

# Introduction to Computer Graphics

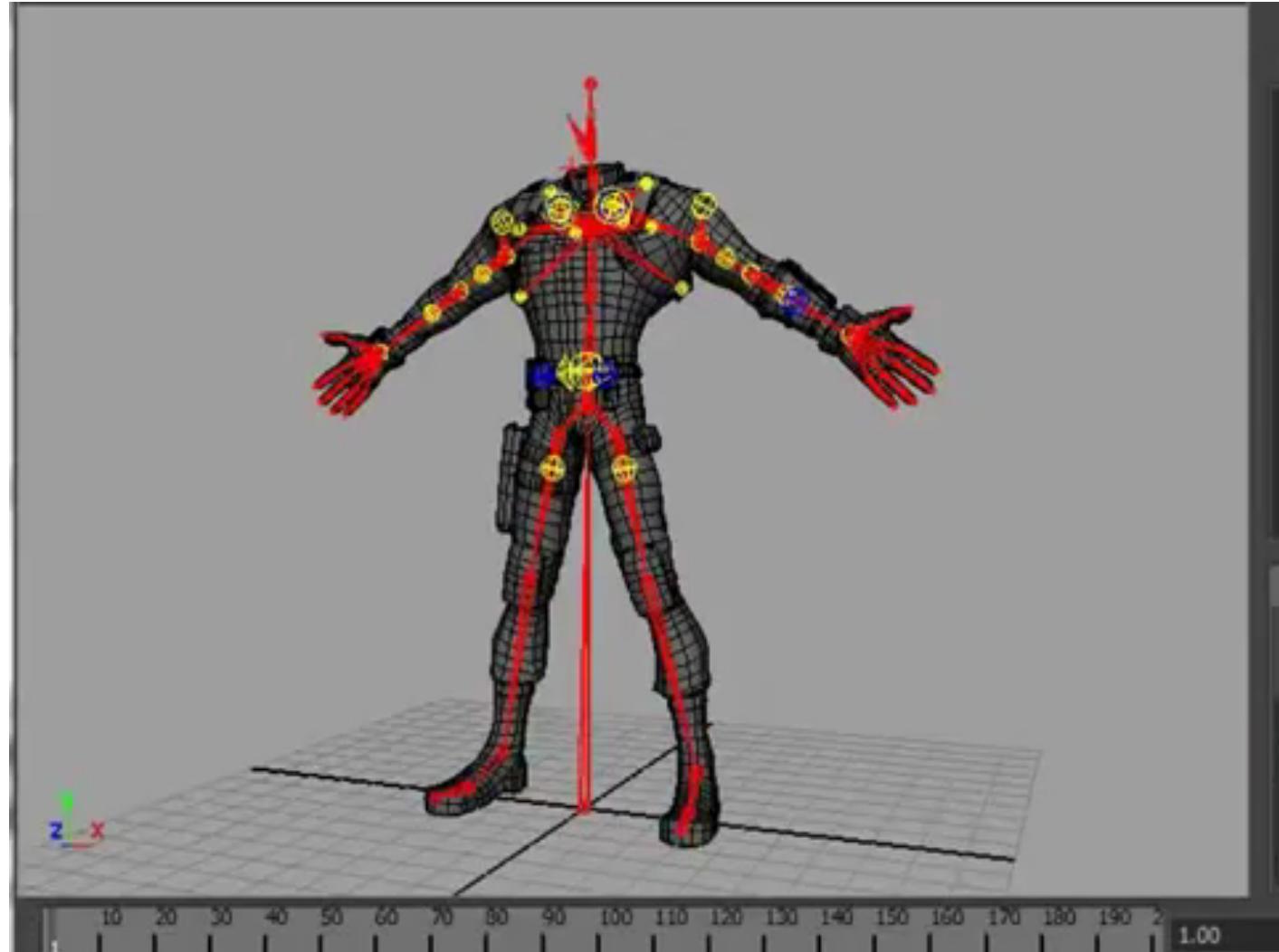
## – Animation (1) –

July 2, 2020

Kenshi Takayama

# Skeleton-based animation

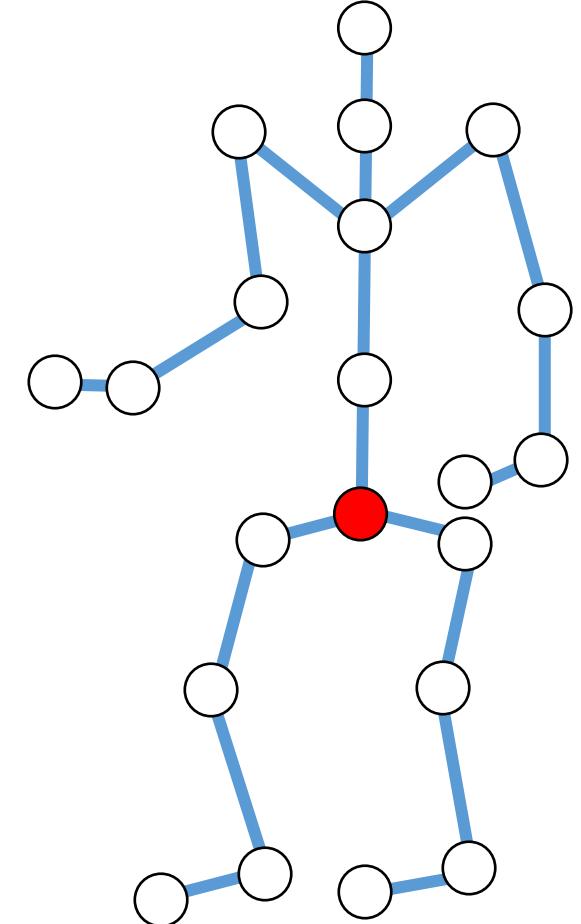
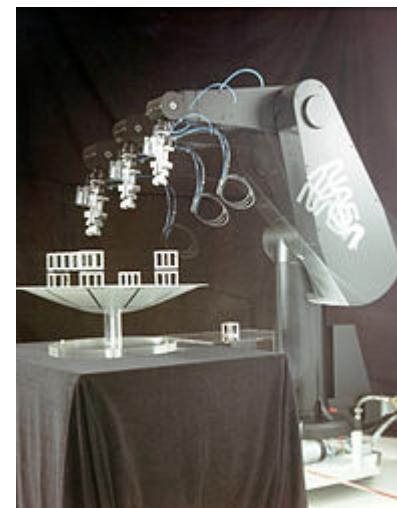
- Simple
- Intuitive
- Low comp. cost



<https://www.youtube.com/watch?v=DsoNab58QVA>

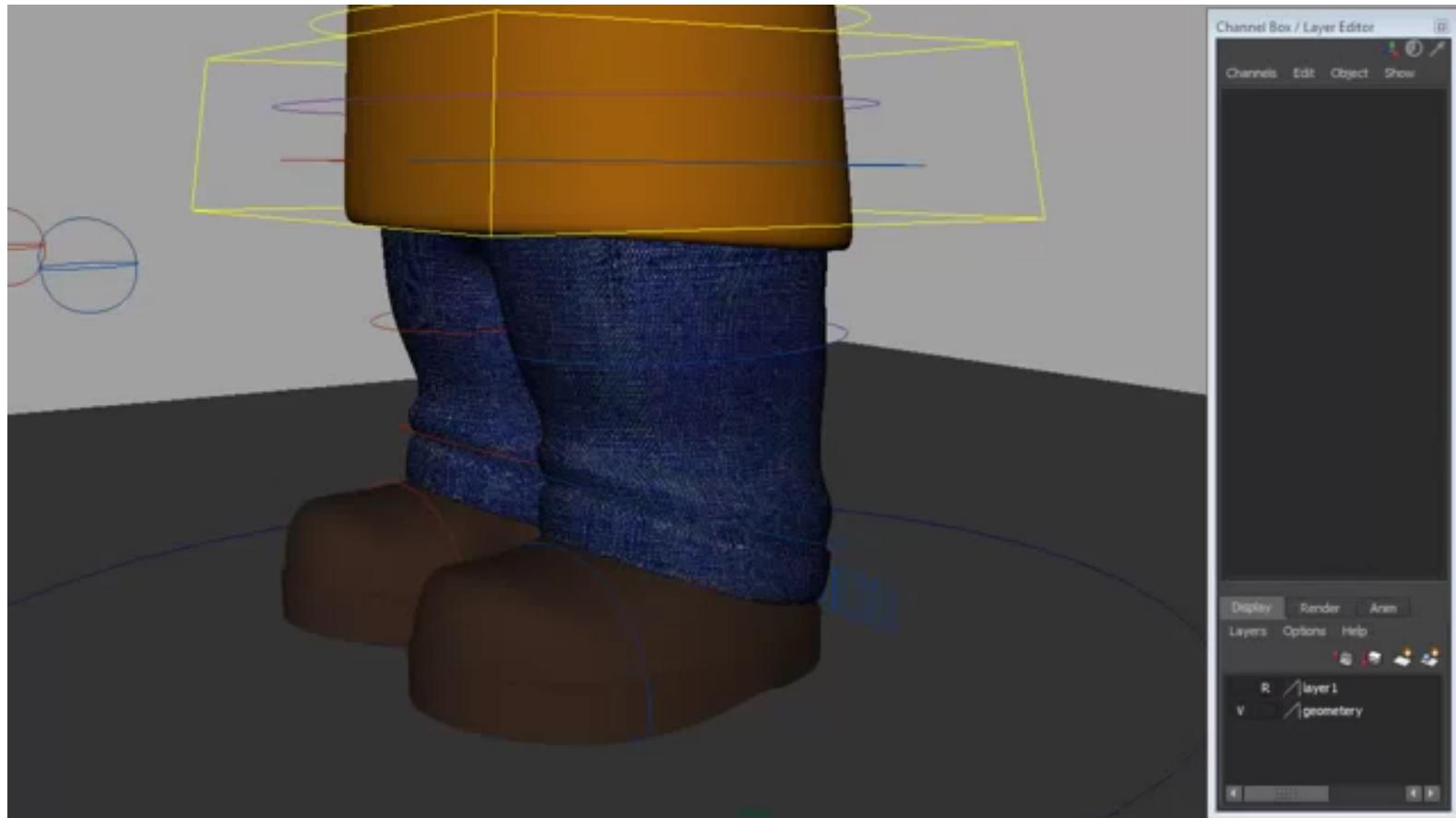
# Representing a pose using skeleton

- Tree structure consisting of bones & joints
- Each bone holds relative rotation angle w.r.t. parent joint
- Whole body pose determined by the set of joint angles  
**(Forward Kinematics)**
- Deeply related to robotics



# Inverse Kinematics

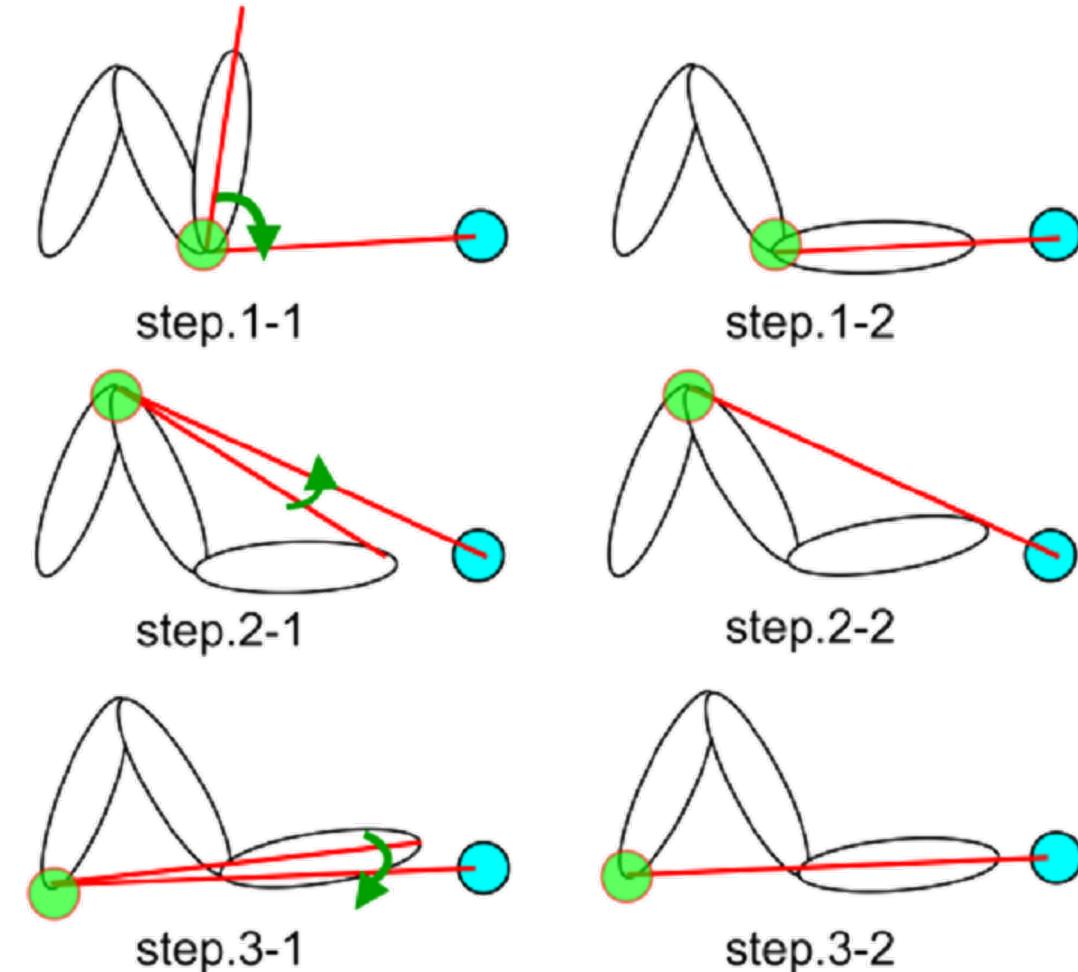
- Find joint angles s.t. an end effector comes at a given goal position
- Typical workflow:
  - Quickly create pose using IK, fine adjustment using FK



[https://www.youtube.com/watch?v=e1qnZ9rV\\_kw](https://www.youtube.com/watch?v=e1qnZ9rV_kw)

# Simple method to solve IK: Cyclic Coordinate Descent

- Change joint angles one by one
  - S.t. the end effector comes as close as possible to the goal position
  - Ordering is important! Leaf → root
- Easy to implement → Basic assignment
- More advanced
  - Jacobi method (directional constraint)
  - Minimizing elastic energy [Jacobson 12]



# IK minimizing elastic energy

## Fast Automatic Skinning Transformations

Alec Jacobson<sup>1</sup>

Ilya Baran<sup>2</sup>

Ladislav Kavan<sup>1</sup>

Jovan Popović<sup>3</sup>

Olga Sorkine<sup>1</sup>

<sup>1</sup>ETH Zurich

<sup>2</sup>Disney Research, Zurich

<sup>3</sup>Adobe Systems, Inc.

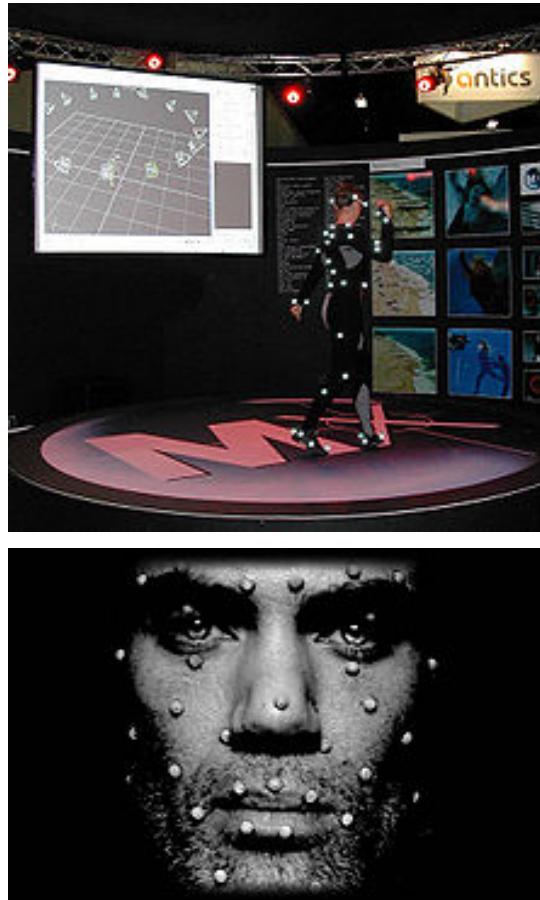
This video contains narration.

<https://www.youtube.com/watch?v=PRcXy2LjI9I>

# Ways to obtain/measure motion data

# Optical motion capture

- Put markers on the actor, record video from many viewpoints (~48)

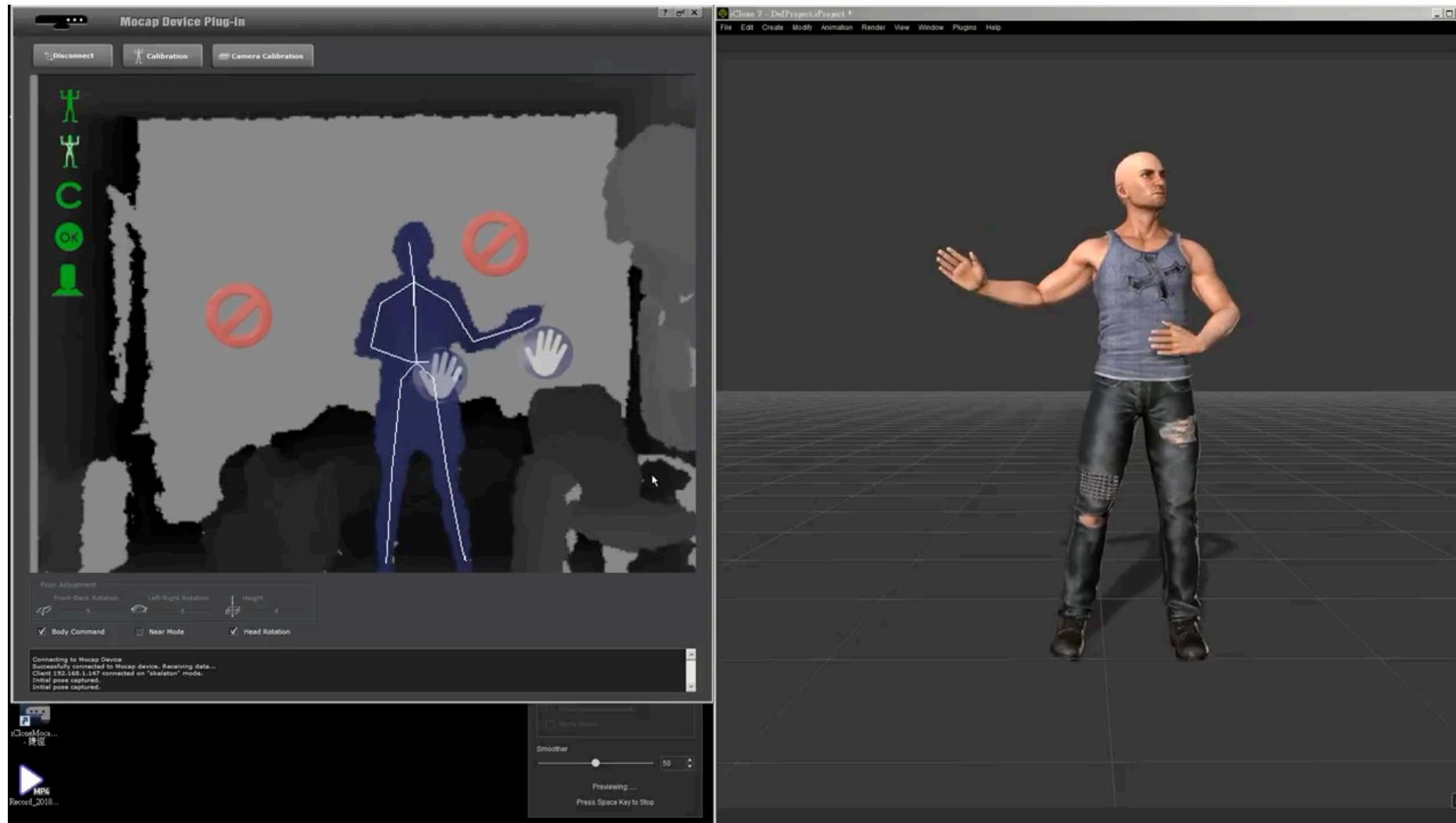


from Wikipedia



<https://www.youtube.com/watch?v=c6X64LhcUyQ>

# Mocap using inexpensive depth camera



<https://www.youtube.com/watch?v=zXDuyMtzunA>

# Mocap designed for outdoor scene

Motion Capture from  
Body-Mounted Cameras



(with audio)

Takaaki Shiratori , Hyun Soo Park , Leonid Sigal ,  
Yaser Sheikh , Jessica K. Hodgins \*

\* Disney Research, Pittsburgh + Carnegie Mellon University

<https://www.youtube.com/watch?v=xbl-NWMfGPs>

# Mocap by drone tracking

Motion planning by  
**FORCES**<sup>PRO</sup>

**Parrot**  
BEBOP DRONE

## Flycon: Environment-independent Human Pose Estimation with Aerial Vehicles

Tobias Nägeli, Samuel Oberholzer, Silvan Plüss, Javier Alonso-Mora, Otmar Hilliges

ACM SIGGRAPH Asia'18



Advanced Interactive  
Technologies

**ETH**

Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

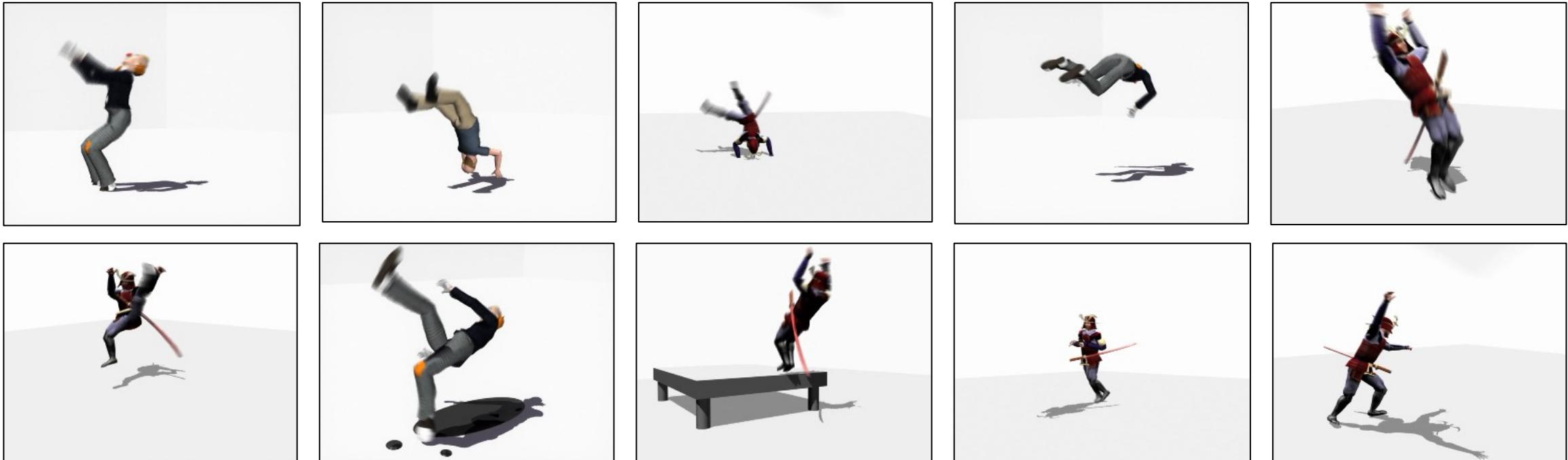
**TU**Delft

Delft  
University of  
Technology

<https://www.youtube.com/watch?v=iSJY-vHDmHQ>

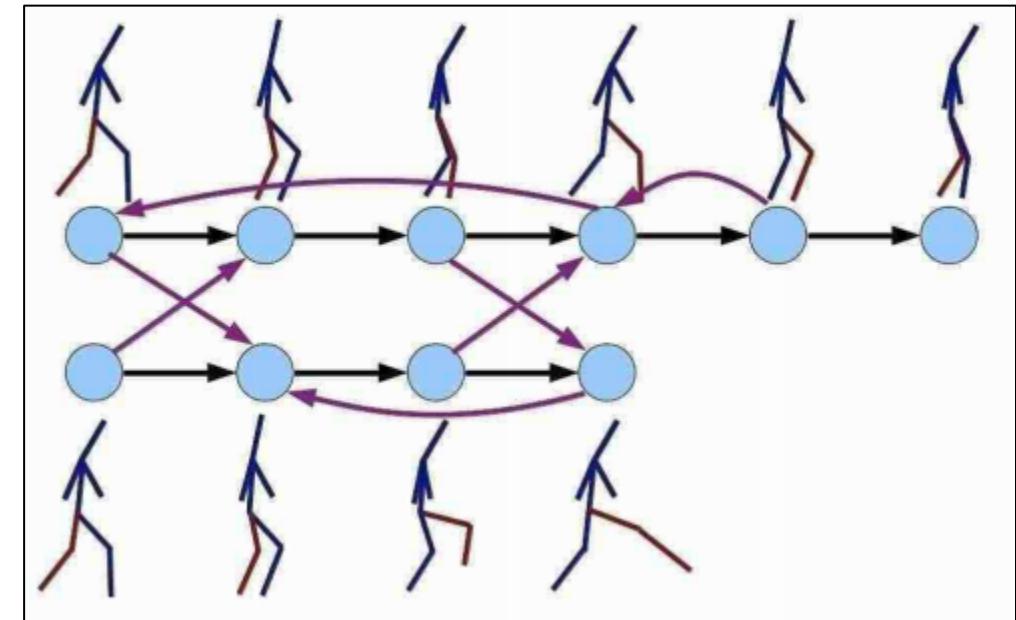
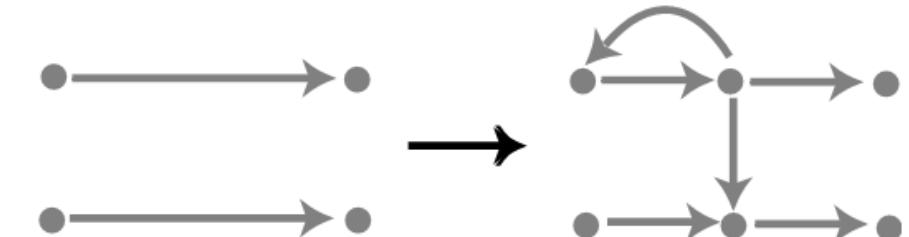
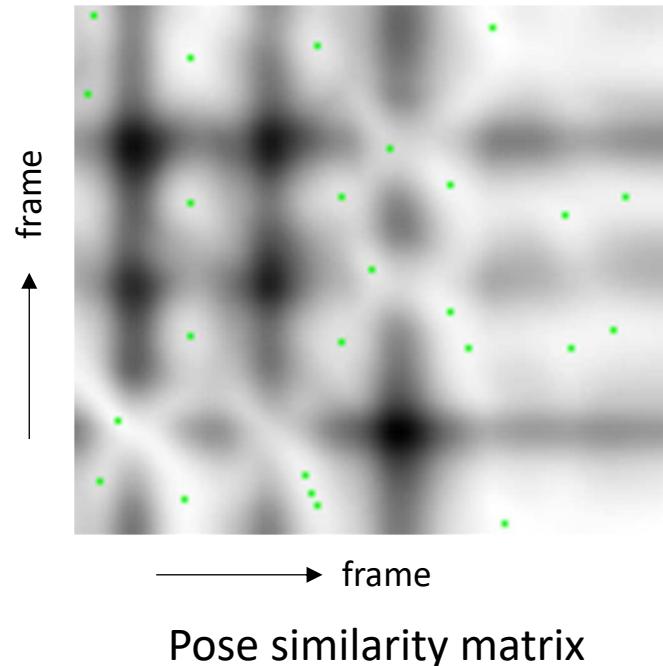
# Motion database

- <http://mocap.cs.cmu.edu/>
- 6 categories, 2605 in total
- Free for research purposes
  - Interpolation, recombination, analysis, search, etc.



# Recombining motions

- Allow transition from one motion to another if poses are similar in certain frame



Motion Graphs [Kovar SIGGRAPH02]

Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data [Lee SIGGRAPH06]

[http://www.tcs.tifr.res.in/~workshop/thapar\\_igga/motiongraphs.pdf](http://www.tcs.tifr.res.in/~workshop/thapar_igga/motiongraphs.pdf)

# Generating motion through simulation

- For creatures unsuitable for mocap
  - Too dangerous, nonexistent, ...
- Natural motion respecting body shape
- Can interact with dynamic environment

## Flexible Muscle-Based Locomotion for Bipedal Creatures

SIGGRAPH ASIA 2013

Thomas Geijtenbeek  
Michiel van de Panne  
Frank van der Stappen

<https://www.youtube.com/watch?v=pgaEE27nsQw>

# Creating poses using special devices

## Tangible and Modular Input Device for Character Articulation

Alec Jacobson<sup>1</sup>

Daniele Panozzo<sup>1</sup>

Oliver Glauser<sup>1</sup>

Cédric Pradalier<sup>2</sup>

Otmar Hilliges<sup>1</sup>

Olga Sorkine-Hornung<sup>1</sup>

<sup>1</sup>ETH Zurich

<sup>2</sup>Georgia Tech Lorraine



*This video contains narration*

# Many topics about character motion

**Keyframe animation  
by  
topology coordinates**

Interaction between  
multiple persons

[https://www.youtube.com/watch?v=1S\\_6wSKI\\_nU](https://www.youtube.com/watch?v=1S_6wSKI_nU)

Synthesis of Detailed Hand Manipulations  
Using Contact Sampling

Yuting Ye C. Karen Liu  
Georgia Institute of Technology

Aggregate Dynamics  
for Dense Crowd Simulation

Submission 0042

Crowd simulation

<https://www.youtube.com/watch?v=pqBSNAOsMDc>

Cost Functions

Grasping motion

<https://www.youtube.com/watch?v=x8c27XYTLTo>

Path planning

<https://vimeo.com/33409868>

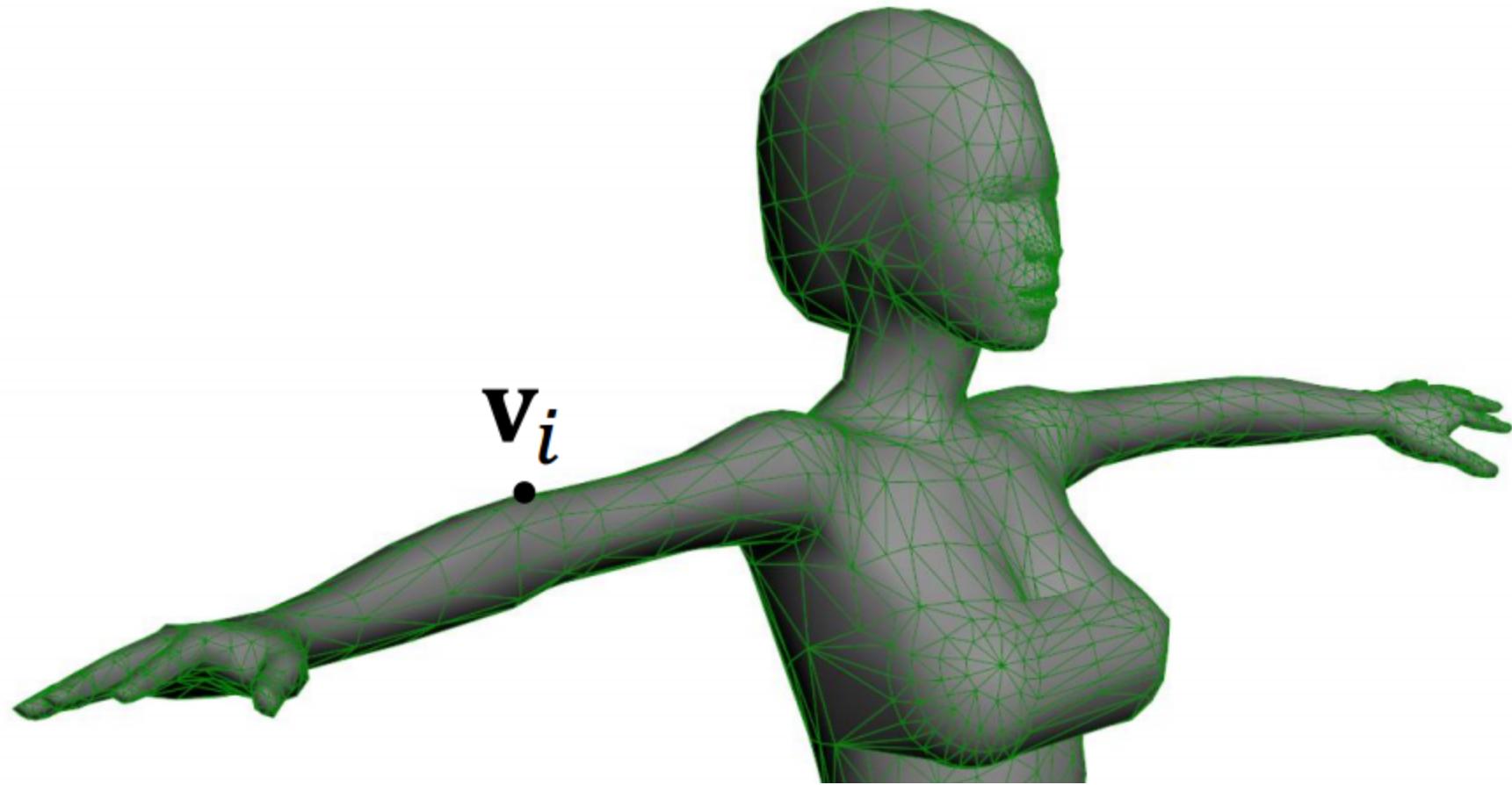
Character motion synthesis by topology coordinates [Ho EG09]

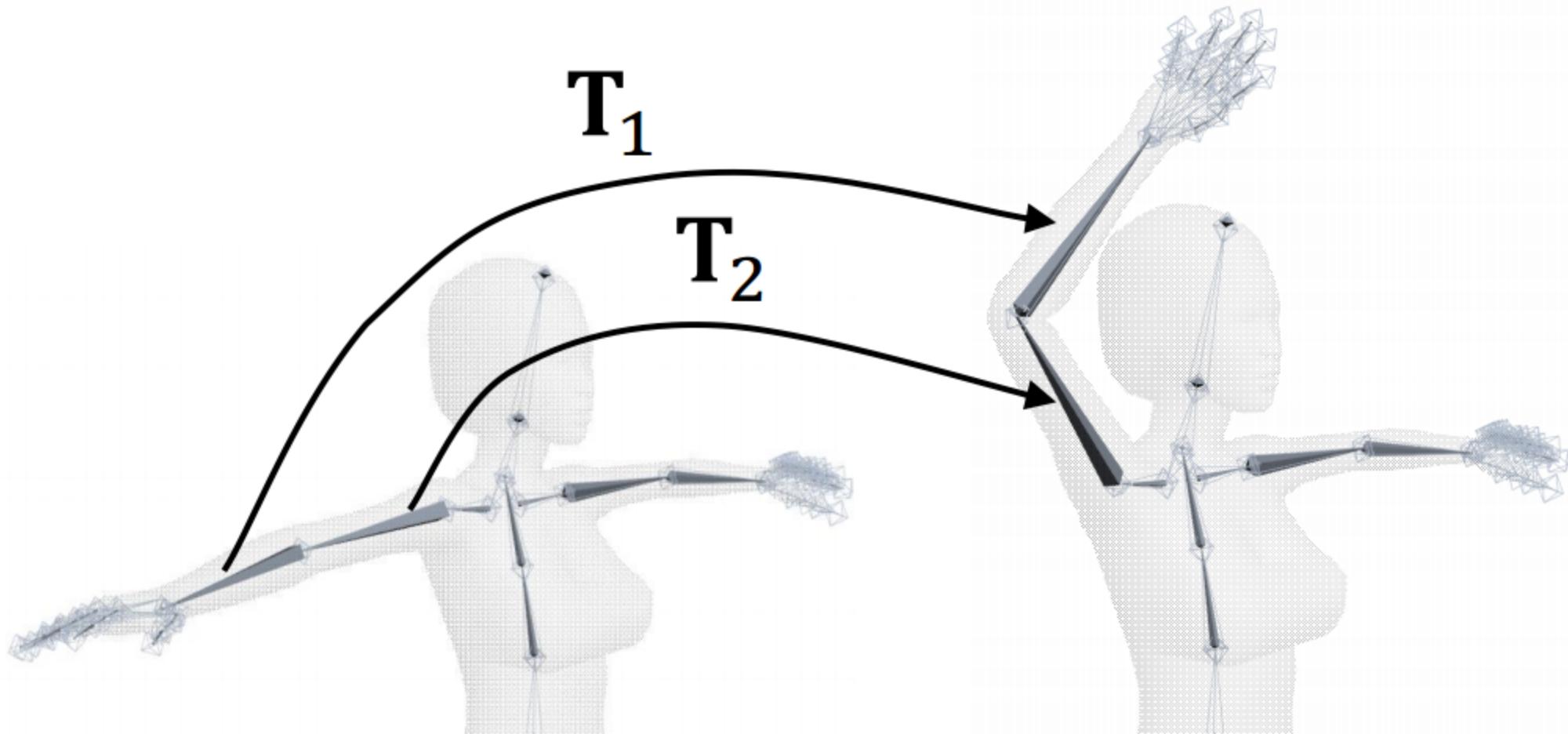
Aggregate Dynamics for Dense Crowd Simulation [Narain SIGGRAPHAsia09]

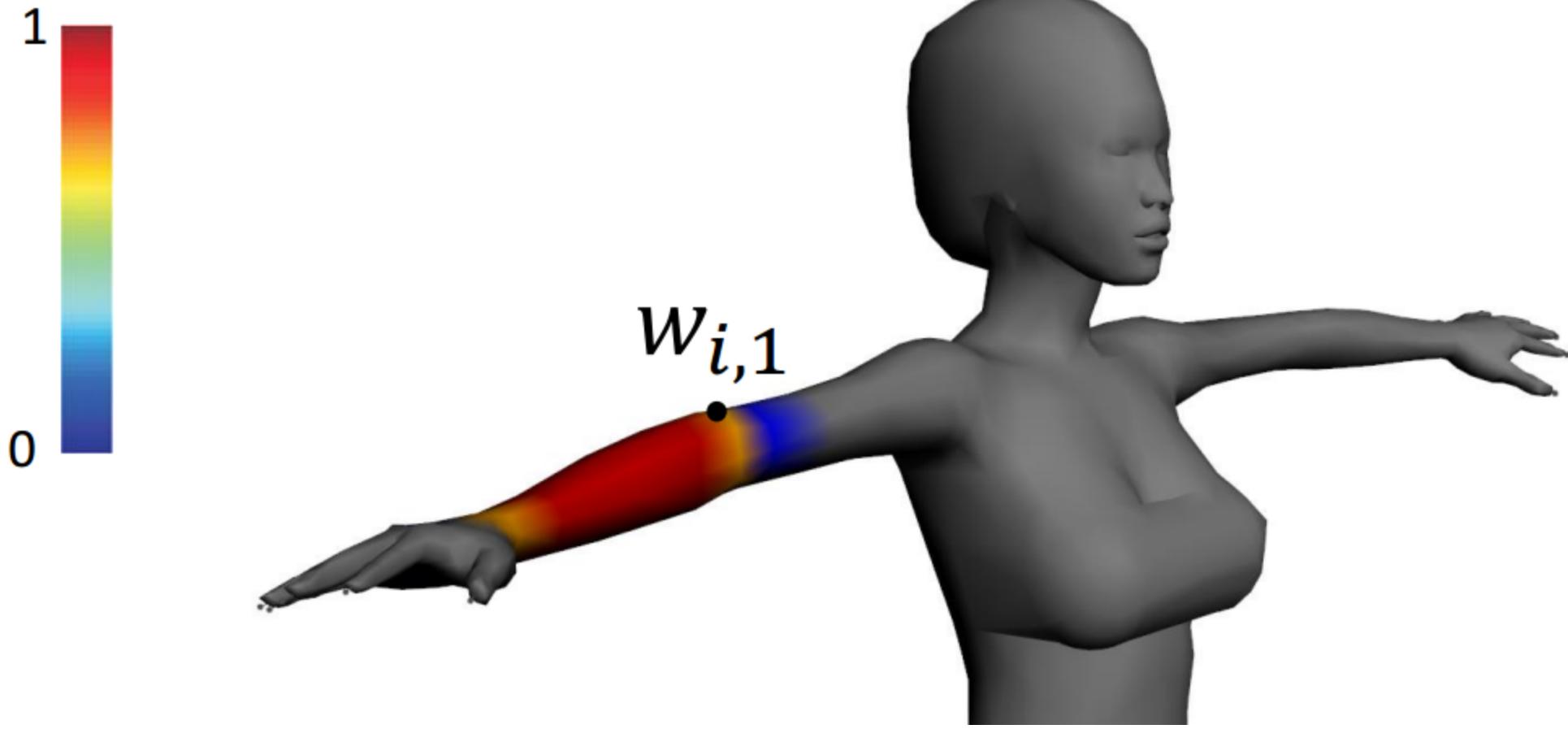
Synthesis of Detailed Hand Manipulations Using Contact Sampling [Ye SIGGRAPH12]

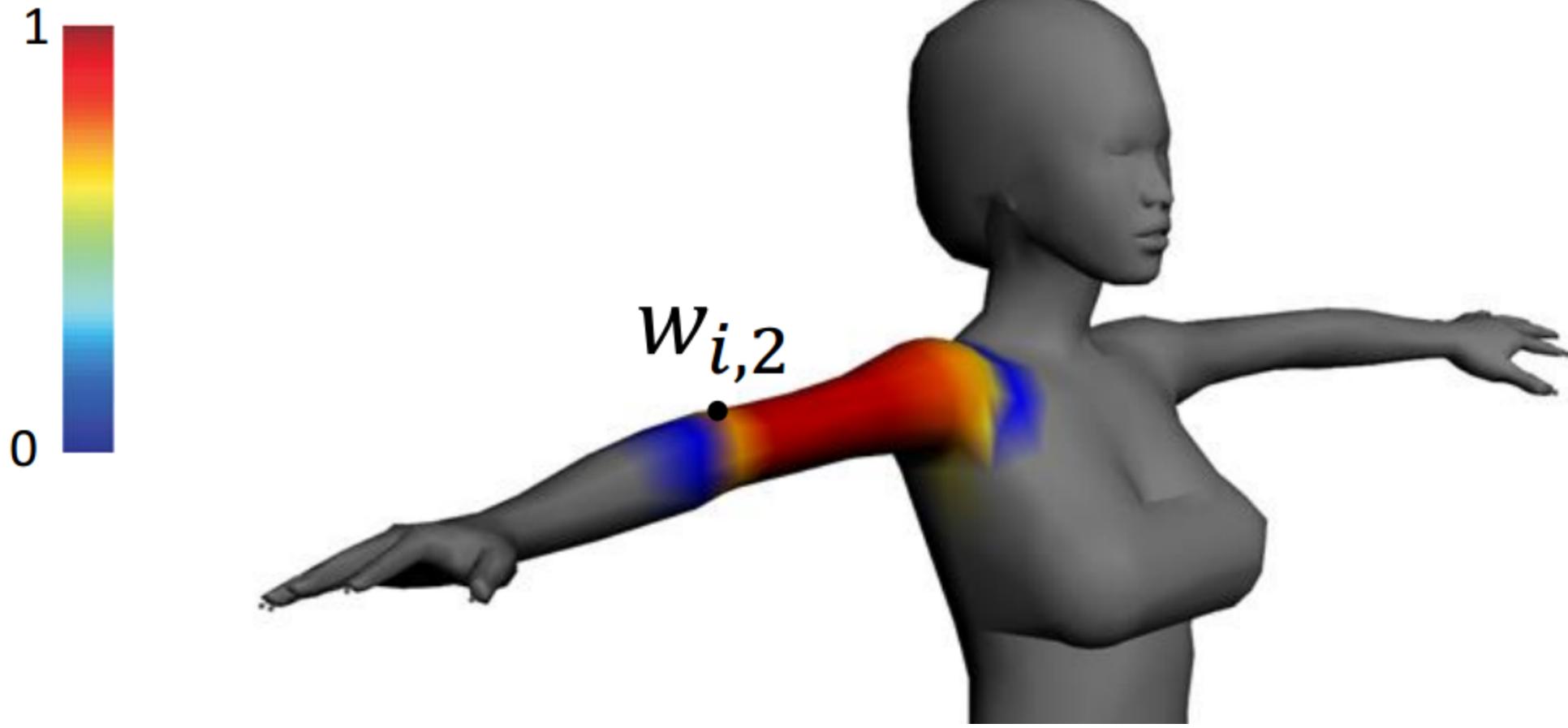
Space-Time Planning with Parameterized Locomotion Controllers.[Levine TOG11]

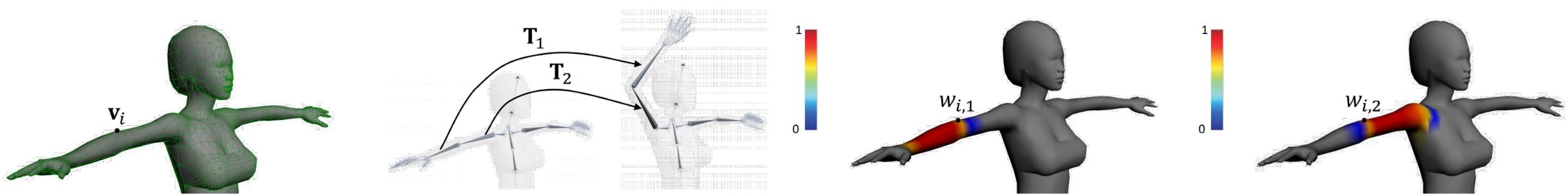
# Skinning











$$\mathbf{v}'_i = \text{blend}(\langle w_{i,1}, \mathbf{T}_1 \rangle, \langle w_{i,2}, \mathbf{T}_2 \rangle, \dots)(\mathbf{v}_i)$$

- Input

- Vertex positions  $\{\mathbf{v}_i\} i = 1, \dots, n$
- Transformation per bone  $\{\mathbf{T}_j\} j = 1, \dots, m$
- Weight from each bone to each vertex  $\{w_{i,j}\} i = 1, \dots, n \ j = 1, \dots, m$

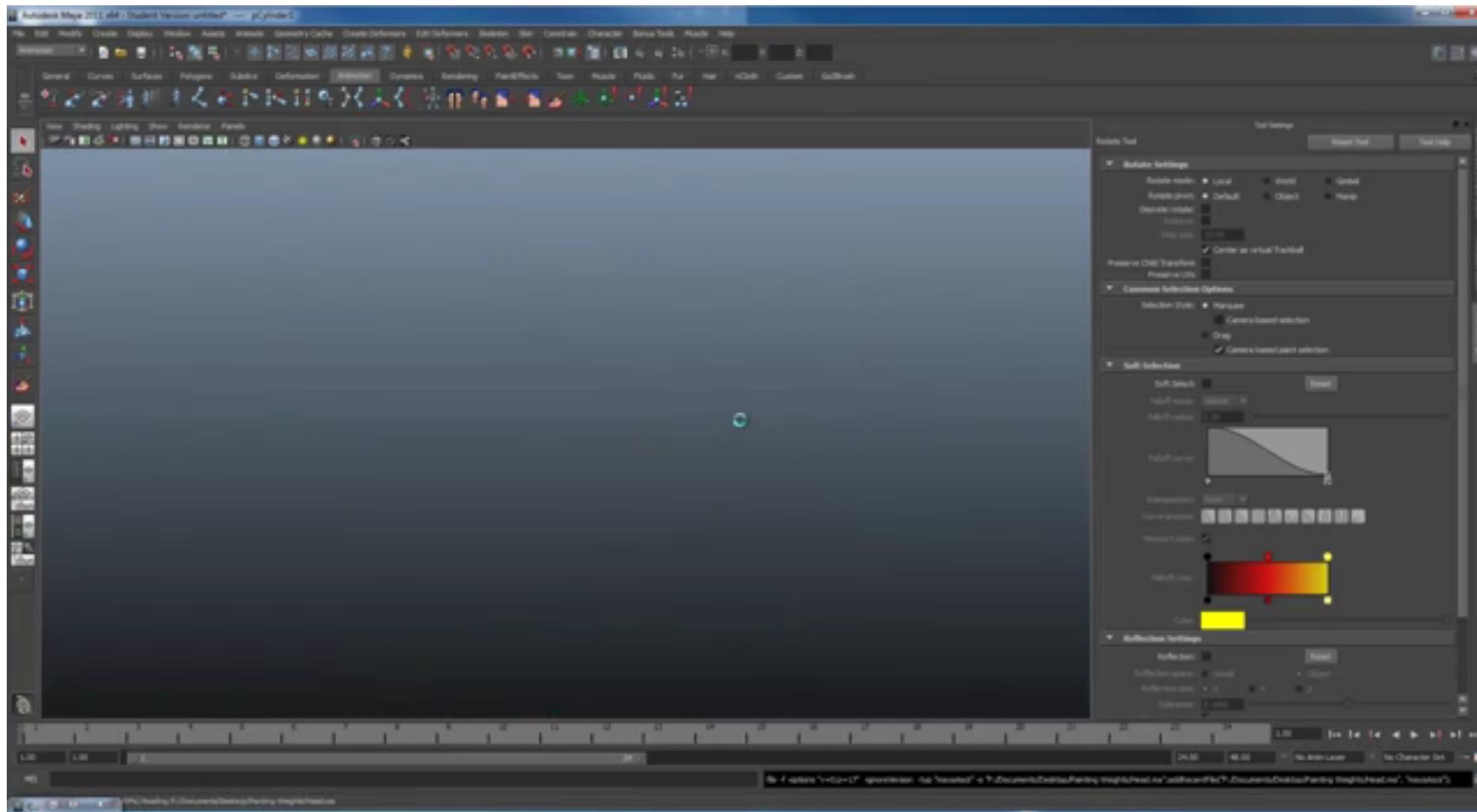
- Output

- Vertex positions after deformation  $\{\mathbf{v}'_i\} i = 1, \dots, n$

- Main focus

- How to define weights  $\{w_{i,j}\}$
- How to blend transformations

# Simple way to define weights: painting



<https://www.youtube.com/watch?v=TACB6bX8SN0>

# Better UI for manual weight editing

## Spline Interface for Intuitive Skinning Weight Editing

Seungbae Bang and Sung-Hee Lee

Korea Advanced Institute of Science and Technology (KAIST)

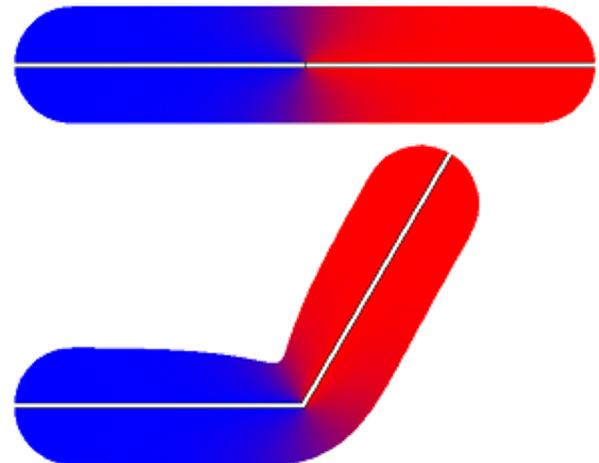
ACM Transactions on Graphics (TOG), 37(5):174, 2018



<https://www.youtube.com/watch?v=mfEP8BIXTgQ>

# Automatic weight computation

- Define weight  $w_j$  as a smooth scalar field that takes 1 on the j-th bone and 0 on the other bones
- Minimize 1<sup>st</sup>-order derivative  $\int_{\Omega} \|\nabla w_j\|^2 dA$  [Baran 07]
  - Approximate solution only on surface → easy & fast
- Minimize 2<sup>nd</sup>-order derivative  $\int_{\Omega} (\Delta w_j)^2 dA$  [Jacobson 11]
  - Introduce inequality constraints  $0 \leq w_j \leq 1$
  - Quadratic Programming over the volume → high-quality



Teddy/Pinocchio demo

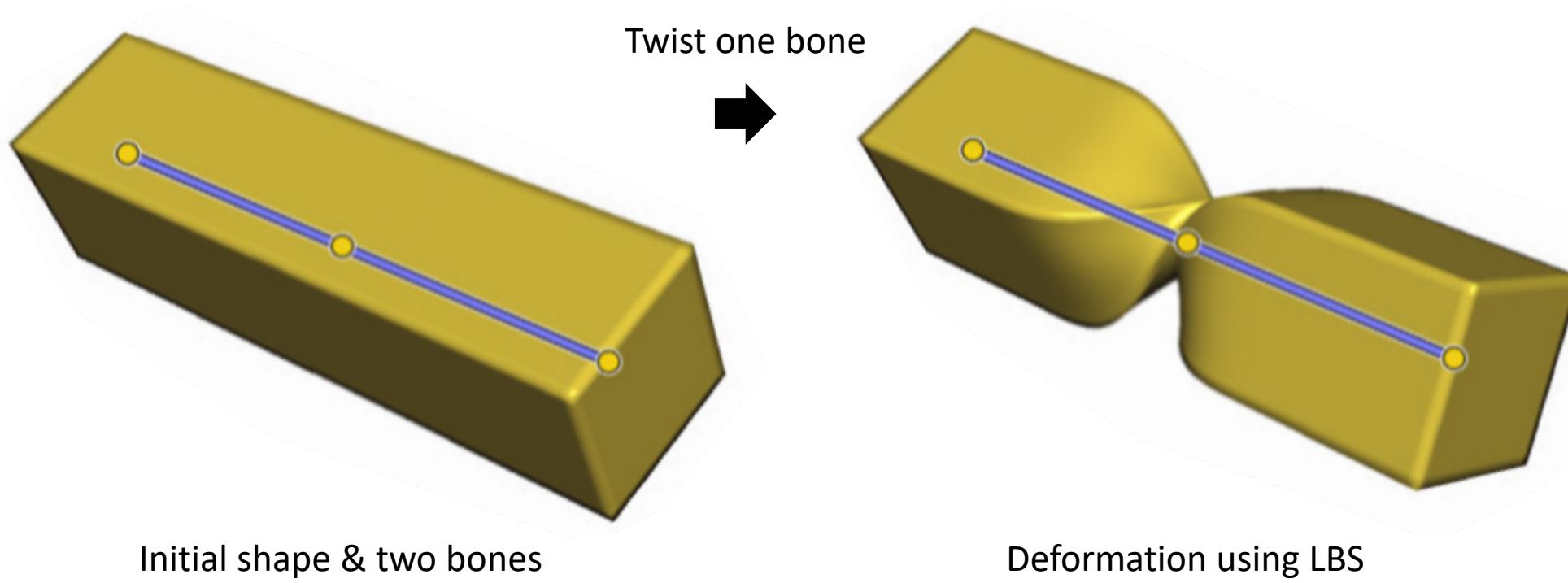
# Simple way to blend transformations: Linear Blend Skinning

- Represent rigid transformation  $\mathbf{T}_j$  as a  $3 \times 4$  matrix consisting of rotation matrix  $\mathbf{R}_j \in \mathbb{R}^{3 \times 3}$  and translation vector  $\mathbf{t}_j \in \mathbb{R}^3$

$$\mathbf{v}'_i = \left( \sum_j w_{i,j} (\mathbf{R}_j \quad \mathbf{t}_j) \right) \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$

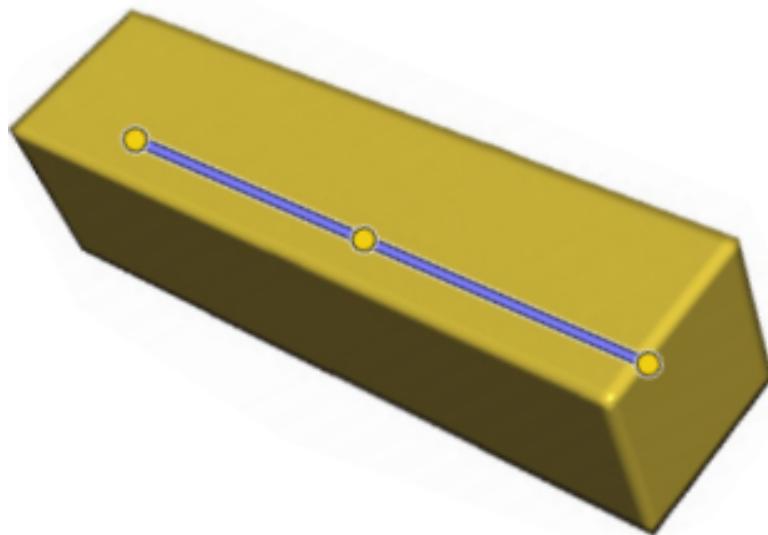
- Simple and fast
  - Implemented using vertex shader: send  $\{\mathbf{v}_i\}$  &  $\{w_{i,j}\}$  to GPU at initialization, send  $\{\mathbf{T}_j\}$  to GPU at each frame
- Standard method

# Artifact of LBS: “candy wrapper” effect

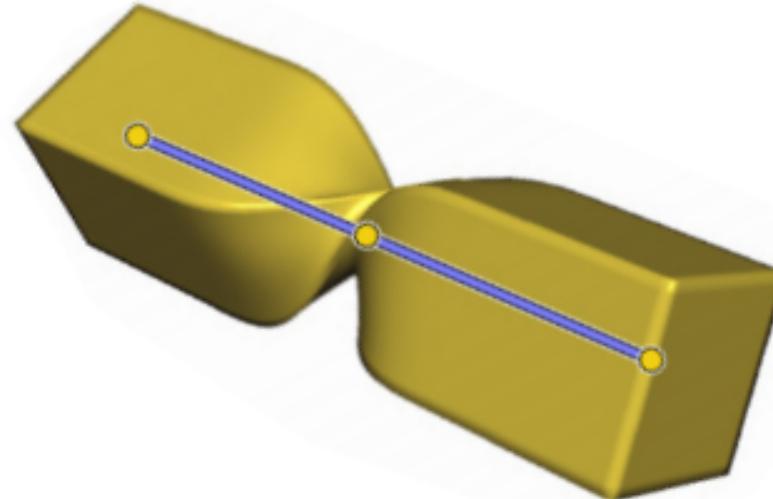


- Linear combination of rigid transformation is not a rigid transformation!
  - Points around joint concentrate when twisted

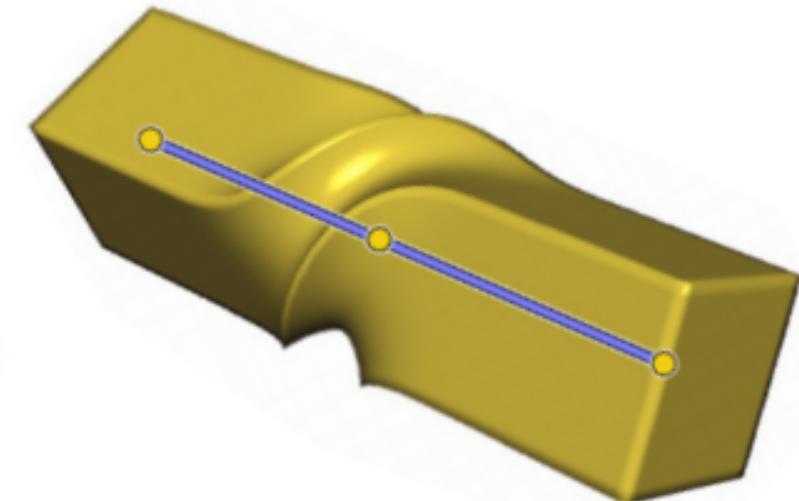
# Alternative to LBS: Dual Quaternion Skinning



Initial shape & two bones



Deformation using LBS



Deformation using DQS

- Idea
  - Quaternion (four numbers) → 3D rotation
  - Dual quaternion (two quaternions) → 3D rigid motion (rotation + translation)

# Dual number & dual quaternion

- Dual number
  - Introduce dual unit  $\varepsilon$  & its arithmetic rule  $\varepsilon^2 = 0$  (cf. imaginary unit  $i$ )
  - Dual number is sum of primal & dual components:
$$\hat{a} := a_0 + \varepsilon a_\varepsilon$$
  - Dual conjugate:
$$\bar{\hat{a}} = \overline{a_0 + \varepsilon a_\varepsilon} = a_0 - \varepsilon a_\varepsilon$$
$$a_0, a_\varepsilon \in \mathbb{R}$$
- Dual quaternion
  - Quaternion whose elements are dual numbers
  - Can be written using two quaternions
$$\hat{\mathbf{q}} := \mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon$$
  - Dual conjugate:
$$\bar{\hat{\mathbf{q}}} = \overline{\mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon} = \mathbf{q}_0 - \varepsilon \mathbf{q}_\varepsilon$$
  - Quaternion conjugate:
$$\hat{\mathbf{q}}^* = (\mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon)^* = \mathbf{q}_0^* + \varepsilon \mathbf{q}_\varepsilon^*$$

# Arithmetic rules for dual number/quaternion

- For dual number  $\hat{a} = a_0 + \varepsilon a_\varepsilon$  :

- Reciprocal

$$\frac{1}{\hat{a}} = \frac{1}{a_0} - \varepsilon \frac{a_\varepsilon}{a_0^2}$$

- Square root

$$\sqrt{\hat{a}} = \sqrt{a_0} + \varepsilon \frac{a_\varepsilon}{2\sqrt{a_0}}$$

- Trigonometric

$$\begin{aligned}\sin \hat{a} &= \sin a_0 + \varepsilon a_\varepsilon \cos a_0 \\ \cos \hat{a} &= \cos a_0 - \varepsilon a_\varepsilon \sin a_0\end{aligned}$$

Easily derived by combining usual arithmetic rules with new rule  $\varepsilon^2 = 0$

From Taylor expansion

- For dual quaternion  $\hat{\mathbf{q}} = \mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon$  :

- Norm

$$\|\hat{\mathbf{q}}\| = \sqrt{\hat{\mathbf{q}}^* \hat{\mathbf{q}}} = \|\mathbf{q}_0\| + \varepsilon \frac{\langle \mathbf{q}_0, \mathbf{q}_\varepsilon \rangle}{\|\mathbf{q}_0\|}$$

Dot product as 4D vectors

- Inverse

$$\hat{\mathbf{q}}^{-1} = \frac{\hat{\mathbf{q}}^*}{\|\hat{\mathbf{q}}\|^2}$$

- Unit dual quaternion satisfies  $\|\hat{\mathbf{q}}\| = 1$

- $\Leftrightarrow \|\mathbf{q}_0\| = 1 \text{ & } \langle \mathbf{q}_0, \mathbf{q}_\varepsilon \rangle = 0$

# Rigid transformation using dual quaternion

- Unit dual quaternion representing rigid motion of translation  $\vec{t} = (t_x, t_y, t_z)$  and rotation  $\mathbf{q}_0$  (unit quaternion) :

$$\hat{\mathbf{q}} = \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{t} \mathbf{q}_0$$

Note: 3D vector is considered as quaternion with zero real part

- Rigid transformation of 3D position  $\vec{v} = (v_x, v_y, v_z)$  using unit dual quaternion  $\hat{\mathbf{q}}$  :

$$\hat{\mathbf{q}}(1 + \varepsilon \vec{v})\overline{\hat{\mathbf{q}}^*} = 1 + \varepsilon \vec{v}'$$

- $\vec{v}'$  : 3D position after transformation

# Rigid transformation using dual quaternion

- $\hat{\mathbf{q}} = \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0$

- $\hat{\mathbf{q}}(1 + \varepsilon \vec{\mathbf{v}}) \overline{\hat{\mathbf{q}}^*} = \left( \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \right) (1 + \varepsilon \vec{\mathbf{v}}) \overline{\left( \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \right)^*}$

$$= \left( \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \right) (1 + \varepsilon \vec{\mathbf{v}}) \overline{\left( \mathbf{q}_0^* + \frac{\varepsilon}{2} (\vec{\mathbf{t}} \mathbf{q}_0)^* \right)}$$

$$= \left( \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \right) (1 + \varepsilon \vec{\mathbf{v}}) \overline{\left( \mathbf{q}_0^* - \frac{\varepsilon}{2} \mathbf{q}_0^* \vec{\mathbf{t}} \right)}$$

$$= \left( \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \right) (1 + \varepsilon \vec{\mathbf{v}}) \left( \mathbf{q}_0^* + \frac{\varepsilon}{2} \mathbf{q}_0^* \vec{\mathbf{t}} \right)$$

$$= \left( \mathbf{q}_0 + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \right) \left( \mathbf{q}_0^* + \varepsilon \vec{\mathbf{v}} \mathbf{q}_0^* + \frac{\varepsilon}{2} \mathbf{q}_0^* \vec{\mathbf{t}} + \frac{\varepsilon^2}{2} \vec{\mathbf{v}} \mathbf{q}_0^* \vec{\mathbf{t}} \right)$$

$$= \mathbf{q}_0 \mathbf{q}_0^* + \frac{\varepsilon}{2} \vec{\mathbf{t}} \mathbf{q}_0 \mathbf{q}_0^* + \varepsilon \mathbf{q}_0 \vec{\mathbf{v}} \mathbf{q}_0^* + \frac{\varepsilon^2}{2} \vec{\mathbf{t}} \mathbf{q}_0 \vec{\mathbf{v}} \mathbf{q}_0^* + \frac{\varepsilon}{2} \mathbf{q}_0 \mathbf{q}_0^* \vec{\mathbf{t}} + \frac{\varepsilon^2}{4} \vec{\mathbf{t}} \mathbf{q}_0 \mathbf{q}_0^* \vec{\mathbf{t}}$$

$$(\vec{\mathbf{t}} \mathbf{q}_0)^* = \mathbf{q}_0^* (\vec{\mathbf{t}})^*$$

$$= -\mathbf{q}_0^* \vec{\mathbf{t}}$$

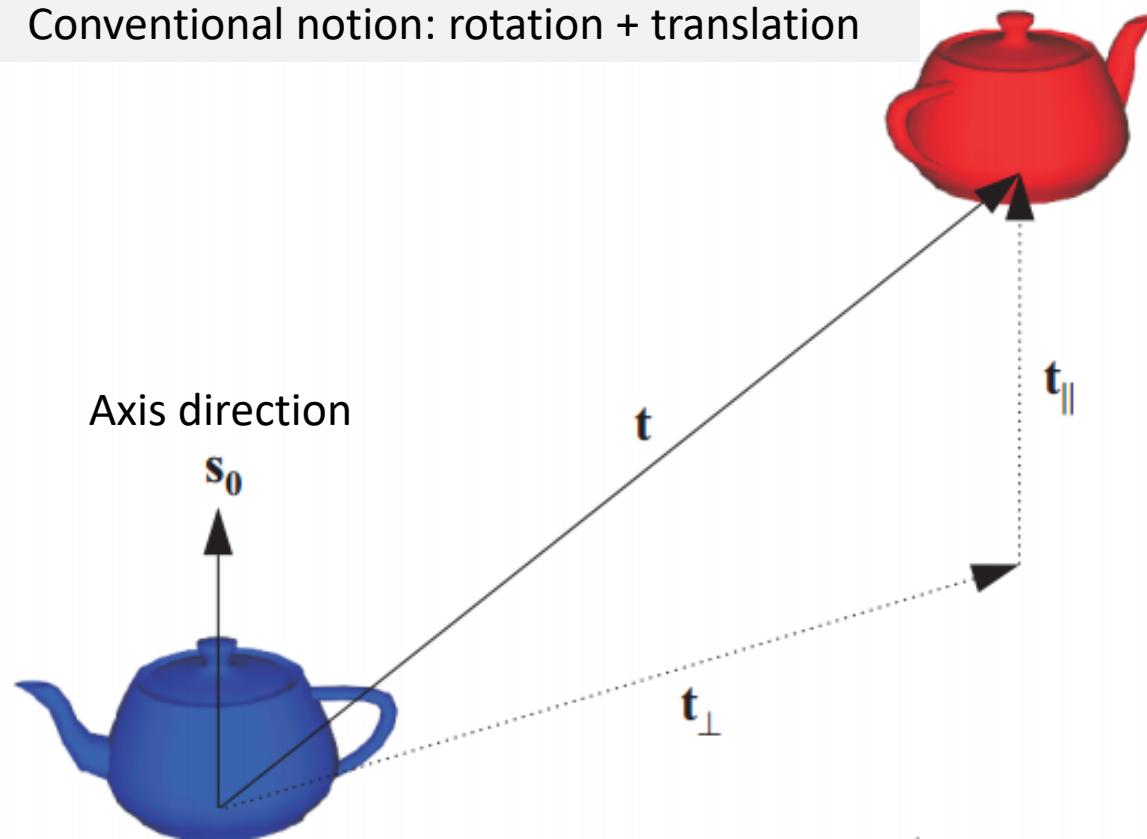
$$= 1 + \varepsilon (\vec{\mathbf{t}} + \mathbf{q}_0 \vec{\mathbf{v}} \mathbf{q}_0^*)$$

3D position  $\vec{\mathbf{v}}$  rotated by quaternion  $\mathbf{q}_0$

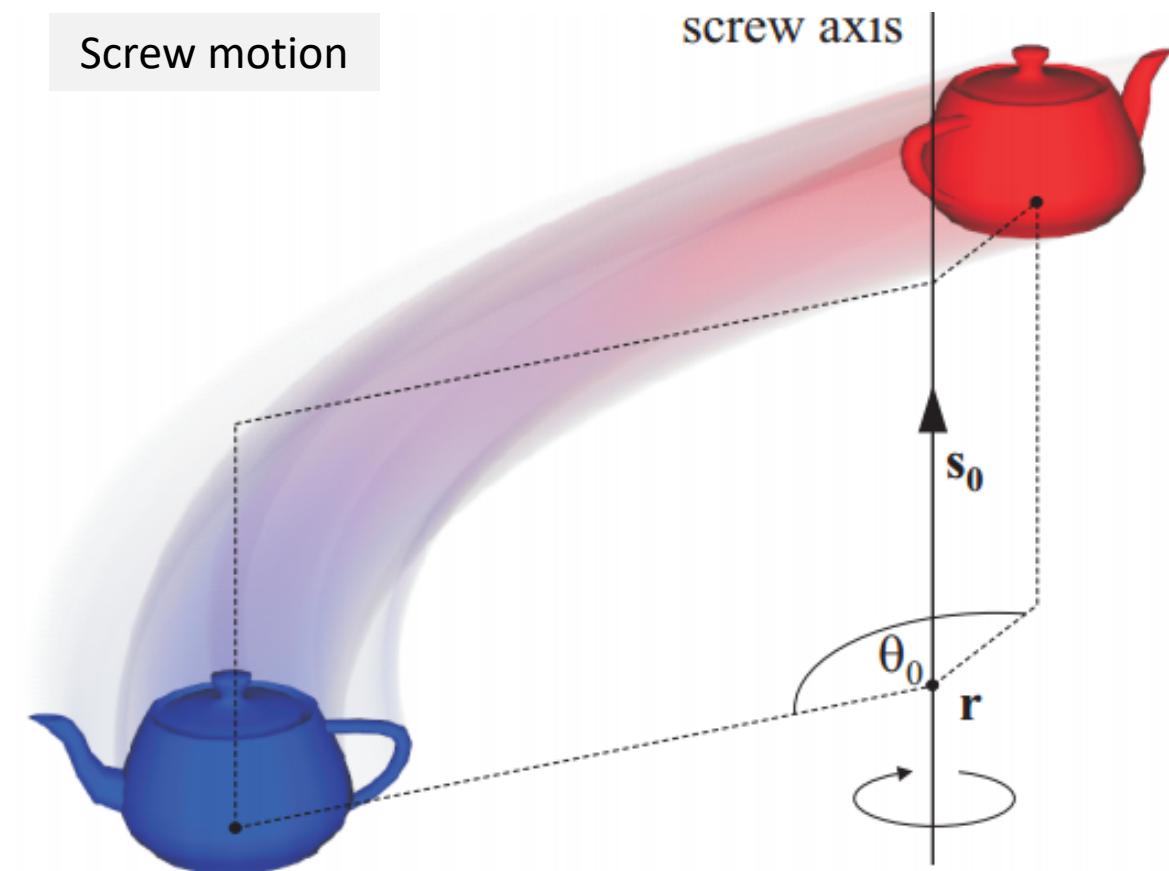
$$\|\mathbf{q}_0\|^2 = 1$$

# Rigid transformation as “screw motion”

Conventional notion: rotation + translation



Screw motion



- Any rigid motion is uniquely described as screw motion
  - (Up to antipodality)

# Screw motion & dual quaternion

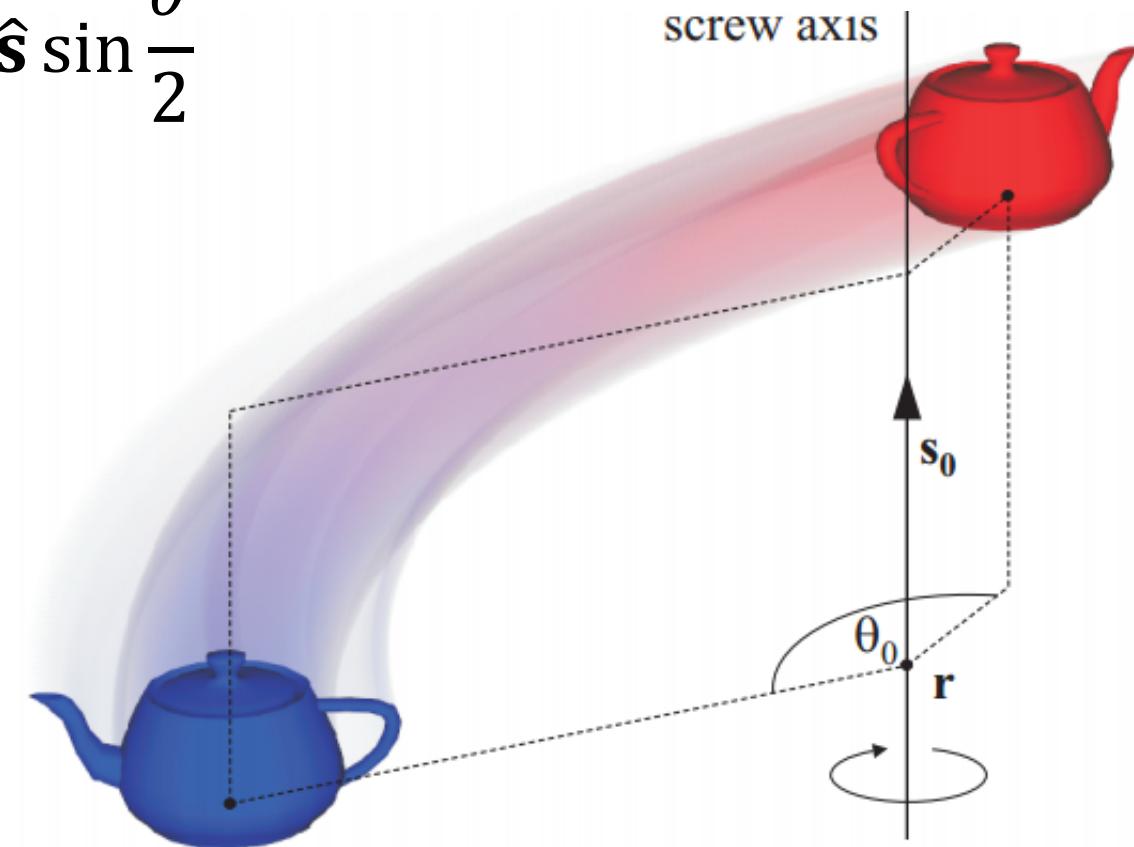
- Unit dual quaternion  $\hat{\mathbf{q}}$  can be written as:

$$\hat{\mathbf{q}} = \cos \frac{\hat{\theta}}{2} + \hat{\mathbf{s}} \sin \frac{\hat{\theta}}{2}$$

- $\hat{\theta} = \theta_0 + \varepsilon \theta_\varepsilon$        $\theta_0, \theta_\varepsilon$  : real number
- $\hat{\mathbf{s}} = \vec{\mathbf{s}}_0 + \varepsilon \vec{\mathbf{s}}_\varepsilon$        $\vec{\mathbf{s}}_0, \vec{\mathbf{s}}_\varepsilon$  : unit 3D vector

- Geometric meaning

- $\vec{\mathbf{s}}_0$  : direction of rotation axis
- $\theta_0$  : amount of rotation
- $\theta_\varepsilon$  : amount of translation parallel to  $\vec{\mathbf{s}}_0$
- $\vec{\mathbf{s}}_\varepsilon$  : when rotation axis passes through  $\vec{\mathbf{r}}$ ,  
it satisfies  $\vec{\mathbf{s}}_\varepsilon = \vec{\mathbf{r}} \times \vec{\mathbf{s}}_0$



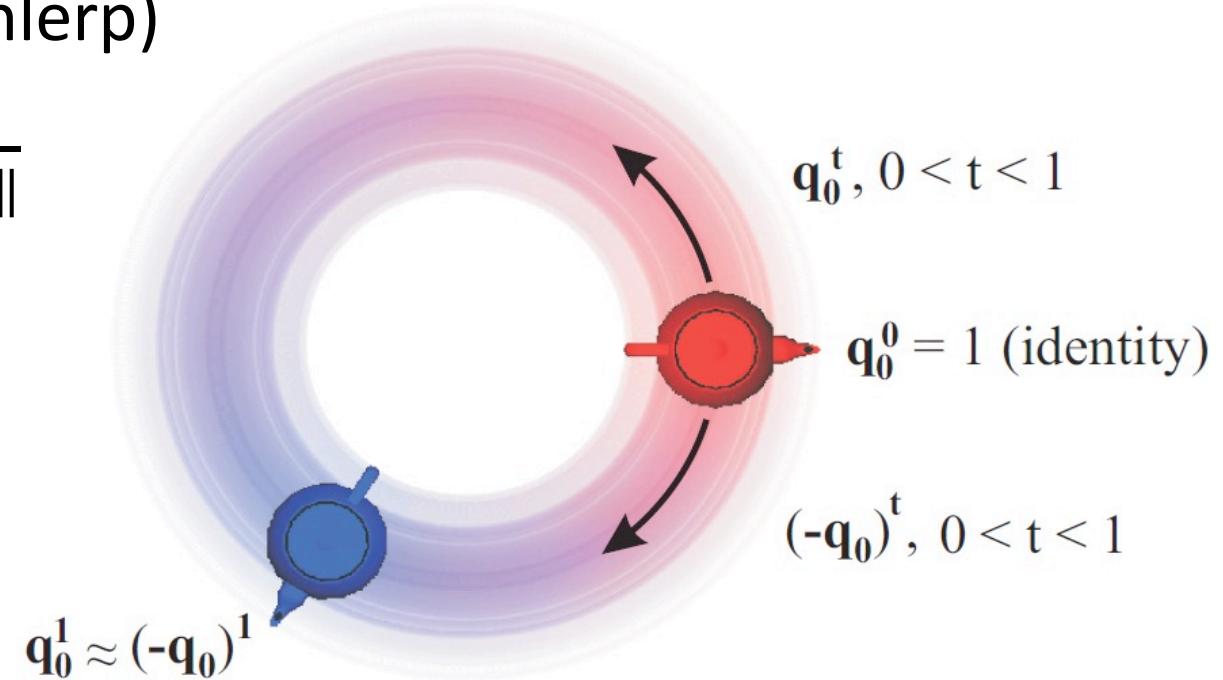
# Interpolating two rigid transformations

- Linear interpolation + normalization (nlerp)

$$\text{nlerp}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, t) := \frac{(1-t)\hat{\mathbf{q}}_1 + t\hat{\mathbf{q}}_2}{\|(1-t)\hat{\mathbf{q}}_1 + t\hat{\mathbf{q}}_2\|}$$

- Note:  $\hat{\mathbf{q}}$  &  $-\hat{\mathbf{q}}$  represent same transformation with opposite path

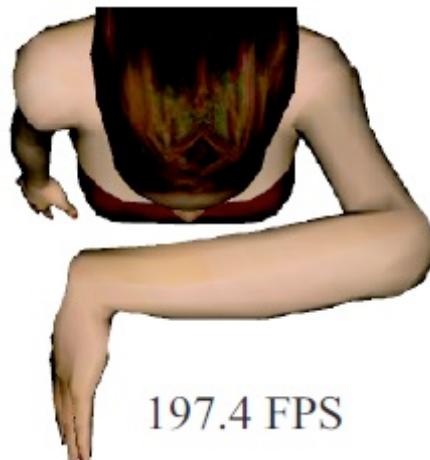
- If 4D dot product of non-dual components of  $\hat{\mathbf{q}}_1$  &  $\hat{\mathbf{q}}_2$  is negative, use  $-\hat{\mathbf{q}}_2$  in the interpolation



# Blending rigid motions using dual quaternion

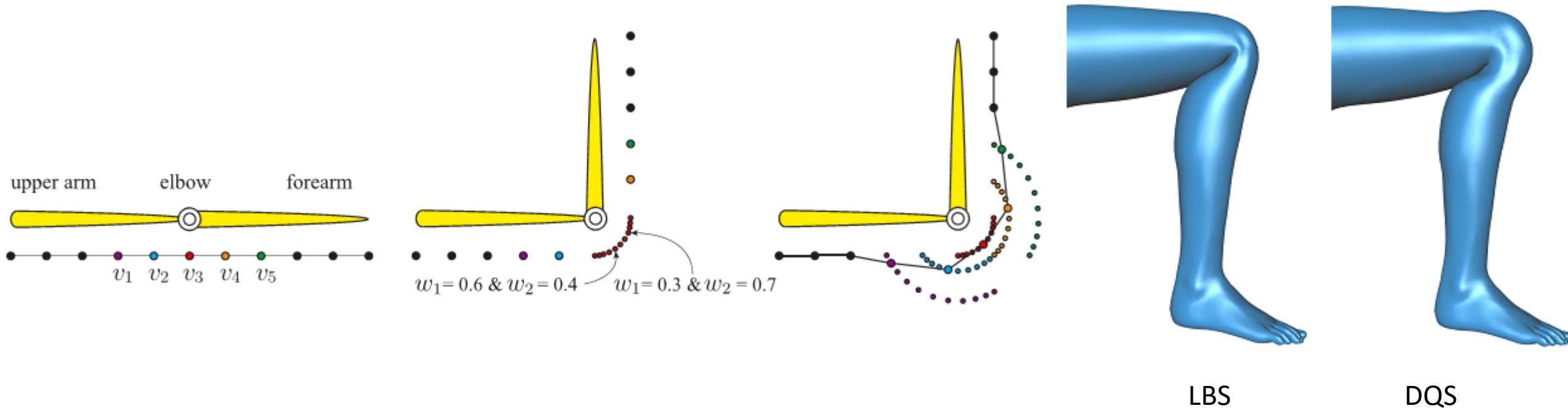
$$\text{blend}(\langle w_1, \hat{\mathbf{q}}_1 \rangle, \langle w_2, \hat{\mathbf{q}}_2 \rangle, \dots) := \frac{w_1 \hat{\mathbf{q}}_1 + w_2 \hat{\mathbf{q}}_2 + \dots}{\|w_1 \hat{\mathbf{q}}_1 + w_2 \hat{\mathbf{q}}_2 + \dots\|}$$

- Akin to blending rotations using quaternion
- Same input format as LBS & low computational cost
- Standard feature in many commercial CG packages

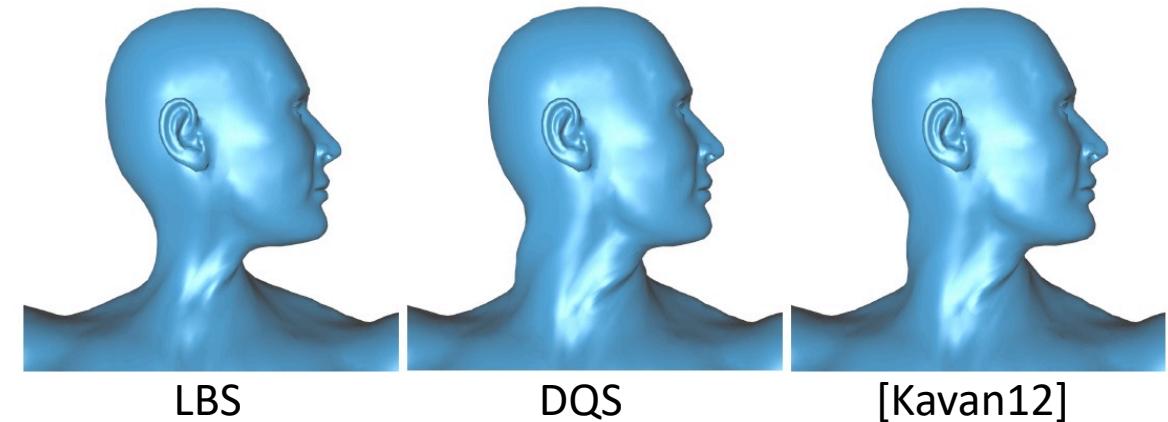
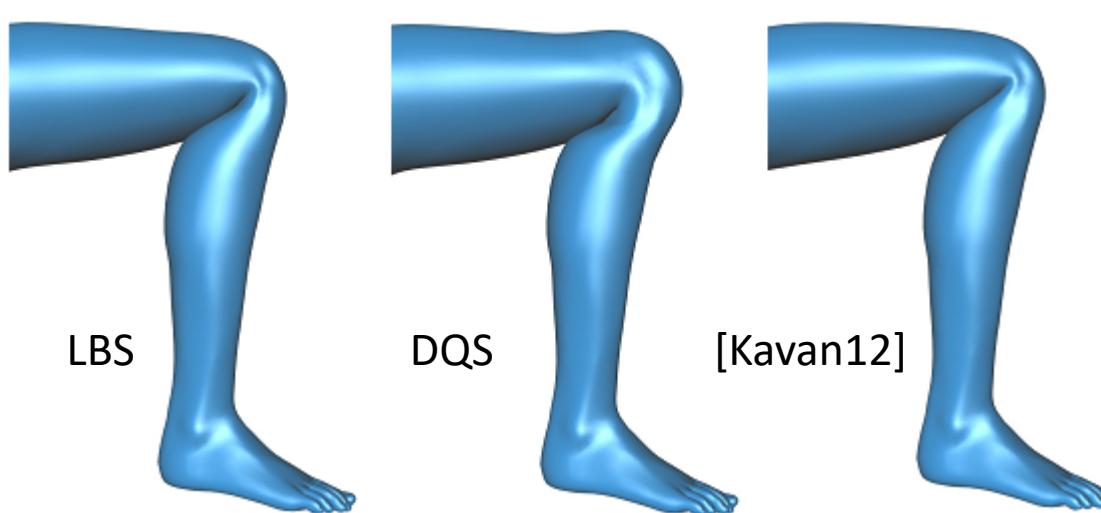


# Artifact of DQS: “bulging” effect

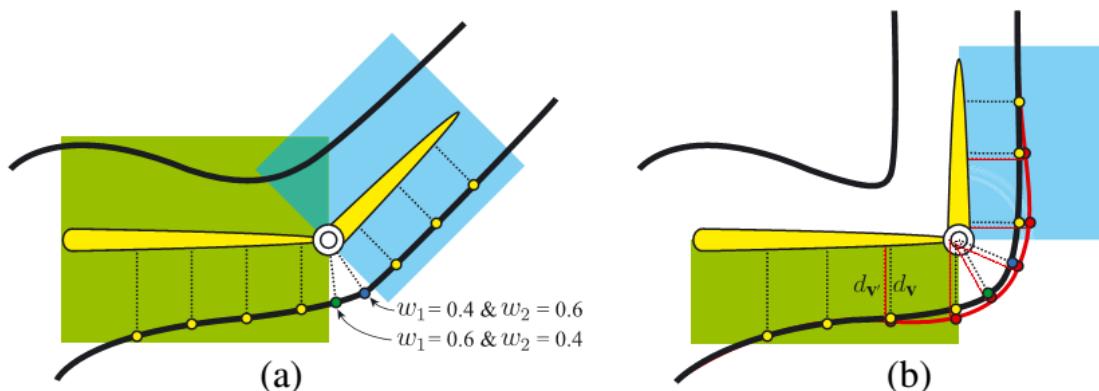
- Produces ball-like shape around the joint when bended



# Overcoming DQS's drawback



Decompose transformation into bend & twist,  
interpolate them separately [Kavan12]



After deforming using DQS,  
offset vertices along normals [Kim14]

# Skinning for avoiding self-intersections

- Make use of implicit functions



<https://www.youtube.com/watch?v=RHySGIqEgyk>

# Other deformation mechanisms than skinning

Unified point/cage/skeleton handles [Jacobson 11]

## Bounded Biharmonic Weights for Real-Time Deformation

Alec Jacobson<sup>1</sup>

Ilya Baran<sup>2</sup>

Jovan Popović<sup>3</sup>

Olga Sorkine<sup>1,4</sup>

<sup>1</sup>New York University

<sup>2</sup>Disney Research, Zurich

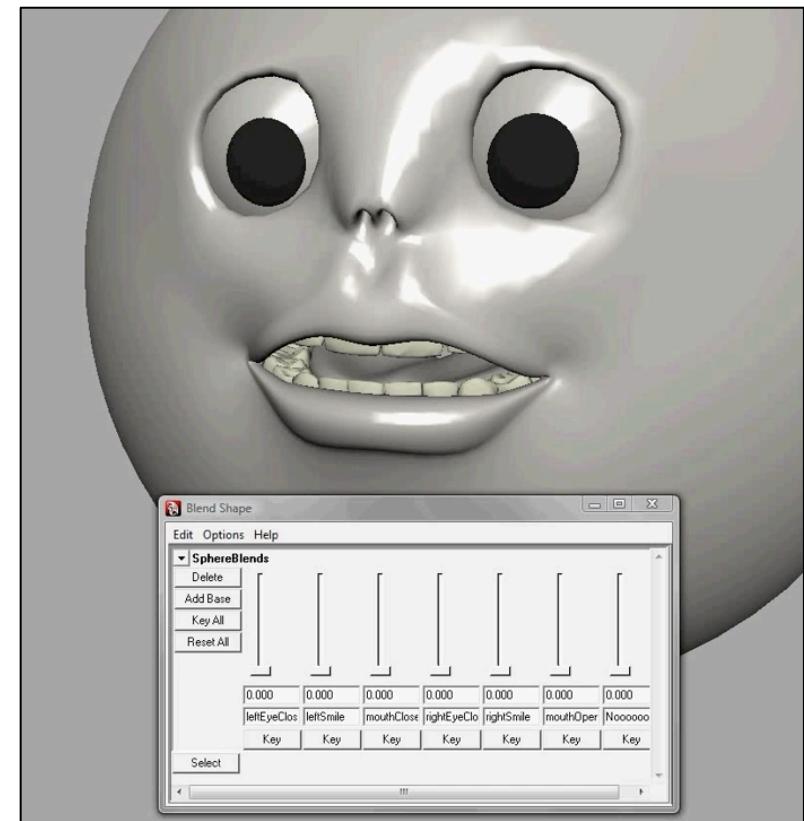
<sup>3</sup>Adobe Systems, Inc.

<sup>4</sup>ETH Zurich

This video contains narration

<https://www.youtube.com/watch?v=P9fqm8vgdB8>

BlendShape



<https://www.youtube.com/watch?v=hb5zH4V37r8>

# References

- [http://en.wikipedia.org/wiki/Motion\\_capture](http://en.wikipedia.org/wiki/Motion_capture)
- <http://skinning.org/>
- <http://mukai-lab.org/category/library/legacy>