Introduction to Computer Graphics

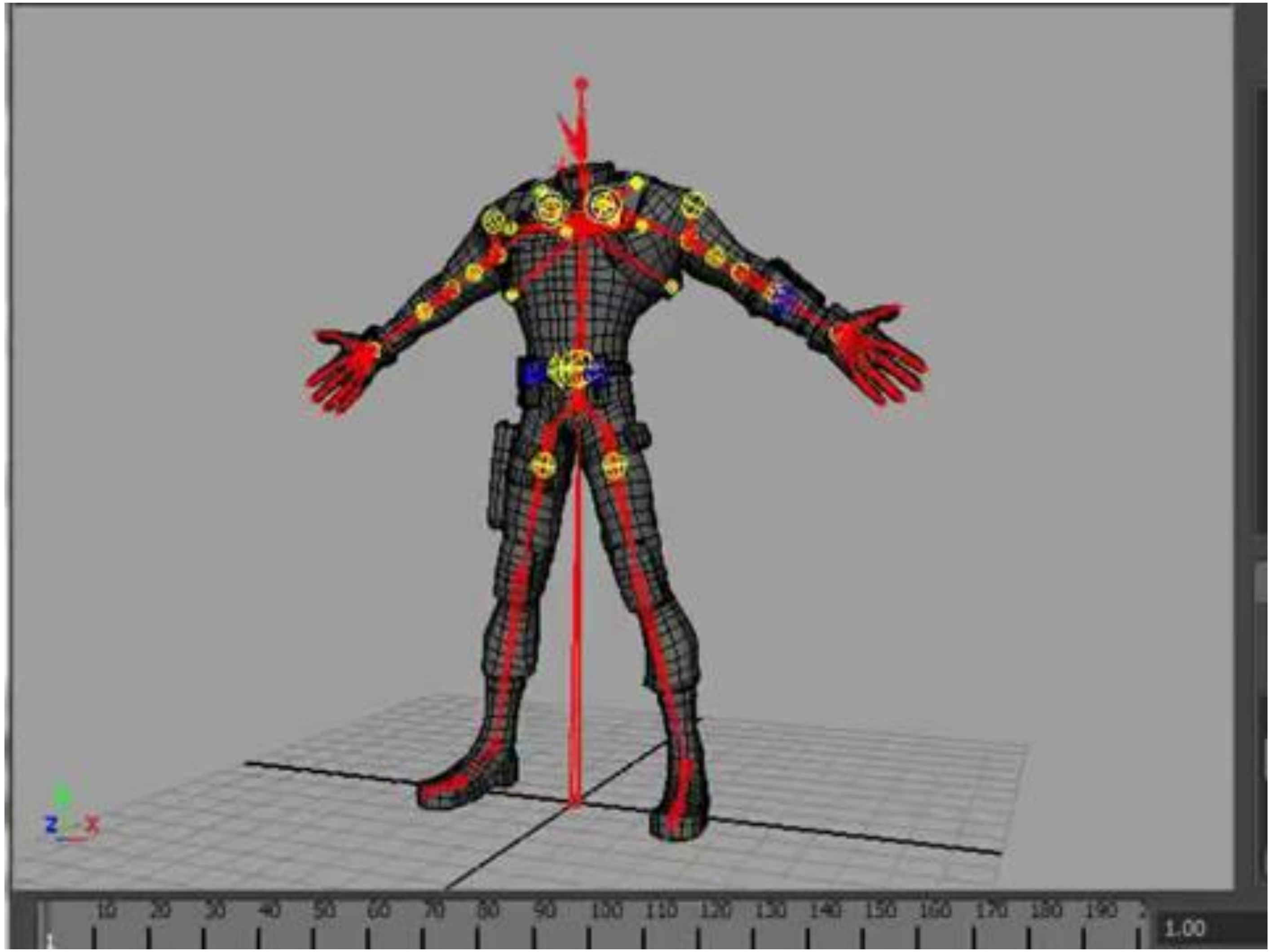
– Animation (1) –

May 23, 2019

Kenshi Takayama

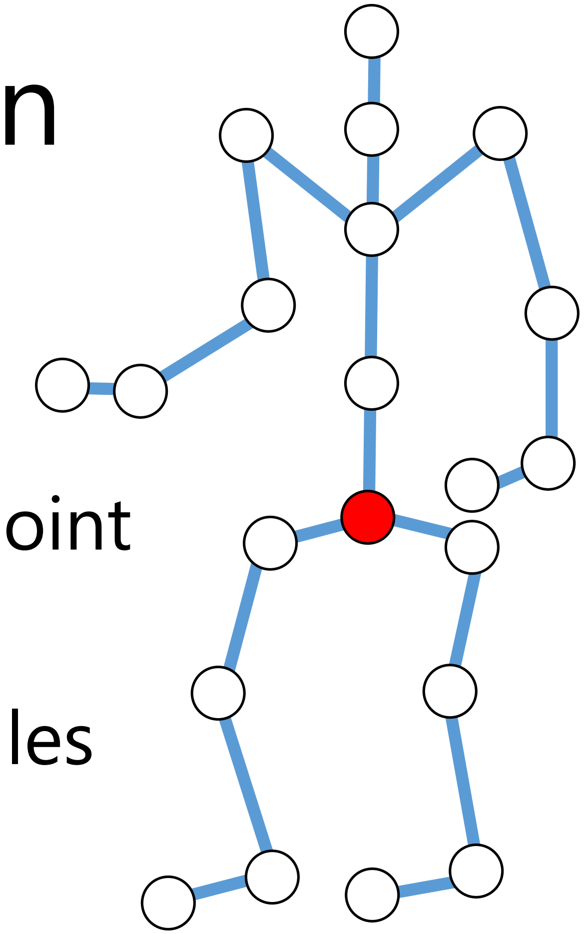
Skeleton-based animation

- Simple
- Intuitive
- Low comp. cost



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

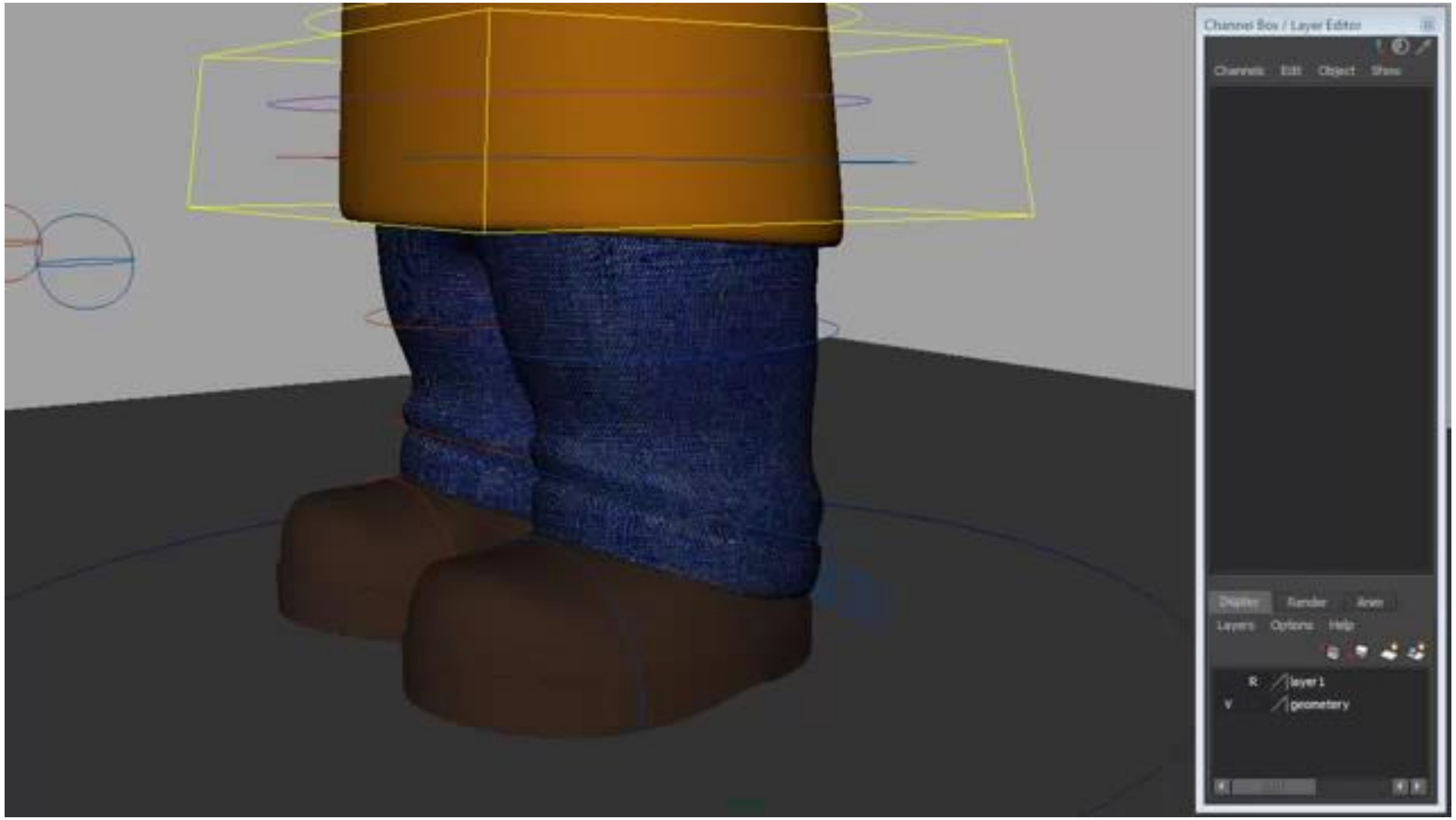
Representing a pose using skeleton



- 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 (**F**orward **K**inematics)
- Deeply related to robotics

Inverse Kinematics

- Find joint angles s.t. an end effector comes at a given goal position

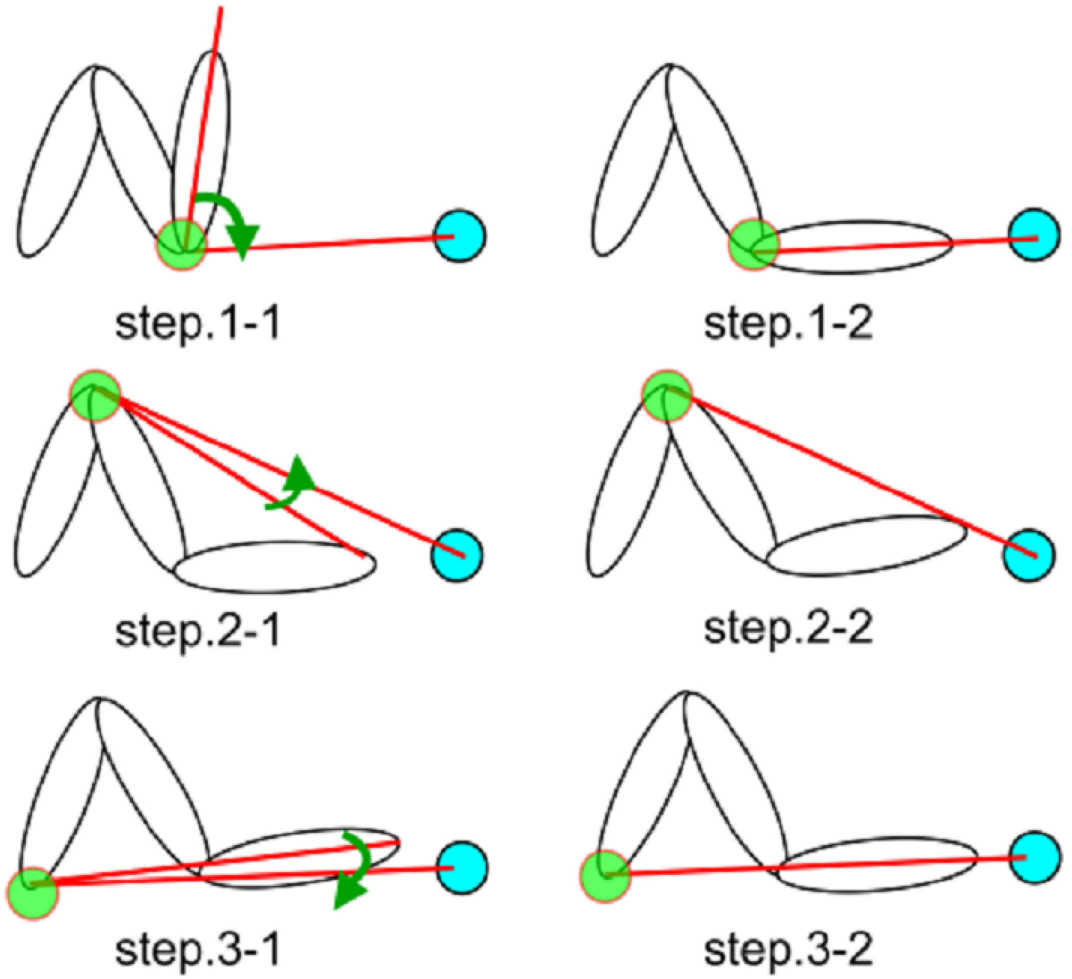


- Typical workflow:
  - Quickly create pose using IK, fine adjustment using FK

https://www.youtube.com/watch?v=e1qnZ9rV_kw

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]



https://mukai-lab.org/content/CcdParticleInverseKinematics.pdf

IK minimizing elastic energy

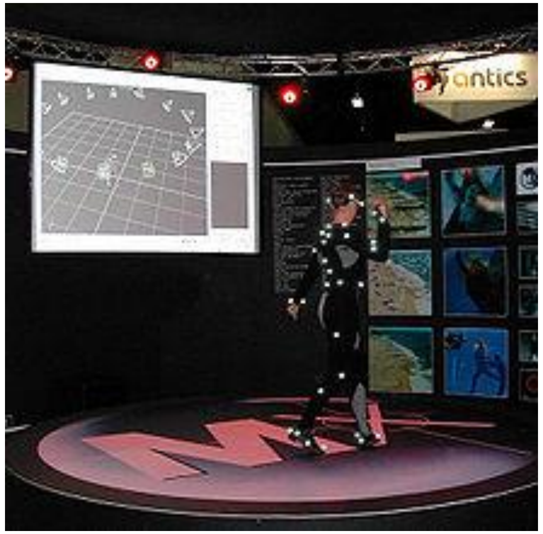
https://www.youtube.com/watch?v=PRcXy2LjI9I

Fast Automatic Skinning Transformations [Jacobson SIGGRAPH12]

Ways to obtain/measure motion data

Optical motion capture

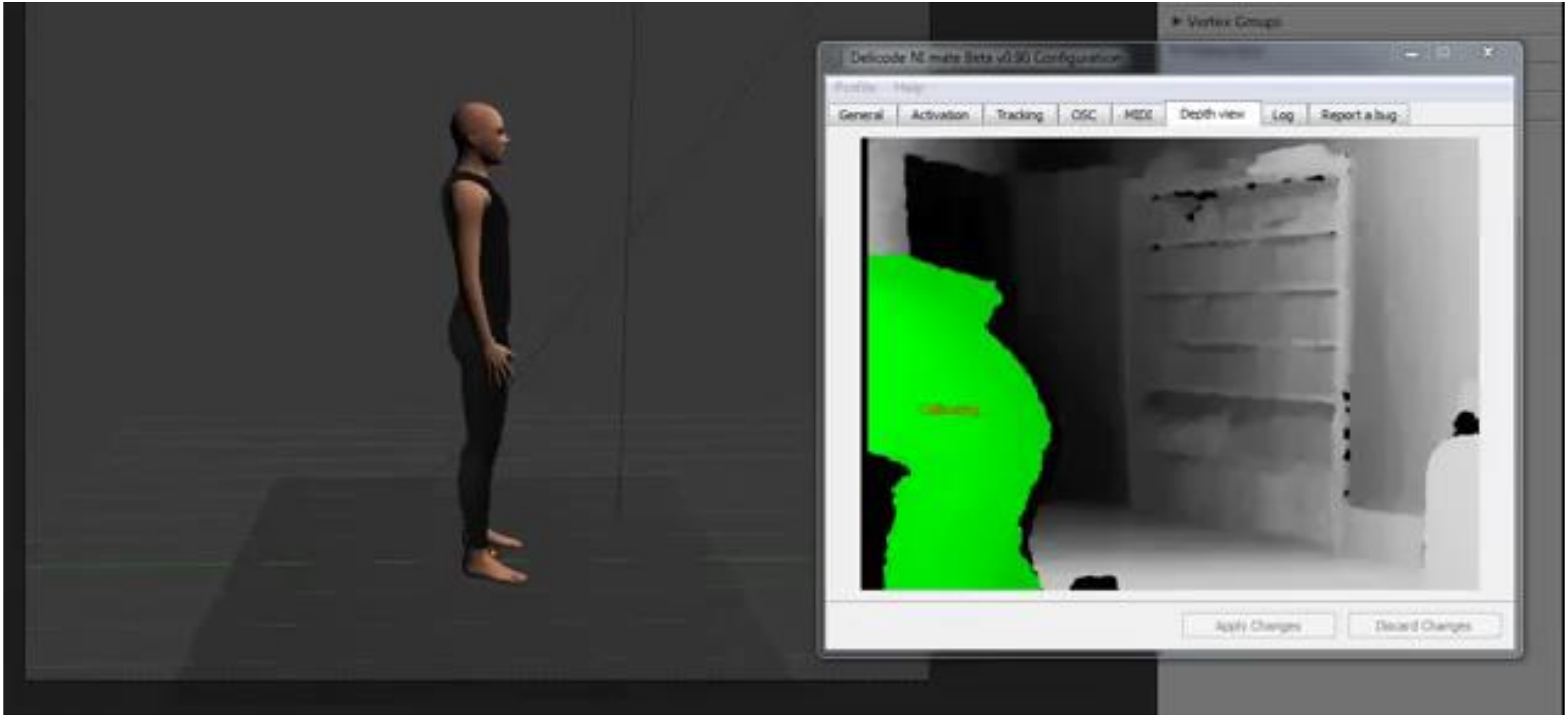
- Put markers on the actor, record video from many viewpoints (~48)



from Wikipedia

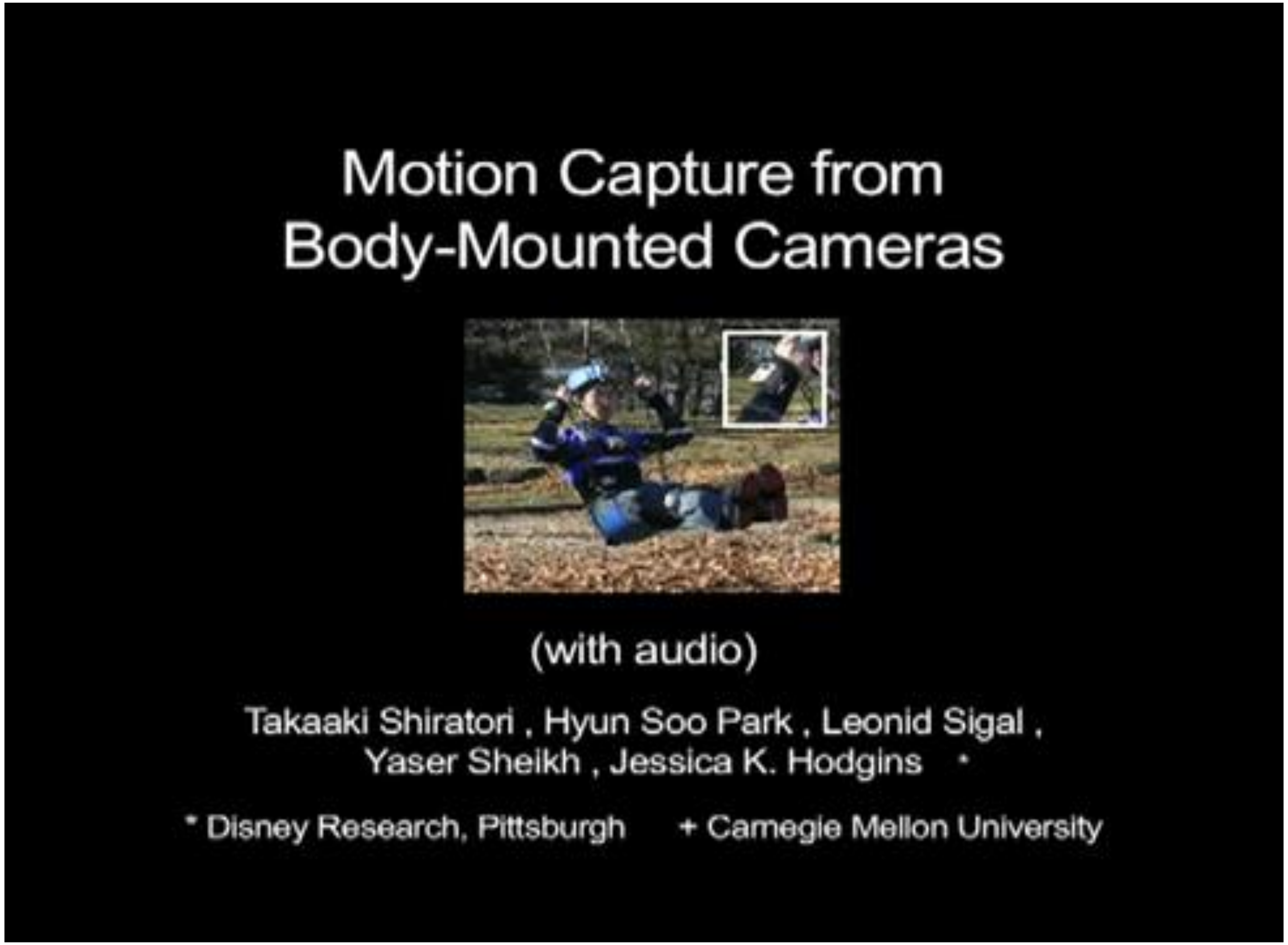
https://www.youtube.com/watch?v=c6X64LhcUyQ

Mocap using inexpensive depth camera



https://www.youtube.com/watch?v=qC-fdgPJhQ8

Mocap designed for outdoor scene

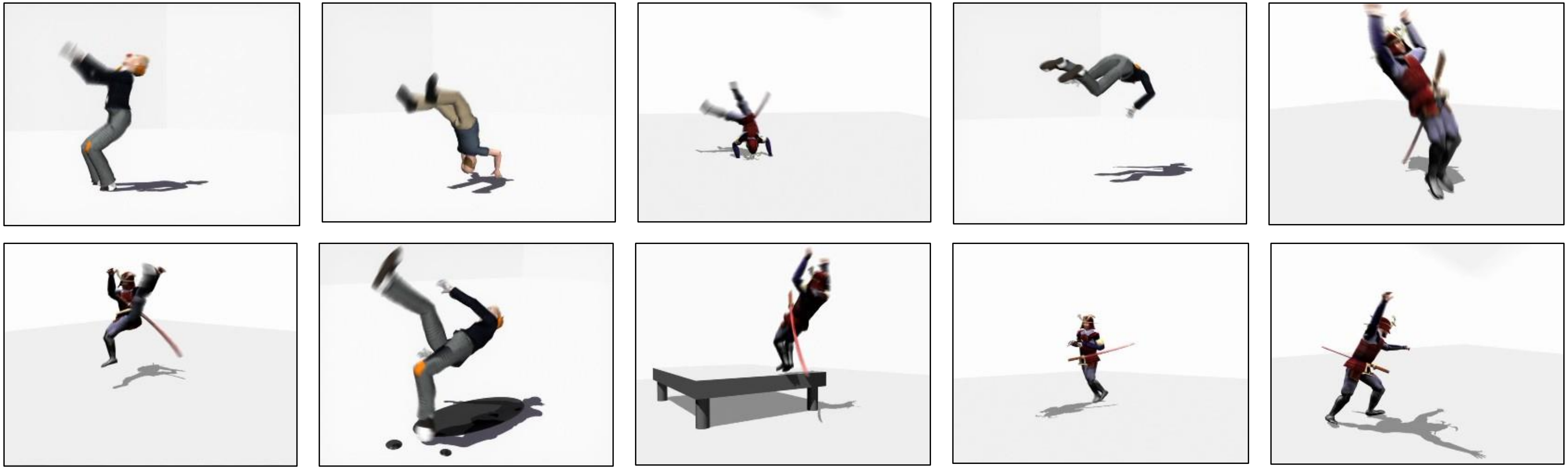


https://www.youtube.com/watch?v=xbI-NWMfGPs

Motion Capture from Body-Mounted Cameras [Shiratori SIGGRAPH11]

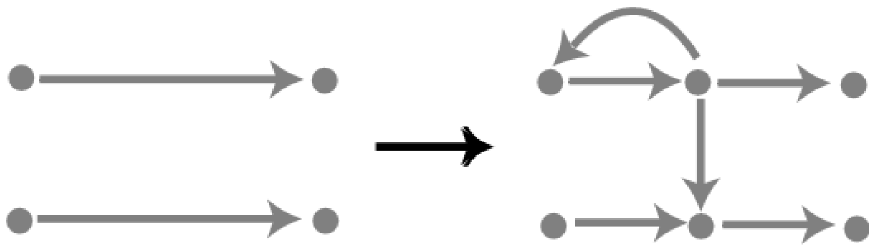
Motion database

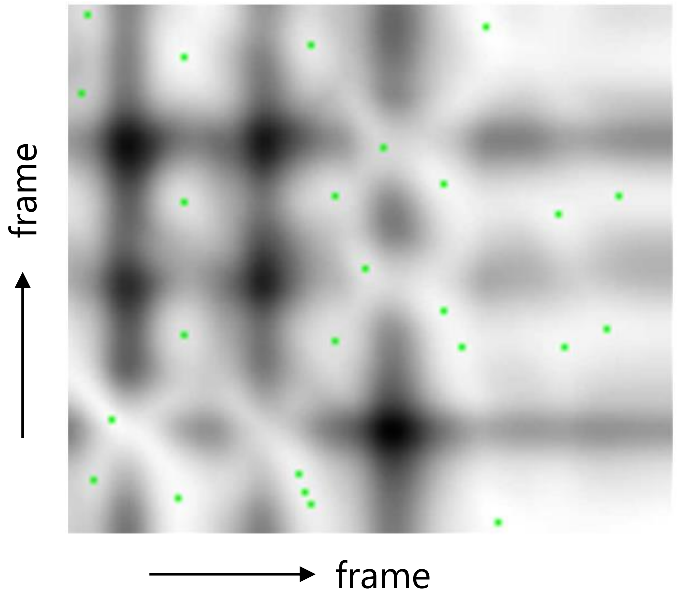
- http://mocap.cs.cmu.edu/
- 6 categories, 2605 in total
- Free for research purposes
  - Interpolation, recombination, analysis, search, etc.



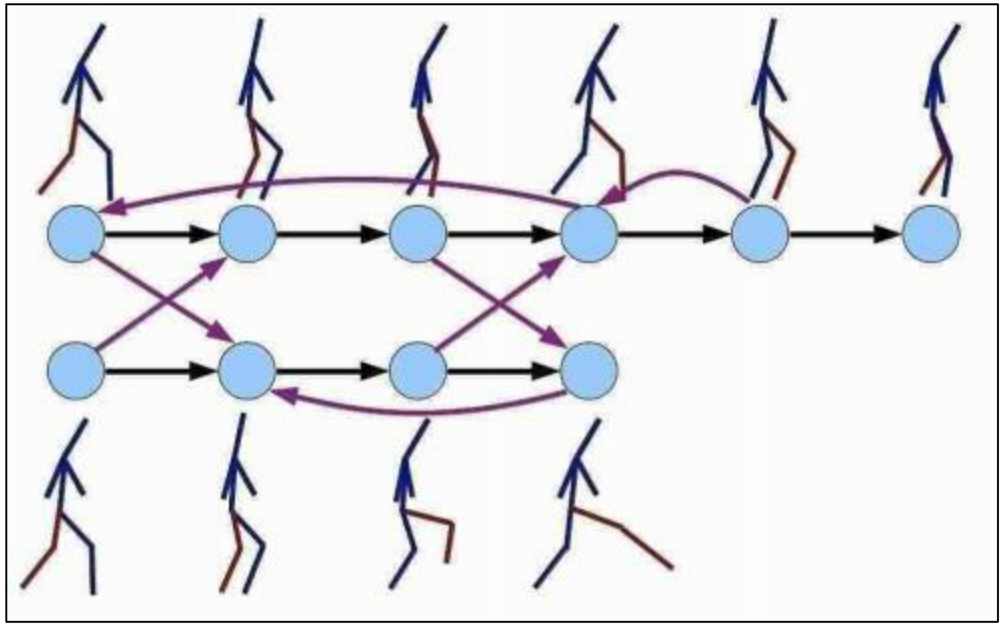
Recombining motions

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





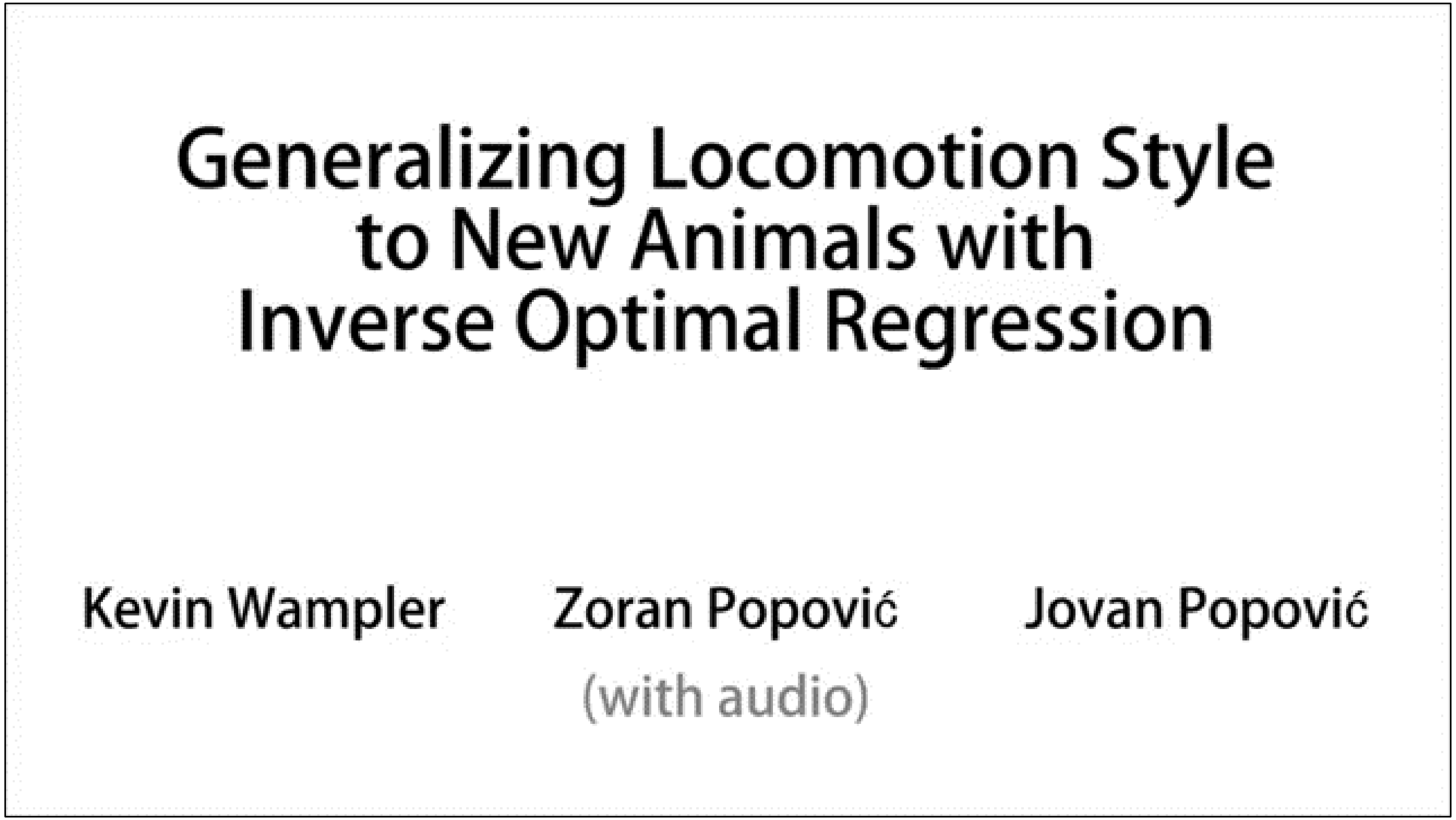
Pose similarity matrix



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

Generating motion through simulation

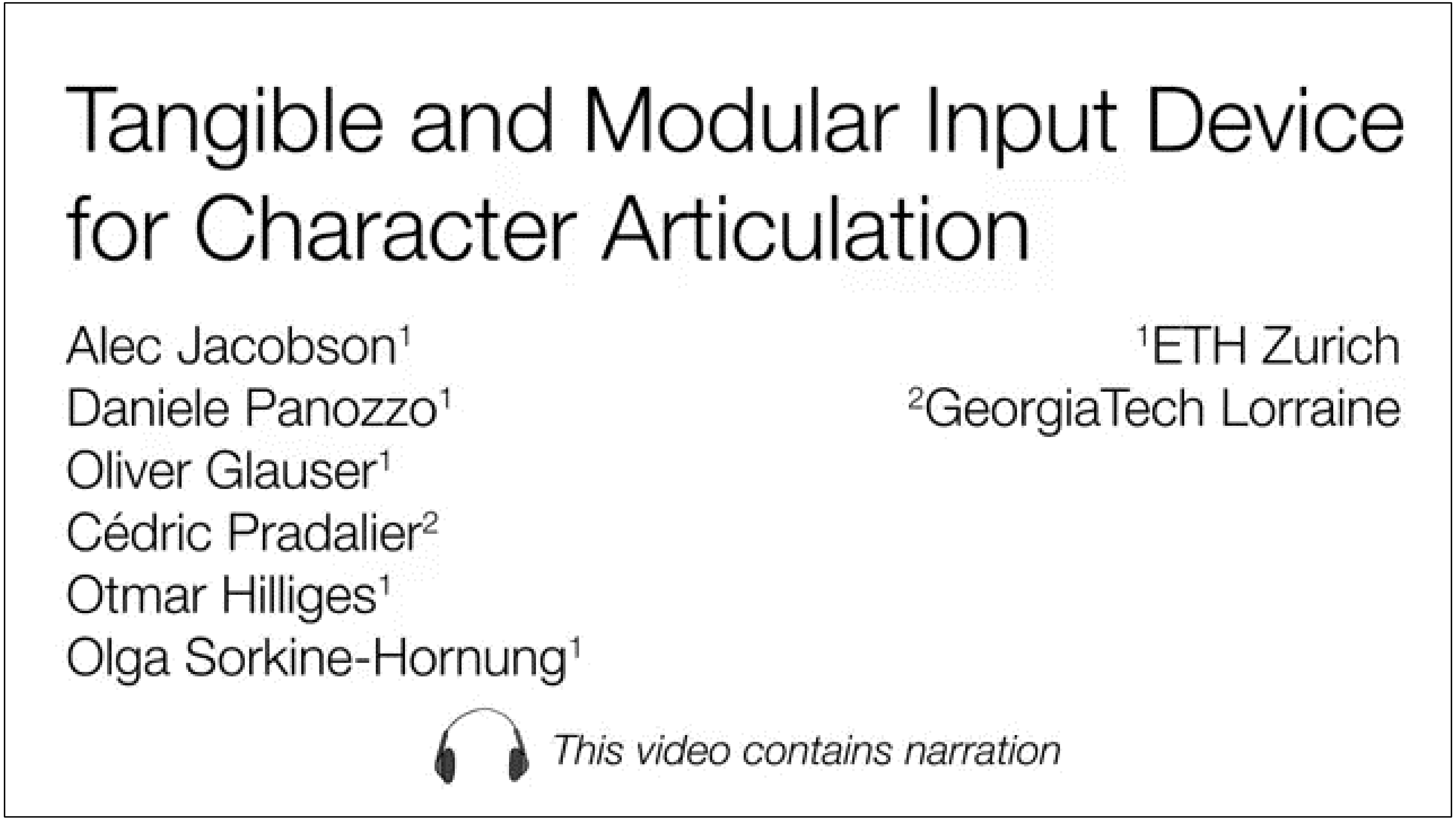
- For creatures unsuitable for mocap
  - Too dangerous, nonexistent, ...
- Natural motion respecting body shape
- Can interact with dynamic environment



https://www.youtube.com/watch?v=KF_a1c7zytw

Generalizing Locomotion Style to New Animals With Inverse Optimal Regression [Wampler SIGGRAPH14]

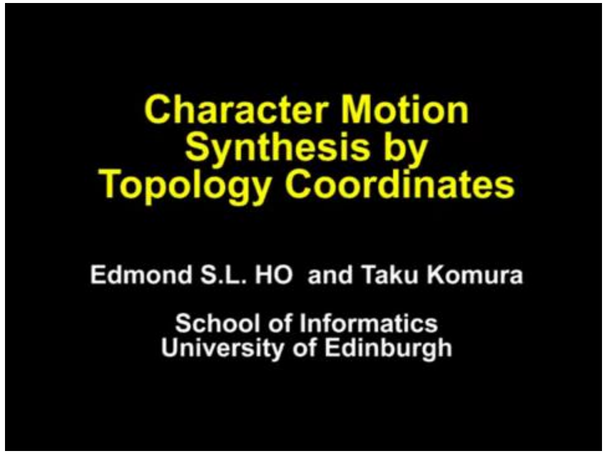
Creating poses using special devices



Tangible and Modular Input Device for Character Articulation [Jacobson SIGGRAPH14]
Rig Animation with a Tangible and Modular Input Device [Glauser SIGGRAPH16]

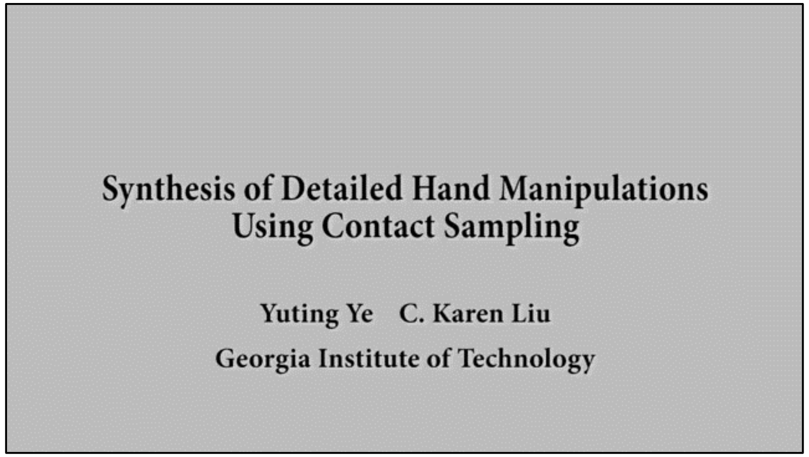
https://www.youtube.com/watch?v=vBX47JamMN0

Many topics about character motion



Interaction between multiple persons

https://www.youtube.com/watch?v=1S_6wSKI_nU



Grasping motion

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

Crowd simulation

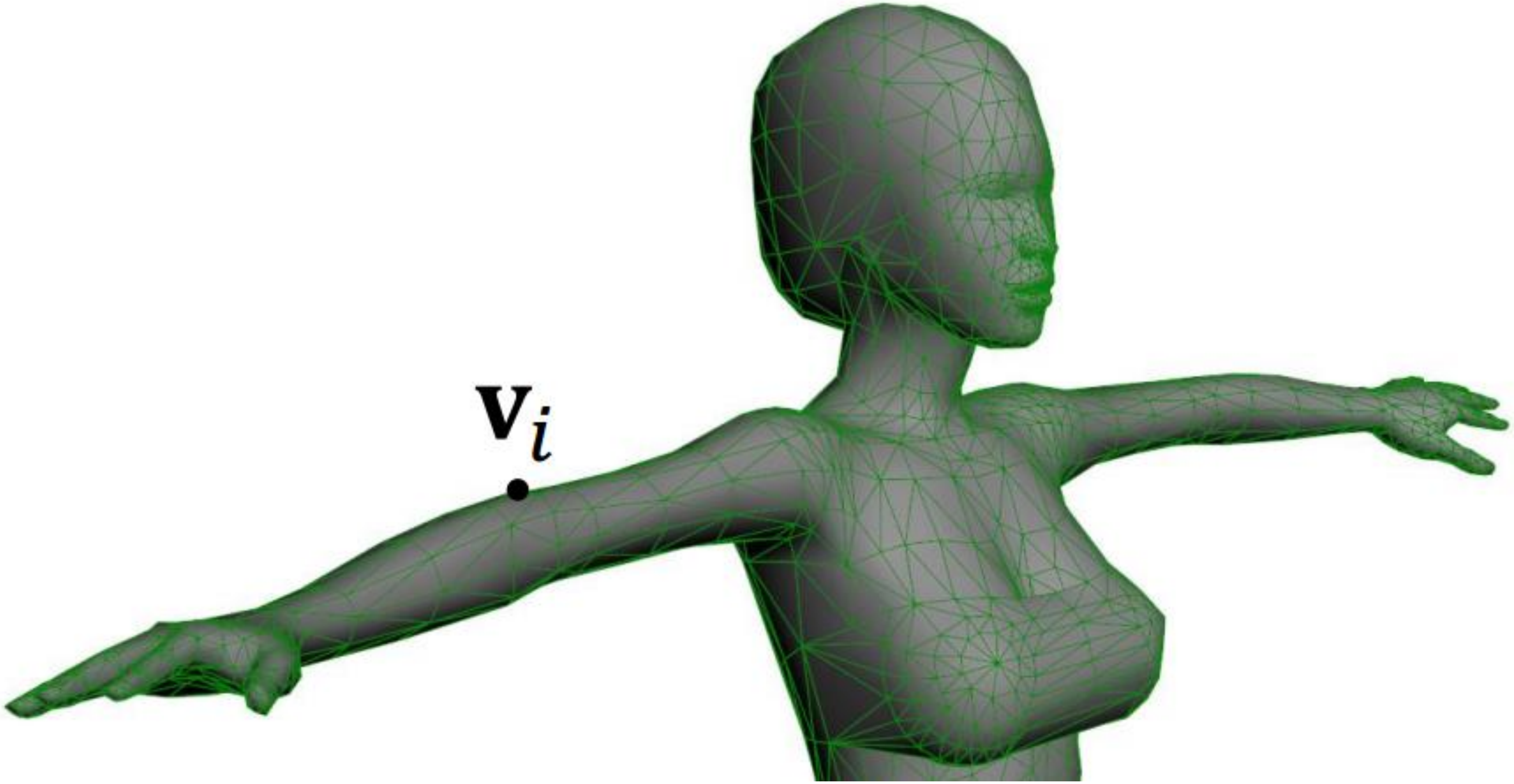
https://www.youtube.com/watch?v=pqBSNAOsMDc

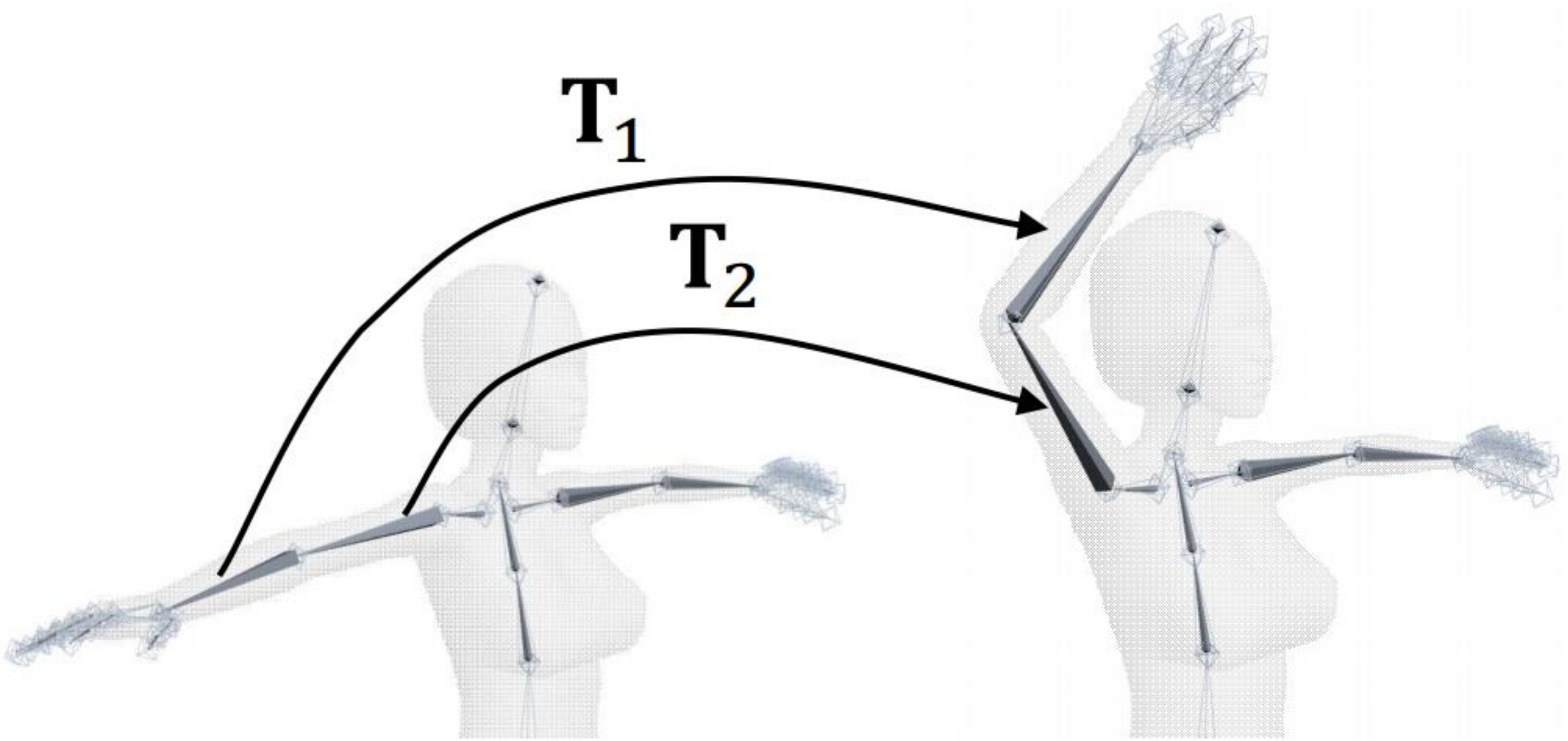
Path planning

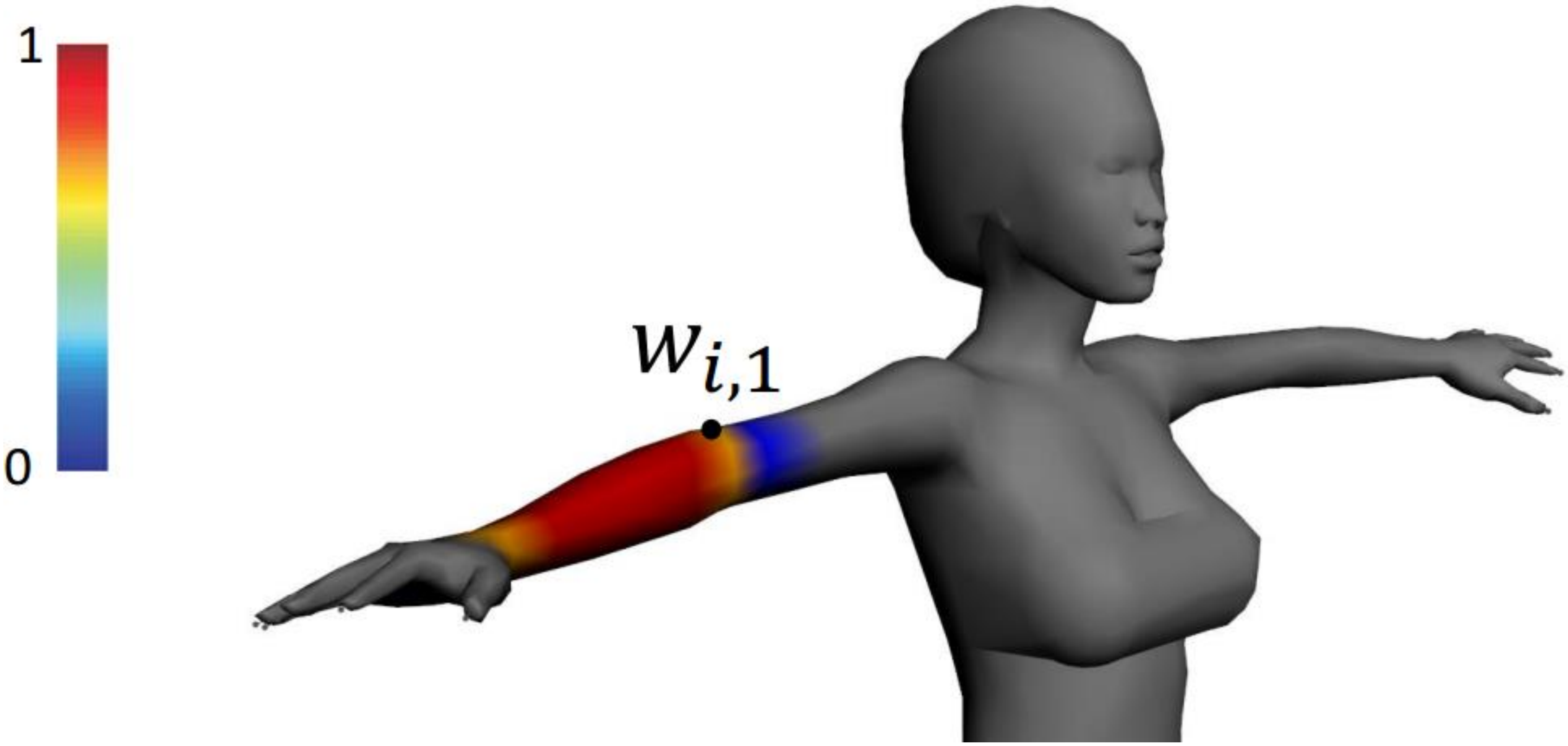
https://vimeo.com/33409868

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]

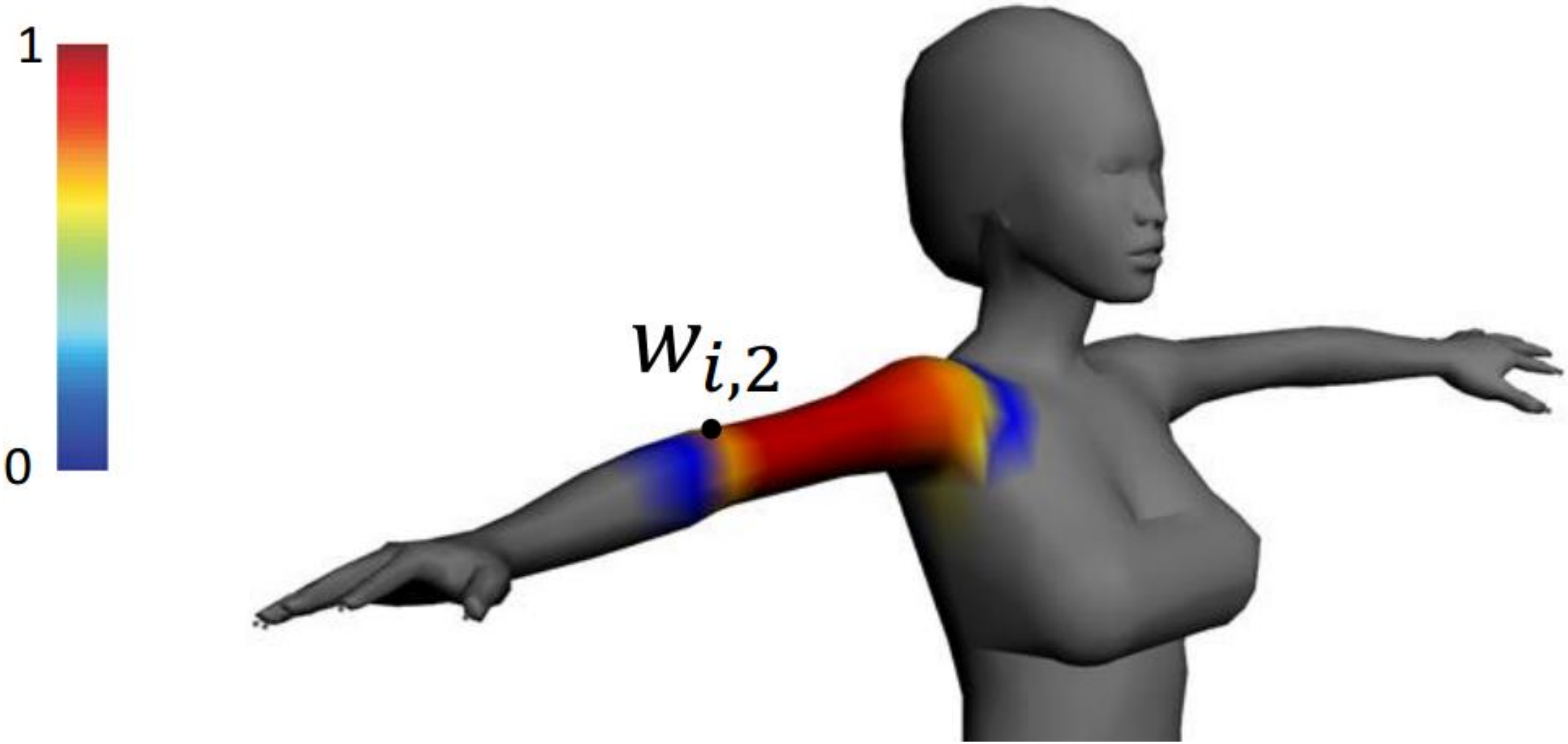
Skinning

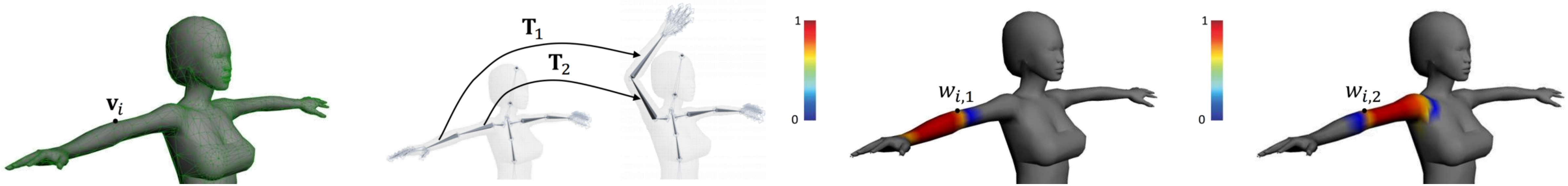

$\mathbf{v}_i$


$\mathbf{T}_1$
$\mathbf{T}_2$



1
0
$w_{i,1}$


1
0
$w_{i,2}$



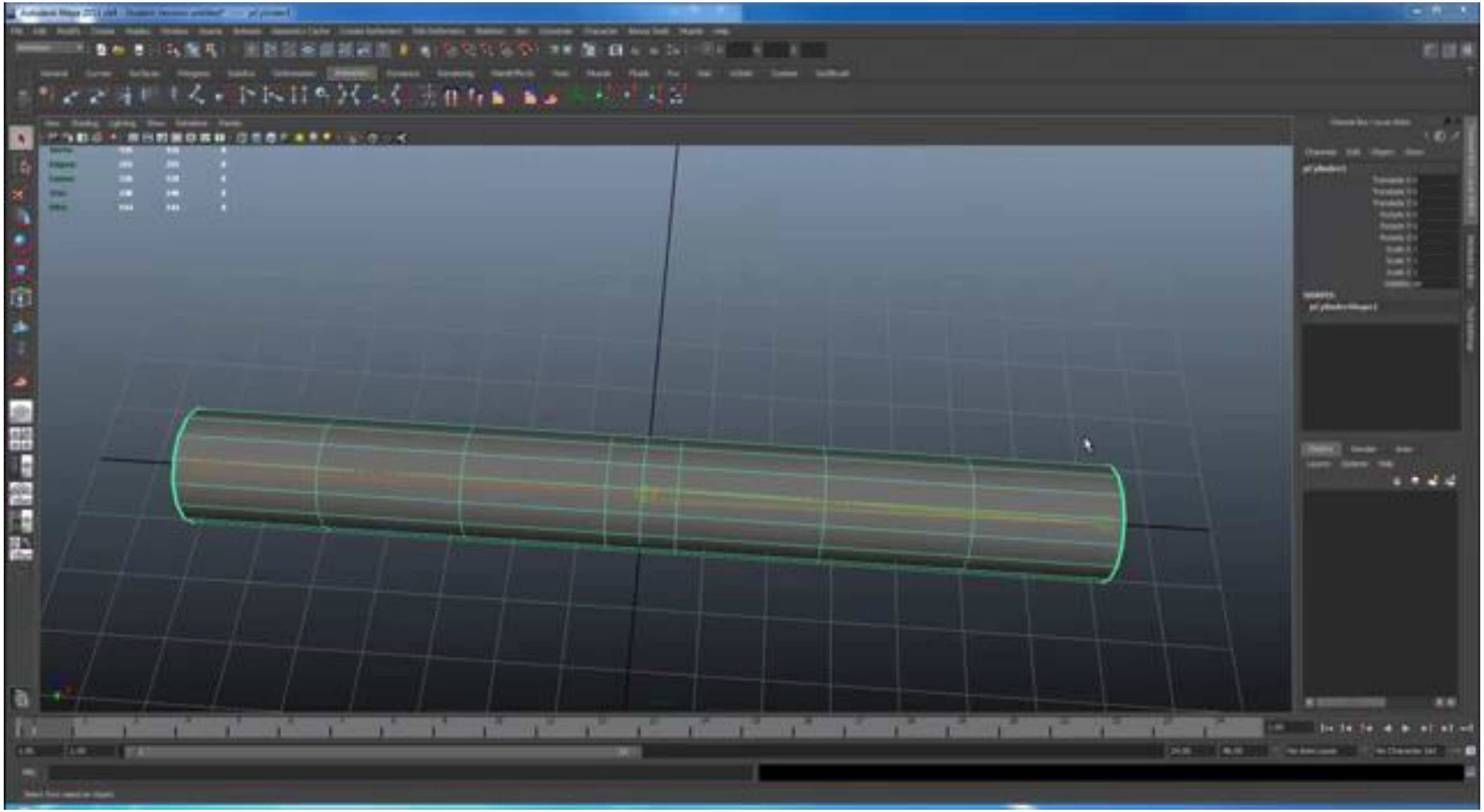
$$\mathbf{v}_i' = \mathrm{blend}\left(\langle w_{i,1}, \mathbf{T}_1 \rangle, \langle w_{i,2}, \mathbf{T}_2 \rangle, \ldots\right)(\mathbf{v}_i)$$

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

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

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

Simple way to define weights: painting



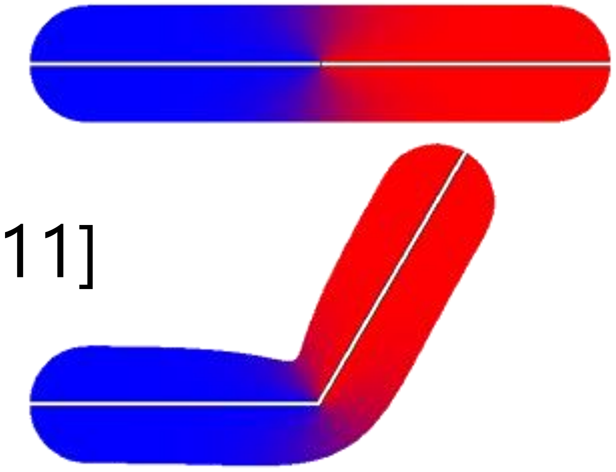
https://www.youtube.com/watch?v=TACB6bX8SN0

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 1st-order derivative $\int_\Omega \left\|\nabla w_j\right\|^2 dA$ [Baran 07]
  - Approximate solution only on surface ➔ easy & fast

- Minimize 2nd-order derivative $\int_\Omega \left(\Delta w_j\right)^2 dA$ [Jacobson 11]
  - Introduce inequality constraints $0 \leq w_j \leq 1$
  - Quadratic Programming over the volume ➔ high-quality



Pinocchio demo

Automatic rigging and animation of 3d characters [Baran SIGGRAPH07]
Bounded Biharmonic Weights for Real-Time Deformation [Jacobson SIGGRAPH11]

Simple way to blend transformations: **L**inear **B**lend **S**kinning

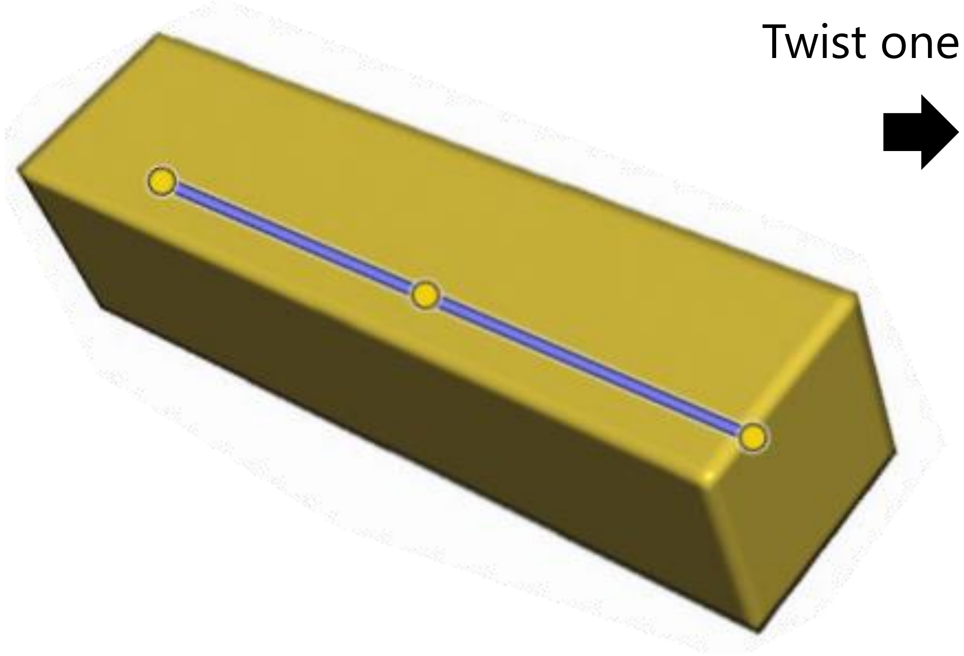
- Represent rigid transformation $\mathbf{T}_j$ as a 3×4 matrix consisting of rotation matrix $\mathbf{R}_j \in \mathbb{R}^{3\times3}$ and translation vector $\mathbf{t}_j \in \mathbb{R}^3$

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

- 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

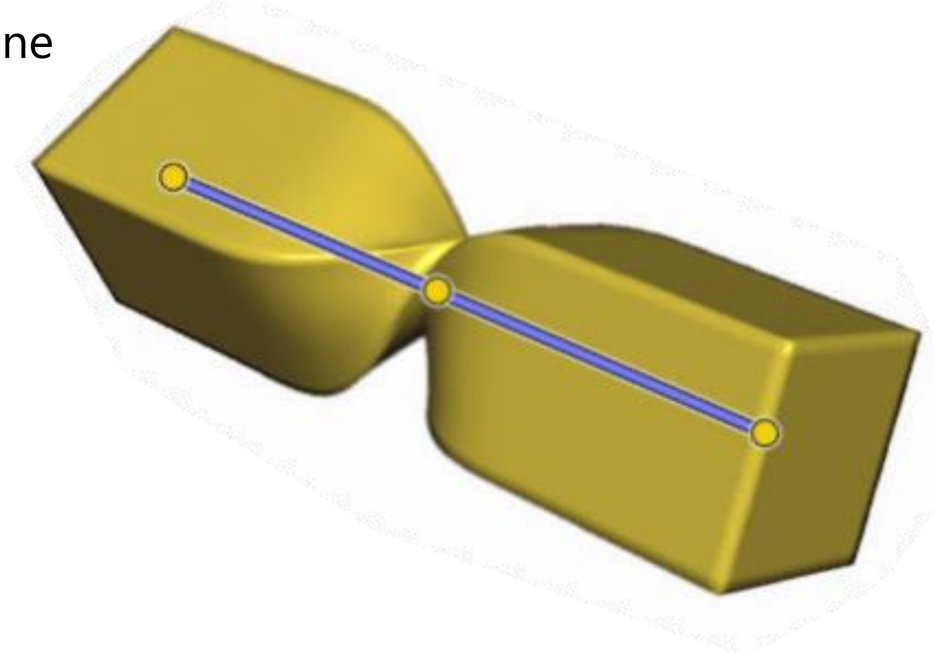
- Standard method

Artifact of LBS: "candy wrapper" effect



Initial shape & two bones

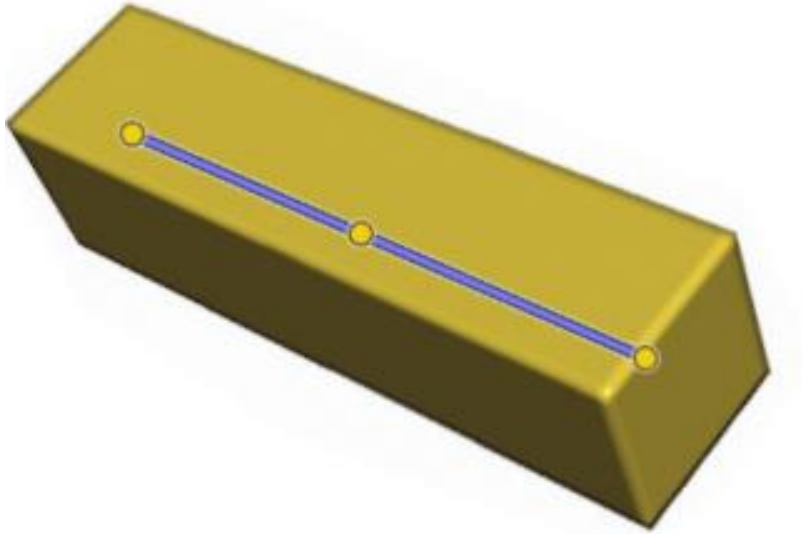
Twist one bone ➡



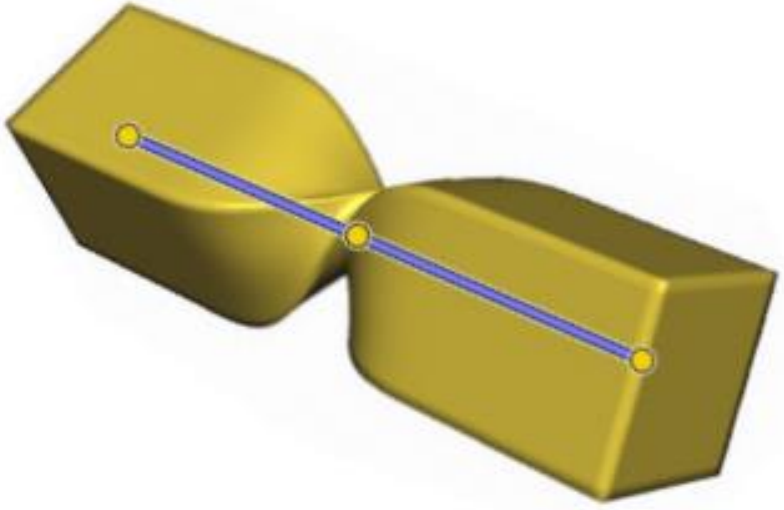
Deformation using LBS

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

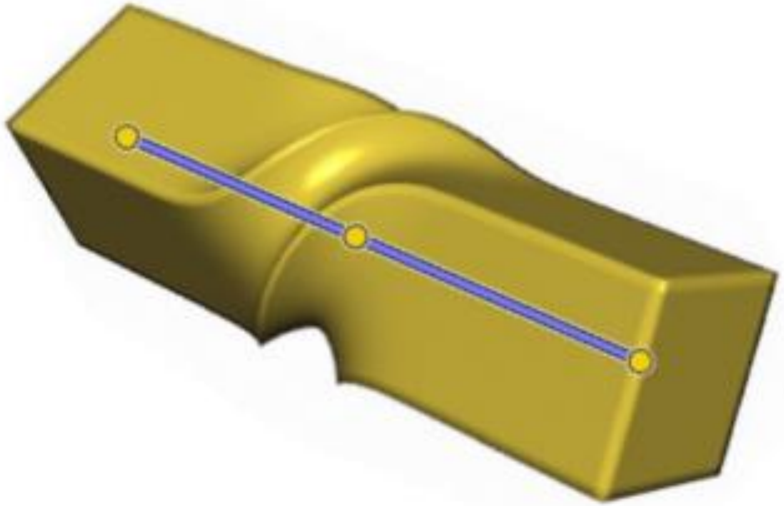
Alternative to LBS: **D**ual **Q**uaternion **S**kinning



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 ε & its arithmetic rule $\varepsilon^2 = 0$ (cf. imaginary unit i)
  - Dual number is sum of primal & dual components:

$$\hat{a} \coloneqq a_0 + \varepsilon a_\varepsilon$$

$$a_0, a_\varepsilon \in \mathbb{R}$$

    - Dual conjugate: $\bar{\hat{a}} = \overline{a_0 + \varepsilon a_\varepsilon} = a_0 - \varepsilon a_\varepsilon$

- Dual quaternion
  - Quaternion whose elements are dual numbers
  - Can be written using two quaternions

$$\hat{\mathbf{q}} \coloneqq \mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon$$

    - Dual conjugate: $\bar{\hat{\mathbf{q}}} = \overline{\mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon} = \mathbf{q}_0 - \varepsilon \mathbf{q}_\varepsilon$
    - Quaternion conjugate: $\hat{\mathbf{q}}^* = (\mathbf{q}_0 + \varepsilon \mathbf{q}_\varepsilon)^* = \mathbf{q}_0^* + \varepsilon \mathbf{q}_\varepsilon^*$

Geometric Skinning with Approximate Dual Quaternion Blending [Kavan TOG08]

Arithmetic rules for dual number/quaternion

- For dual number $\hat{a} = a_0 + \varepsilon a_\varepsilon$:
  - Reciprocal $\quad \frac{1}{\hat{a}} = \frac{1}{a_0} - \varepsilon \frac{a_\varepsilon}{a_0^2}$
  - Square root $\quad \sqrt{\hat{a}} = \sqrt{a_0} + \varepsilon \frac{a_\varepsilon}{2\sqrt{a_0}}$
  - Trigonometric $\quad \sin \hat{a} = \sin a_0 + \varepsilon a_\varepsilon \cos a_0$
    $\cos \hat{a} = \cos a_0 - \varepsilon a_\varepsilon \sin a_0$

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

    Dot product as 4D vectors
  - Inverse $\quad \hat{\mathbf{q}}^{-1} = \frac{\hat{\mathbf{q}}^*}{\|\hat{\mathbf{q}}\|^2}$
  - Unit dual quaternion satisfies $\|\hat{\mathbf{q}}\| = 1$
    - $\Leftrightarrow \|\mathbf{q}_0\| = 1 \;\&\; \langle \mathbf{q}_0, \mathbf{q}_\varepsilon \rangle = 0$

Geometric Skinning with Approximate Dual Quaternion Blending [Kavan TOG08]

Rigid transformation using dual quaternion

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

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

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

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

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

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

Geometric Skinning with Approximate Dual Quaternion Blending [Kavan TOG08]

Rigid transformation using dual quaternion

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

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

$$\left((0+\vec{\mathbf{t}})\mathbf{q}_0\right)^* = \mathbf{q}_0^*(0+\vec{\mathbf{t}})^* = -\mathbf{q}_0^*\vec{\mathbf{t}}$$

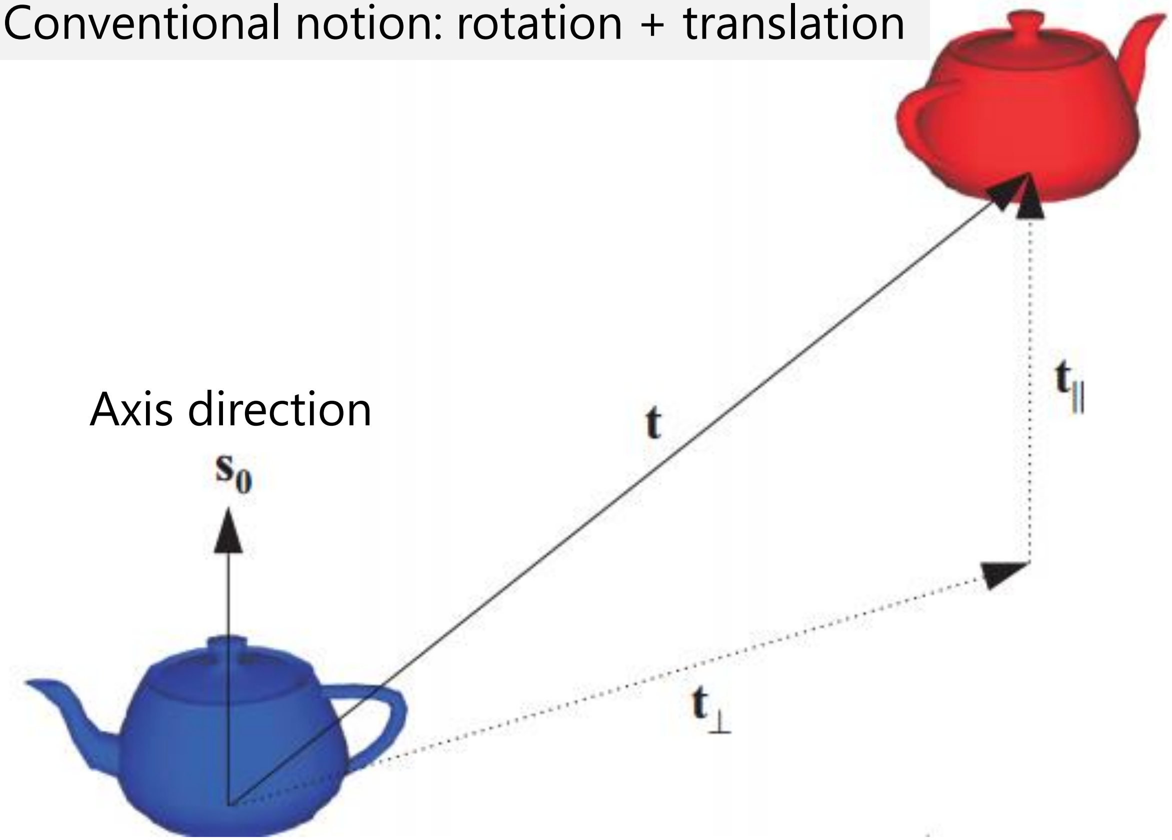
$$\|\mathbf{q}_0\|^2 = 1$$

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

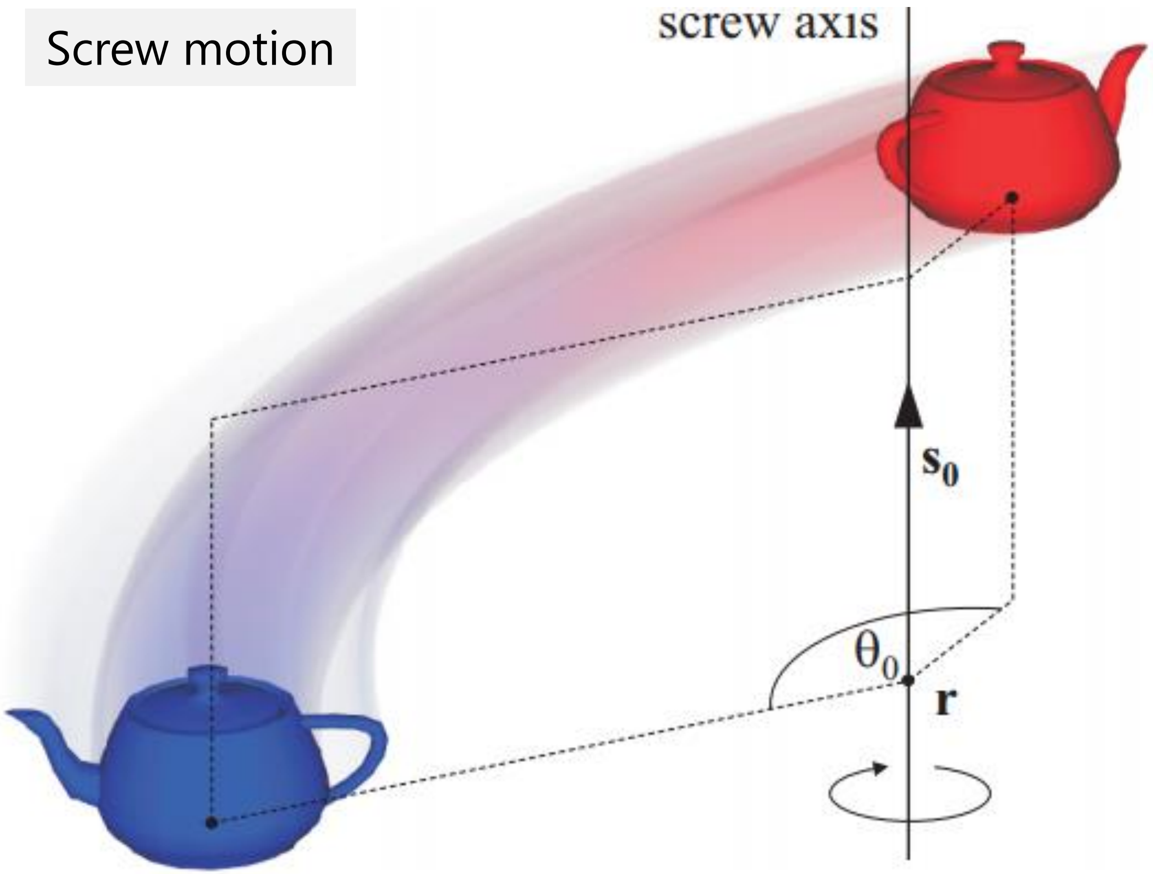
Rigid transformation as “screw motion”

Conventional notion: rotation + translation

Axis direction



Screw motion



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

Geometric Skinning with Approximate Dual Quaternion Blending [Kavan TOG08]

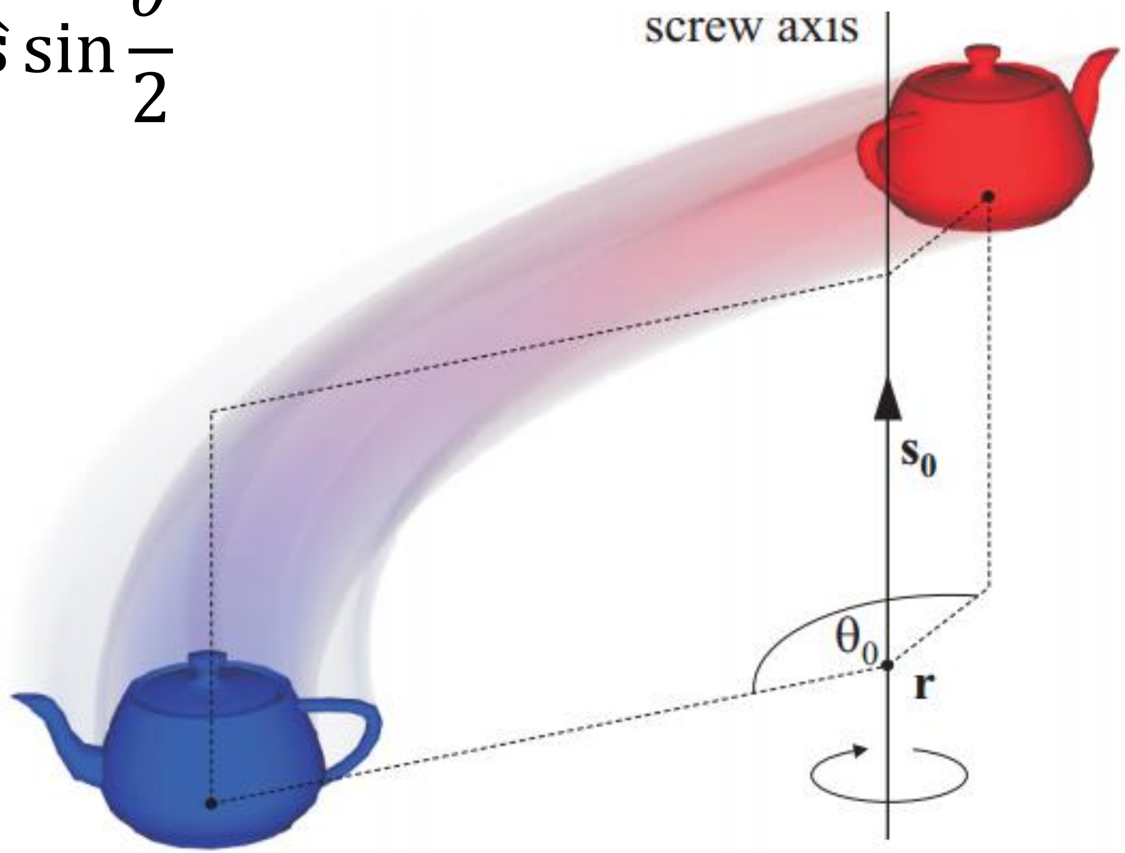
Screw motion & dual quaternion

- 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
  - $\hat{\mathbf{s}} = \overrightarrow{\mathbf{s}_0} + \varepsilon\overrightarrow{\mathbf{s}_\varepsilon}$ — $\overrightarrow{\mathbf{s}_0}, \overrightarrow{\mathbf{s}_\varepsilon}$: unit 3D vector

- Geometric meaning
  - $\overrightarrow{\mathbf{s}_0}$: direction of rotation axis
  - θ_0 : amount of rotation
  - θ_ε : amount of translation parallel to $\overrightarrow{\mathbf{s}_0}$
  - $\overrightarrow{\mathbf{s}_\varepsilon}$: when rotation axis passes through $\vec{\mathbf{r}}$, it satisfies $\overrightarrow{\mathbf{s}_\varepsilon} = \vec{\mathbf{r}} \times \overrightarrow{\mathbf{s}_0}$

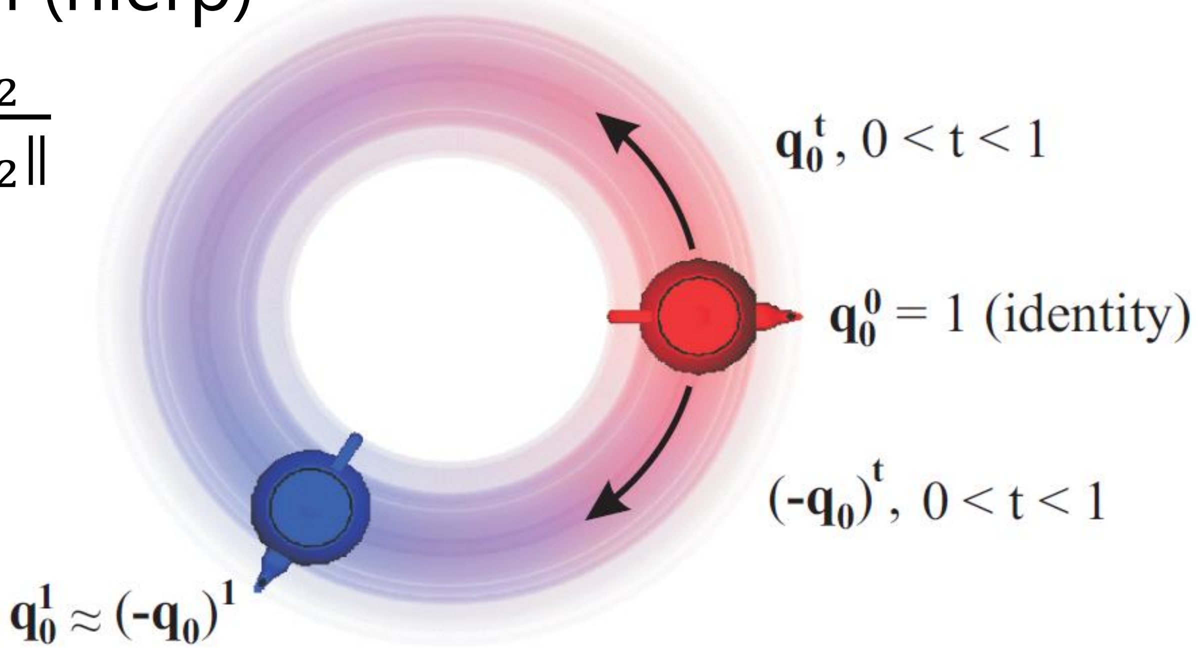


Geometric Skinning with Approximate Dual Quaternion Blending [Kavan TOG08]

Interpolating two rigid transformations

- Linear interpolation + normalization (nlerp)

$$\mathrm{nlerp}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, t) \coloneqq \frac{(1-t)\hat{\mathbf{q}}_1 + t\hat{\mathbf{q}}_2}{\|(1-t)\hat{\mathbf{q}}_1 + t\hat{\mathbf{q}}_2\|}$$



- 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

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

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

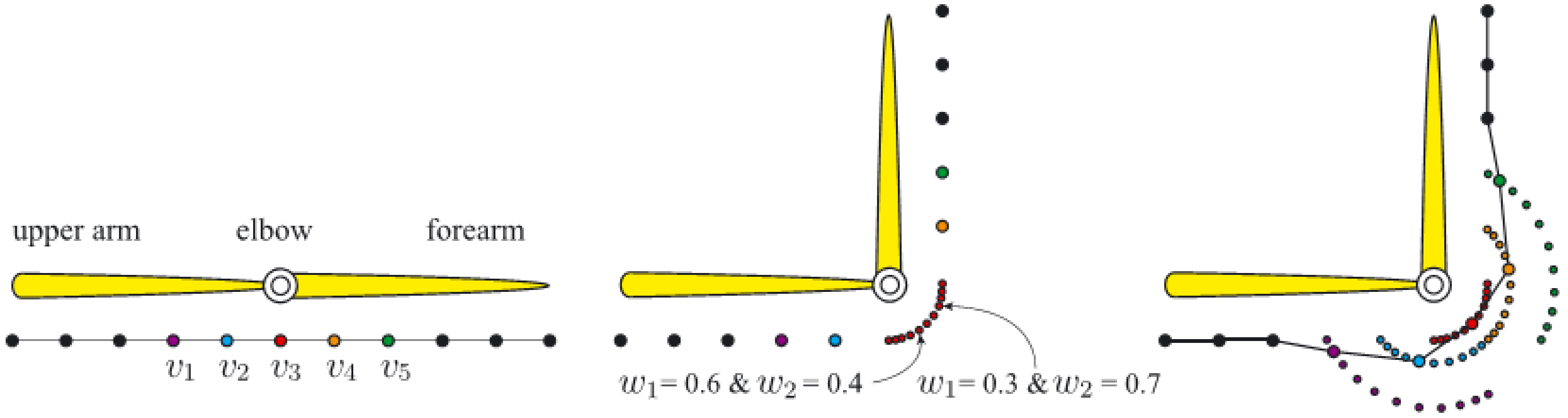
197.4 FPS

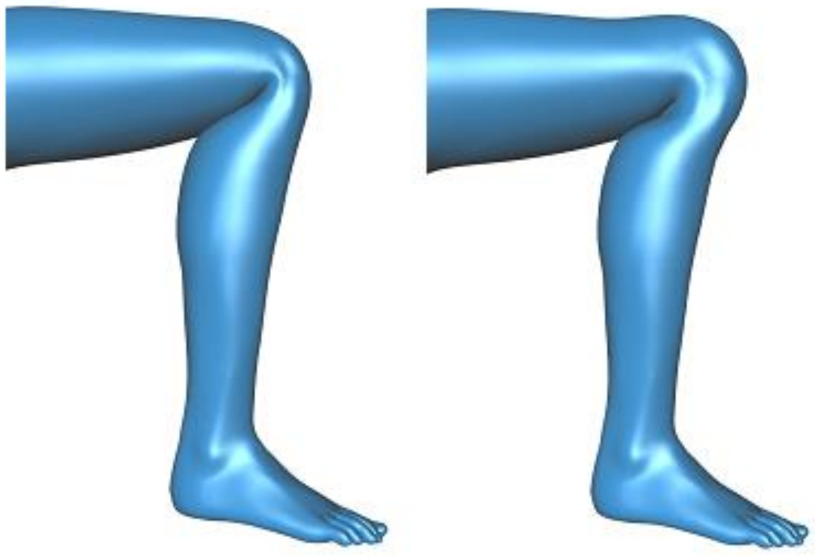
122 FPS

Geometric Skinning with Approximate Dual Quaternion Blending [Kavan TOG08]

Artifact of DQS: "bulging" effect

- Produces ball-like shape around the joint when bended

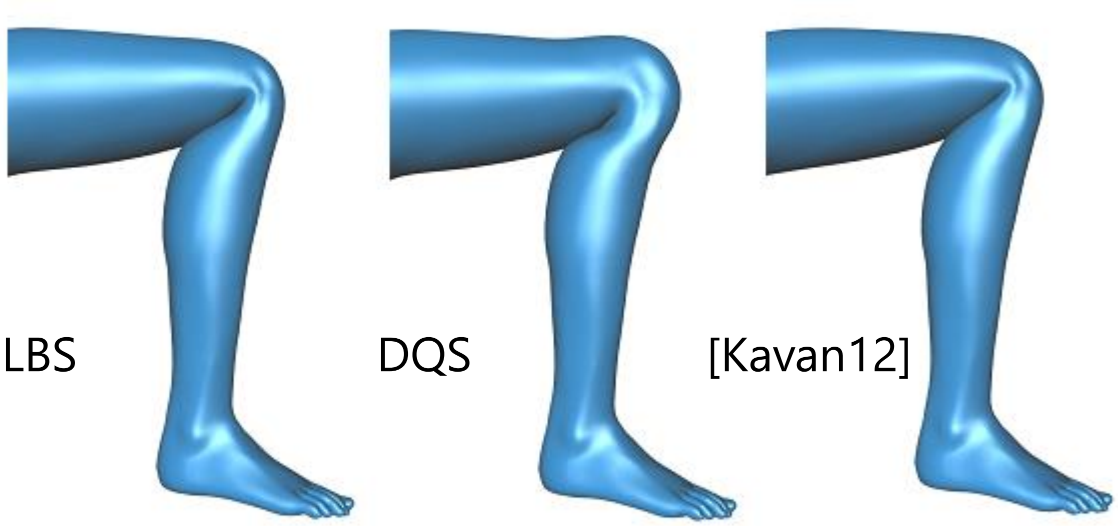




LBS DQS

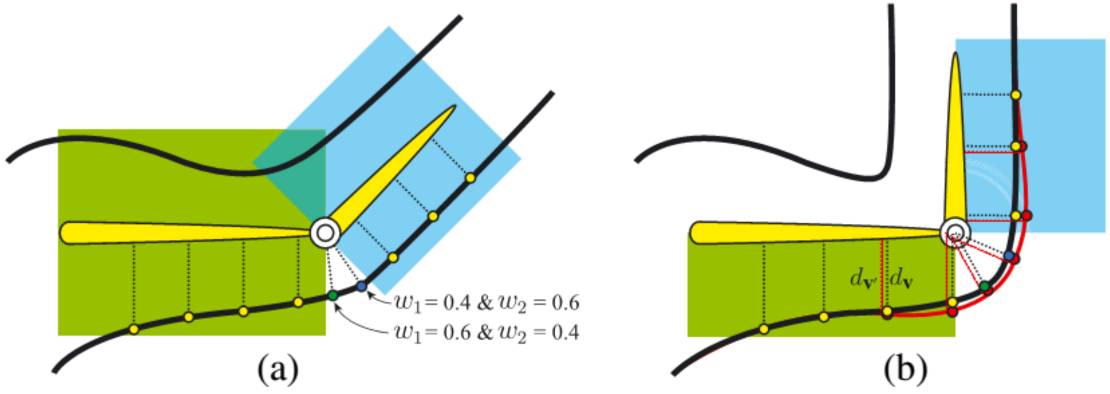
Elasticity-Inspired Deformers for Character Articulation [Kavan SIGGRAPHAsia12]
Bulging-free dual quaternion skinning [Kim CASA14]

Overcoming DQS's drawback



LBS DQS [Kavan12]

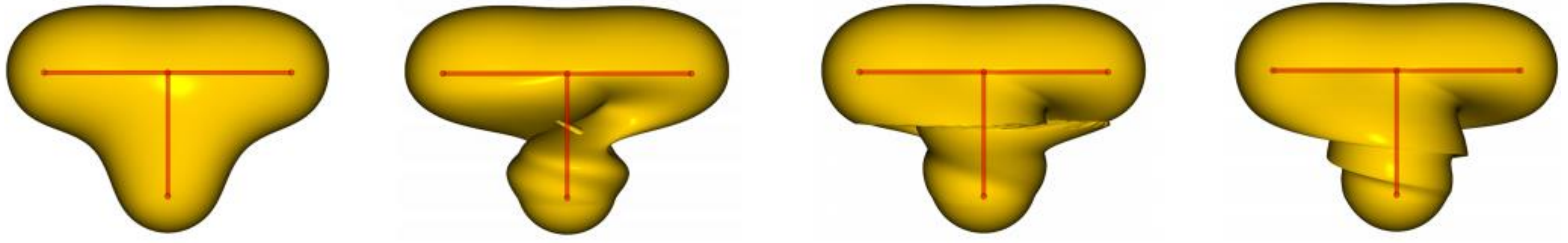
Decompose transformation into bend & twist, interpolate them separately [Kavan12]



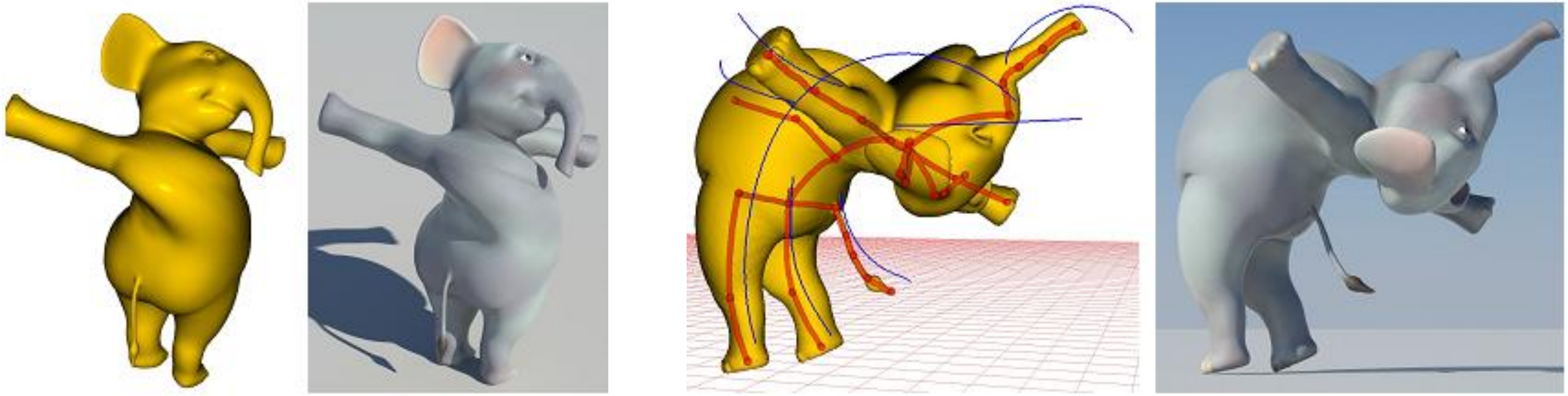
After deforming using DQS, offset vertices along normals [Kim14]

Elasticity-Inspired Deformers for Character Articulation [Kavan SIGGRAPHAsia12]
Bulging-free dual quaternion skinning [Kim CASA14]

Limitation of DQS: Cannot represent rotation by more than 360°



Rest pose | Linear blending | Dual quaternion blending | Differential blending



Differential Blending for Sketch-based Expressive Posing [Oztireli SCA13]

Skinning for avoiding self-intersections

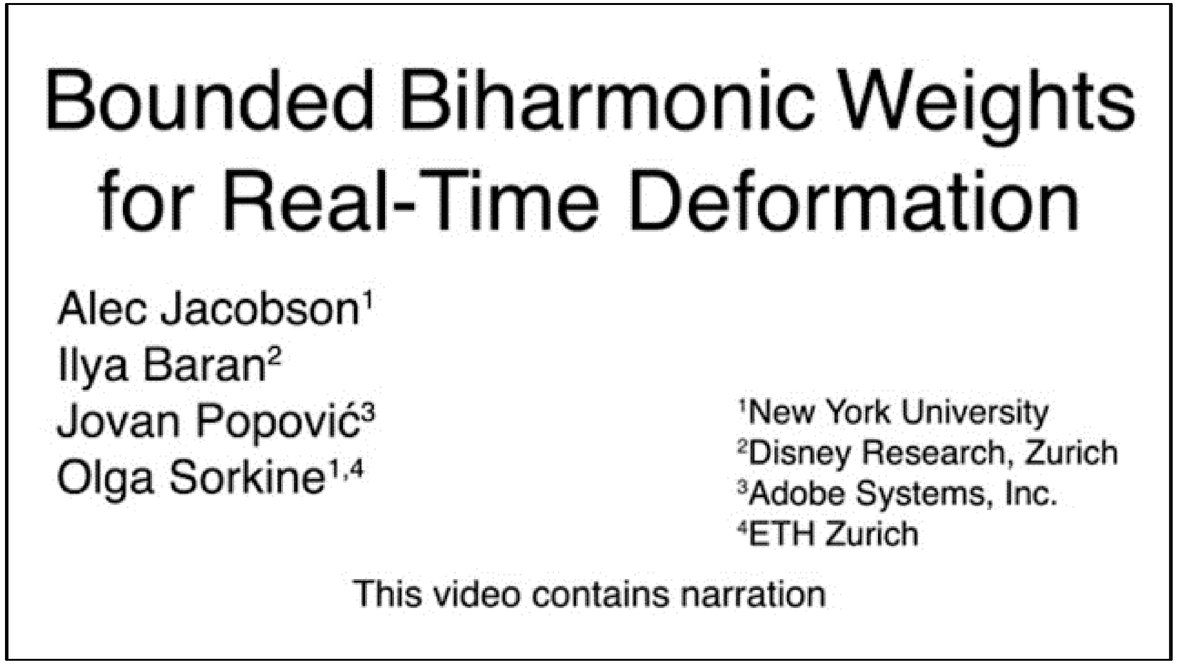
- Make use of implicit functions

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

Implicit Skinning; Real-Time Skin Deformation with Contact Modeling [Vaillant SIGGRAPH13]

Other deformation mechanisms than skinning

Unified point/cage/skeleton handles [Jacobson 11]



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

BlendShape

https://www.youtube.com/watch?v=BFPAIU8hwQ4

Bounded Biharmonic Weights for Real-Time Deformation [Jacobson SIGGRAPH11]

References

- http://en.wikipedia.org/wiki/Motion_capture
- http://skinning.org/
- http://mukai-lab.org/category/library/legacy