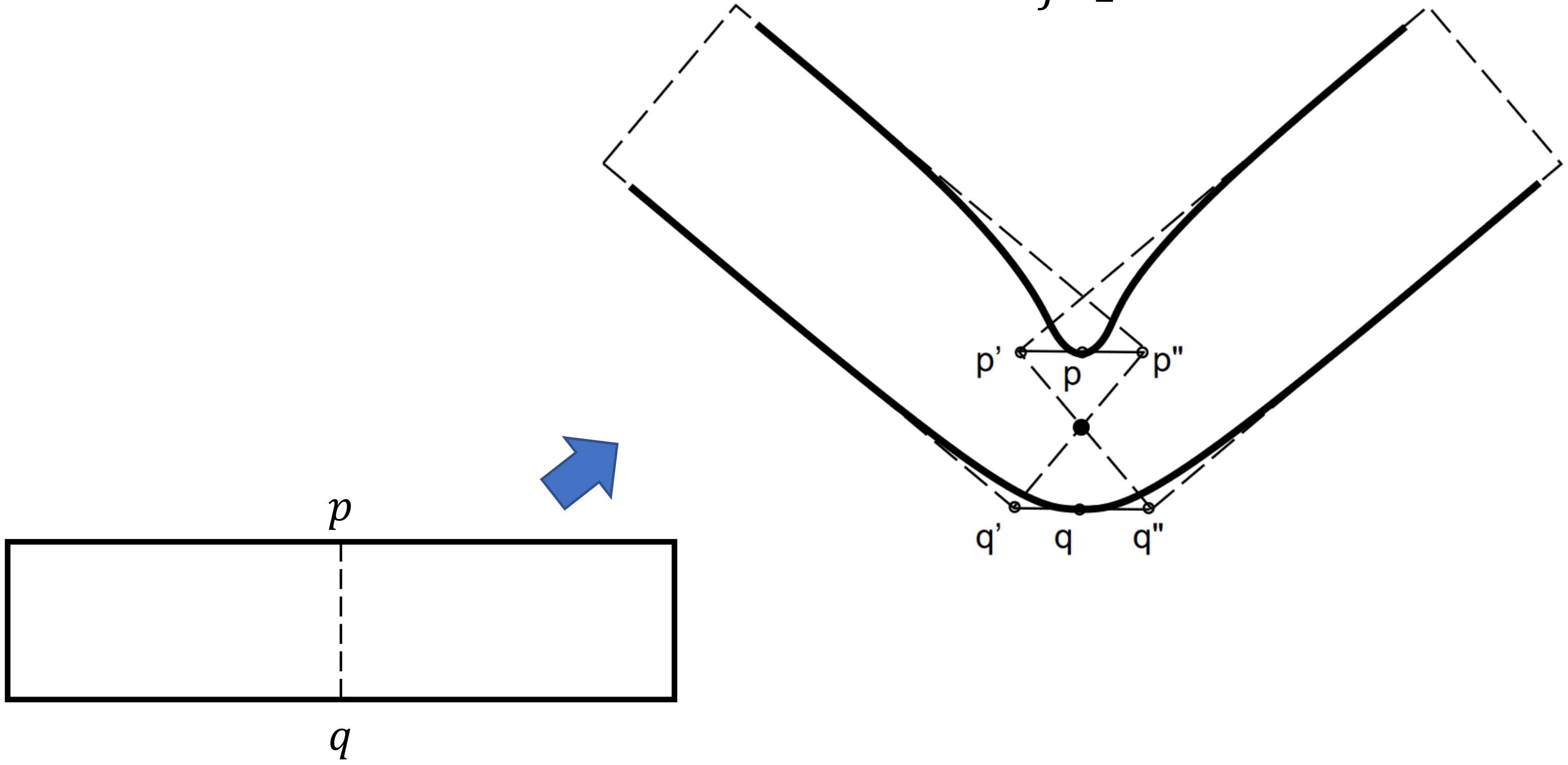
Budging Artifact of DQS



http://rodolphe-vaillant.fr/entry/29/dual-quaternions-skinning-tutorial-and-c-codes

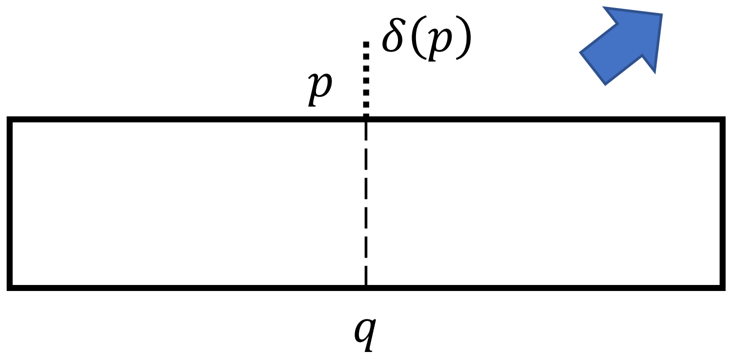
How to Correct LBS?

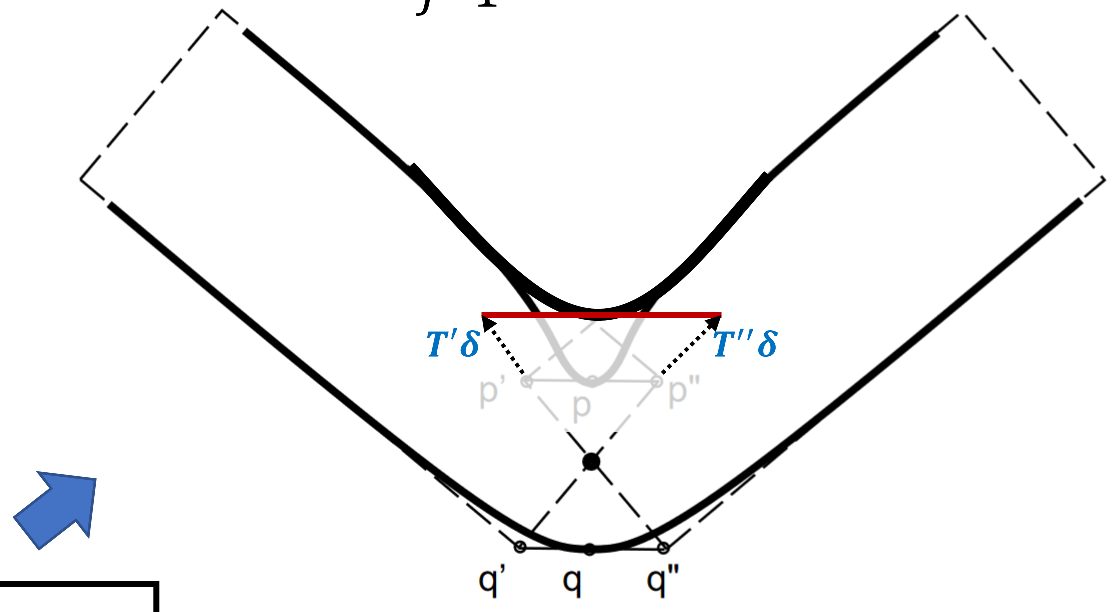
$$\boldsymbol{x}' = \sum_{j=1}^{m} \omega_j T_j \boldsymbol{x}$$



How to Correct LBS?

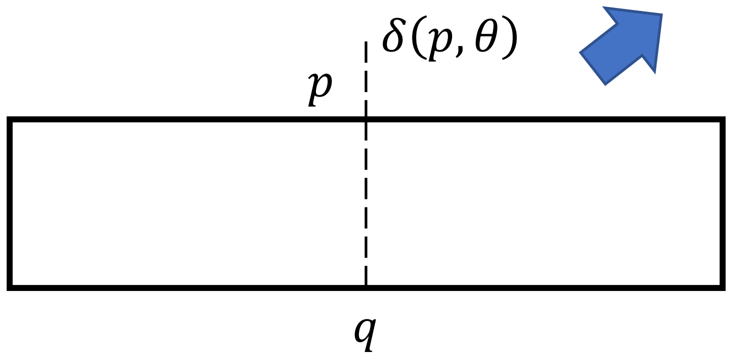
$$\boldsymbol{x}' = \sum_{j=1}^{m} \omega_j T_j\big(\boldsymbol{x} + \boldsymbol{\delta}(\boldsymbol{x})\big)$$

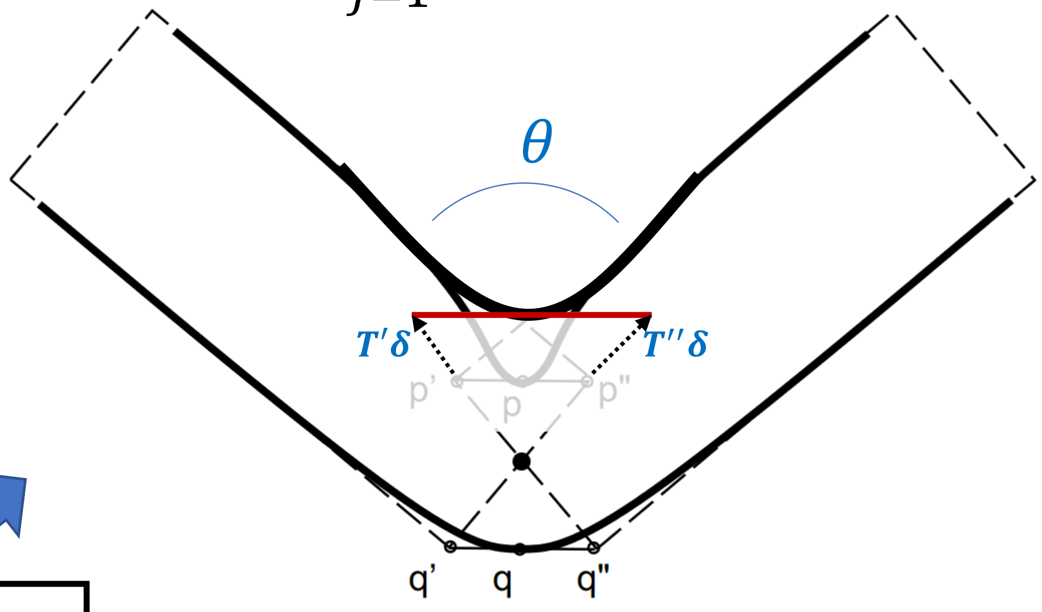




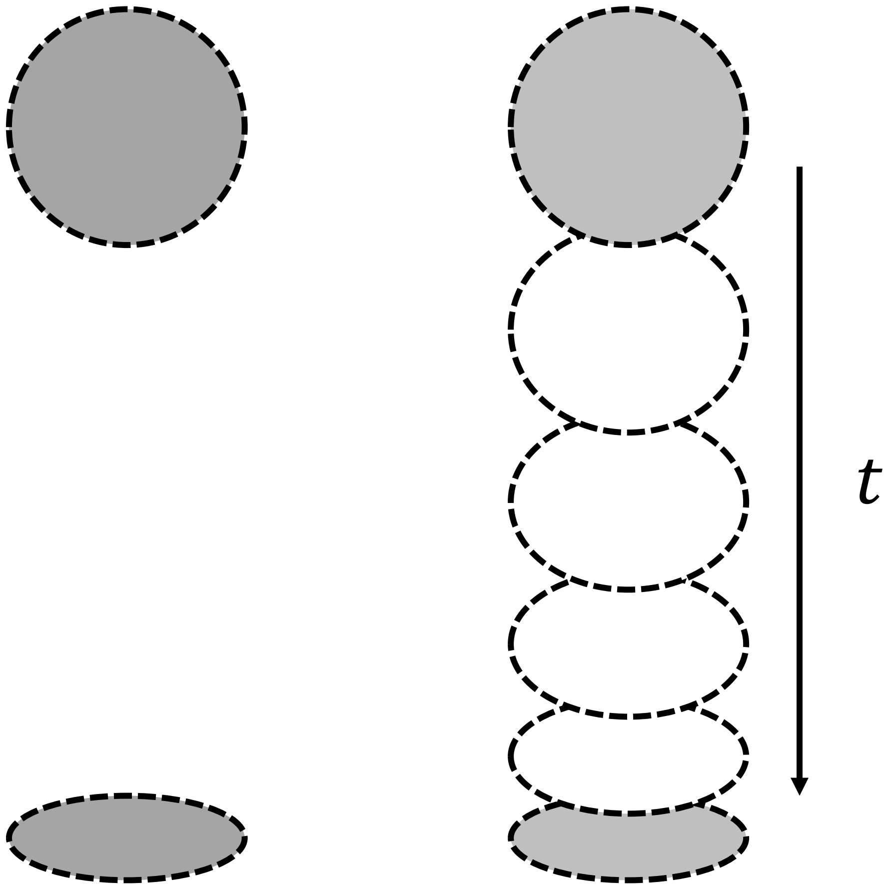
How to Correct LBS?

$$\boldsymbol{x}' = \sum_{j=1}^{m} \omega_j T_j\left(\boldsymbol{x} + \boldsymbol{\delta}(\boldsymbol{x}, \boldsymbol{\theta})\right)$$

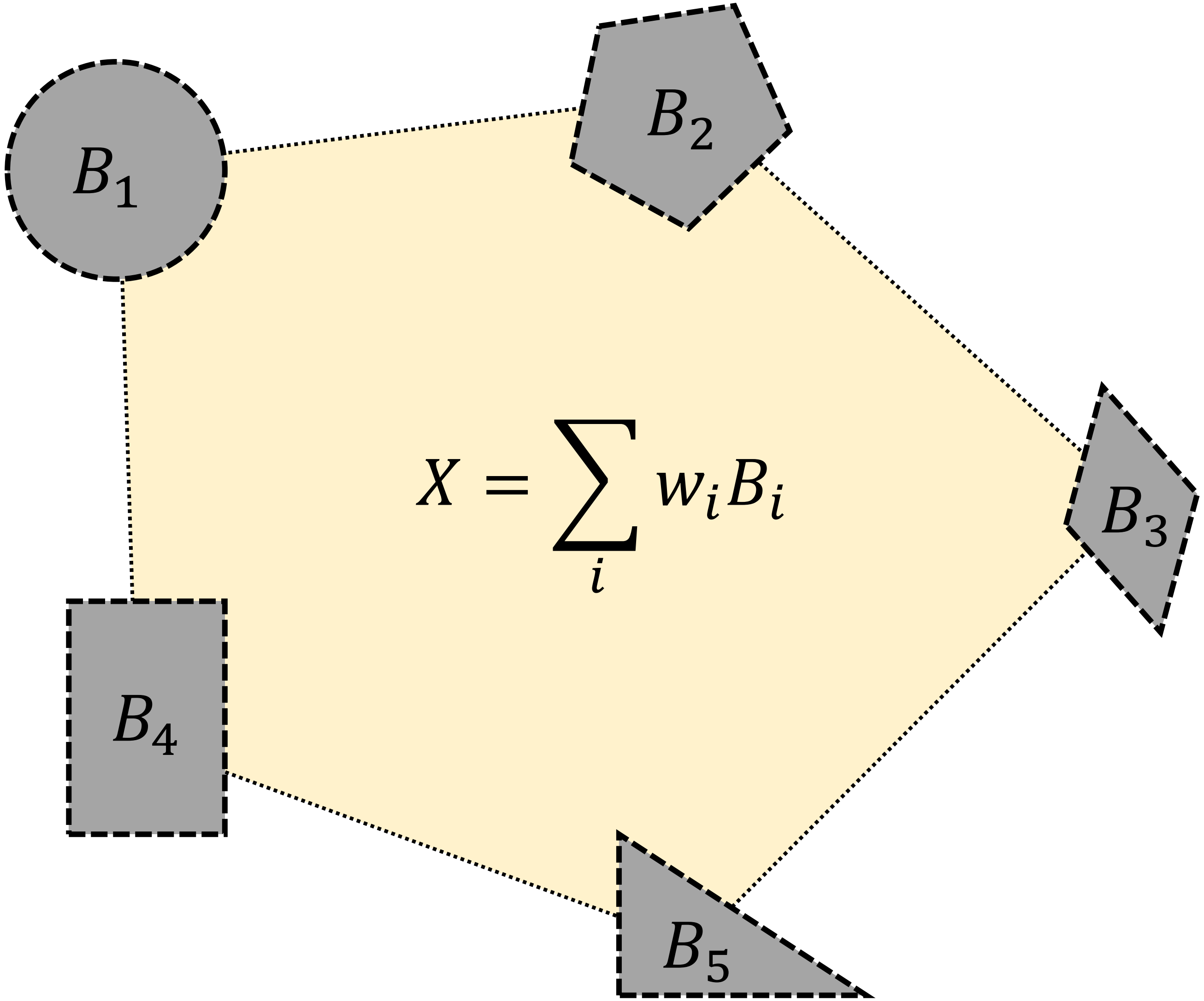




Example-based Shape Deformation

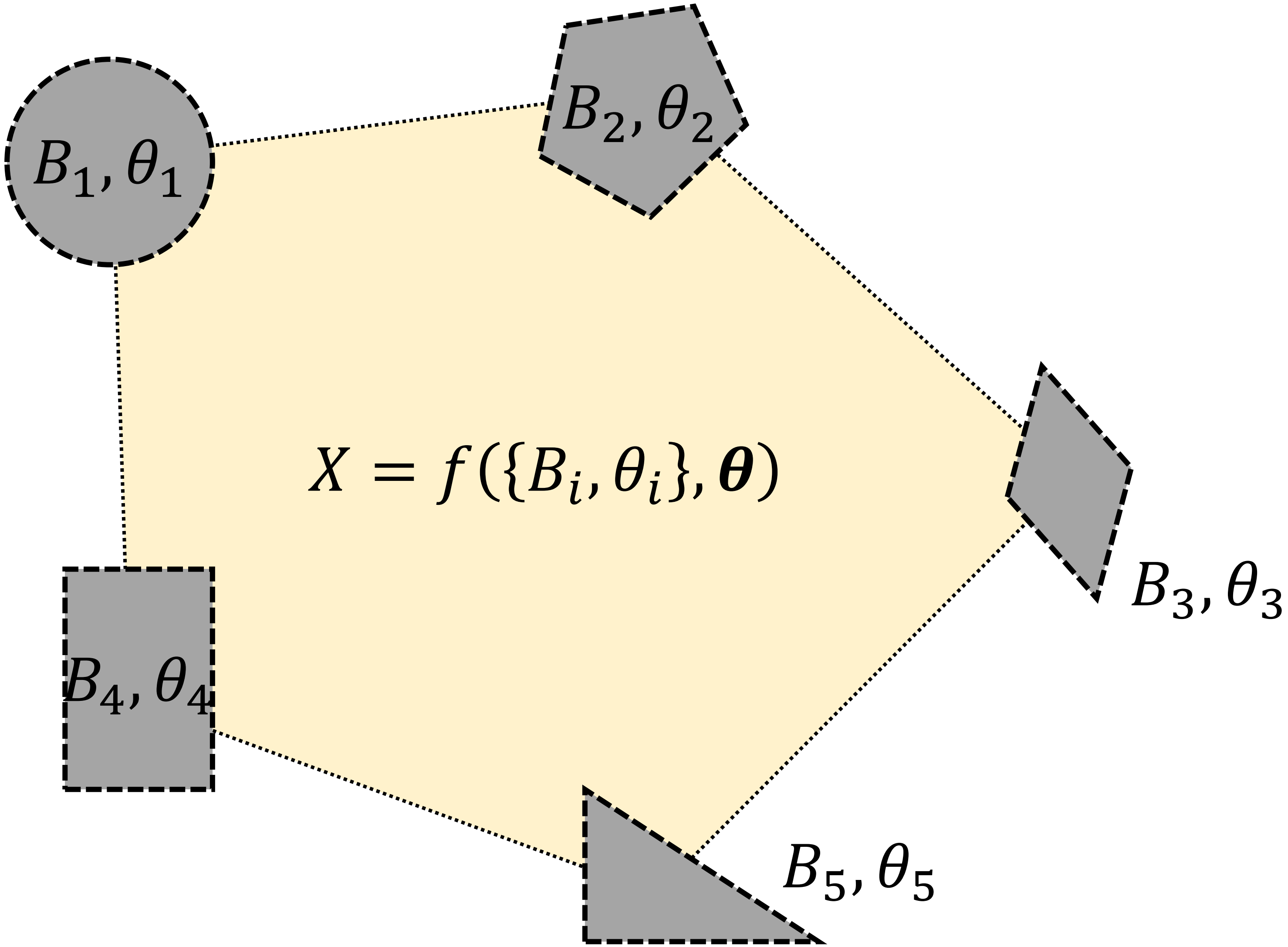


Blendshapes / Blend Space

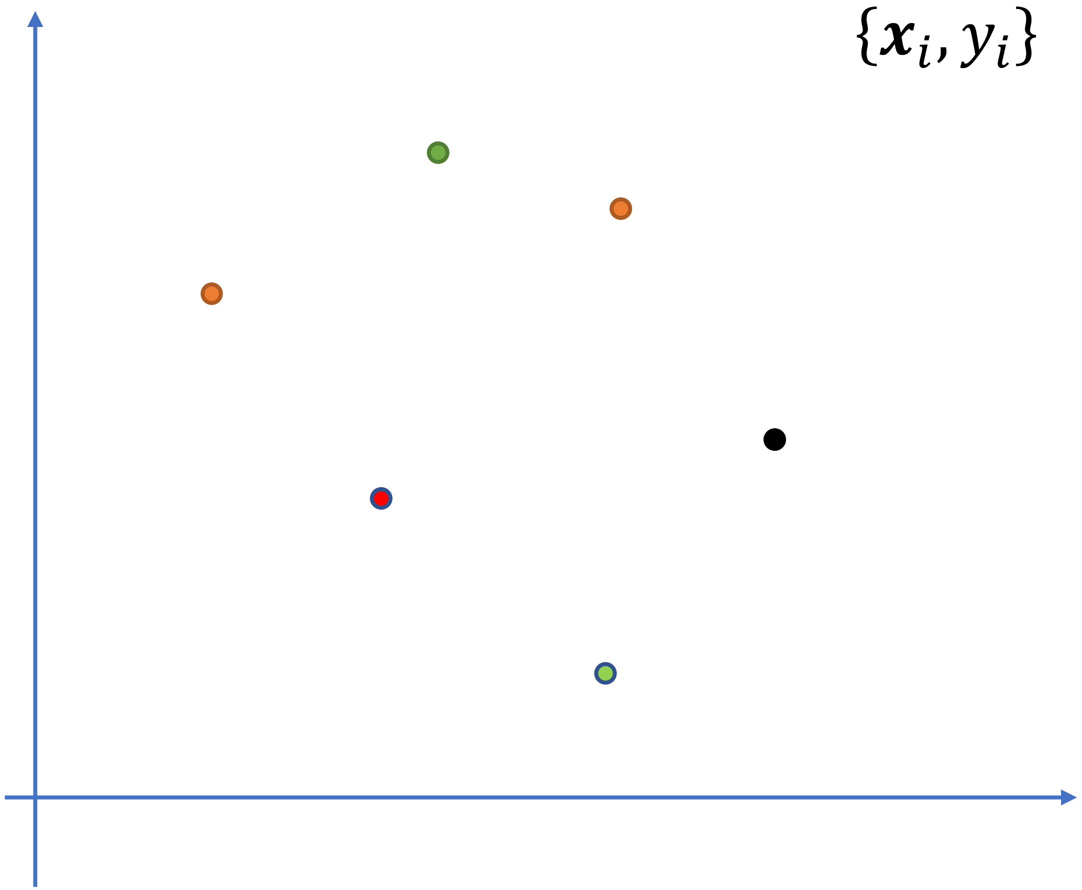


$$X = \sum_i w_i B_i$$

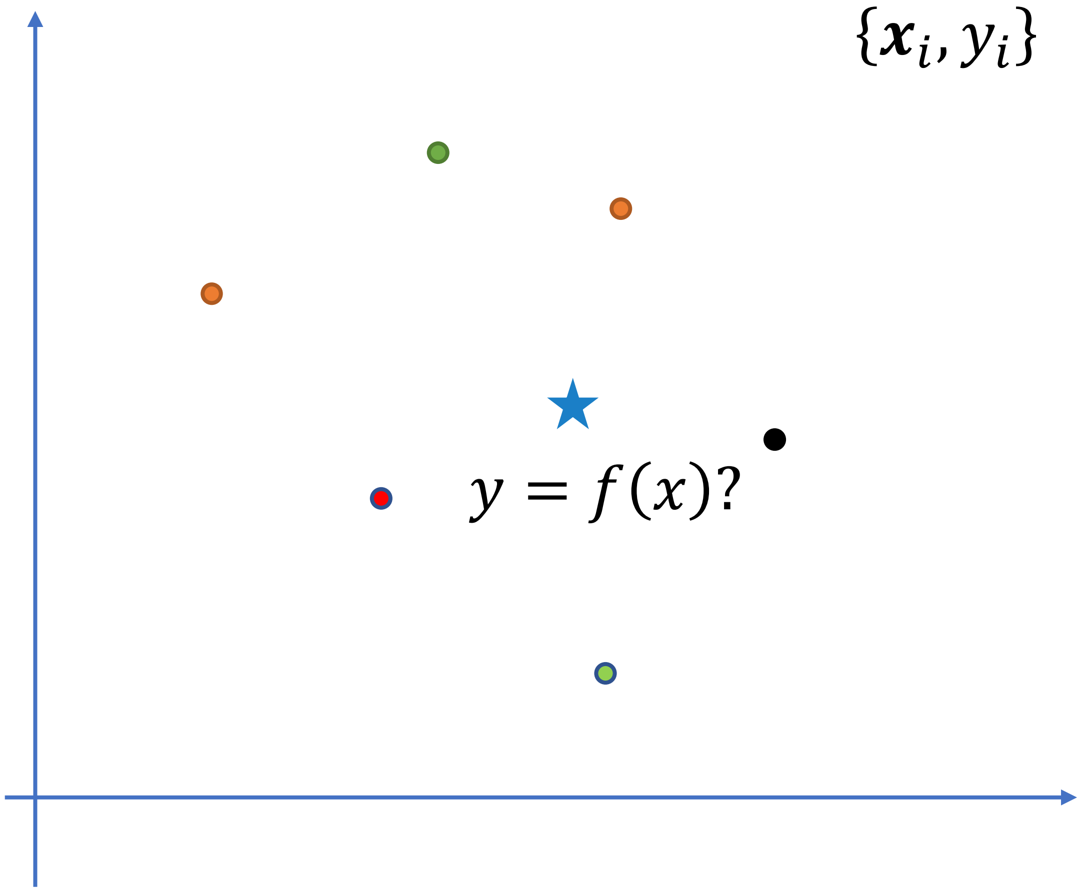
Pose Space Deformation



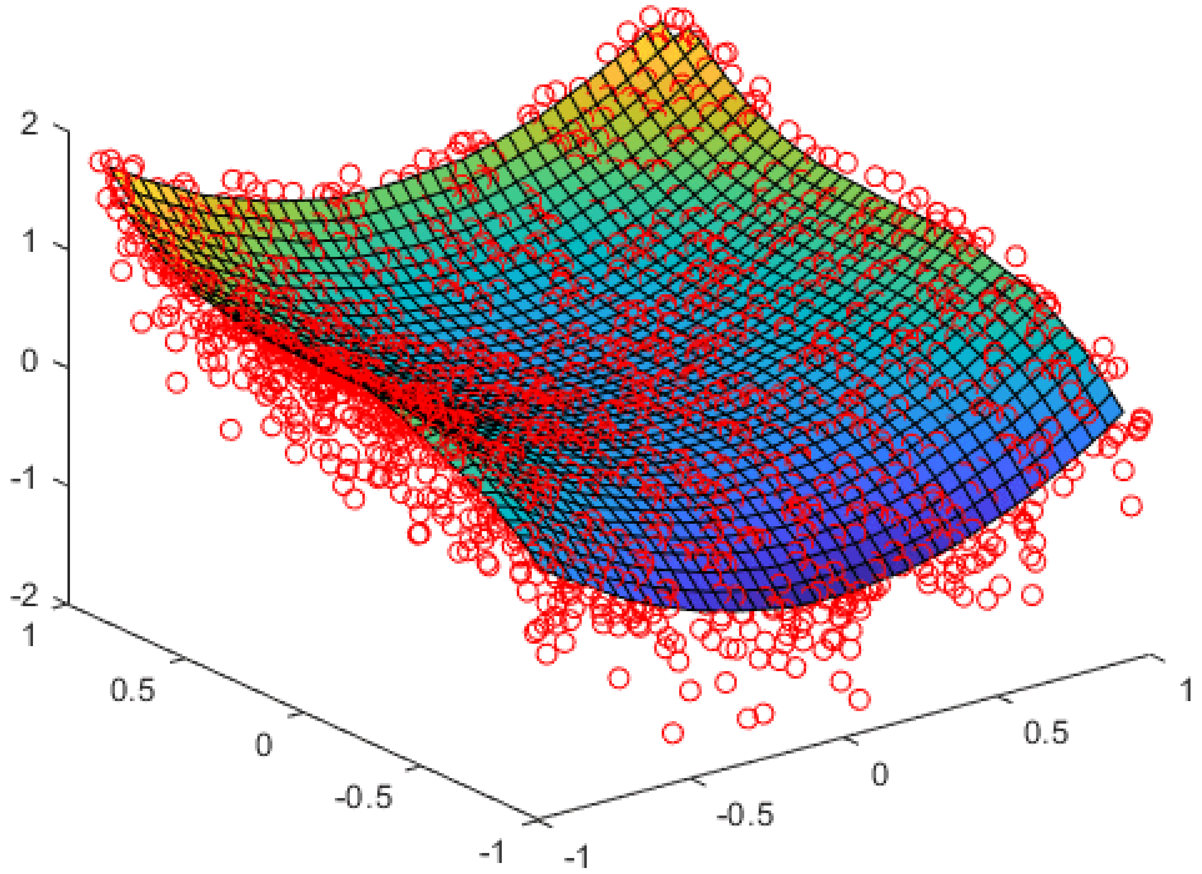
Scattered Data Interpolation



Scattered Data Interpolation



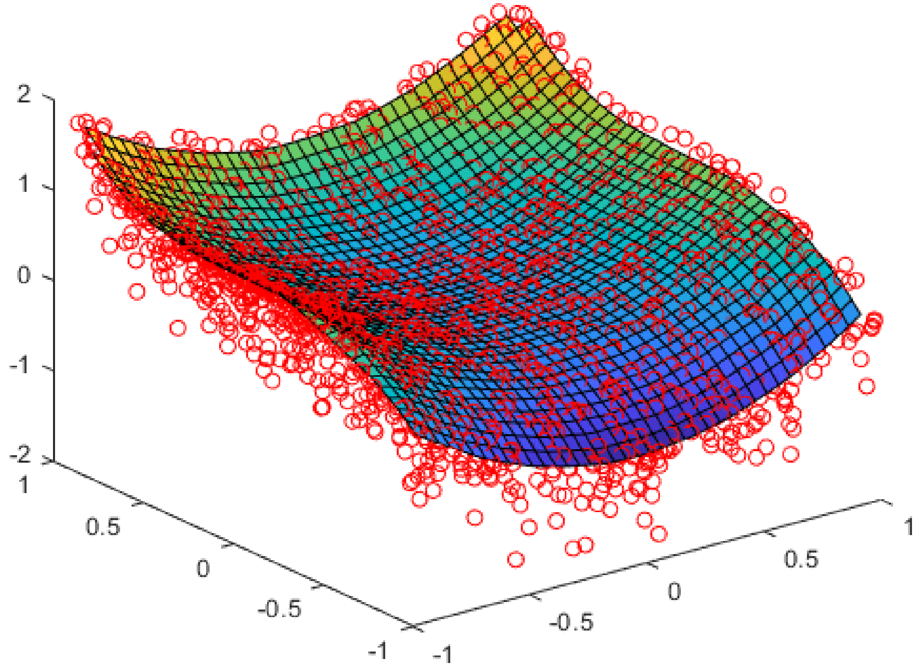
Scattered Data Interpolation



https://www.mathworks.com/help/matlab/ref/griddata.html

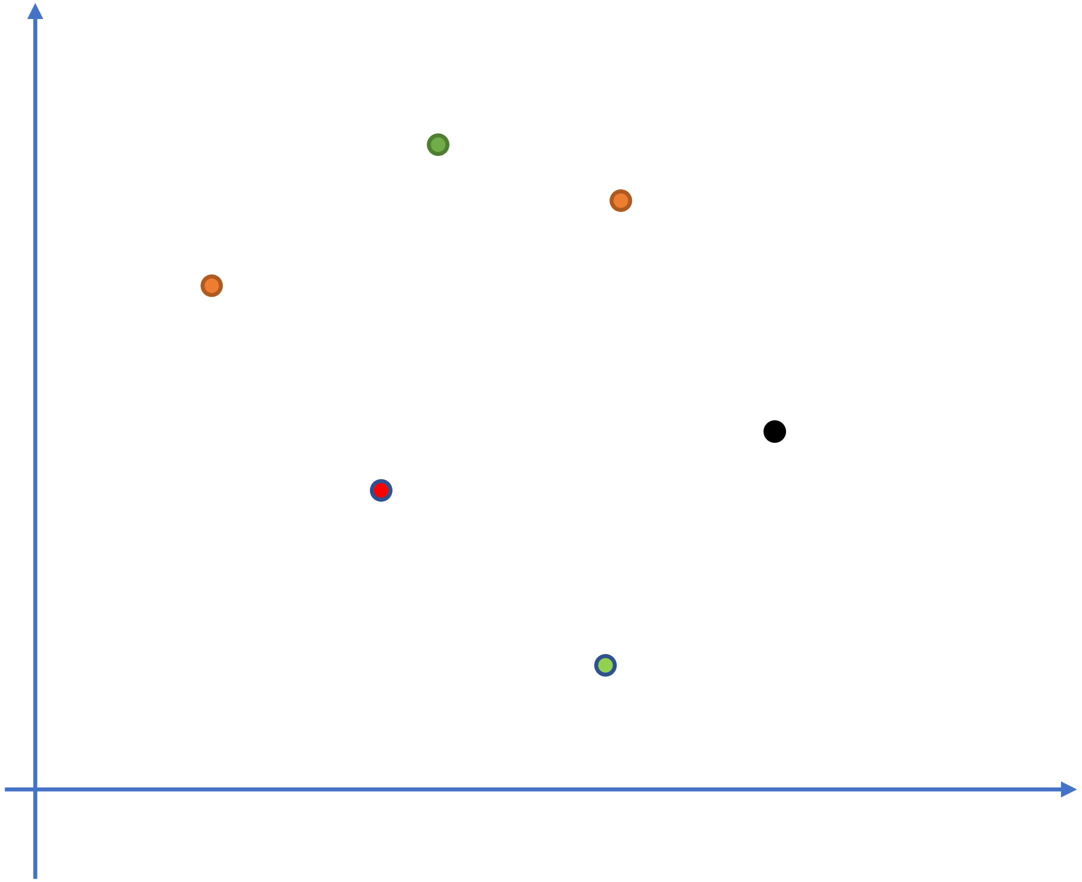
Scattered Data Interpolation

- Linear
  - Least squares
- Splines
- Inverse distance weighting
- Gaussian process
- Radial Basis Function
- ……

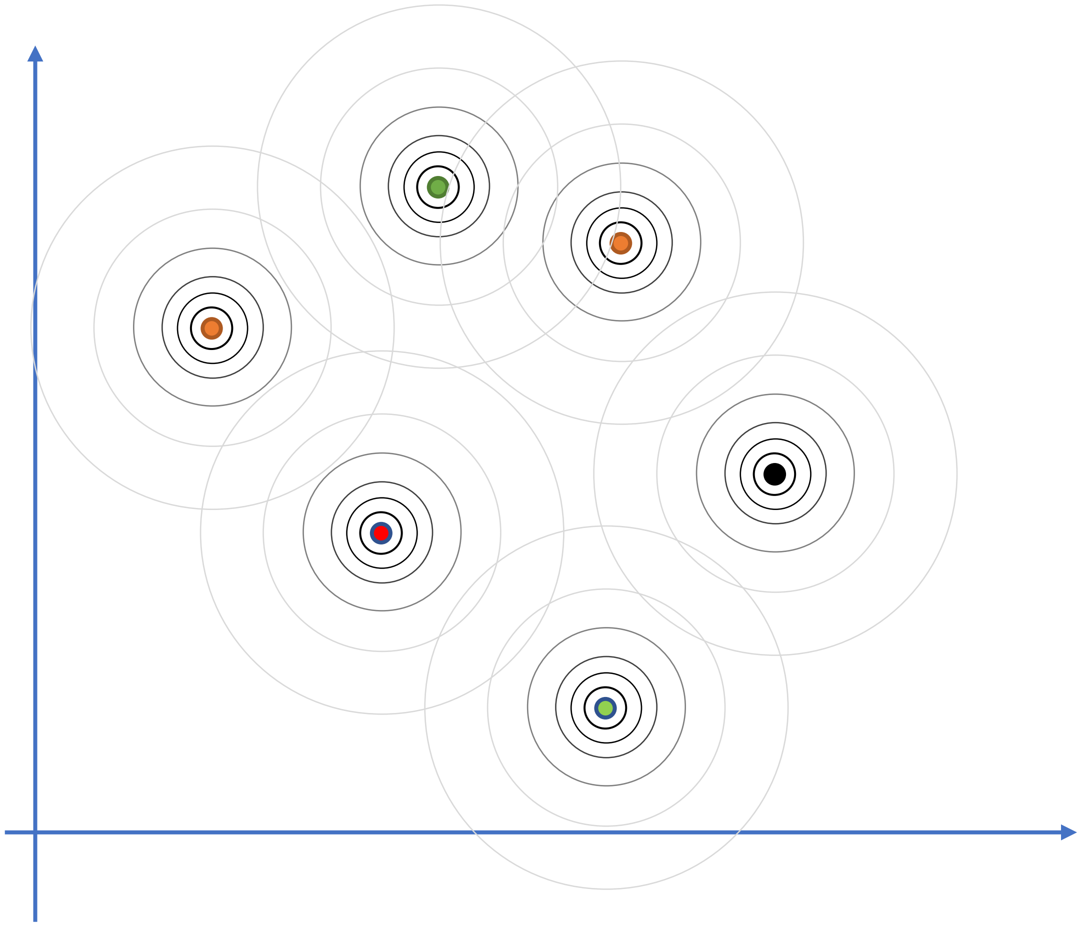


https://www.mathworks.com/help/matlab/ref/griddata.html

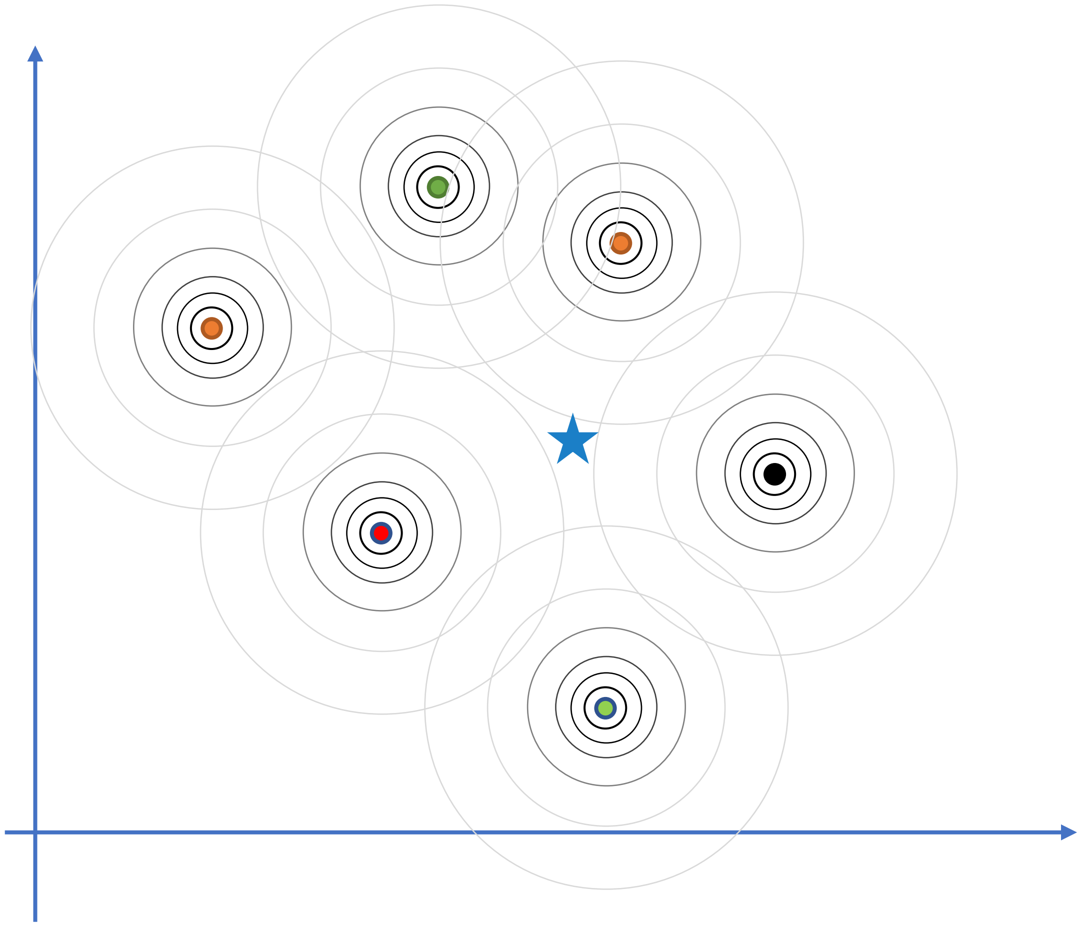
Radial Basis Function (RBF) Interpolation



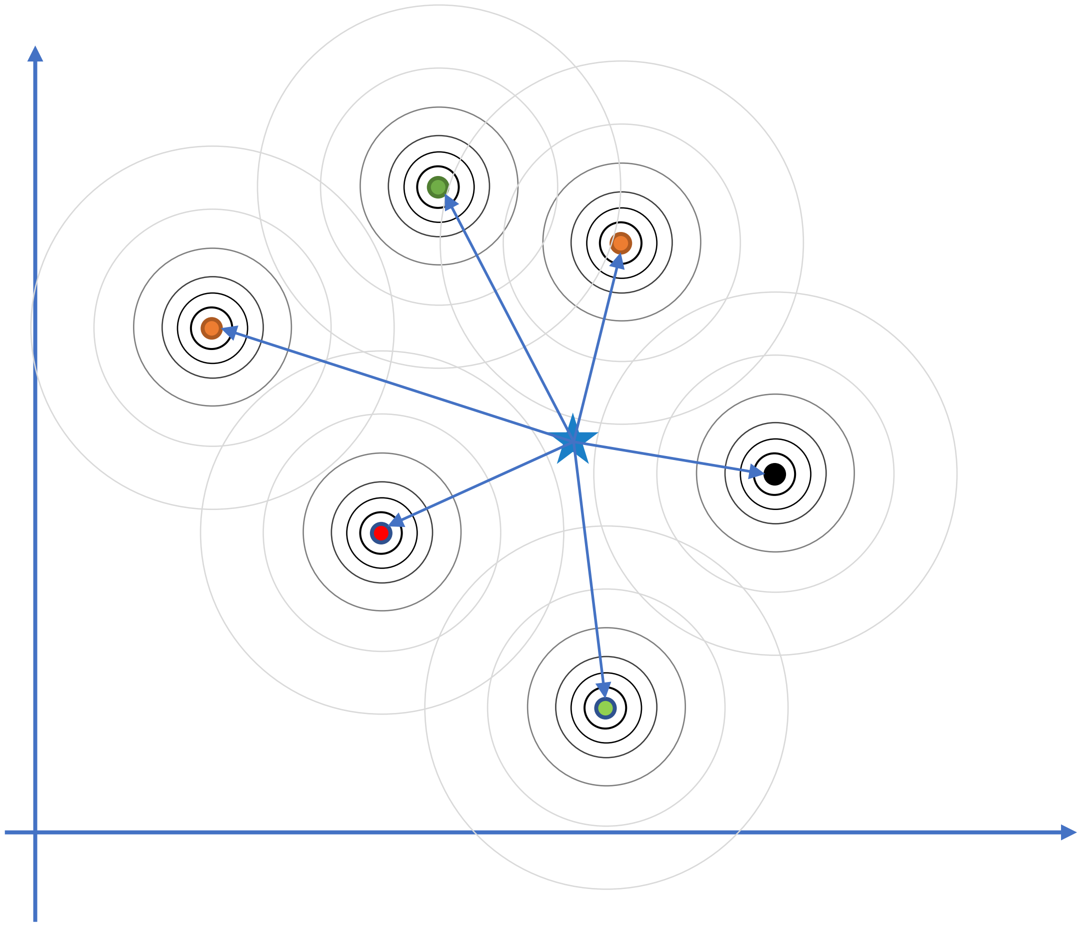
Radial Basis Function (RBF) Interpolation



Radial Basis Function (RBF) Interpolation

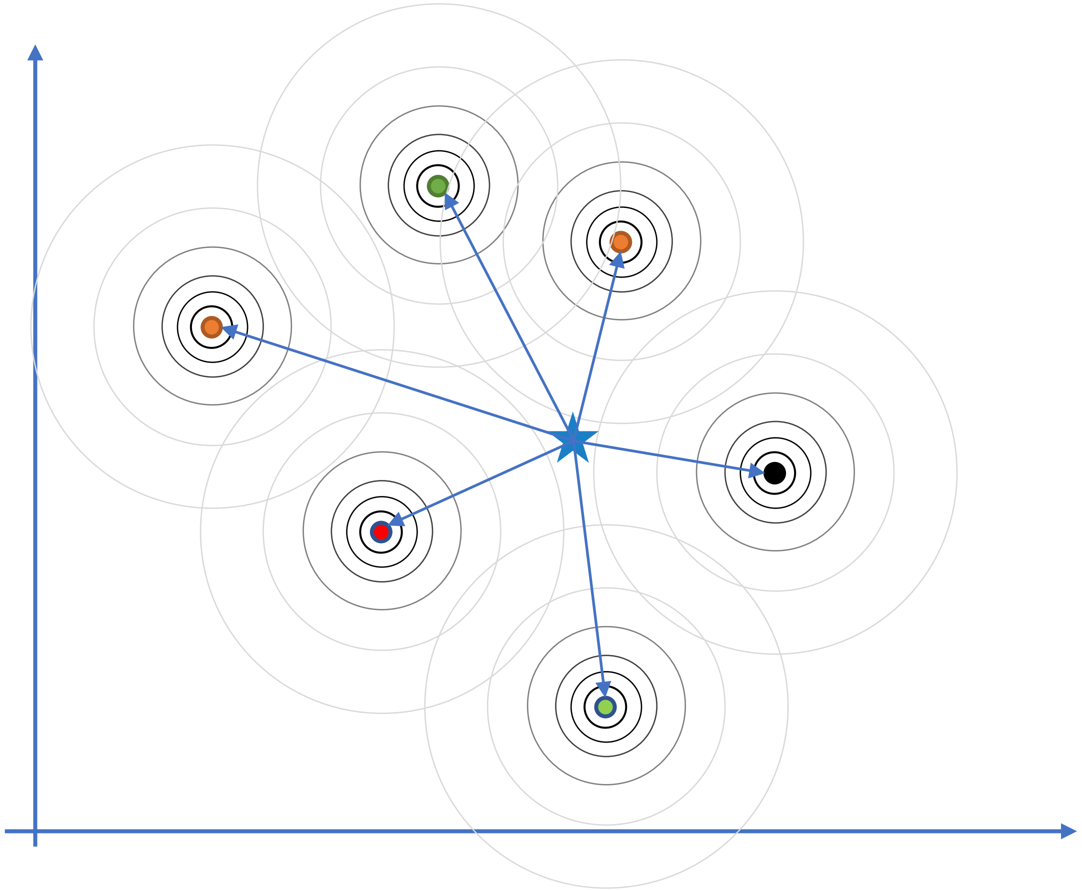


Radial Basis Function (RBF) Interpolation



Radial Basis Function (RBF) Interpolation

$$y = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$$



Radial Basis Function (RBF) Interpolation

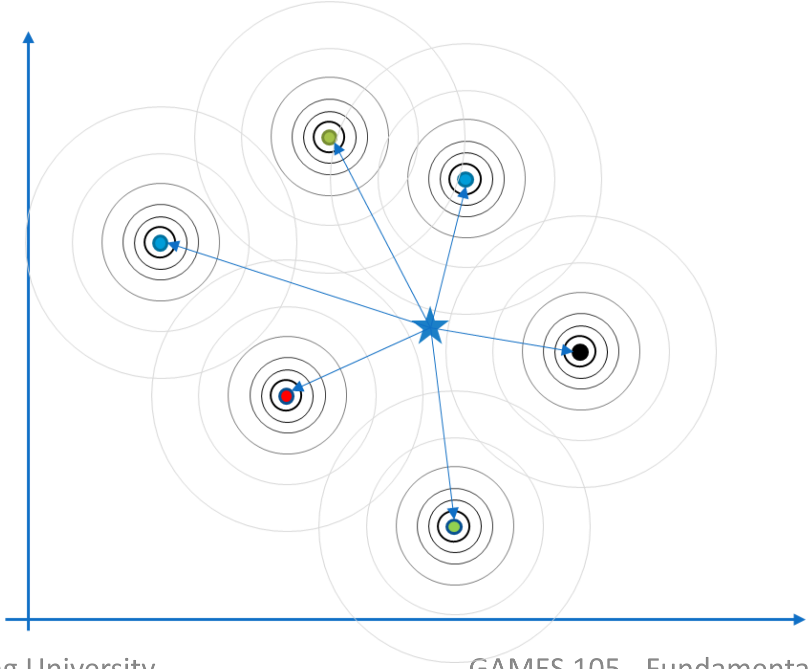
$$y = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$$

How to compute w_i ?

Radial Basis Function (RBF) Interpolation

$$y = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$$

How to compute w_i ? We need $f(\boldsymbol{x}_i) = y_i$



$$\begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,K} \\ R_{2,1} & R_{2,2} & & \vdots \\ \vdots & & \ddots & \vdots \\ R_{K,1} & \cdots & \cdots & R_{K,K} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$$

$$R_{i,j} = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$$

Radial Basis Function (RBF)

$$y = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$$

- Gaussian: $\varphi(r) = e^{-(r/c)^2}$
- Inverse multiquadric: $\varphi(r) = \frac{1}{\sqrt{r^2+c^2}}$
- Thin plate spline: $\varphi(r) = r^2 \log r$
- Polyharmonic splines: $\varphi(r) = \begin{cases} r^k, & k = 2n+1 \\ r^k \log r, & k = 2n \end{cases}$

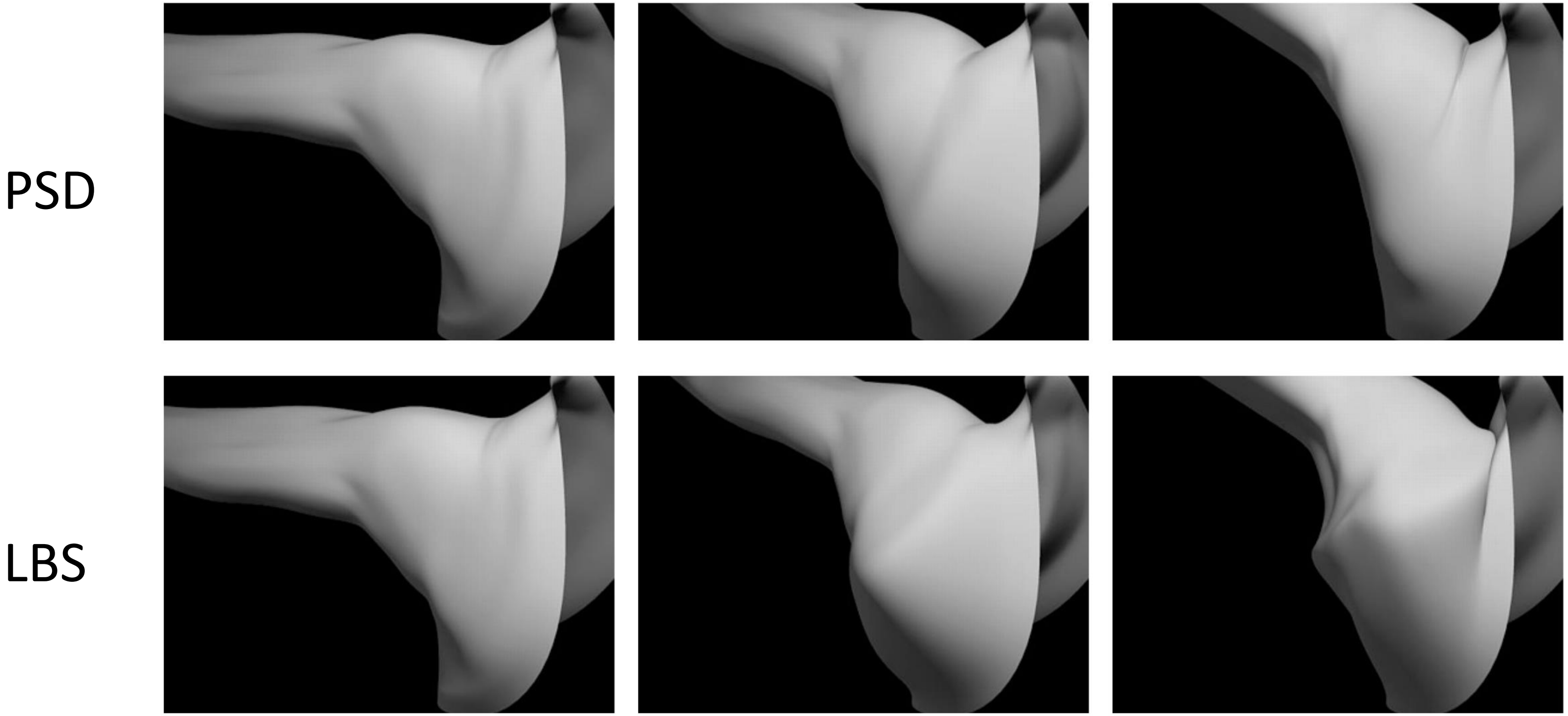
Pose Space Deformation

$$\boldsymbol{x}' = \sum_{j=1}^{m} \omega_j T_j\left(\boldsymbol{x} + \boldsymbol{\delta}(\boldsymbol{x}, \theta)\right)$$

- $\boldsymbol{x}' = SKIN\left(PSD(\boldsymbol{x})\right)$
- PSD is implemented as RBF interpolation
- Example shapes can be created manually
  - Or by 3D scanning real people → the SMPL model

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. ***Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation***. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '00), ACM Press/Addison-Wesley Publishing Co., USA, 165–172.

Pose Space Deformation



J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. ***Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation***. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '00), ACM Press/Addison-Wesley Publishing Co., USA, 165–172.

Issues

- Per-shape or per-vertex interpolation
  - Should we interpolate a shape as a whole?
- Local or global interpolation?
  - Should a vertex be affected by all joints?
- Interpolation algorithm?
  - Is RBF the only choice?

SIGGRAPH Course 2014 — Skinning: Real-time Shape Deformation

Example-based Skinning (EBS) vs. Skeleton Subspace Deformation (SSD)

*EBS: PSD

- Good: Easy to control
- Good: Good quality
- Good: Pose-dependent details (e.g. bulging muscle and extruding veins)

- Bad: Creating examples can be cumbersome
- Bad: Extra storage for examples
- Bad: Interpolation needs careful tuning

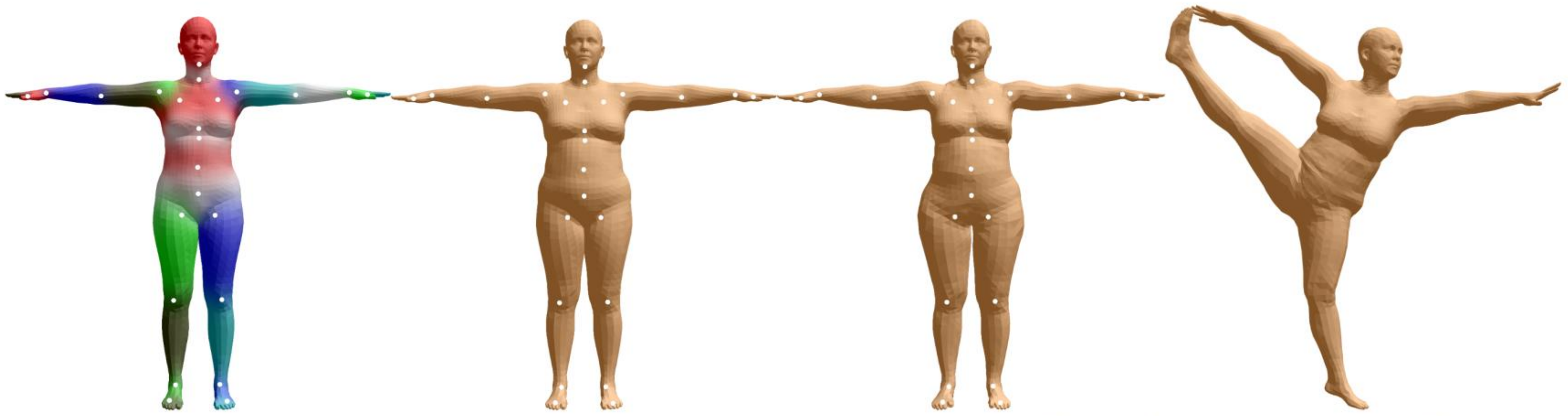
*SSD: LBS, DQS, etc.

- Good: Easy to implement
- Good: Fast and GPU friendly

- Bad: Various artifacts
- Bad: Skinning weights needs careful tuning
- Bad: Hard to create pose-dependent details

Example: SMPL Model

- A widely adopted human model in ML/CV
- Learned on real scan data
- Combines SSD and EBS techniques

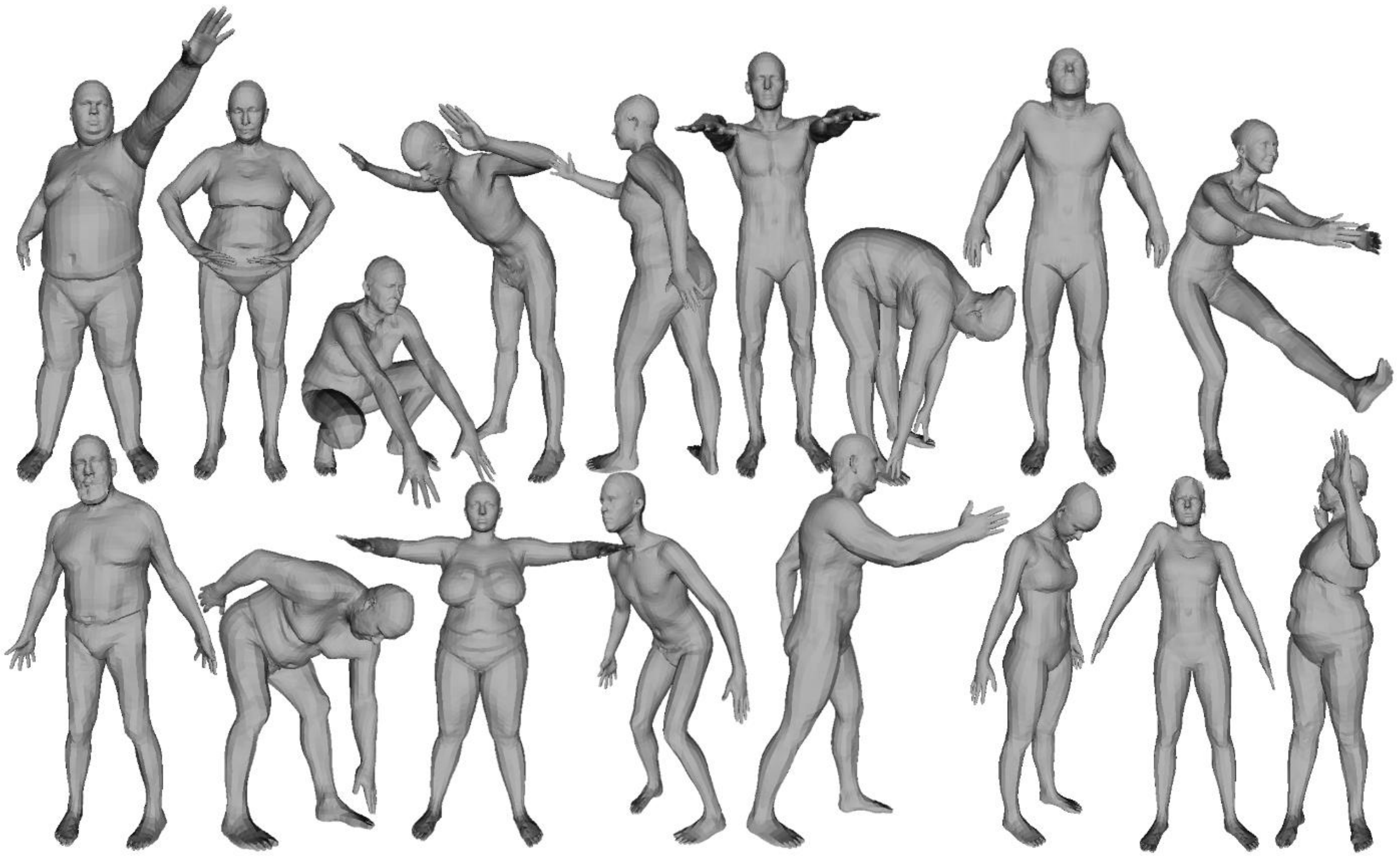


(a) $\bar{\mathbf{T}}, \mathcal{W}$ (b) $\bar{\mathbf{T}} + B_S(\vec{\beta}), J(\vec{\beta})$ (c) $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$ (d) $W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

Figure 3: SMPL model. *(a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blendshape contribution only; vertex and joint locations are linear in shape vector $\vec{\beta}$. (c) With the addition of of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by dual quaternion skinning for the split pose.*

[SMPL: A Skinned Multi-Person Linear Model]

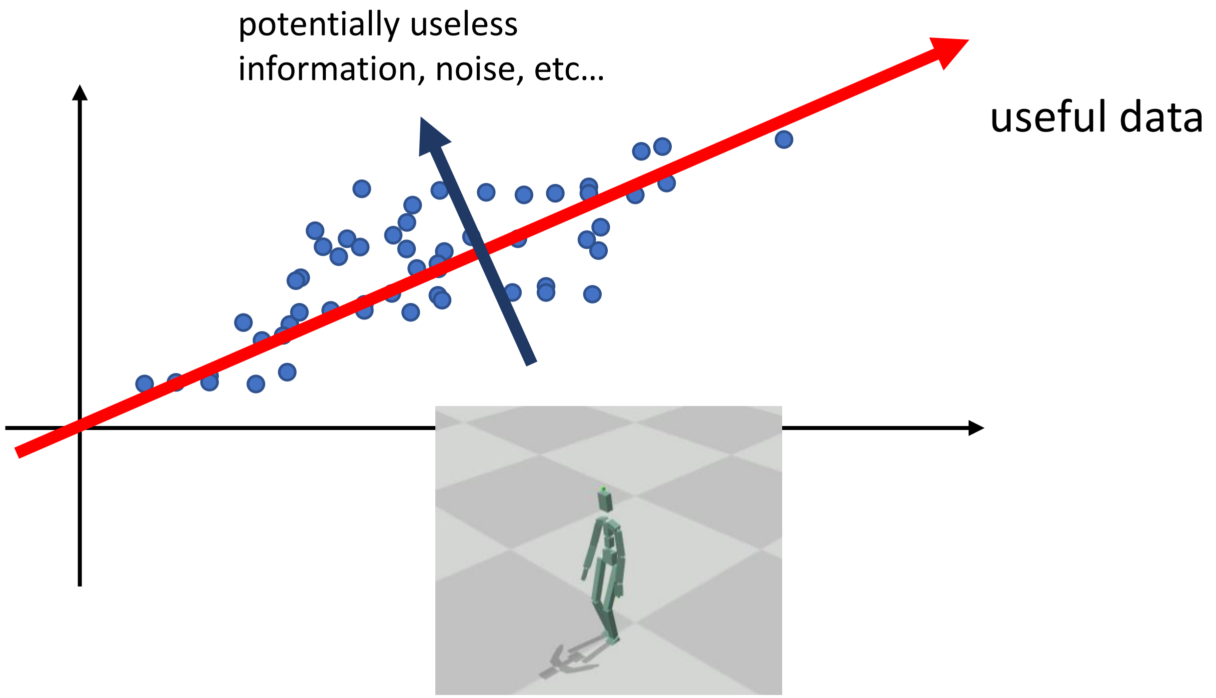
How to deal with massive examples?

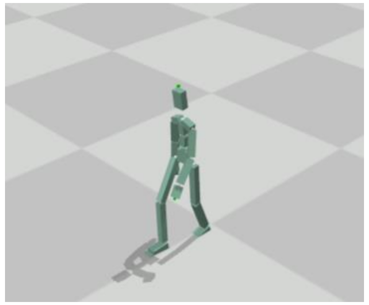


[SMPL: A Skinned Multi-Person Linear Model]

Recall: Principal Component Analysis (PCA)

- A technique for
  - finding out the correlations among dimensions
  - dimensionality reduction





Recall: Principal Component Analysis (PCA)

- Given a dataset $\{\boldsymbol{x}_i\}, \boldsymbol{x}_i \in \mathbb{R}^N$, then PCA gives

$$\boldsymbol{x}_i = \bar{\boldsymbol{x}} + \sum_{k=1}^{n} w_{i,k} \boldsymbol{u}_{\boldsymbol{k}}$$

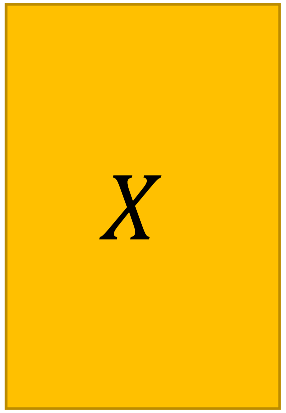
- $\boldsymbol{u}_{\boldsymbol{k}}$ is the k-th principal component
  - A direction in $\mathbb{R}^N$ along which the projection of $\{\boldsymbol{x}_i\}$ has the k-th maximal **variance**
- $w_{i,k} = (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \cdot \boldsymbol{u}_k$ is the score of $\boldsymbol{x}_i$ on $\boldsymbol{u}_k$

Recall: Principal Component Analysis (PCA)

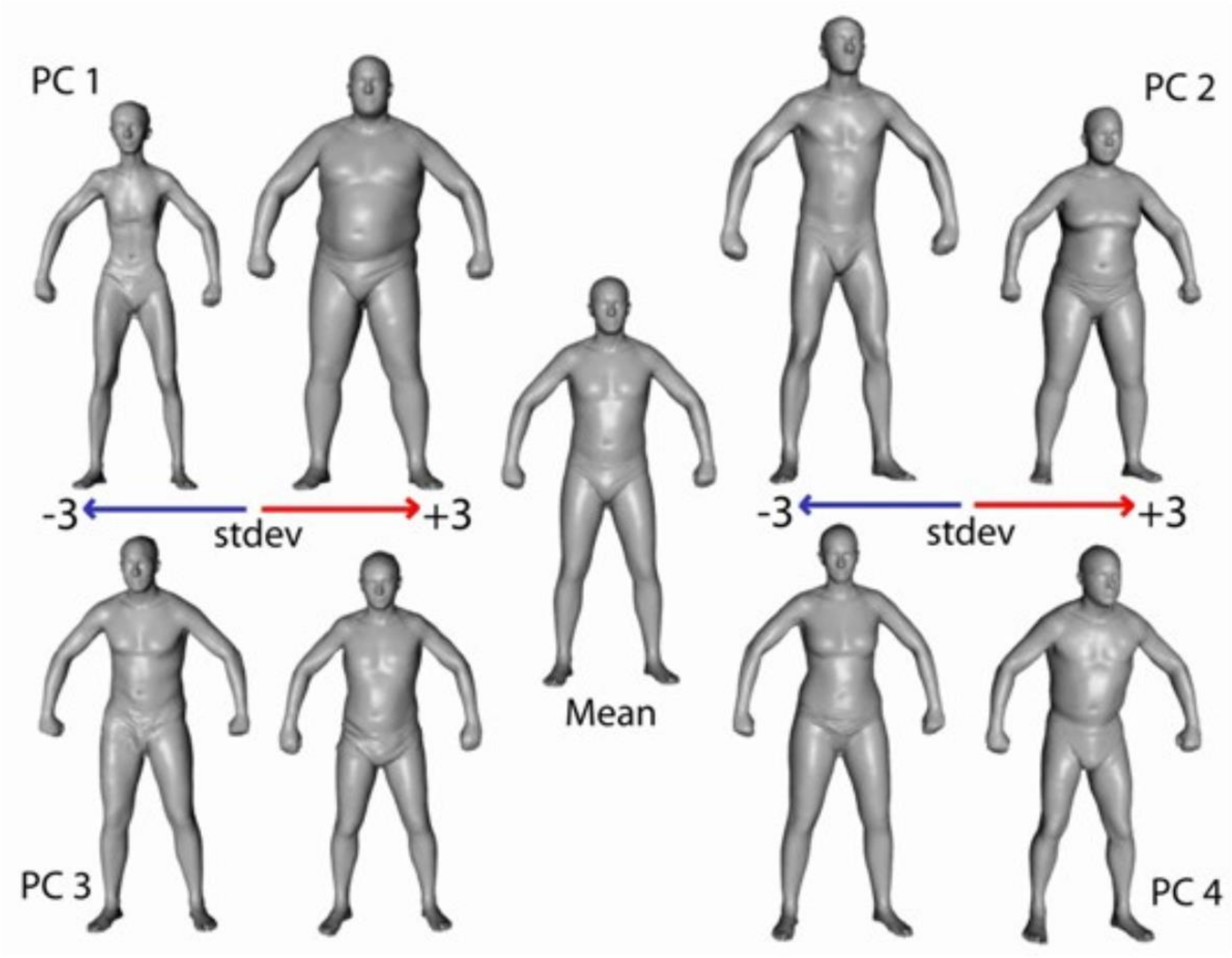
- Given a dataset $\{\boldsymbol{x}_i\}, \boldsymbol{x}_i \in \mathbb{R}^N$, the PCA can be computed by apply **eigen decomposition** on the covariance matrix

$$\Sigma = X^T X = U \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_N^2 \end{bmatrix} U^T$$

- $X = [\boldsymbol{x}_0 - \overline{\boldsymbol{x}}, \boldsymbol{x}_1 - \overline{\boldsymbol{x}}, \dots, \boldsymbol{x}_N - \overline{\boldsymbol{x}}]^T$
- $\sigma_i \geq \sigma_j \geq 0$ when $i < j$, corresponds to the Explained Variance
- $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_N]$



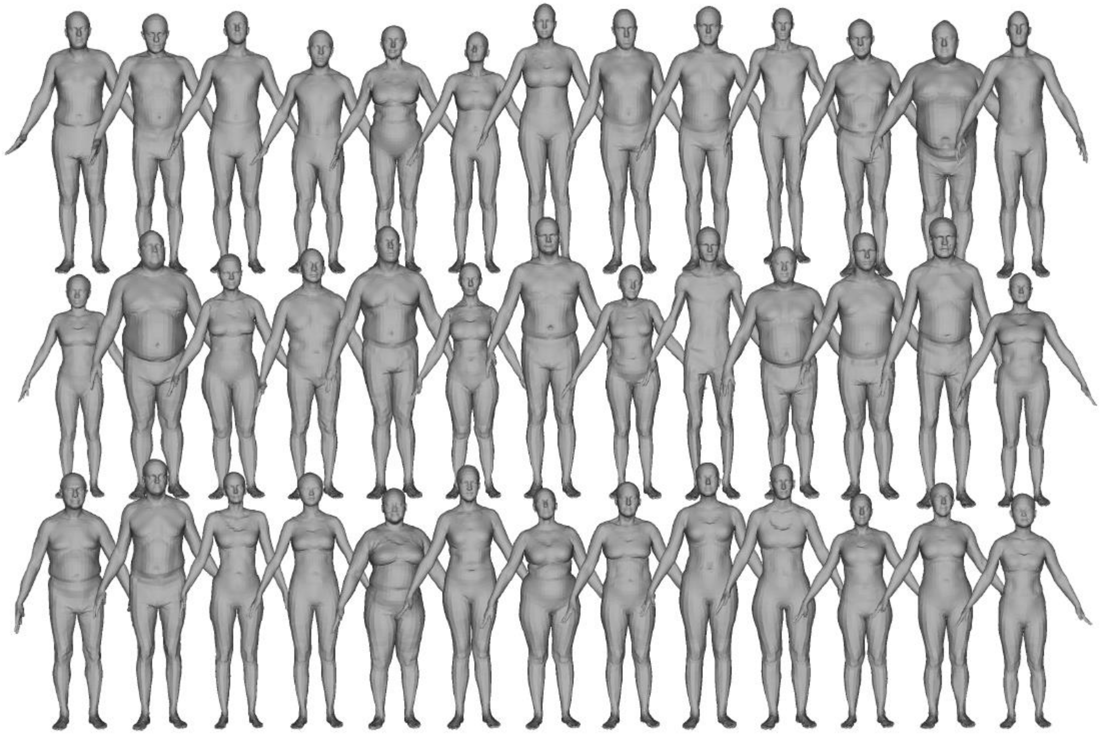
PCA over Body Shapes



Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. ***SCAPE: shape completion and animation of people***. *ACM Trans. Graph.* 24, 3 (July 2005), 408–416.

SMPL Model: Body Shape

$$T(\beta) = \bar{T} + \sum_{m=1}^{|\beta|} \beta_m S_n$$

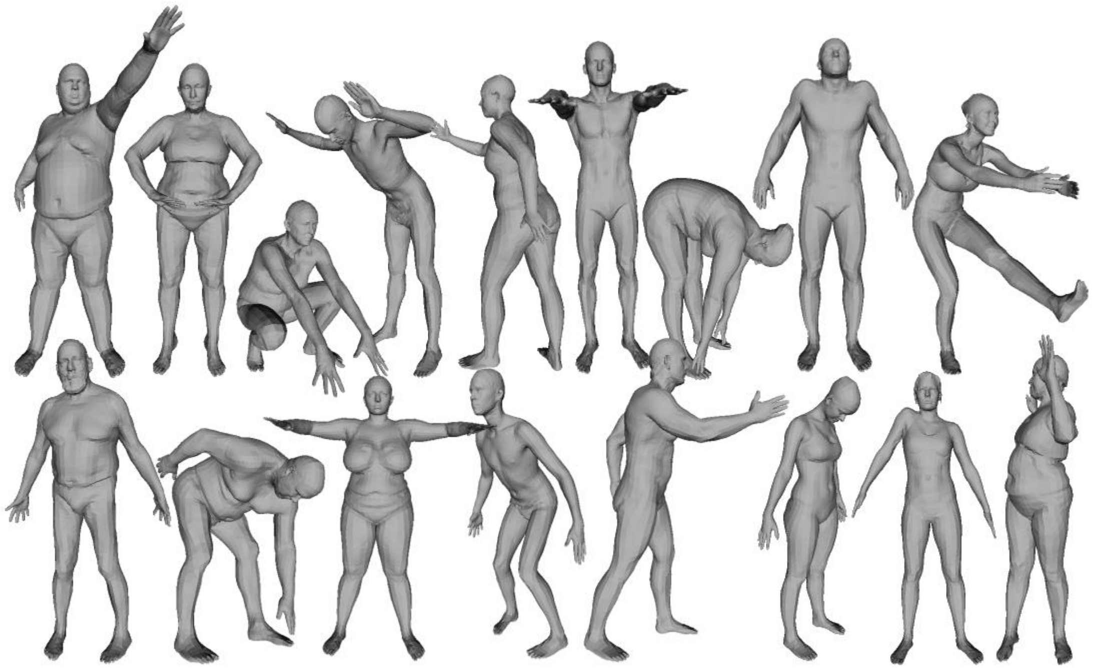


PCA

[SMPL: A Skinned Multi-Person Linear Model]

SMPL Model: Pose Blend Shapes

$$T(\beta,\theta) = \bar{T} + \sum_{m=1}^{|\beta|} \beta_m S_n + \sum_{n=1}^{|\theta|} \theta_n p_n$$



Optimize

[SMPL: A Skinned Multi-Person Linear Model]

SMPL Model: Deformation

$$T(\beta, \theta) = \bar{T} + \sum_{m=1}^{|\beta|} \beta_m S_n + \sum_{n=1}^{|\theta|} \theta_n p_n$$

$$x = SKIN(T(\beta, \theta), \theta, \mathcal{W})$$

$SKIN$: LBS, DQS, etc…

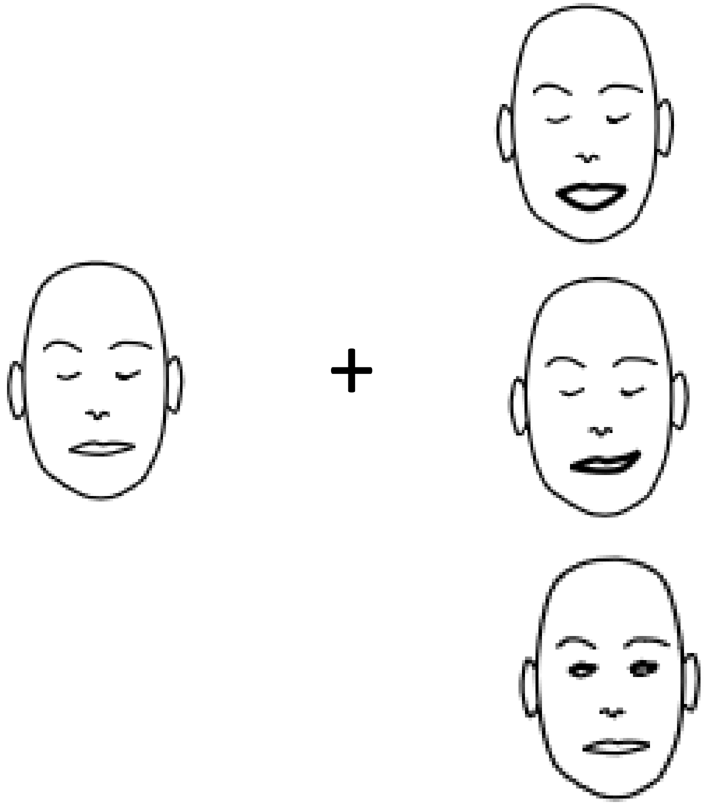
[SMPL: A Skinned Multi-Person Linear Model]

Example: Facial Animation

[UnrealEngine]

Facial Animation

Facial Animation = Identity + Expression



Facial Animation

Facial Animation = Identity + Expression

$$X = X_0 + \sum_i \beta_i B_i^{\mathrm{ID}} + \sum_j \theta_j B_j^{\mathrm{Exp}}$$

Facial Animation

Facial Animation = Identity + Expression

The "Average Face"

Facial Expression

$$X = X_0 + \sum_i \beta_i B_i^{\mathrm{ID}} + \sum_j \theta_j B_j^{\mathrm{Exp}}$$

Face Customization

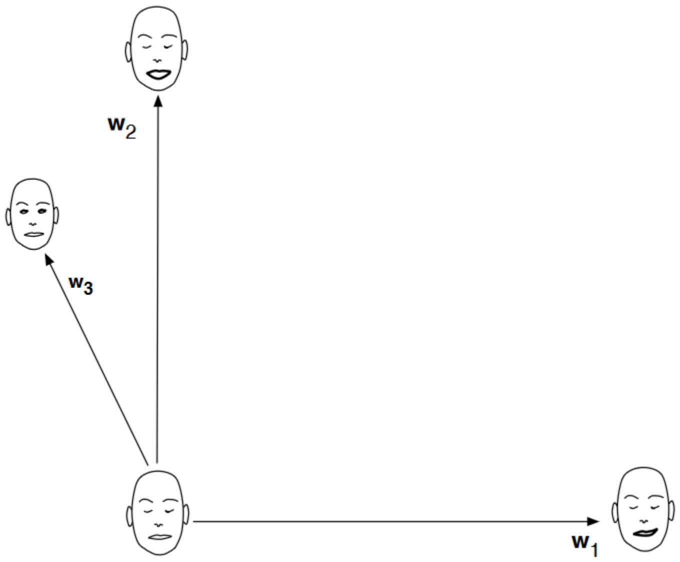
Facial Blendshapes

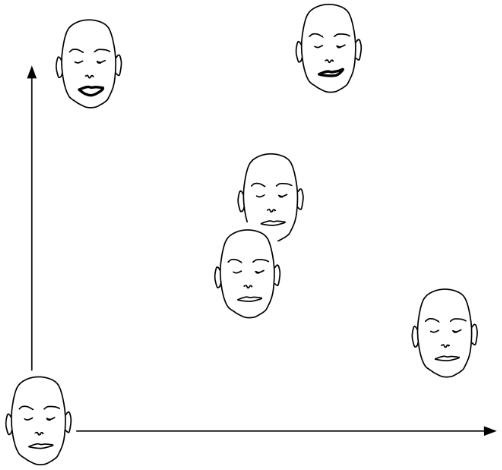
http://www.santhoshkoneru.com/facial-blendshapes

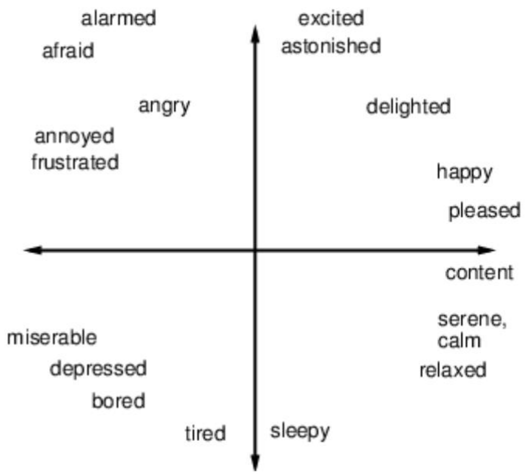
A Typical Set of Blendshapes (ARKit)

- eyeBlinkLeft
- eyeLookDownLeft
- eyeLookInLeft
- eyeLookOutLeft
- eyeLookUpLeft
- eyeSquintLeft
- eyeWideLeft
- eyeBlinkRight
- eyeLookDownRight
- eyeLookInRight
- eyeLookOutRight
- eyeLookUpRight
- eyeSquintRight
- eyeWideRight
- jawForward
- jawLeft
- jawRight
- jawOpen
- mouthClose
- mouthFunnel
- mouthPucker
- mouthRight
- mouthLeft
- mouthSmileLeft
- mouthSmileRight
- mouthFrownRight
- mouthFrownLeft
- mouthDimpleLeft
- mouthDimpleRight
- mouthStretchLeft
- mouthStretchRight
- mouthRollLower
- mouthRollUpper
- mouthShrugLower
- mouthShrugUpper
- mouthPressLeft
- mouthPressRight
- mouthLowerDownLeft
- mouthLowerDownRight
- mouthUpperUpLeft
- mouthUpperUpRight
- browDownLeft
- browDownRight
- browInnerUp
- browOuterUpLeft
- browOuterUpRight
- cheekPuff
- cheekSquintLeft
- cheekSquintRight
- noseSneerLeft
- noseSneerRight
- tongueOut
- gemfield

Blendshapes vs. Example-based Skinning

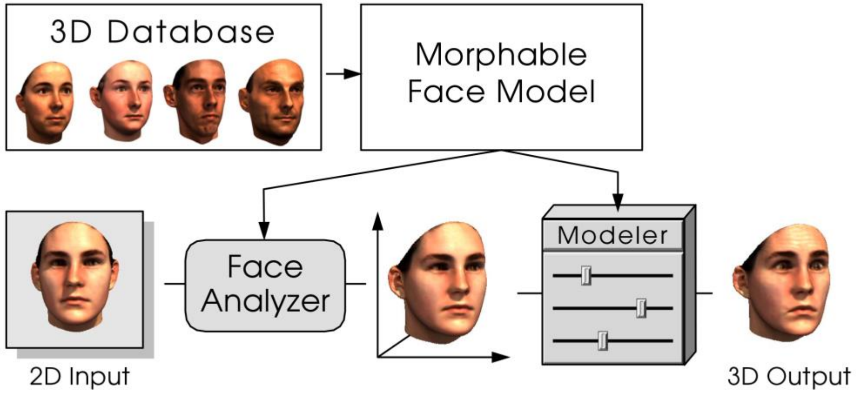


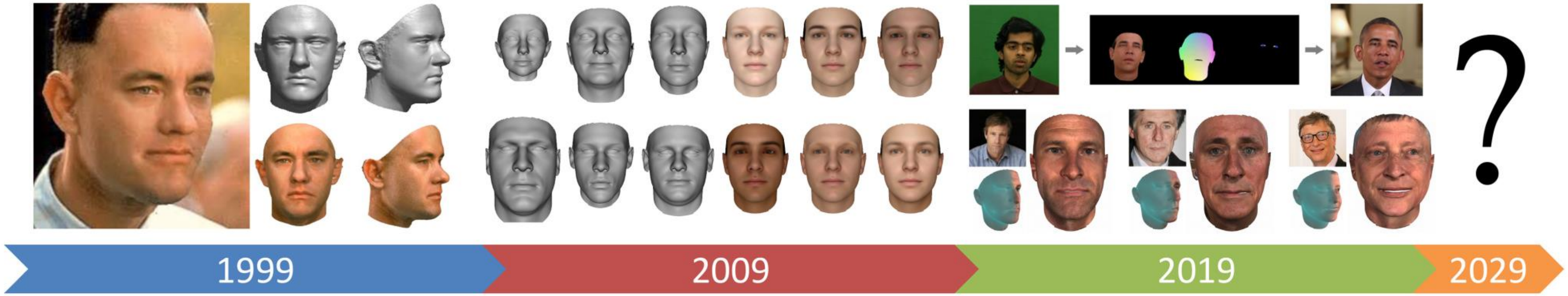




Morphable Face Models

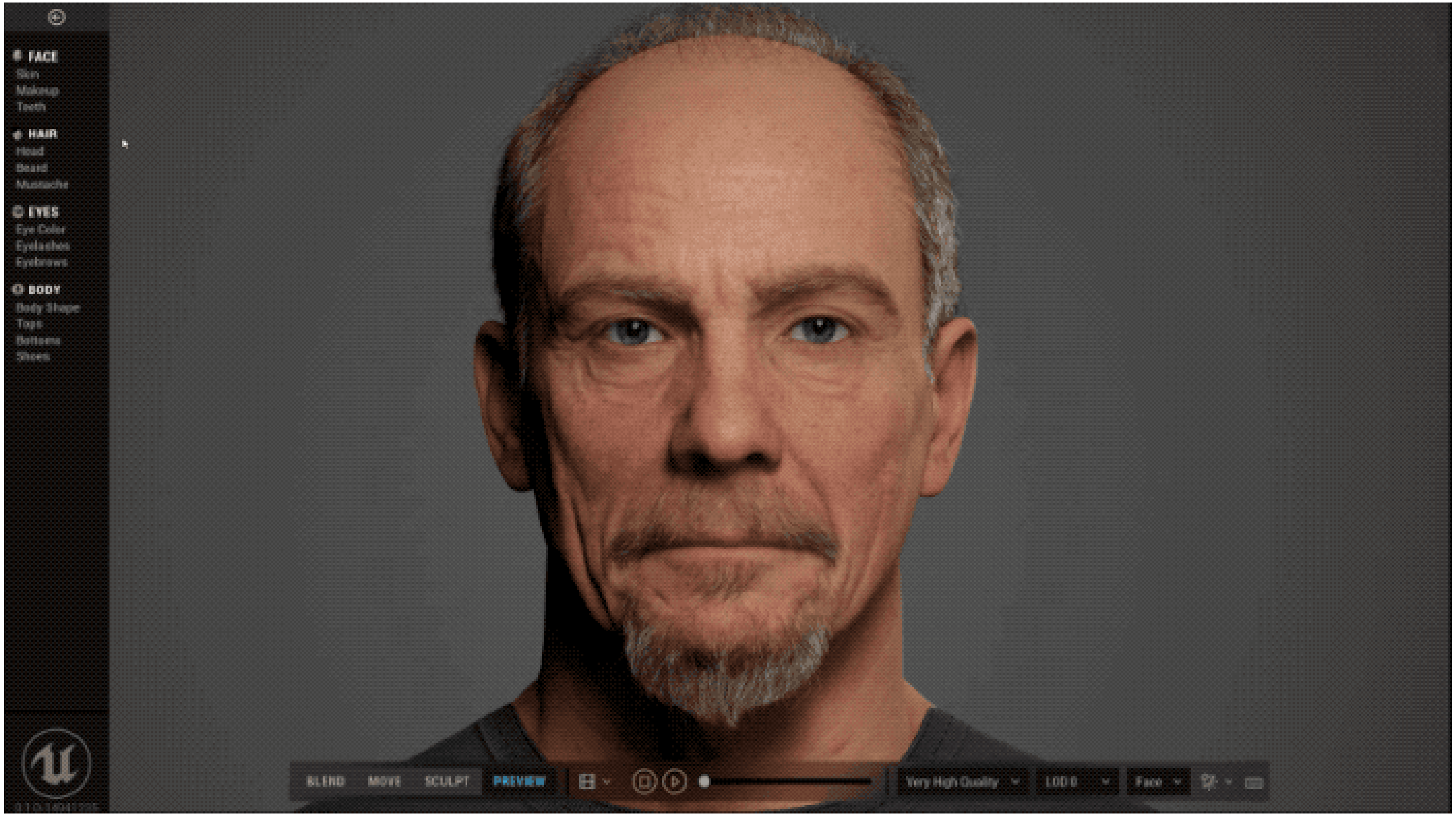
$$X = X_0 + \sum_i \beta_i B_i^{\mathrm{ID}} + \sum_j \theta_j B_j^{\mathrm{Exp}}$$





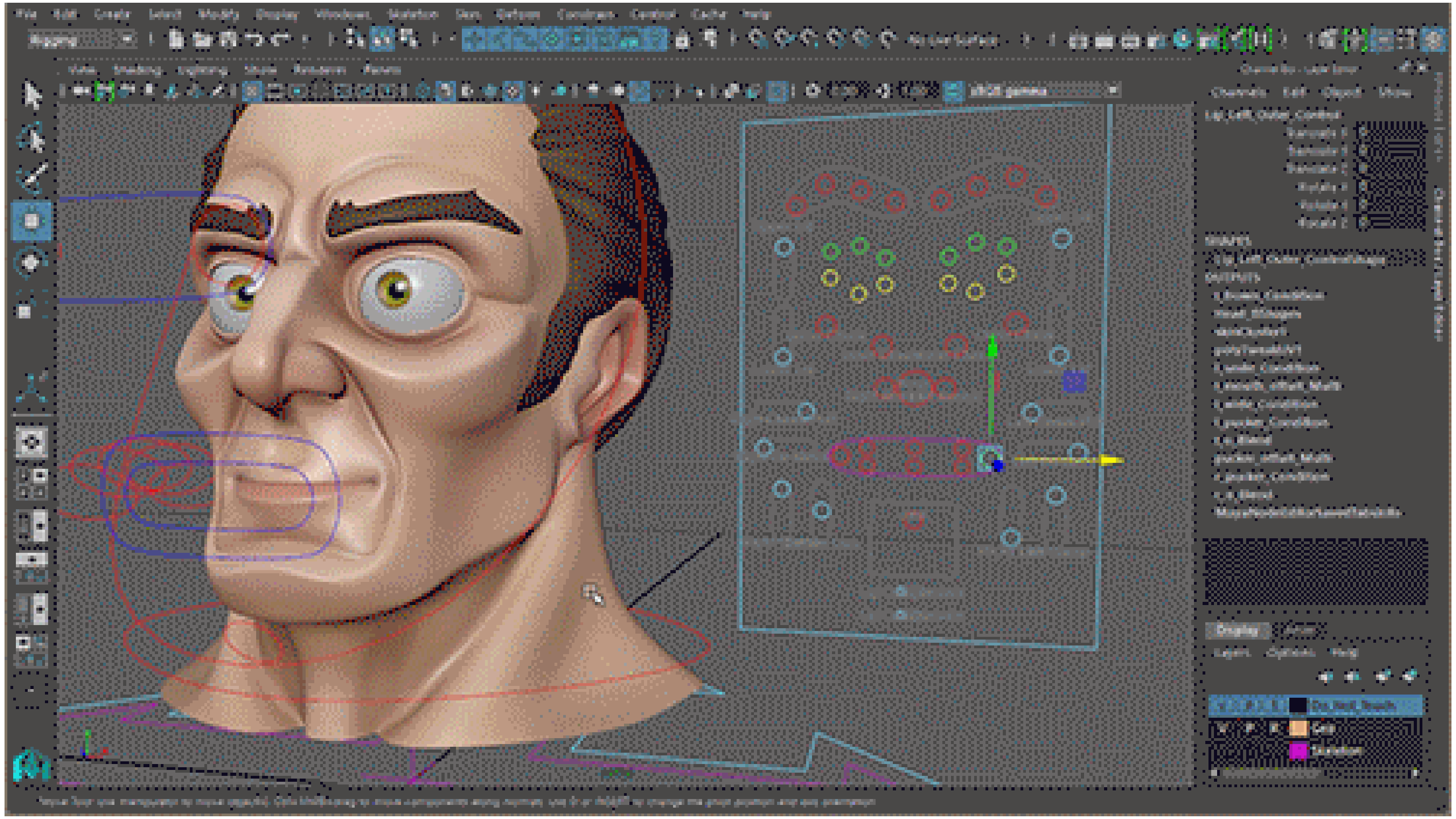
Egger et al. 2020. ***3D Morphable Face Models - Past, Present, and Future.*** *ACM Trans. Graph.* 39, 5 (June 2020), 157:1-157:38.

Morphable Face Models



Meta Human - UnrealEngine

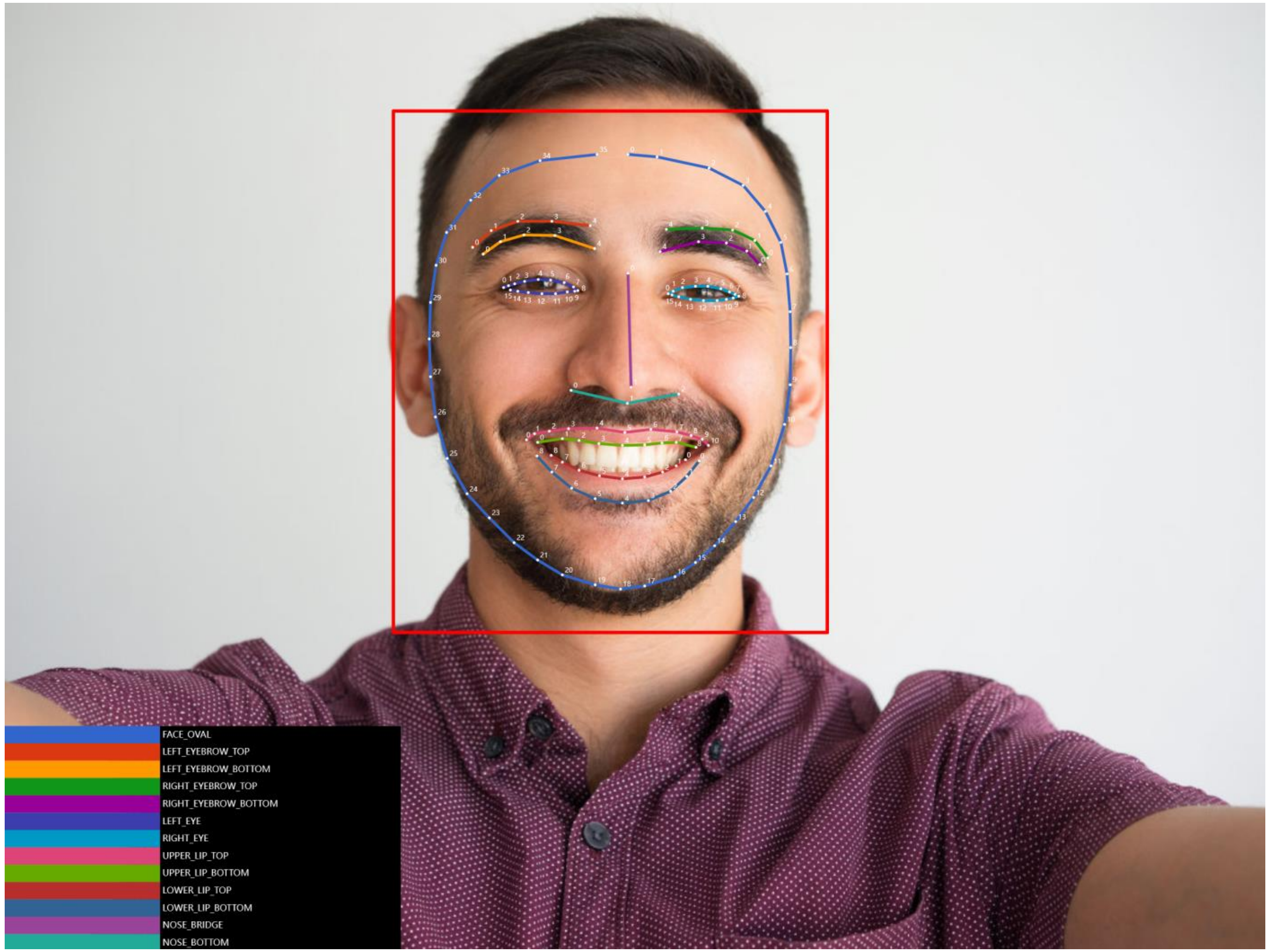
How to Animate a Face?



Face Tracking

https://developers.google.com/ml-kit/vision/face-detection

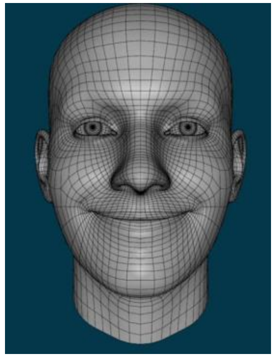
Face Tracking

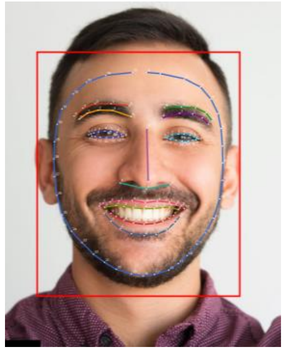


https://developers.google.com/ml-kit/vision/face-detection

Face Tracking

$$\min_{\beta_i,\theta_j} \left\| X_0 + \sum_i \beta_i B_i^{\mathrm{ID}} + \sum_j \theta_j B_j^{\mathrm{Exp}} - Y \right\| + E(\beta_i, \theta_j)$$



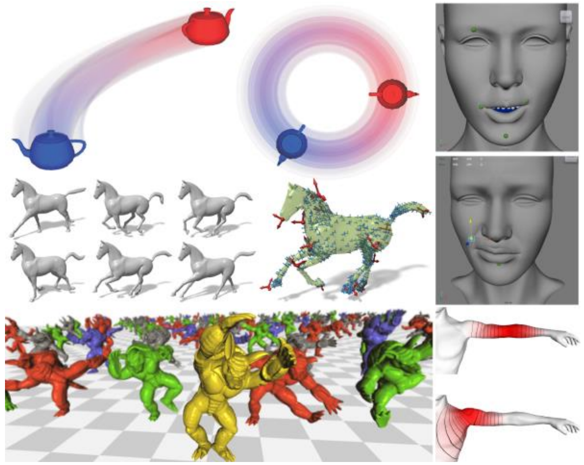


Speech-driven

Outline

- Skinning
  - Linear Blend Skinning (LBS)
  - Dual Quaternion Skinning (DQS)
  - Blendshapes

- Examples:
  - The SMPL model
  - Facial Animation



Many images are from: https://skinning.org/
Alec Jacobson, Zhigang Deng, Ladislav Kavan, and J. P. Lewis. 2014.
***Skinning: real-time shape deformation**.*
In ACM SIGGRAPH 2014 Courses (SIGGRAPH '14)

Questions?