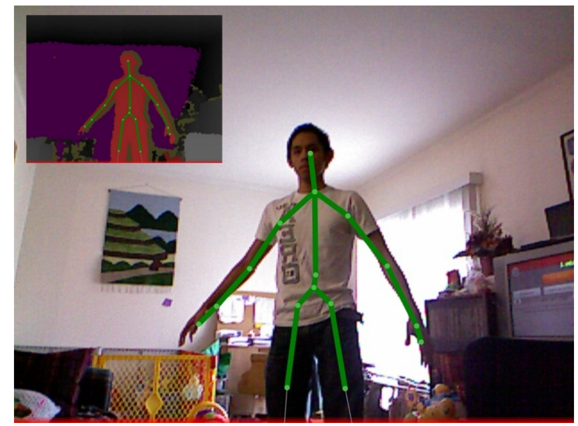
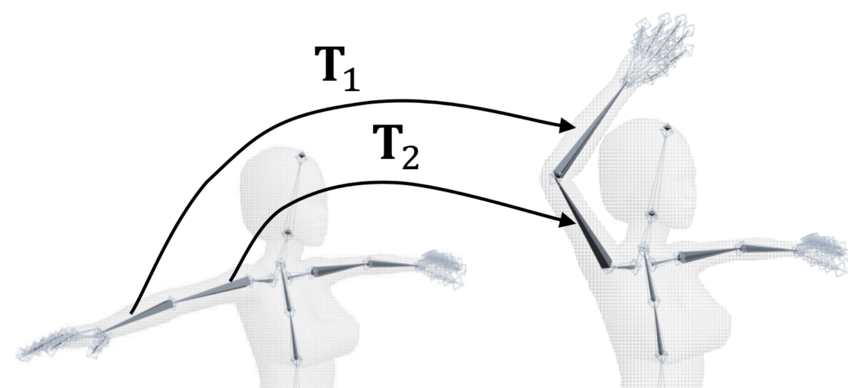
Skeleton-based deformations





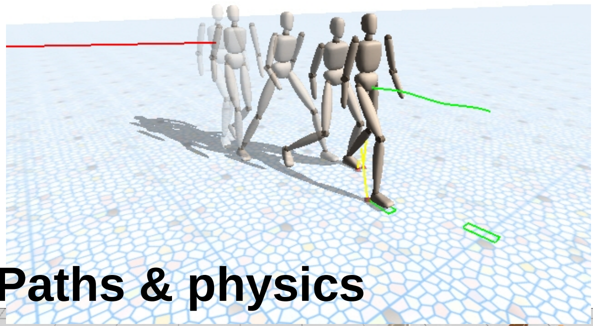
Why ?

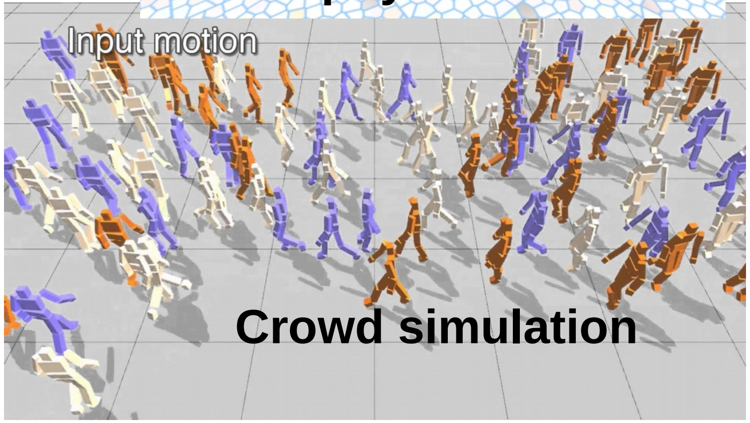
Why ?

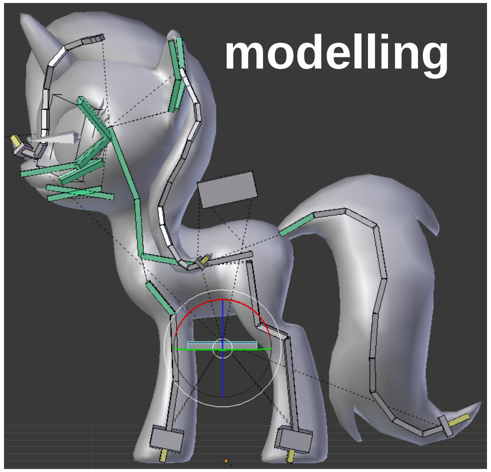
Where ?

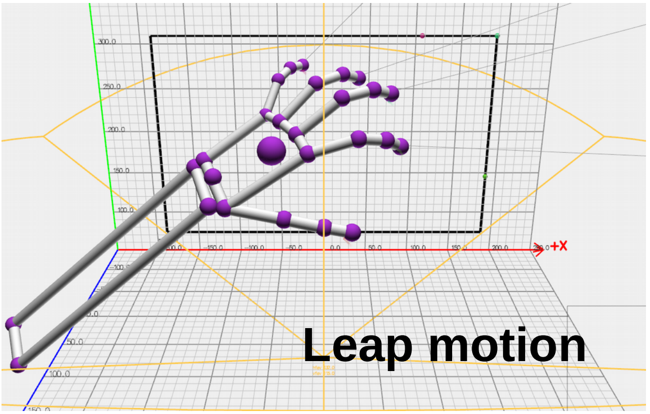
Halo3

Bolt





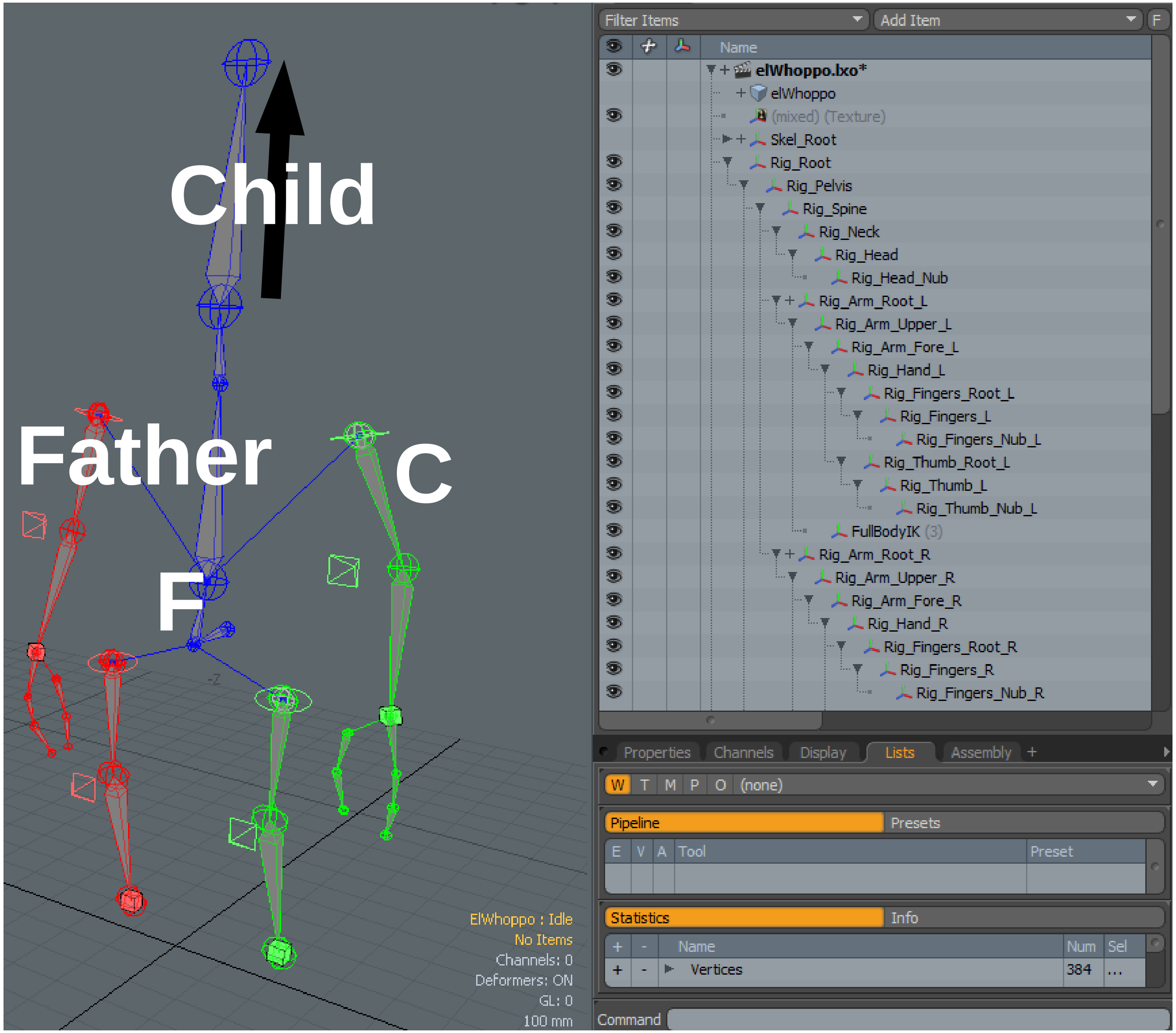




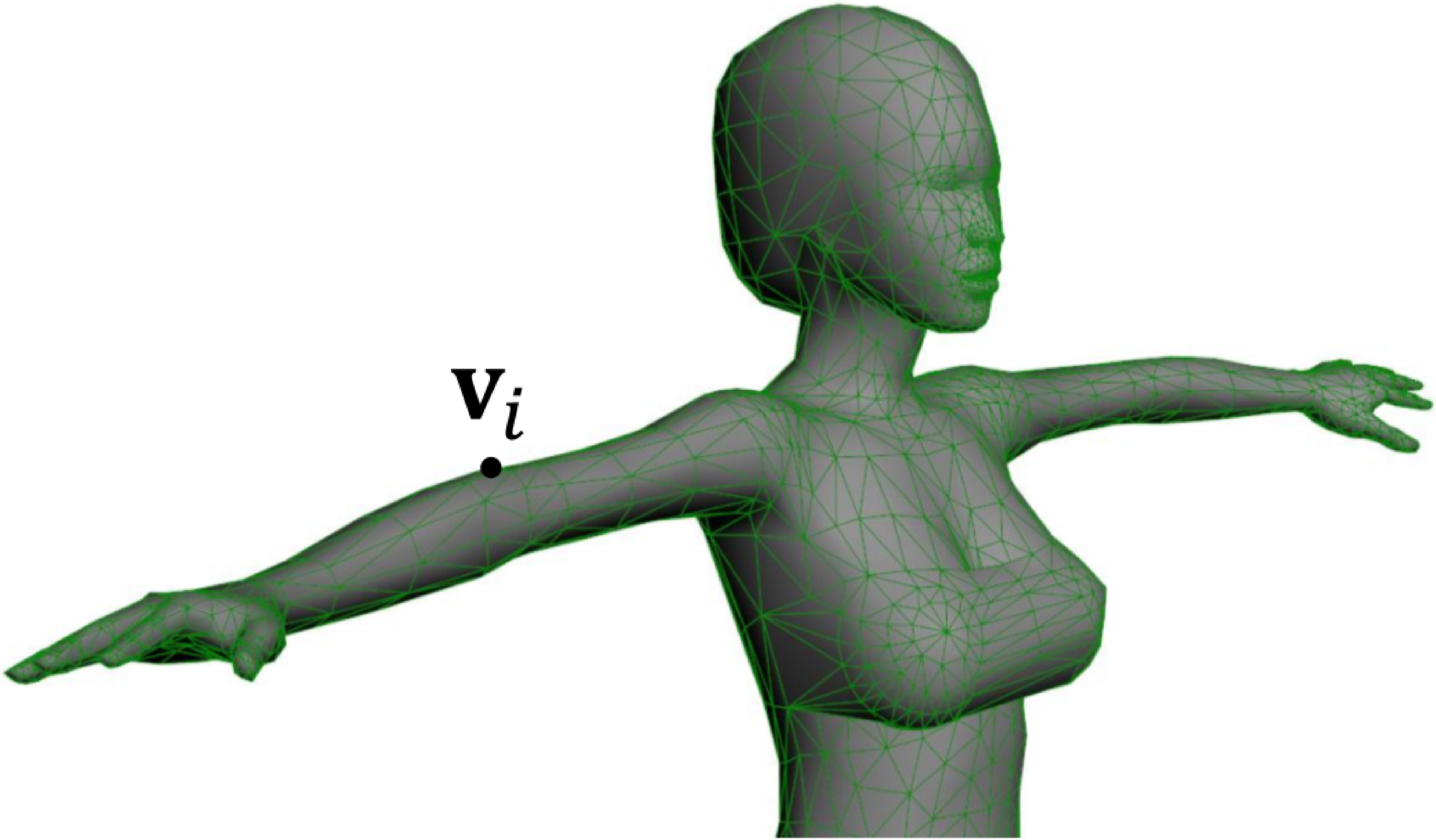
problematics

- What is a skeleton ?
- How to define what is its transformation ?
- How to transfer its deformation to the mesh ?
- How to manipulate it easily ?
- ...

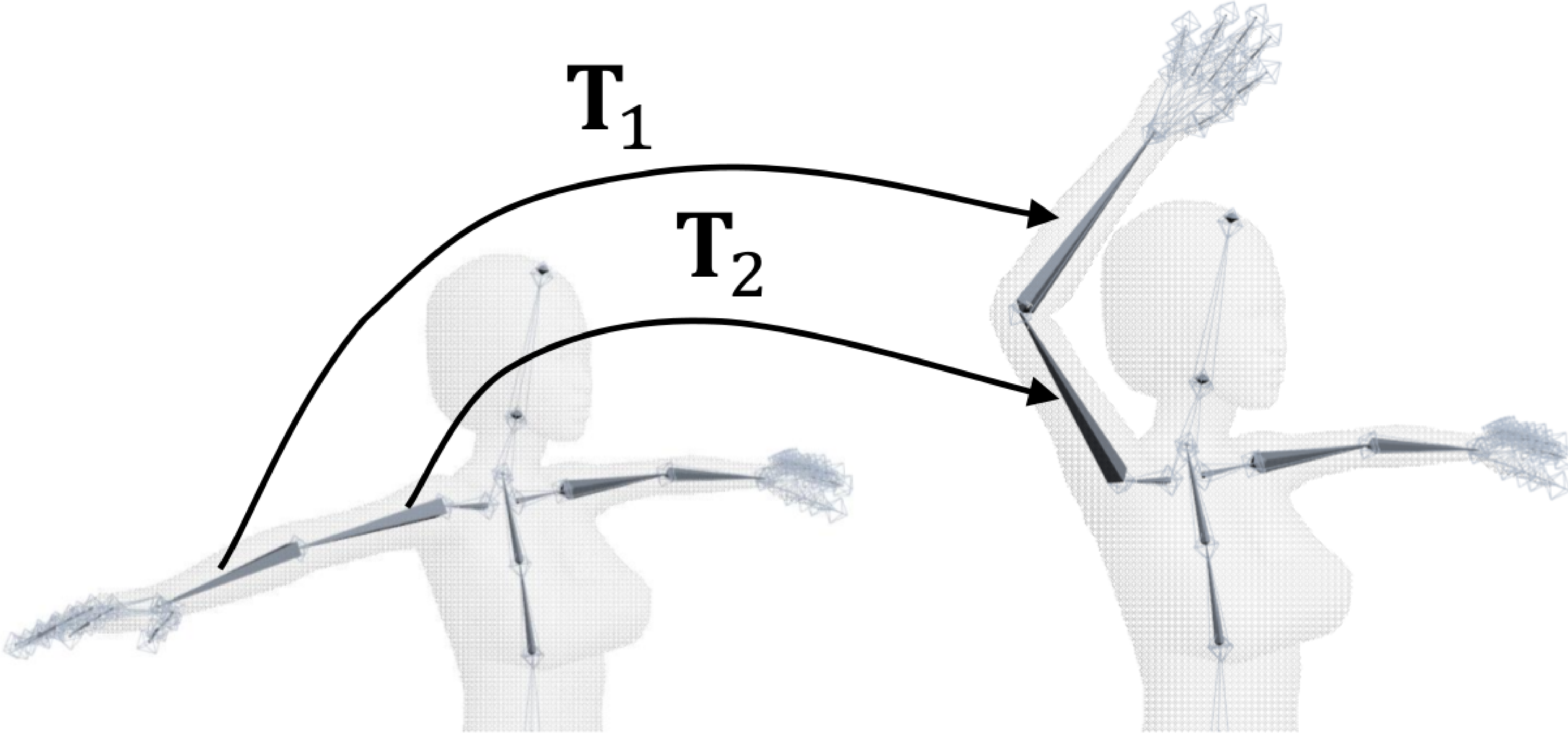
Skeleton structure



Linear blend skinning

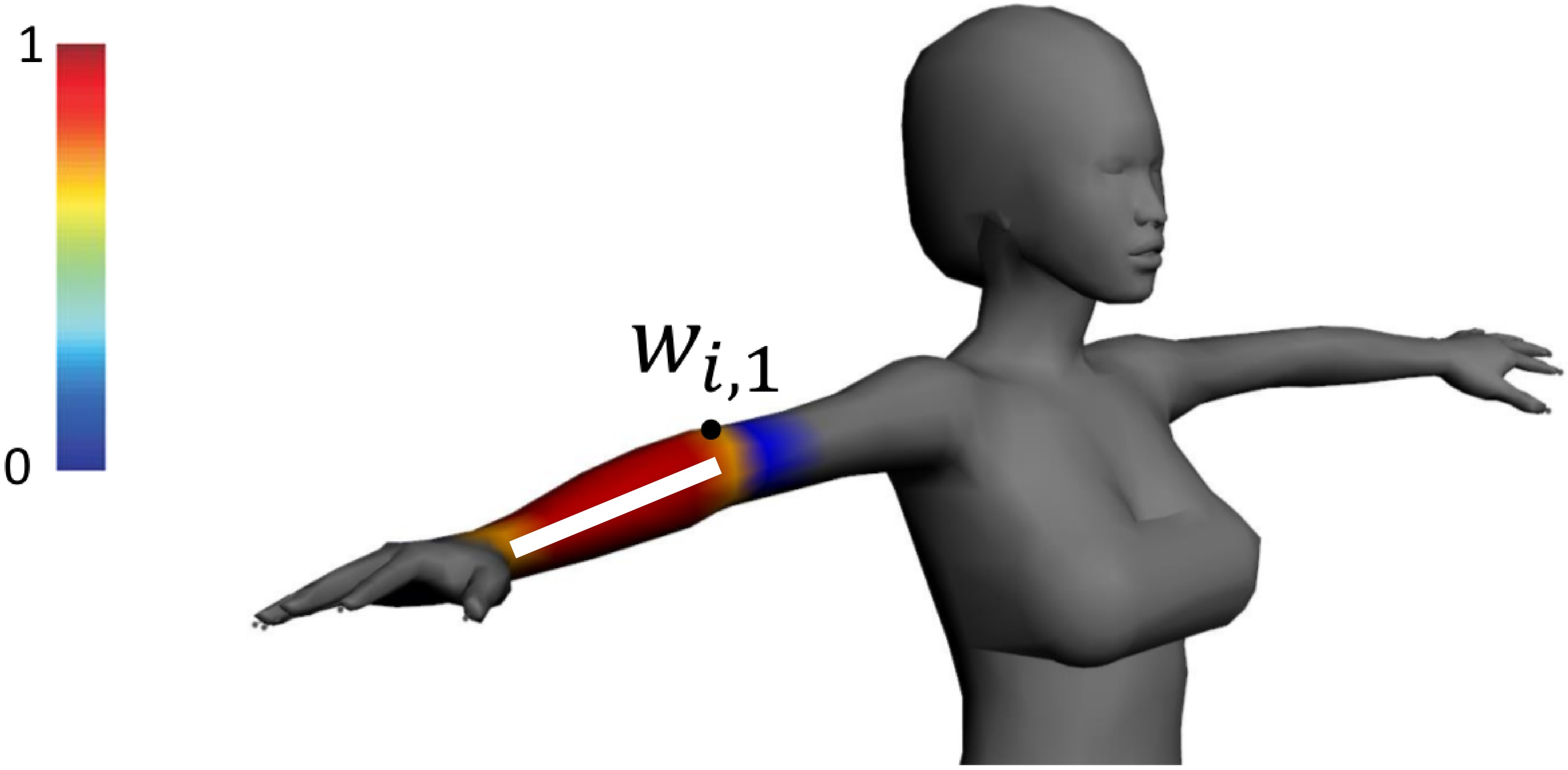


Linear blend skinning

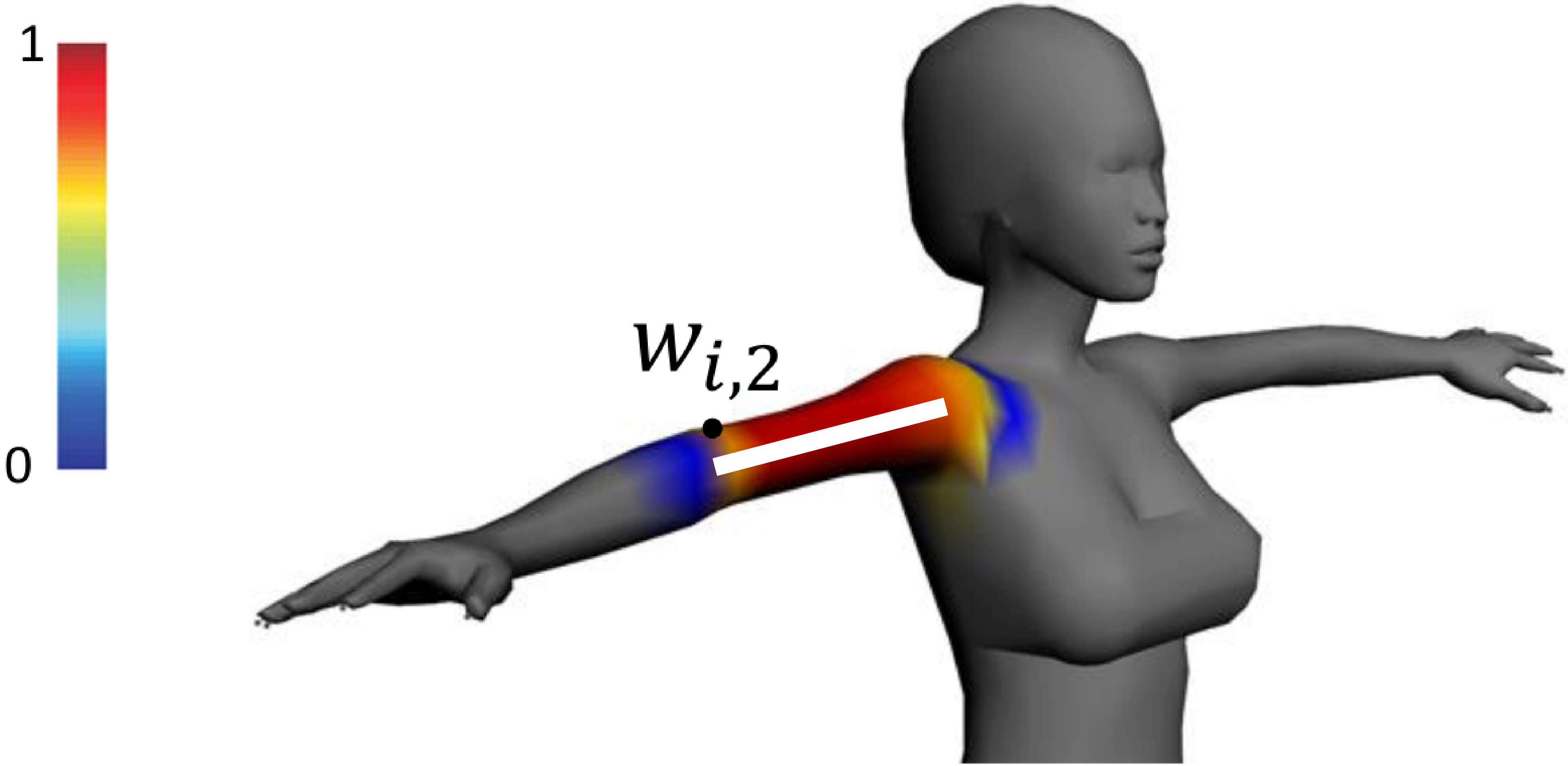


We need to define the influence of the bones onto the mesh vertices

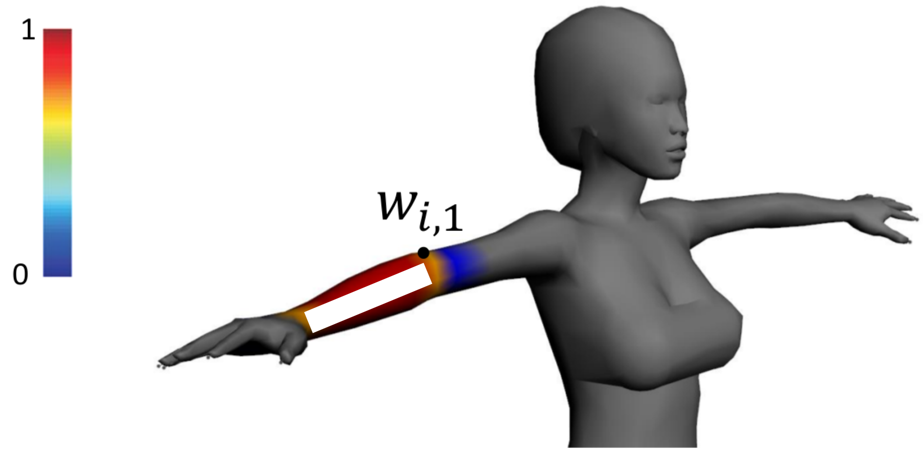
Linear blend skinning

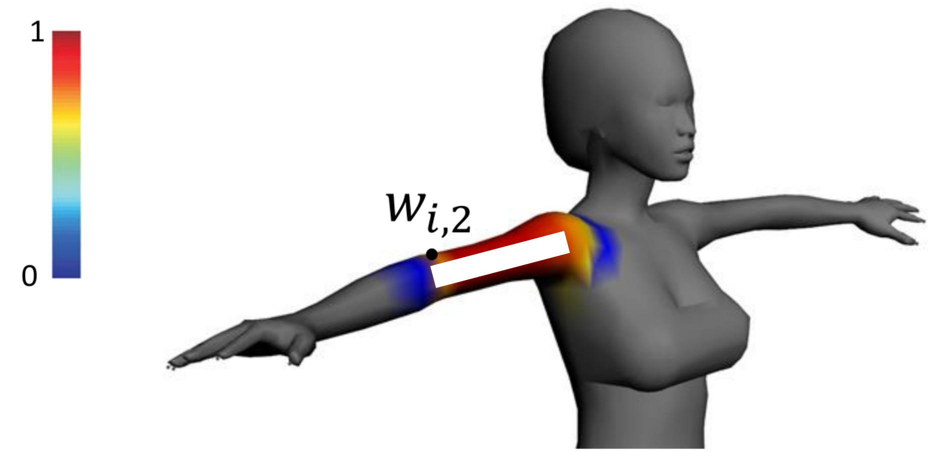


Linear blend skinning



Linear blend skinning





Weight of bone j over vertex i

Vertex i

$$f : v_i \rightarrow \sum_j w_{ij}(R_j . v_i + T_j)$$

Deformation function

Bones influencing vertex i

Rotation of bone j

Translation of bone j

Skinning weights properties

$$f : v_i \rightarrow \sum_j w_{ij} \left(R_j . v_i + T_j \right)$$

- Positivity $\quad w_{ij} \geqslant 0$
- Affinity $\quad \sum_j w_{ij} = 1$

Why ?

Skinning weights properties

$$f : v_i \rightarrow \sum_j w_{ij}(R_j . v_i + T_j)$$

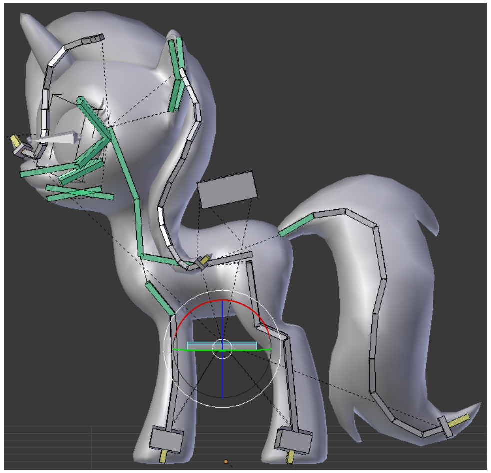
- Sparsity : only a few $w_{ij} \succ 0$

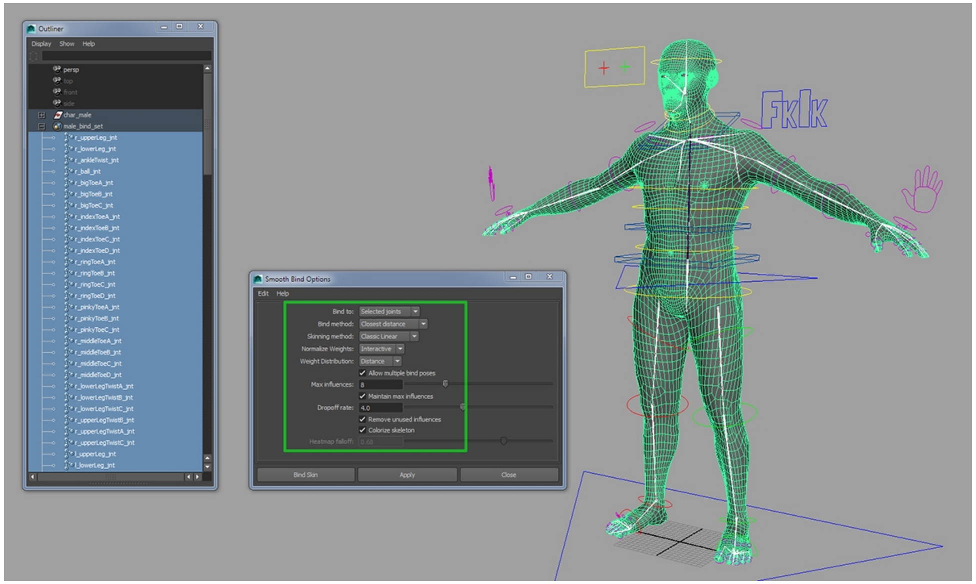
Why ?

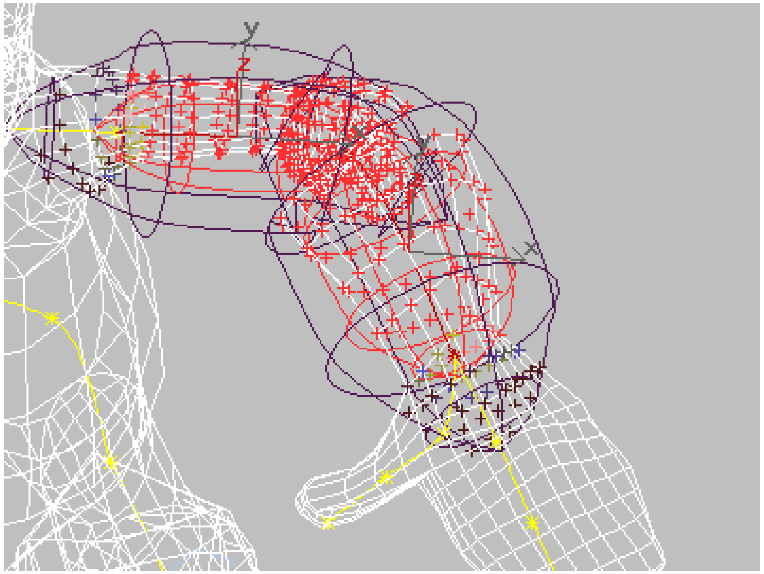
Skinning in modelling tools

- Blender
- Maya
- 3DSMax

DEMO

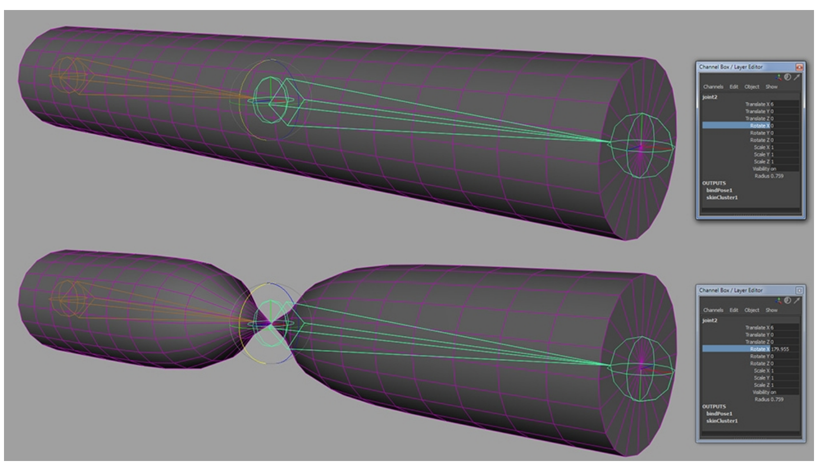






LBS alternatives

$$\text{LBS :} \quad f : v_i \rightarrow \sum_j w_{ij} (R_j . v_i + T_j)$$



180 degrees → « candy wrapper » effect

Alternatives :

- Dual quaternion skinning (DQS)
- Spline skinning
- Differential blending

Blending transformations

$$f : v_i \rightarrow \sum_j w_{ij} \left(R_j . v_i + T_j \right)$$

Blend « the transformed vertices »

Blending transformations

$$f : v_i \rightarrow \underbrace{\left(\sum_j w_{ij} R_j\right)}_{?} . v_i + \left(\sum_j w_{ij} T_j\right)$$

Blend « the transformations »

$$1/2\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + 1/2\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Rotation by 0	Rotation by π	Not a rotation

Blending transformations

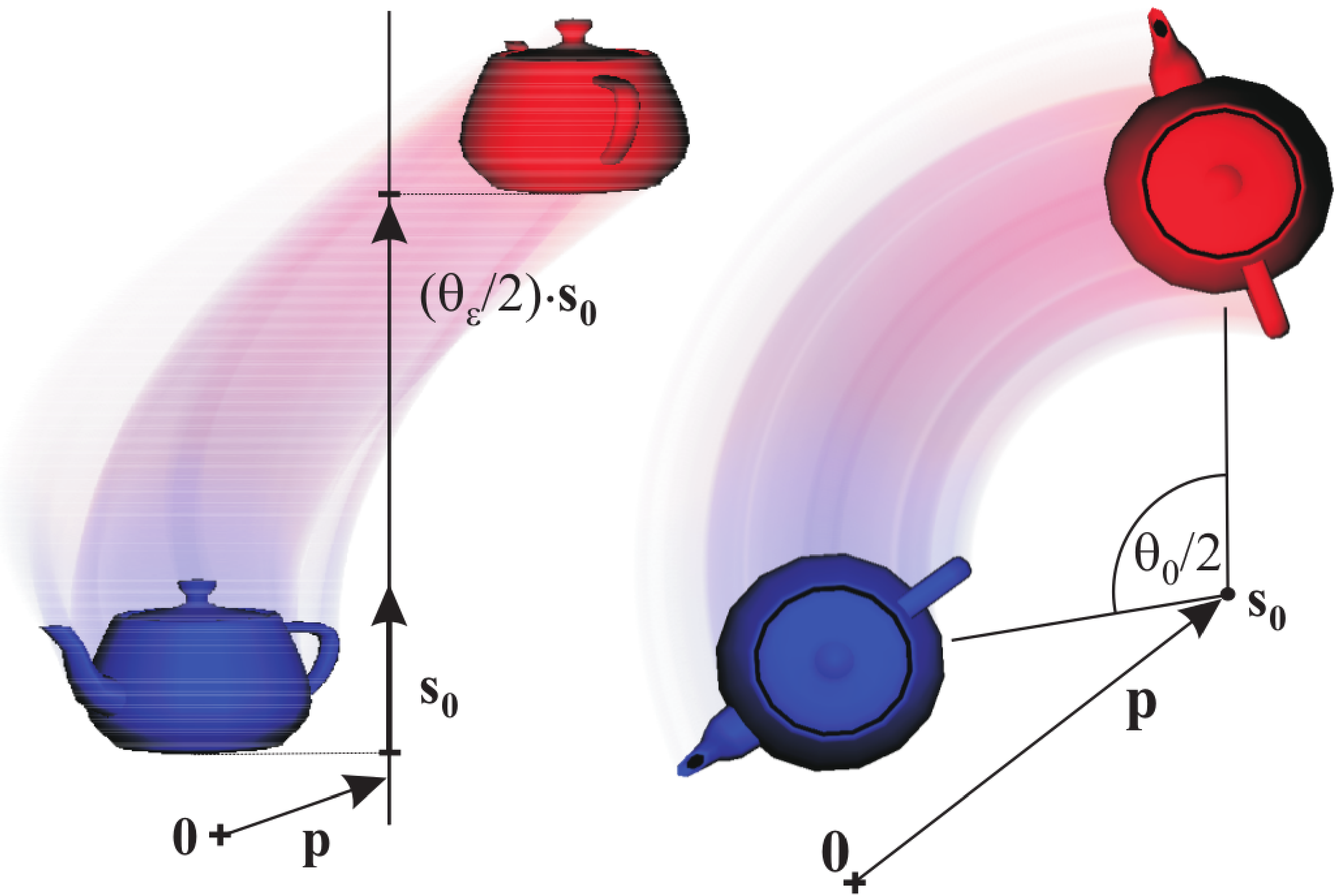
$$f : v_i \rightarrow \underbrace{\left(\sum_j w_{ij} R_j\right)}_{?} . v_i + \left(\sum_j w_{ij} T_j\right)$$

Blend « the transformations »

$$1/2\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad + \quad 1/2\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \text{ « = » } \sqrt{(2)}/2\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Rotation by 0	Rotation by π	Rotation by π/2

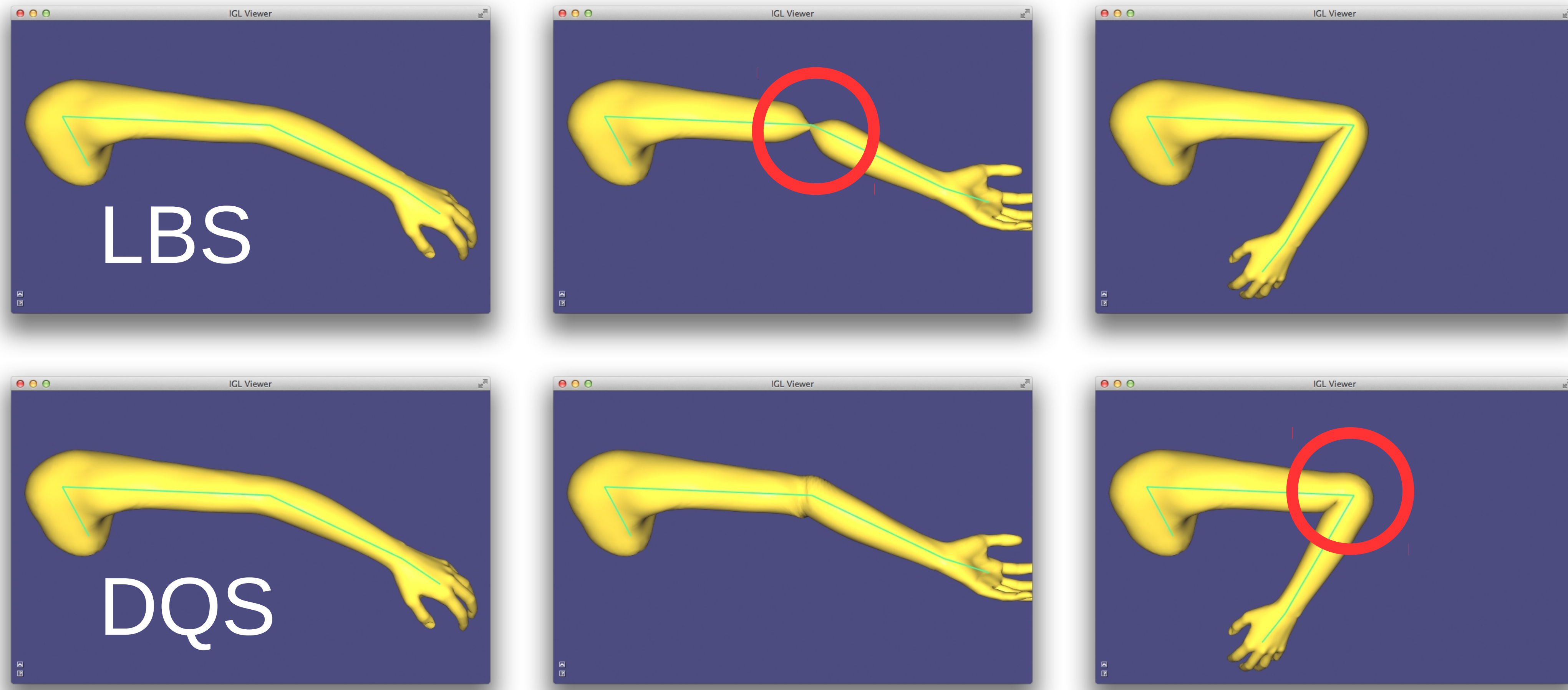
Dual quaternion Skinning



[Kavan et al.] : Skinning with Dual Quaternions

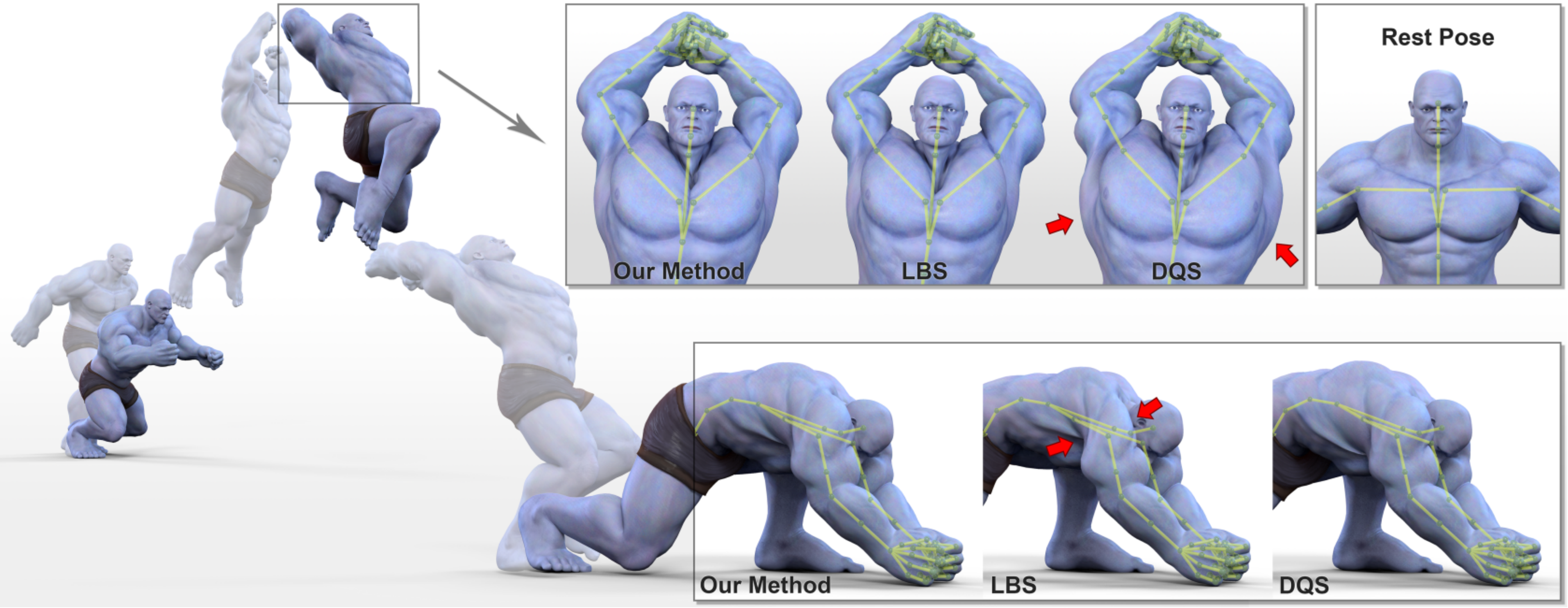
Dual quaternion Skinning

« candy-wrapper »



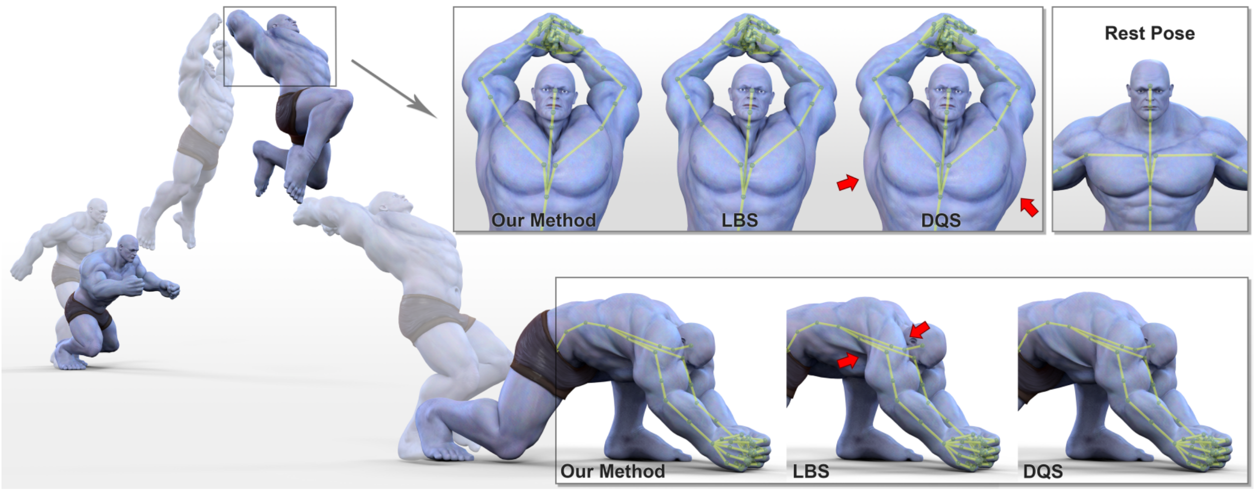
« bulge »

Disney's CoRs



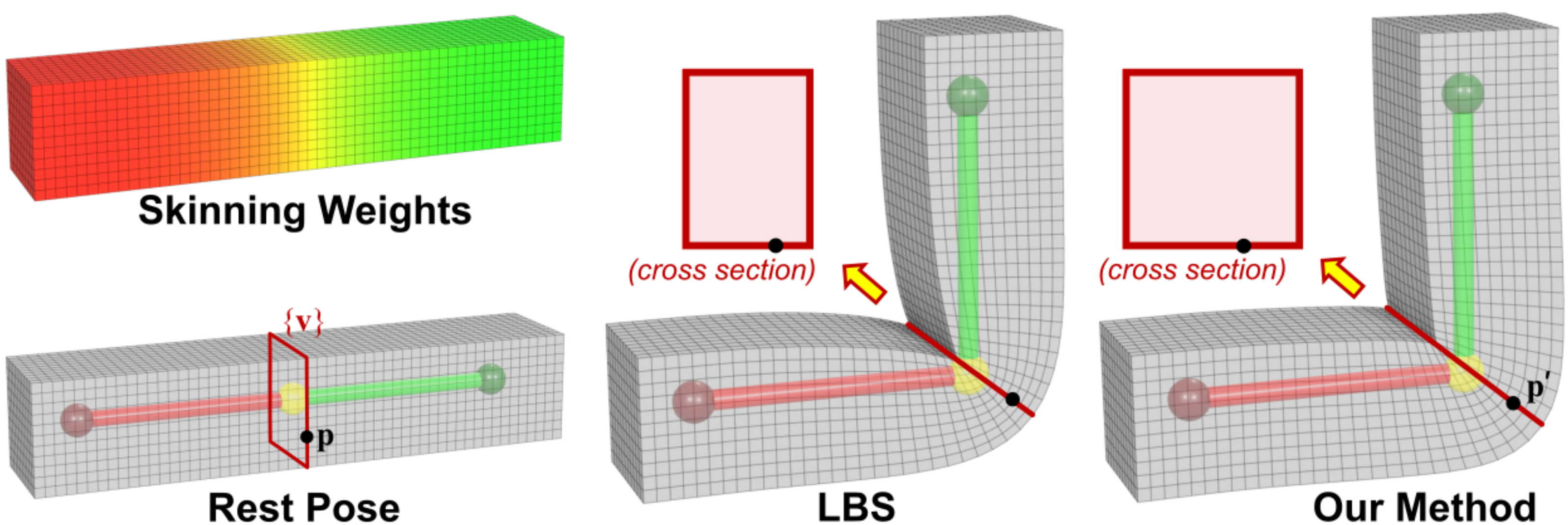
Key ideas

- LBS and DQS have « orthogonal » problems
- DQS is good at blending the rotations
- The bulge effect is due to a non-optimized translation (or a non-optimized center of rotation)



What is a good CoR ?

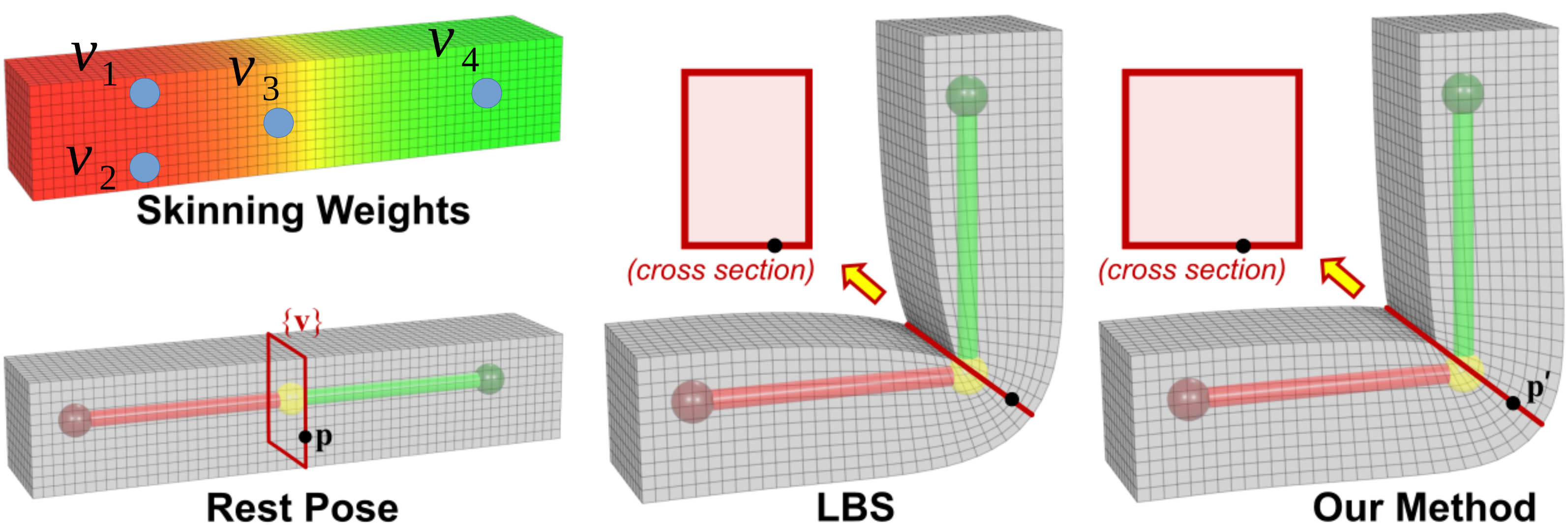
- Vertices with similar weights will have a similar rotation
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.



What is a good CoR ?

- Vertices with similar weights will have a similar rotation
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
- Requires a similarity function between weights

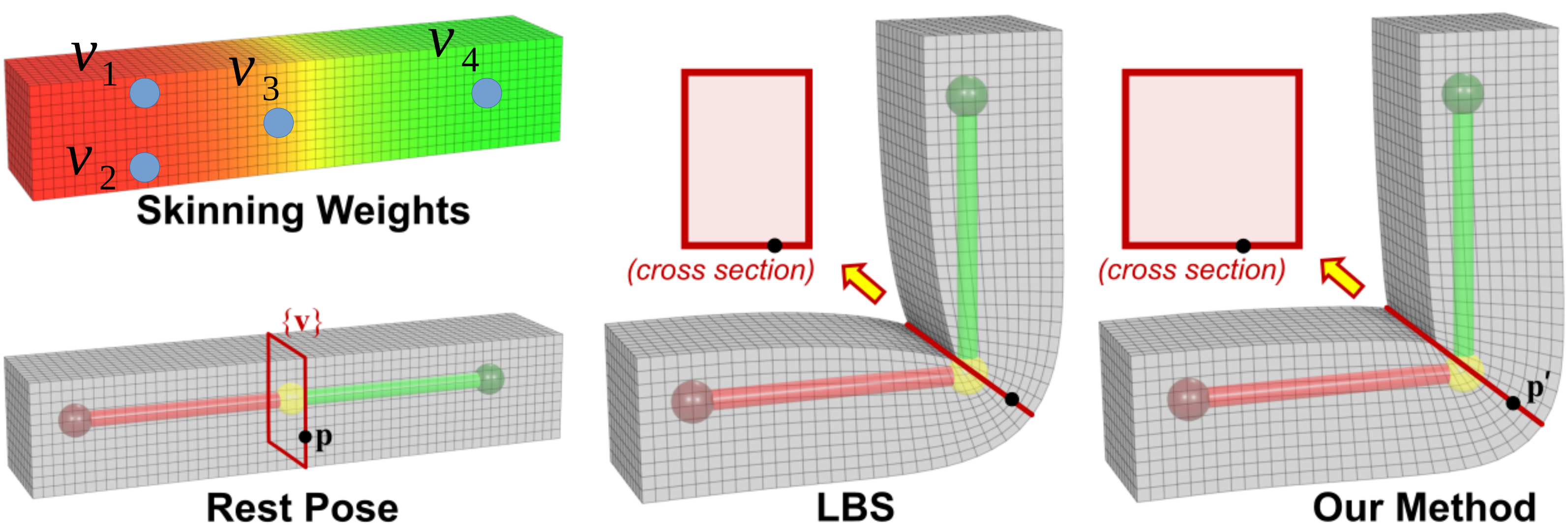
$s(w_1, w_2) = 1$ $s(w_1, w_3) = 0.01$ $s(w_1, w_4) = 0$



What is a good CoR ?

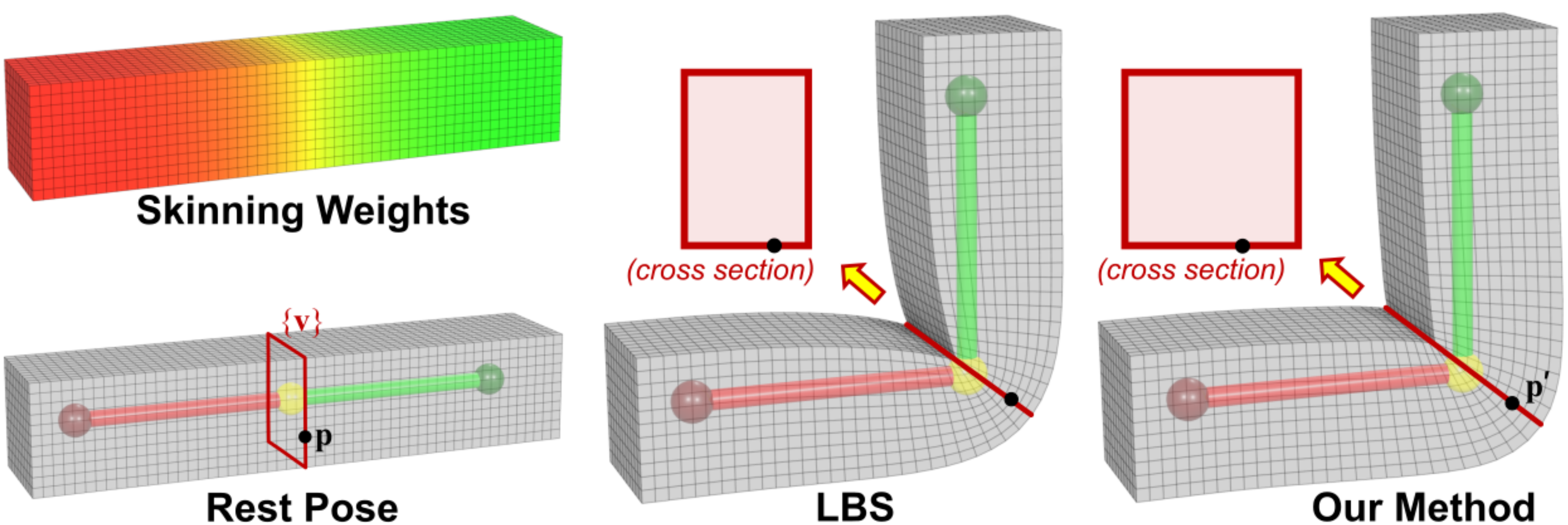
- Vertices with similar weights will have a similar rotation
- They are organized as cross sections orthogonal to the bones, and ideally should be transformed rigidly.
- Requires a similarity function between weights

$$s(\mathbf{w_p}, \mathbf{w_v}) = \sum_{\forall j \neq k} \mathbf{w_{p}}_j \mathbf{w_{p}}_k \mathbf{w_{v}}_j \mathbf{w_{v}}_k \, e^{-\frac{(\mathbf{w_{p}}_j \mathbf{w_{v}}_k - \mathbf{w_{p}}_k \mathbf{w_{v}}_j)^2}{\sigma^2}}$$



Idea

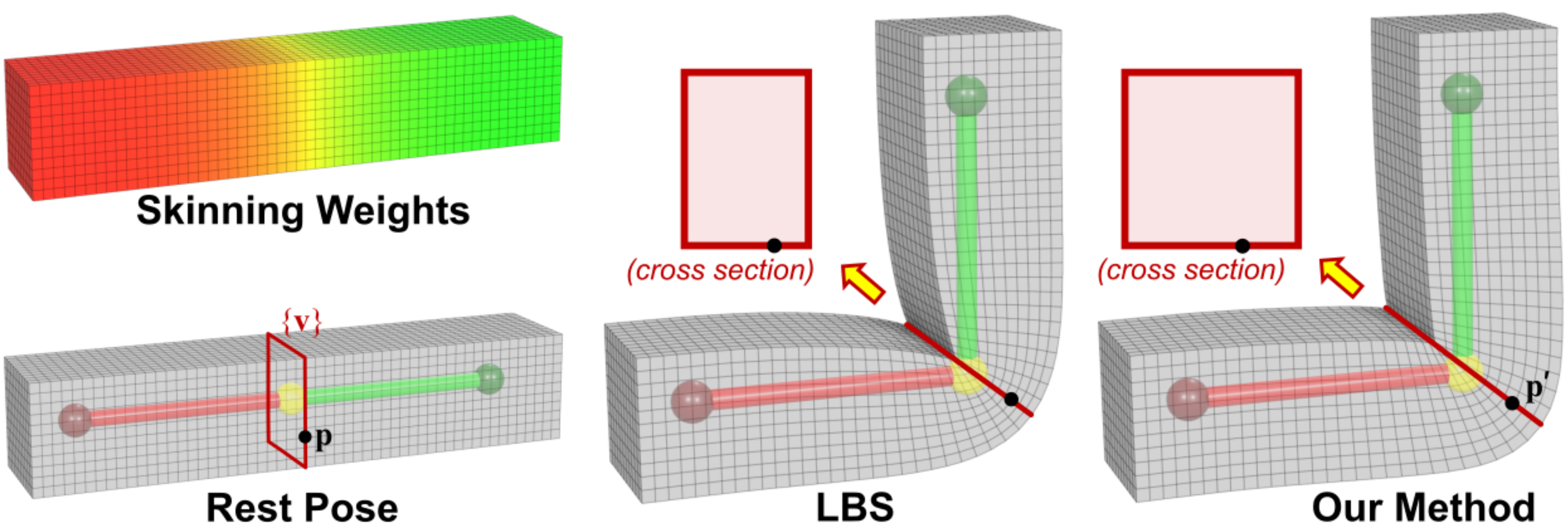
- Consider the LBS transformation of the mesh
- Use the DQS rotation for each vertex
- Optimize per-vertex translation to fit the LBS deformation while enforcing rigid sections.



Idea

$$\mathbf{t_p} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \left\|\mathbf{R_p}\mathbf{v} + \mathbf{t} - \widetilde{\mathbf{v}}\right\|_2^2 \mathrm{d}\mathbf{v}$$

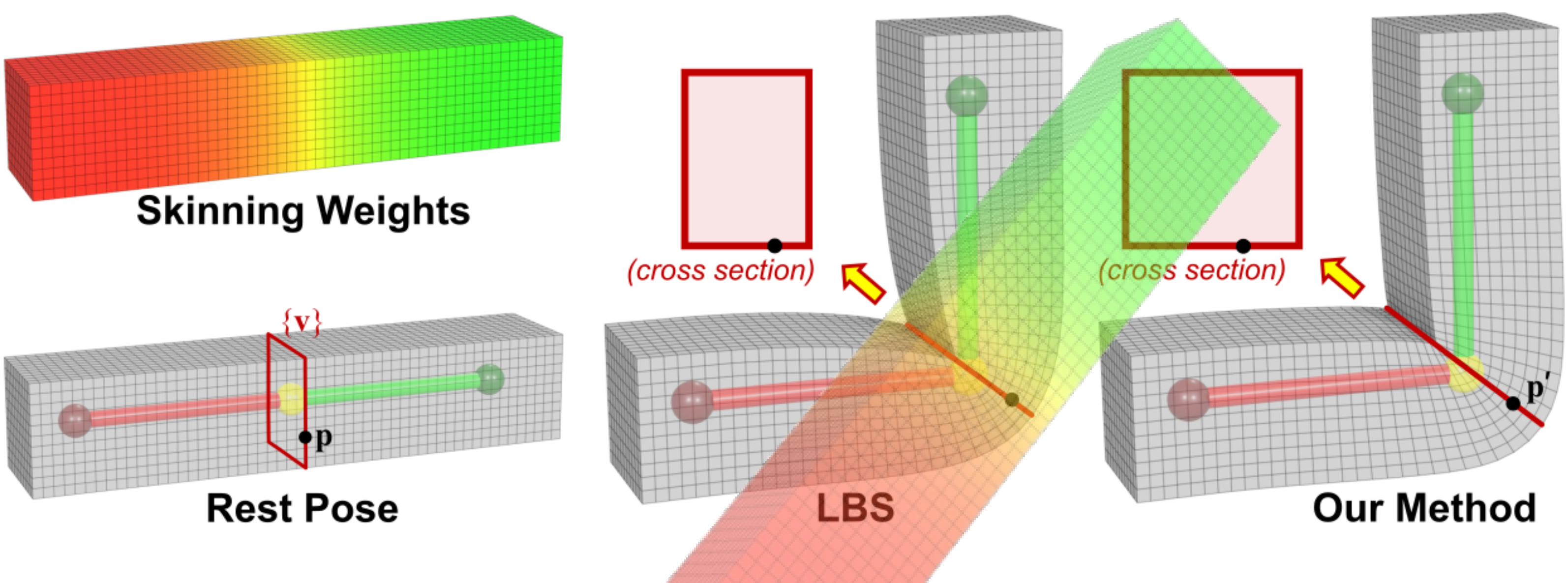
$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j\mathbf{v} + \mathbf{t}_j\right)$$



Idea

$$\mathbf{t}_{\mathbf{p}} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w}_{\mathbf{p}}, \mathbf{w}_{\mathbf{v}}) \left\| \mathbf{R}_{\mathbf{p}}\mathbf{v} + \mathbf{t} - \widetilde{\mathbf{v}} \right\|_2^2 \mathrm{d}\mathbf{v}$$

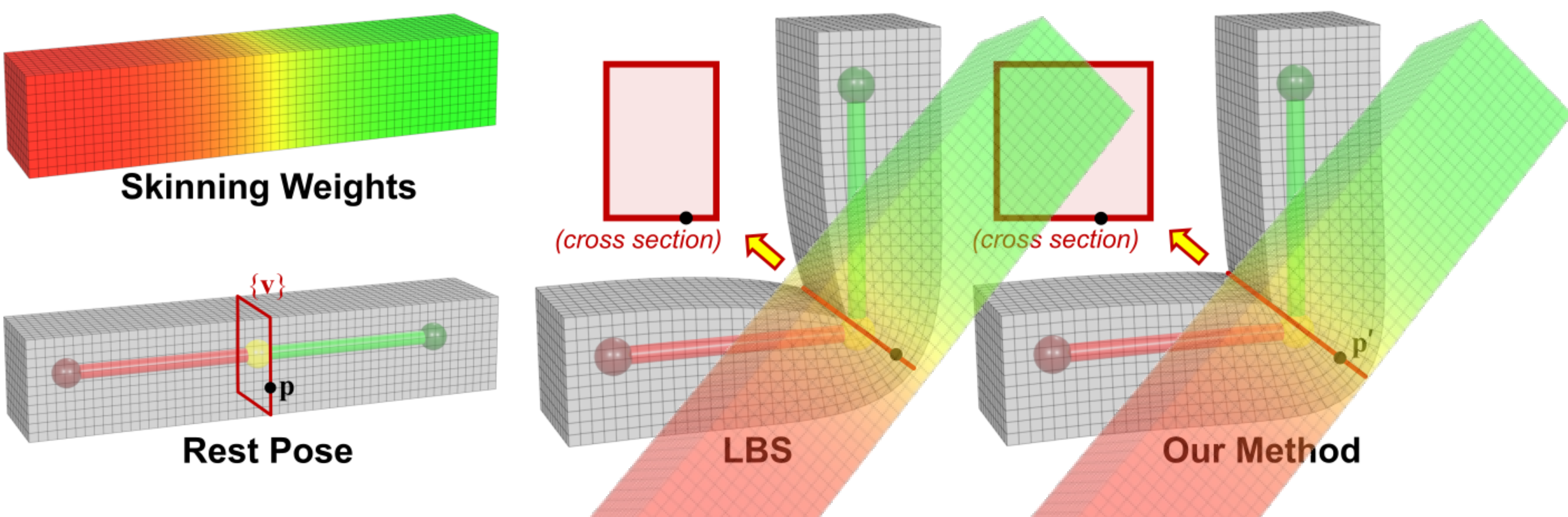
$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \mathbf{v} + \mathbf{t}_j \right)$$



Idea

$$\mathbf{t}_\mathbf{p} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w}_\mathbf{p}, \mathbf{w}_\mathbf{v}) \left\| \mathbf{R}_\mathbf{p}\mathbf{v} + \mathbf{t} - \widetilde{\mathbf{v}} \right\|_2^2 \mathrm{d}\mathbf{v}$$

$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \mathbf{v} + \mathbf{t}_j \right)$$



Computation

$$\mathbf{t}_{\mathbf{p}} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w}_{\mathbf{p}}, \mathbf{w}_{\mathbf{v}}) \, \|\mathbf{R}_{\mathbf{p}}\mathbf{v} + \mathbf{t} - \widetilde{\mathbf{v}}\|_2^2 \, \mathrm{d}\mathbf{v}$$

$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j\mathbf{v} + \mathbf{t}_j\right)$$

Computation

$$\mathbf{t_p} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \, \|\mathbf{R_p}\mathbf{v} + \mathbf{t} - \widetilde{\mathbf{v}}\|_2^2 \, \mathrm{d}\mathbf{v}$$

$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \mathbf{v} + \mathbf{t}_j\right)$$

$$\mathbf{t_p} = \frac{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \left(\widetilde{\mathbf{v}} - \mathbf{R_p}\mathbf{v}\right) \mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \, \mathrm{d}\mathbf{v}}$$

Computation

$$\mathbf{t_p} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \, \|\mathbf{R_p}\mathbf{v} + \mathbf{t} - \widetilde{\mathbf{v}}\|_2^2 \, \mathrm{d}\mathbf{v}$$

$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \mathbf{v} + \mathbf{t}_j\right)$$

$$\mathbf{t_p} = \frac{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \left(\widetilde{\mathbf{v}} - \mathbf{R_p}\mathbf{v}\right) \mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \, \mathrm{d}\mathbf{v}}$$

$$= \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \frac{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \mathbf{v} \, \mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \, \mathrm{d}\mathbf{v}} + \mathbf{t}_j \right) - \mathbf{R_p} \frac{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \mathbf{v} \, \mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\boldsymbol{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \, \mathrm{d}\mathbf{v}}$$

Computation

$$\mathbf{t_p} = \arg\min_{\mathbf{t}} \int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \left\|\mathbf{R_p v} + \mathbf{t} - \widetilde{\mathbf{v}}\right\|_2^2 \mathrm{d}\mathbf{v}$$

$$\text{where: } \widetilde{\mathbf{v}} = \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \mathbf{v} + \mathbf{t}_j\right)$$

$$\mathbf{t_p} = \frac{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \left(\widetilde{\mathbf{v}} - \mathbf{R_p v}\right) \mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \,\mathrm{d}\mathbf{v}}$$

$$= \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \frac{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v})\mathbf{v} \,\mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \,\mathrm{d}\mathbf{v}} + \mathbf{t}_j\right) - \mathbf{R_p} \frac{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v})\mathbf{v} \,\mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \,\mathrm{d}\mathbf{v}}$$

$$= \sum_{j=1}^{m} w_{\mathbf{p}j} \left(\mathbf{R}_j \mathbf{p}^* + \mathbf{t}_j\right) - \mathbf{R_p p}^*$$

$$\text{where: } \mathbf{p}^* = \frac{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v})\mathbf{v} \,\mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v}) \,\mathrm{d}\mathbf{v}}$$

Algorithm

Algorithm 1 Skeletal Skinning with Optimized Centers of Rotation

Input: n vertices, vertex i includes:

- Rest pose position $\mathbf{v}_i \in \mathbb{R}^3$
- Skinning weights $\mathbf{w}_i \in \mathbb{R}^m$
- CoR $\mathbf{p}_i^* \in \mathbb{R}^3$ computed by Eq. (1) and Eq. (4)

m bones, bone j transformation is $[\mathbf{R}_j \ \mathbf{t}_j] \in \mathbb{R}^{3\times 4}$

Output: Deformed position $\mathbf{v}'_i \in \mathbb{R}^3$ for all vertices $i = 1..n$

1: **for each** bone j **do**
2: Convert rotation matrix $\mathbf{R}_j$ to unit quaternion $\mathbf{q}_j$
3: **end for**
4: **for each** vertex i **do**
5: $\mathbf{q} \leftarrow w_{i1}\mathbf{q}_1 \oplus w_{i2}\mathbf{q}_2 \oplus \ldots \oplus w_{im}\mathbf{q}_m$

where: $\mathbf{q}_a \oplus \mathbf{q}_b = \begin{cases} \mathbf{q}_a + \mathbf{q}_b & \text{if } \mathbf{q}_a \cdot \mathbf{q}_b \geq 0 \\ \mathbf{q}_a - \mathbf{q}_b & \text{if } \mathbf{q}_a \cdot \mathbf{q}_b < 0 \end{cases}$

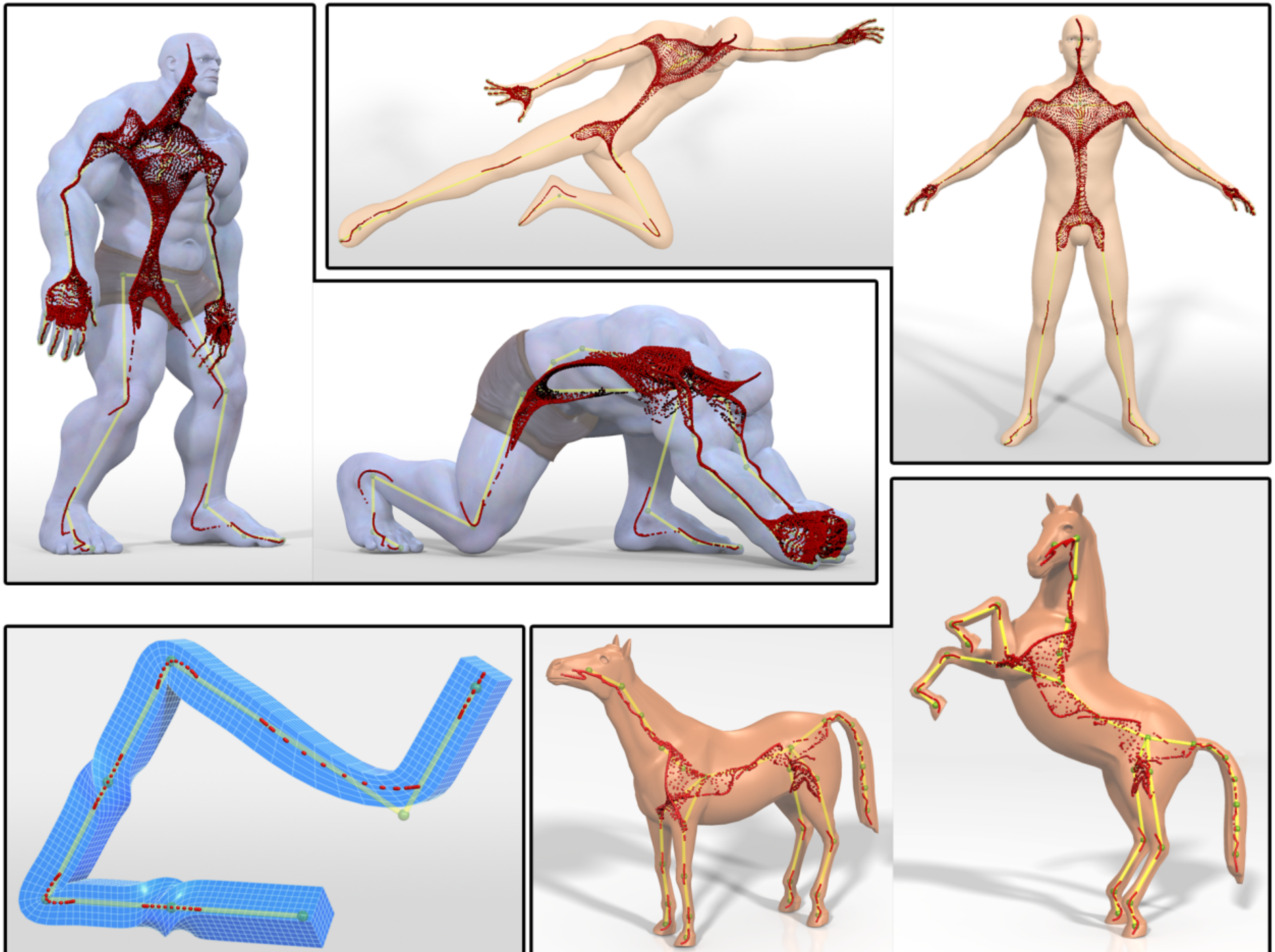
($\mathbf{q}_a \cdot \mathbf{q}_b$ denotes the vector dot product)

6: Normalize and convert $\mathbf{q}$ to rotation matrix $\mathbf{R}$
7: LBS: $[\widetilde{\mathbf{R}} \ \widetilde{\mathbf{t}}] \leftarrow \sum_{j=1}^{m} w_{ij}[\mathbf{R}_j \ \mathbf{t}_j]$
8: Compute translation: $\mathbf{t} \leftarrow \widetilde{\mathbf{R}}\mathbf{p}_i^* + \widetilde{\mathbf{t}} - \mathbf{R}\mathbf{p}_i^*$ (Eq. (3b))
9: $\mathbf{v}'_i \leftarrow \mathbf{R}\mathbf{v}_i + \mathbf{t}$
10: **end for**

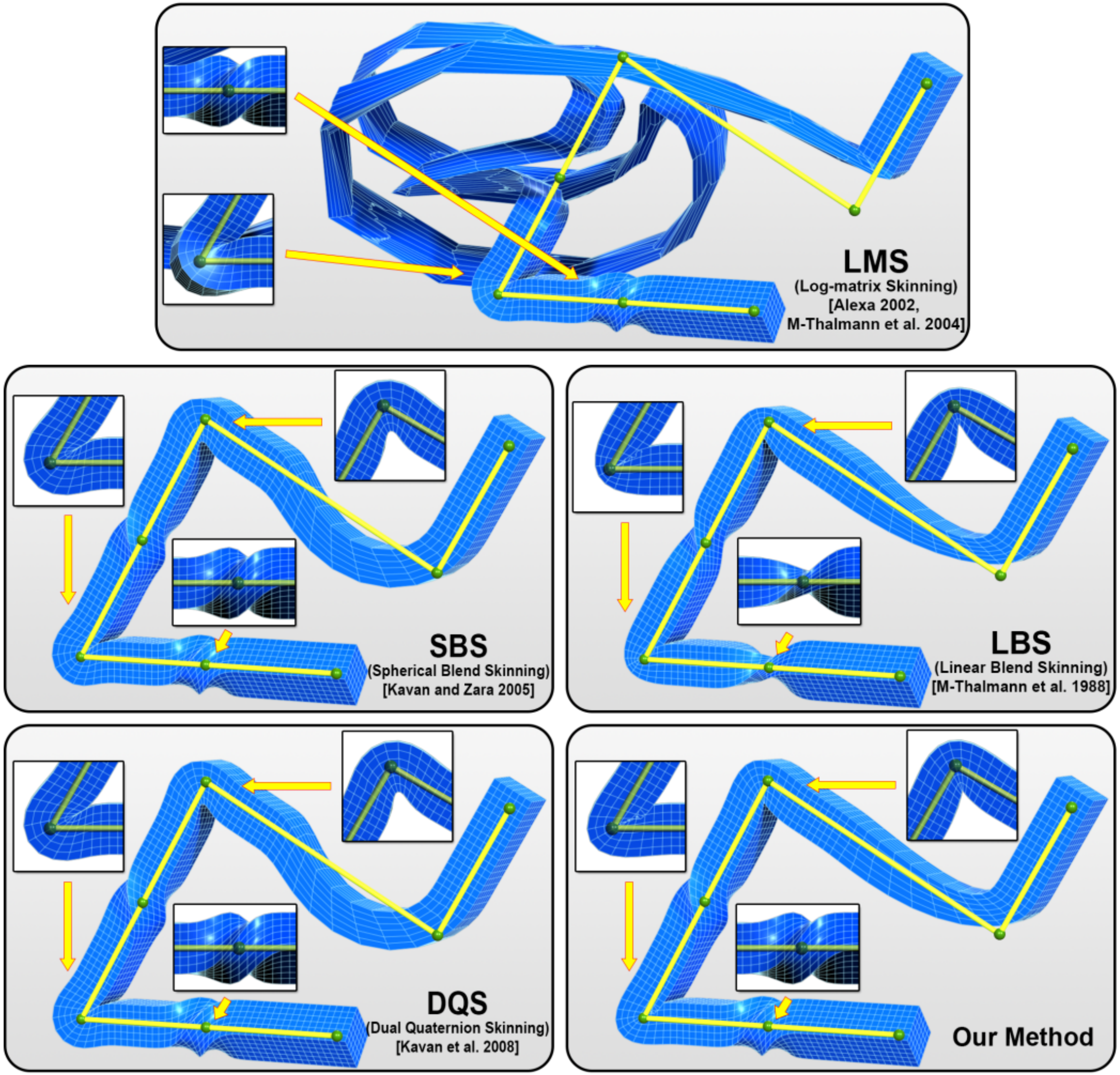
$$\mathbf{p}^* = \frac{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w}_\mathbf{p}, \mathbf{w}_\mathbf{v})\mathbf{v}\,\mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w}_\mathbf{p}, \mathbf{w}_\mathbf{v})\,\mathrm{d}\mathbf{v}}$$

CoRs

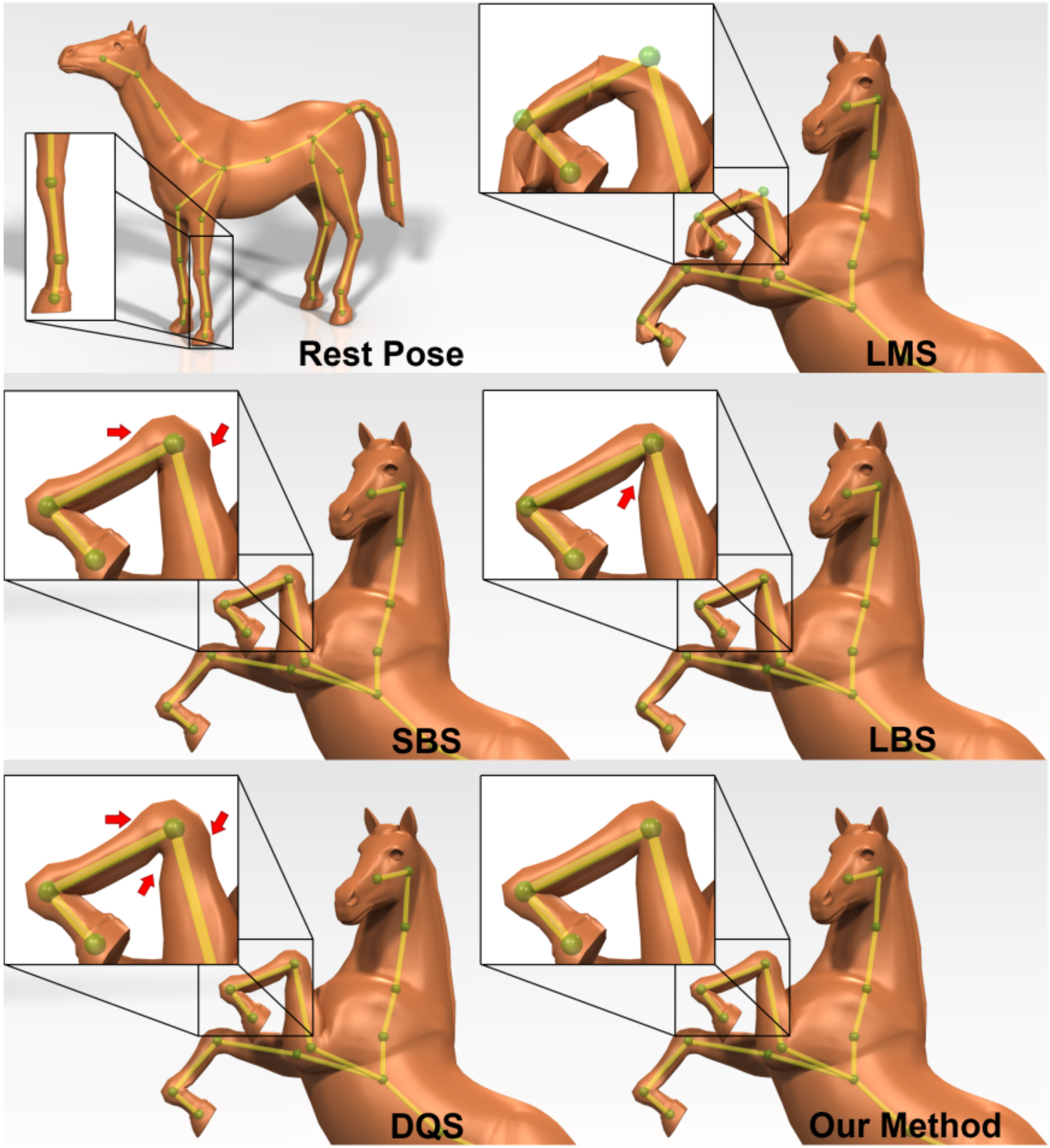
$$\mathbf{p}^* = \frac{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v})\mathbf{v}\,\mathrm{d}\mathbf{v}}{\int_{\mathbf{v}\in\mathbf{\Omega}} s(\mathbf{w_p}, \mathbf{w_v})\,\mathrm{d}\mathbf{v}}$$



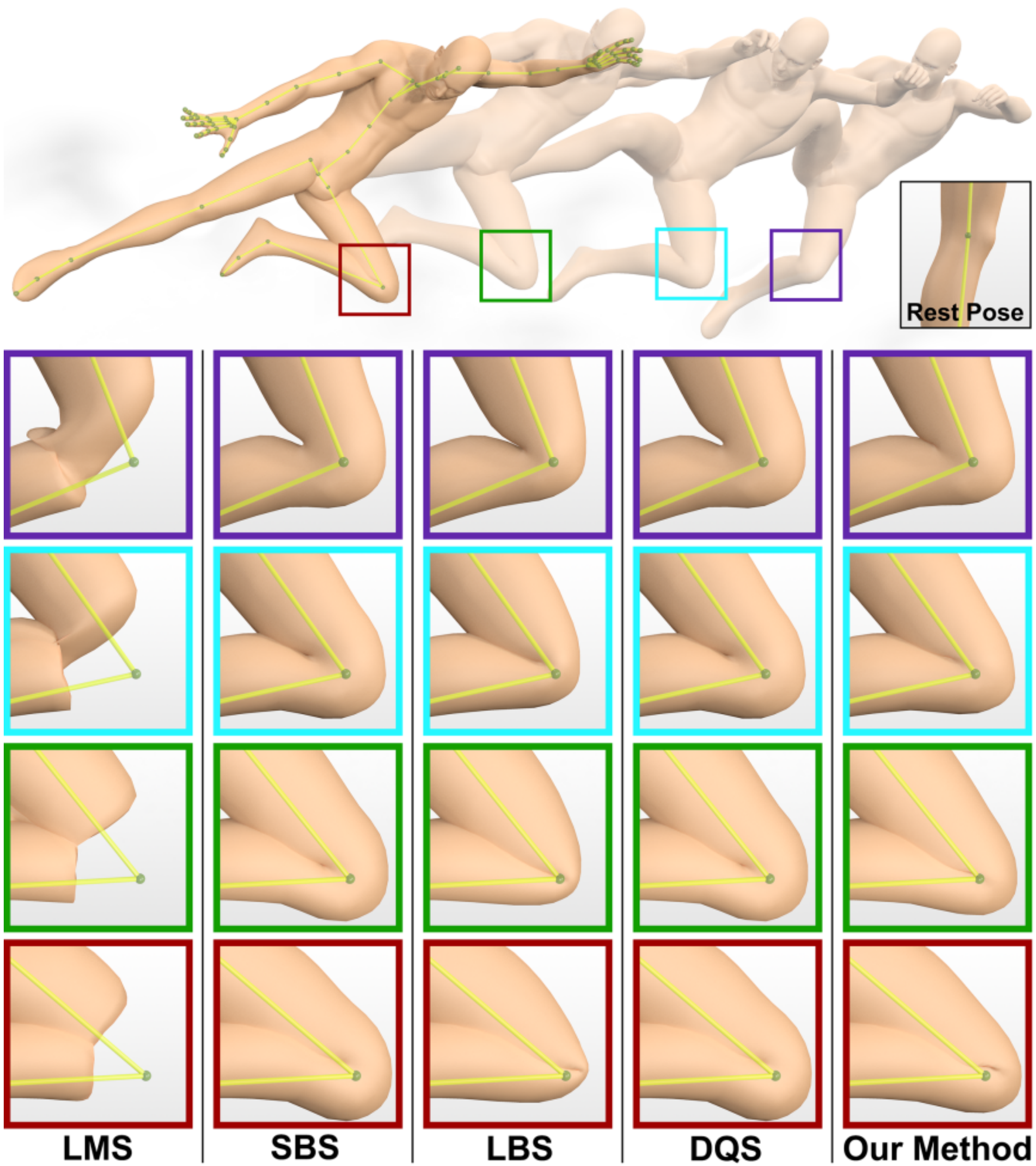
Results


LMS
(Log-matrix Skinning)
[Alexa 2002,
M-Thalmann et al. 2004]
SBS
(Spherical Blend Skinning)
[Kavan and Zara 2005]
LBS
(Linear Blend Skinning)
[M-Thalmann et al. 1988]
DQS
(Dual Quaternion Skinning)
[Kavan et al. 2008]
Our Method

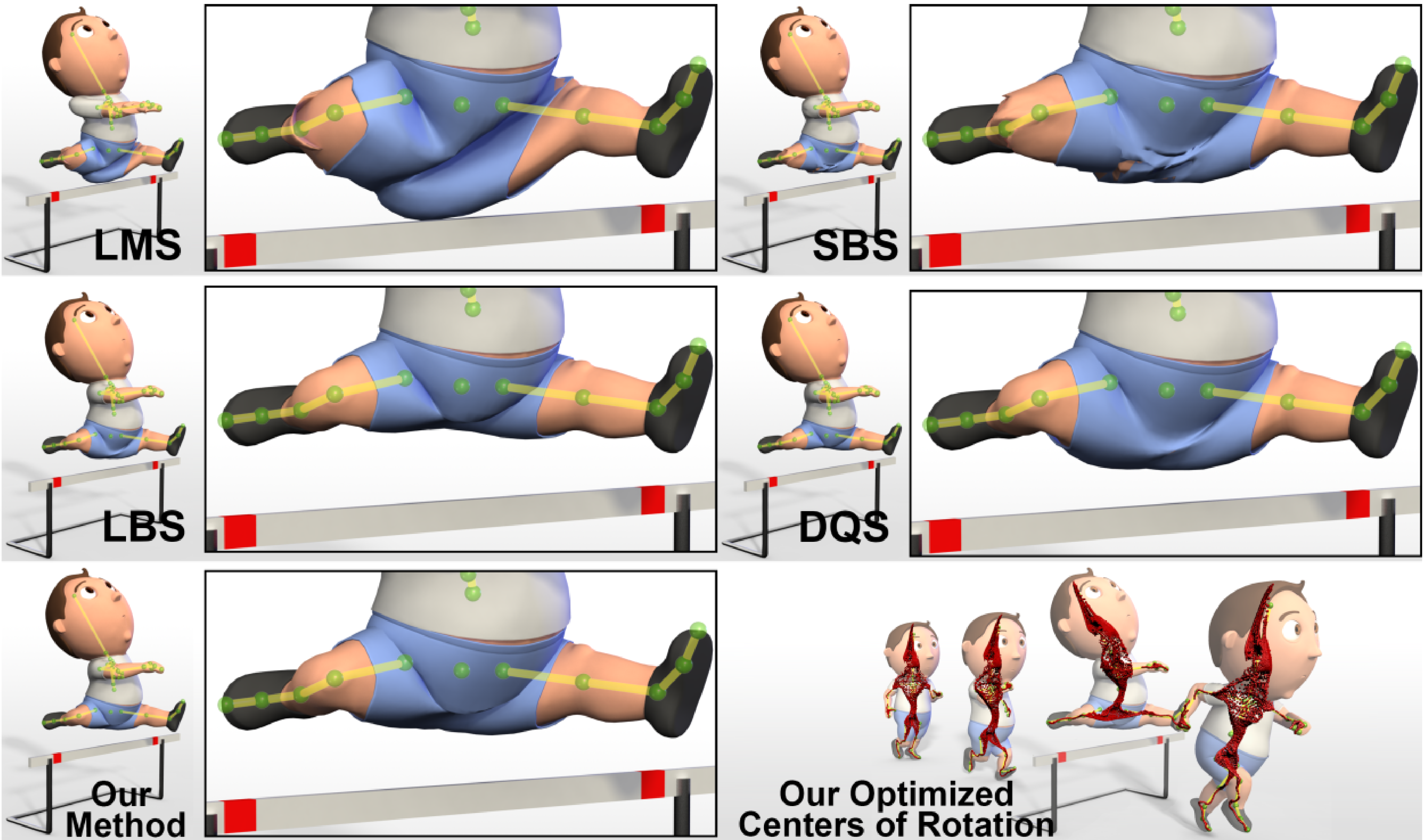
Results



Results



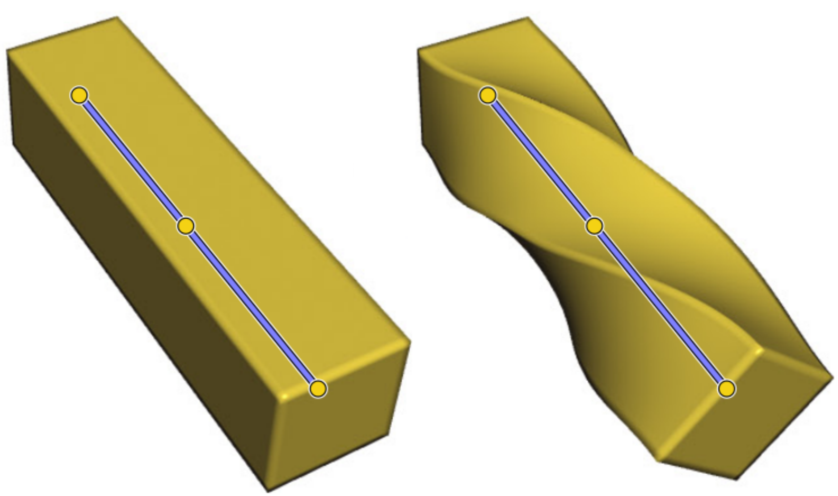
Results


LMS
SBS
LBS
DQS
Our
Method
Our Optimized
Centers of Rotation

Finally : LBS with Complex bones

Before : $$f : v_i \mapsto \sum_{j \in B(i)} w_{ij}(Rj \cdot v_i + Tj)$$

Now : $$f : v_i \mapsto \sum_{j \in B(i)} w_{ij}(R_j(v_i) \cdot v_i + T_j(v_i))$$

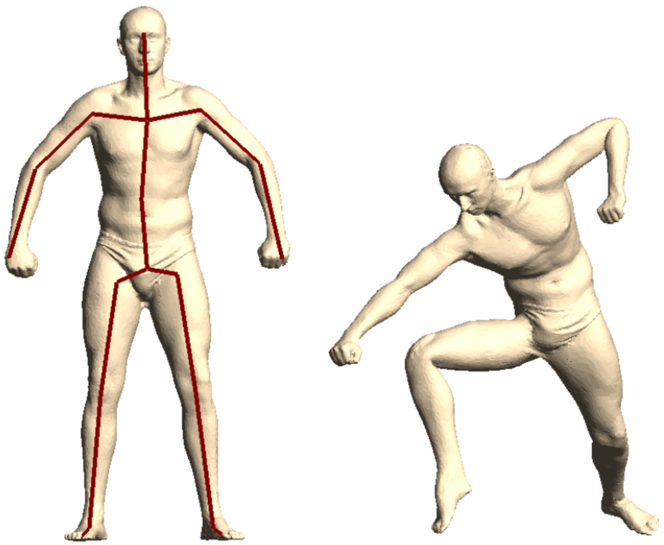


Automatic weights computation methods

- Input :
  - Mesh
  - Skeleton
- Output :
  - Skinning weights for each mesh vertex

HeatBones

- Rather simple
- Very fast
- Lightweight implementation



[Baran&Popovitch2007]

[Baran & Popovic] : Automatic rigging and animation of 3d characters

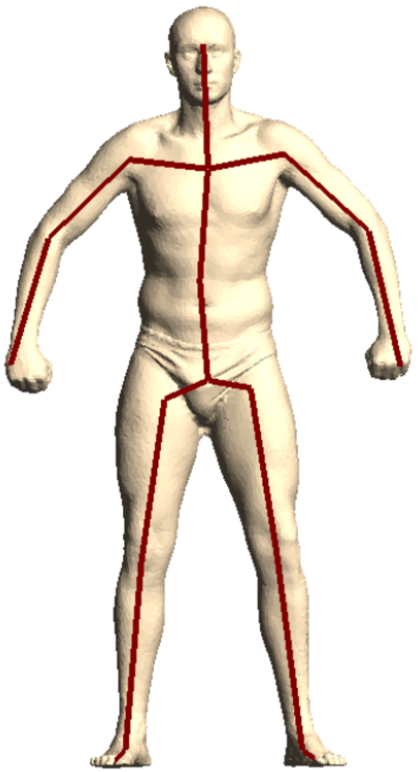
HeatBones : principle

$$-\boldsymbol{\Delta} w_{\bullet j} + H w_{\bullet j} = H \chi_{\bullet j}$$

Laplacian matrix

Stiffness diagonal matrix

Voronoi indicative function

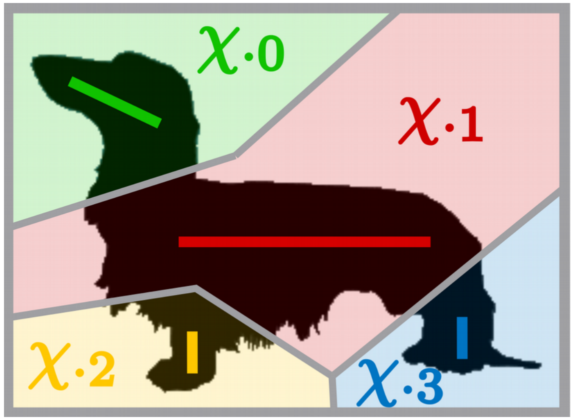


$$H_{jj} = c/d(j)^2$$

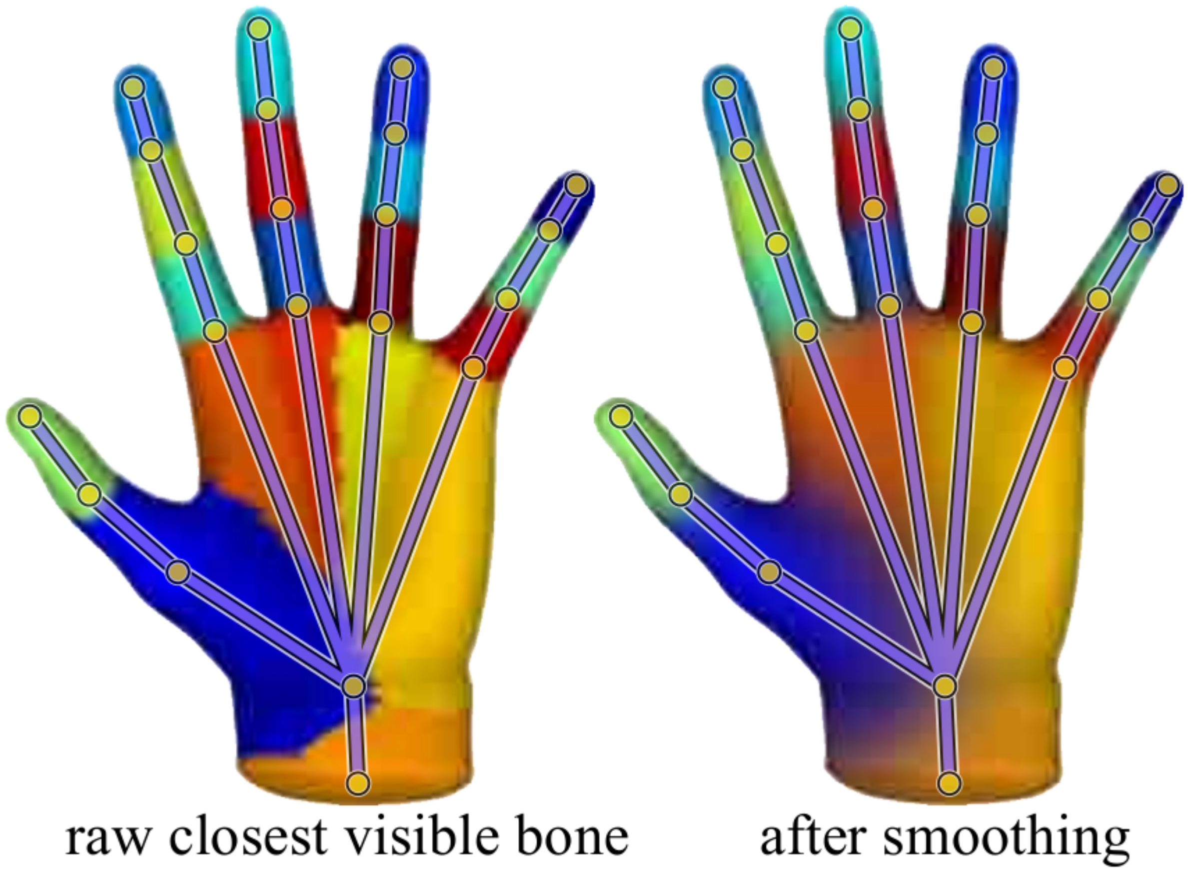
Solve a linear equation, for each bone j.

Positivity and affinity naturally fulfilled.

Intersections with kd-tree.

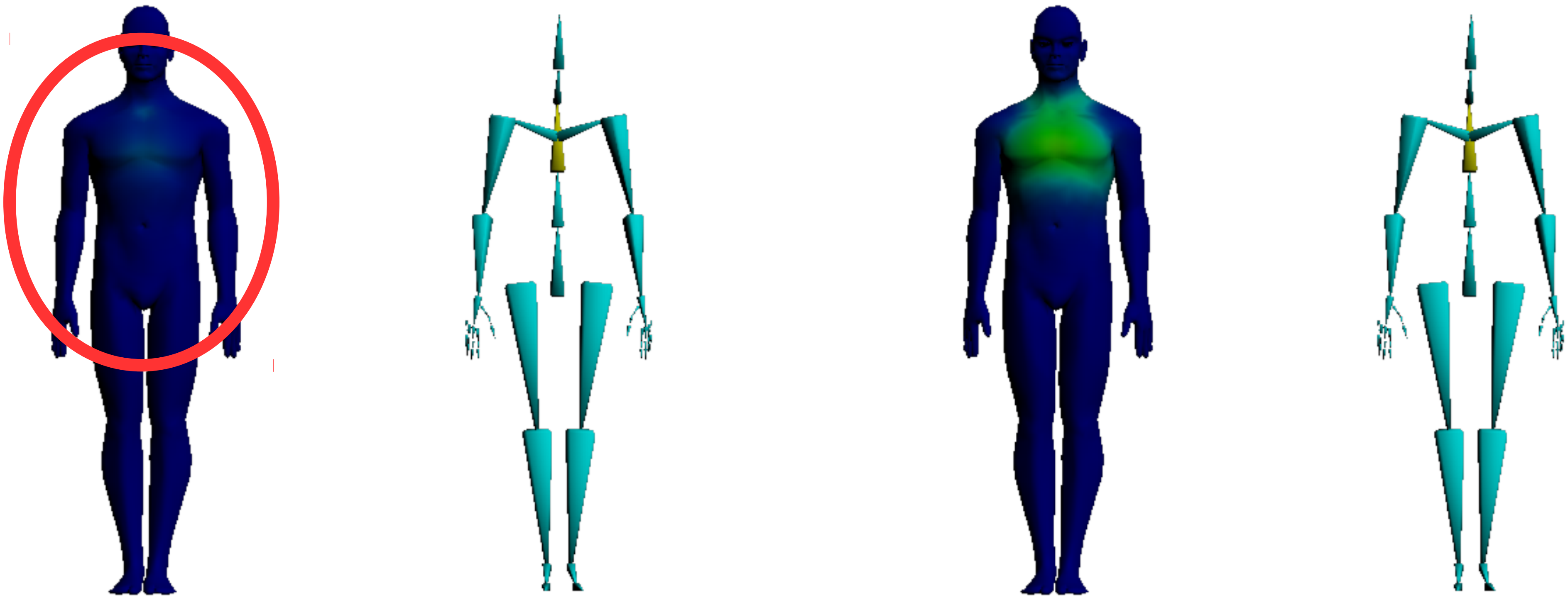


HeatBones : principle



What the algorithm does is simple in spirit : it takes the Voronoi indicative functions, and it blurs them.

BoneGlow : variant of HeatBones

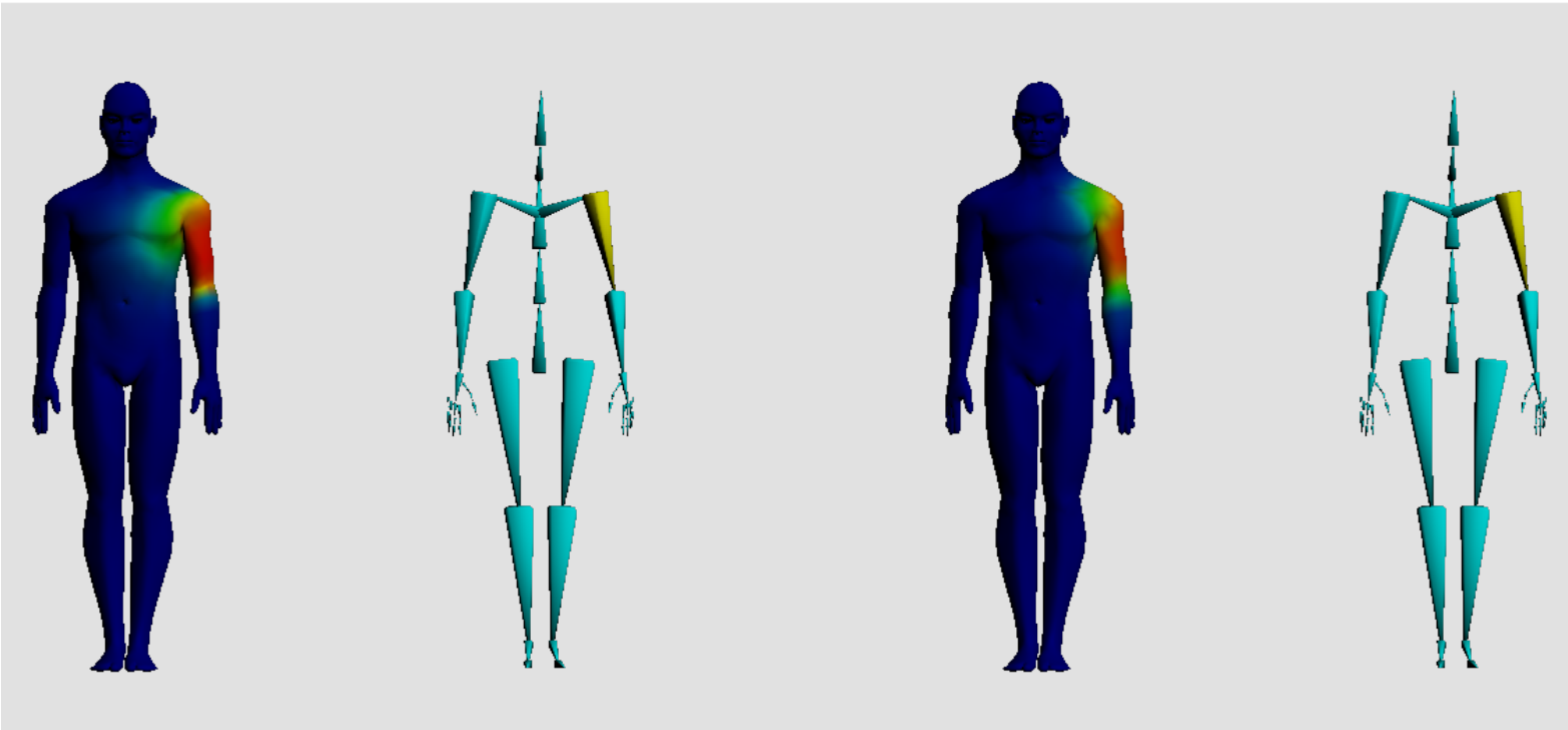


$$-\mathbf{\Delta} w_{\bullet j} + H w_{\bullet j} = H \chi_{\bullet j}$$

[Wareham & Lasenby] : Bone Glow: An Improved Method for the Assignment of Weights for Mesh Deformation

BoneGlow : variant of HeatBones

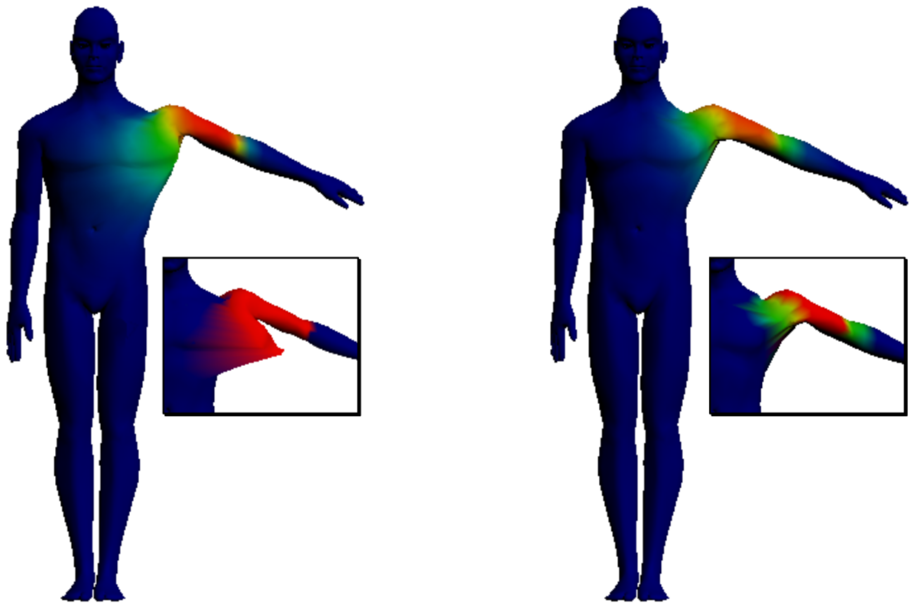
Bone Heat | **Bone Glow**



BoneGlow : variant of HeatBones

$$-\mathbf{\Delta} w_{\bullet j} + H w_{\bullet j} = H \chi_{\bullet j}$$

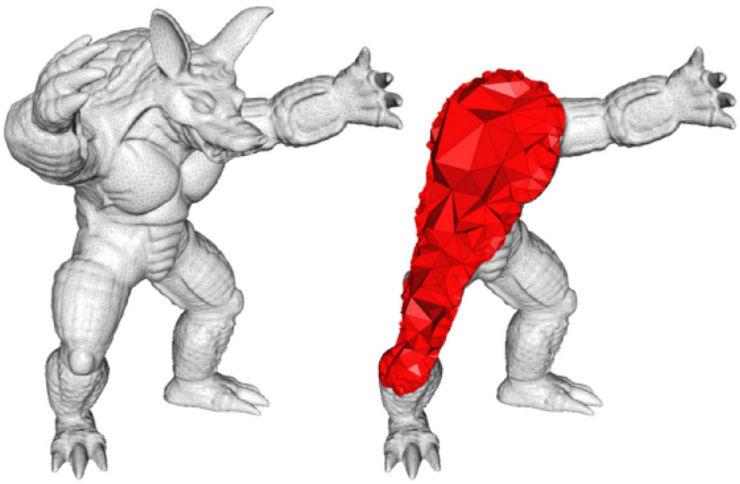
Replace the binary Voronoi indicative function by a softer bone visibility test



(a) Bone Heat (b) Bone Glow

Automatic weights (2)
Bounded Biharmonic Weights (BBW)

- Rather simple
- Rather slow
- Difficult to implement if positivity constraints are enforced



[Jacobson et al.2011]

[Jacobson et al.] : Bounded Biharmonic Weights for Real-Time Deformation

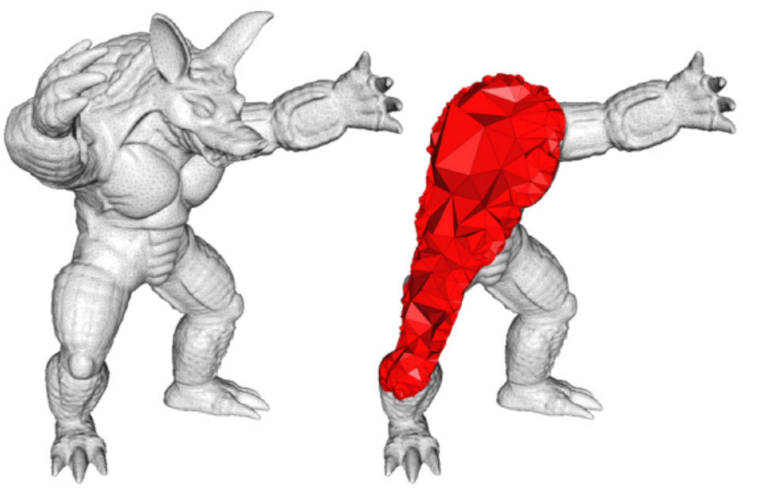
BBW : principle

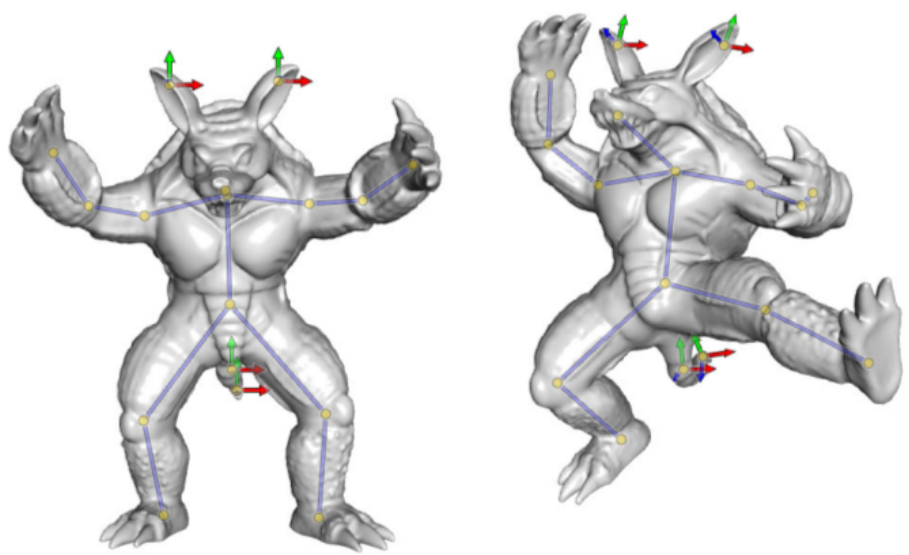
$$\underset{w_j,\ \ j=1,\ldots,m}{\arg\min} \sum_{j=1}^{m} \frac{1}{2} \int_{\Omega} \|\Delta w_j\|^2 dV$$

$$\text{subject to:} \quad w_j|_{H_k} = \delta_{jk}$$

$$\sum_{j=1}^{m} w_j(\mathbf{p}) = 1$$

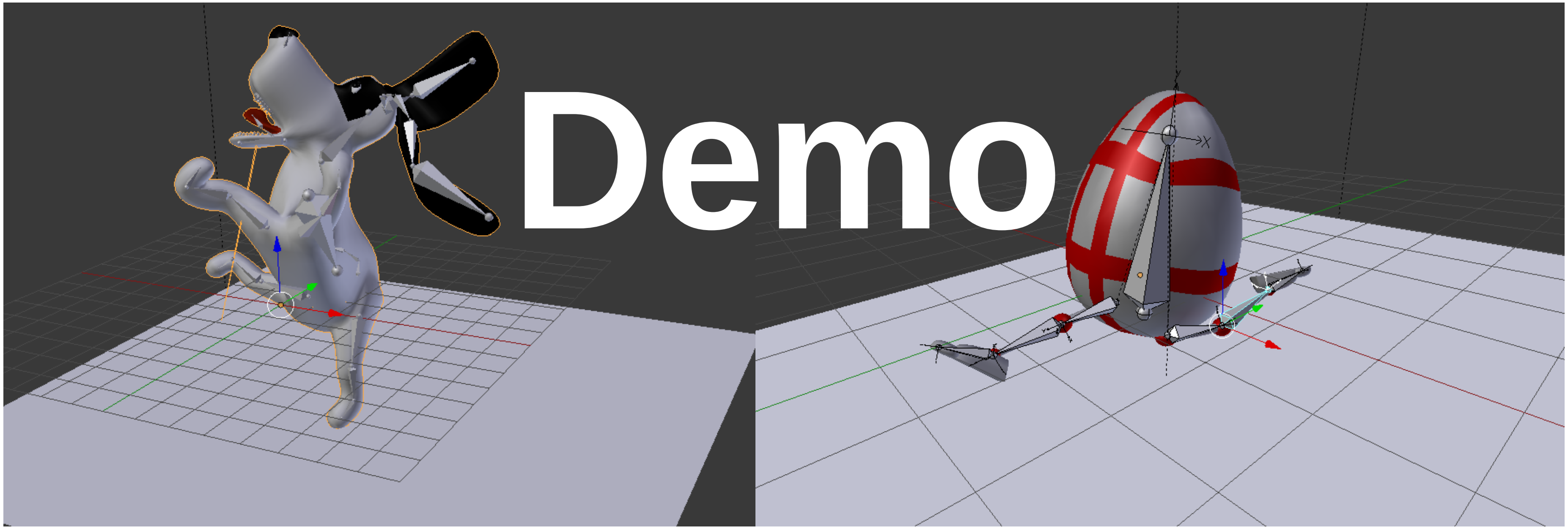
$$0 \le w_j(\mathbf{p}) \le 1$$





Minimize the bi-Laplacian on a tetrahedral mesh, with linear inequalities. → slow

Automatic methods

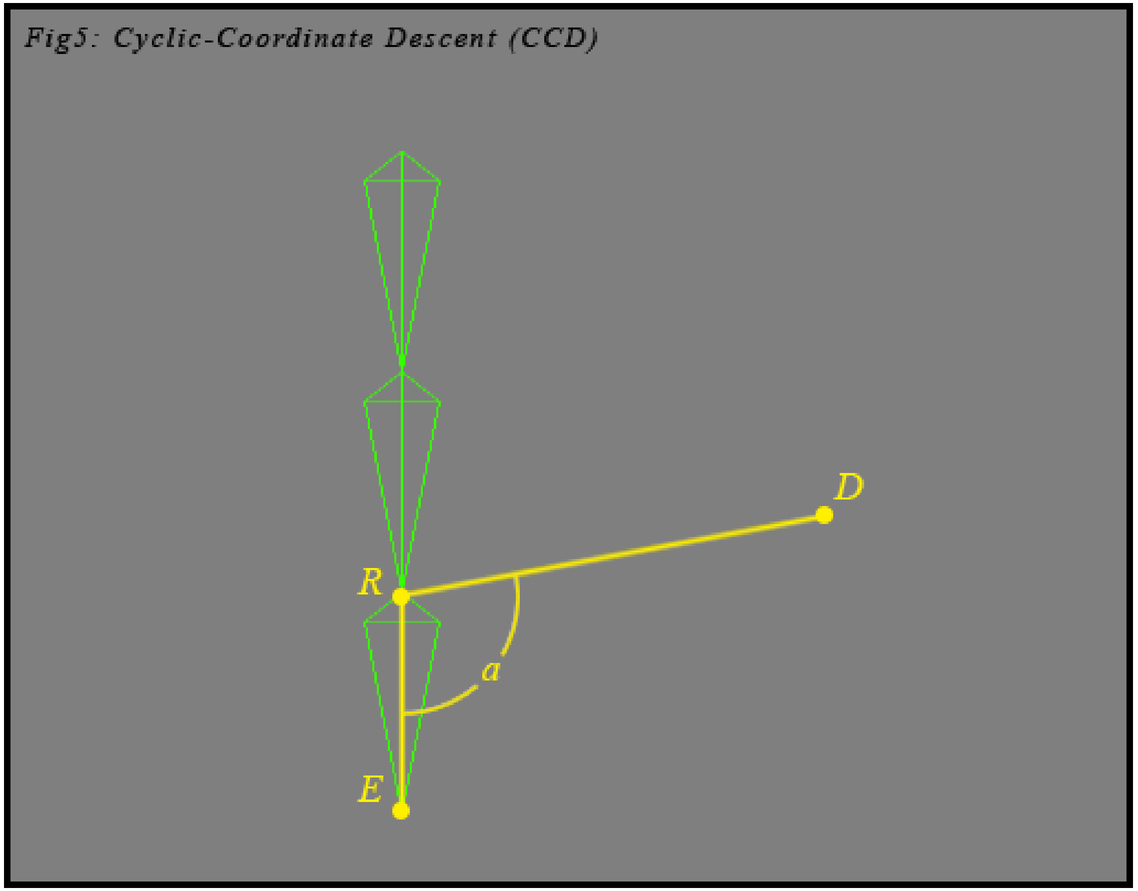


Inverse kinematics

One type of IK Solutions

Cyclic-Coordinate Descent

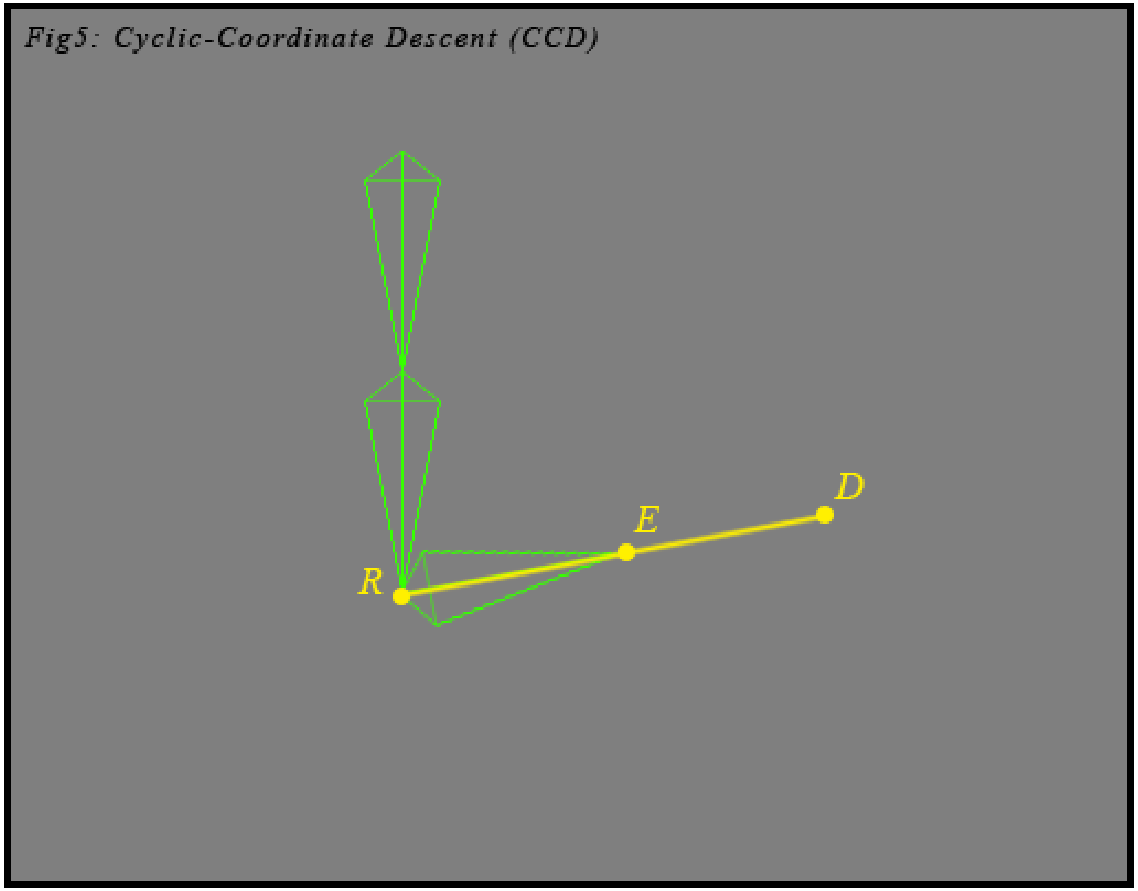
- Starting with the root of our effector, R, to our current endpoint, E.
- Next, we draw a vector from R to our desired endpoint, D
- The inverse cosine of the dot product gives us the angle between the vectors: cos(a) = RD ● RE



One type of IK Solutions

Cyclic-Coordinate Descent

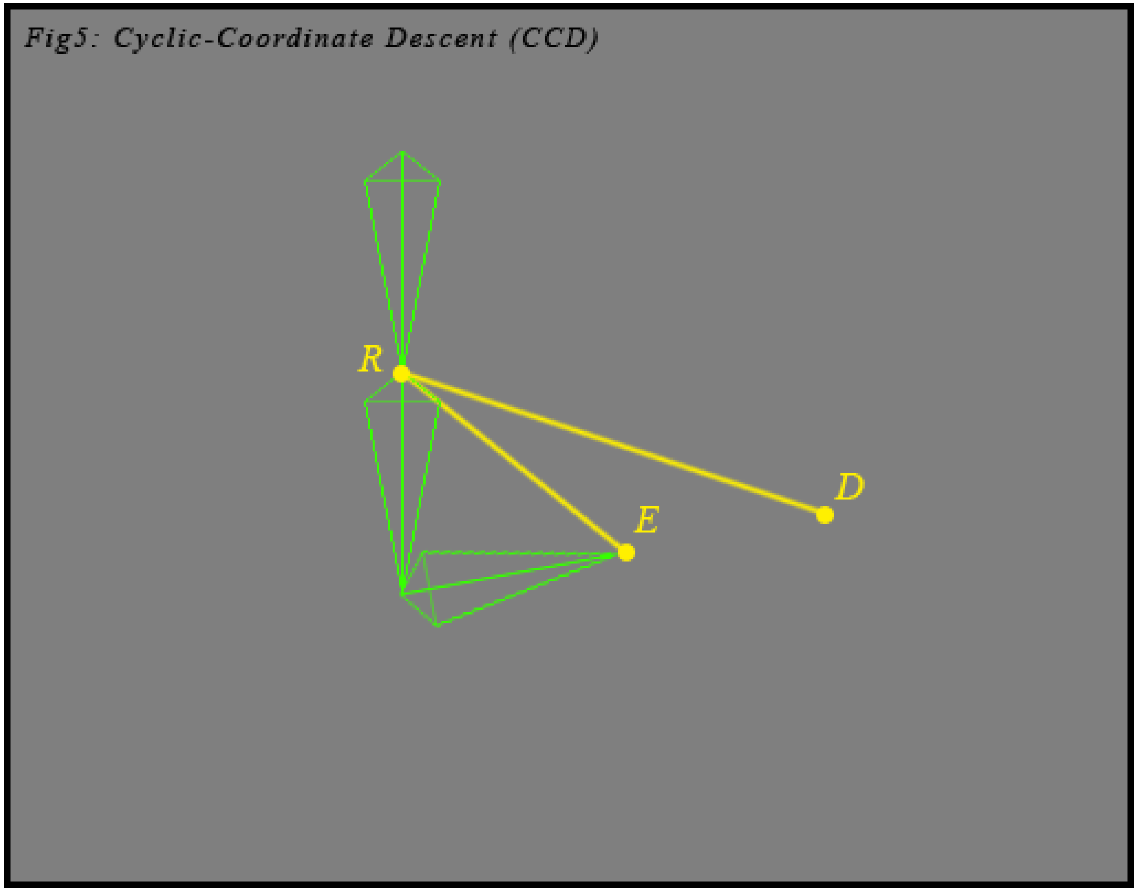
Rotate our link so that RE falls on RD



One type of IK Solutions

Cyclic-Coordinate Descent

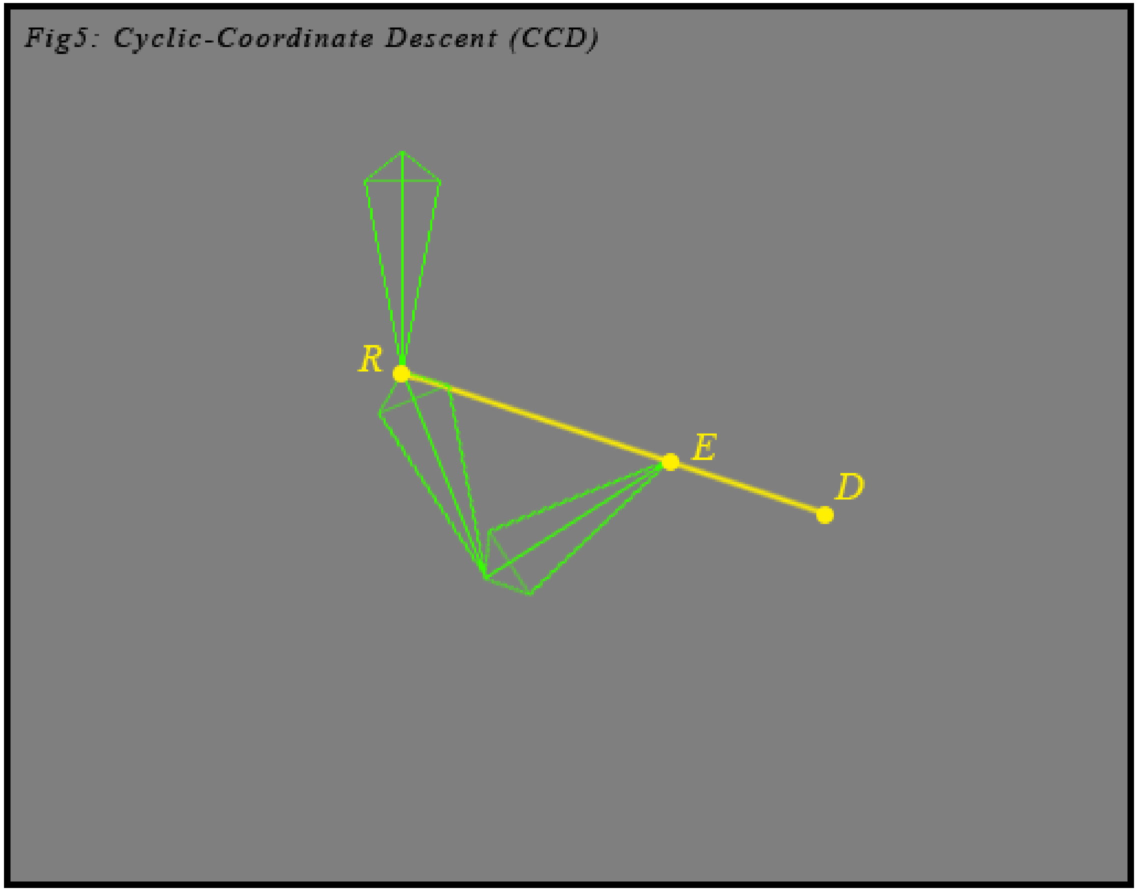
Move one link up the chain, and repeat the process



One type of IK Solutions

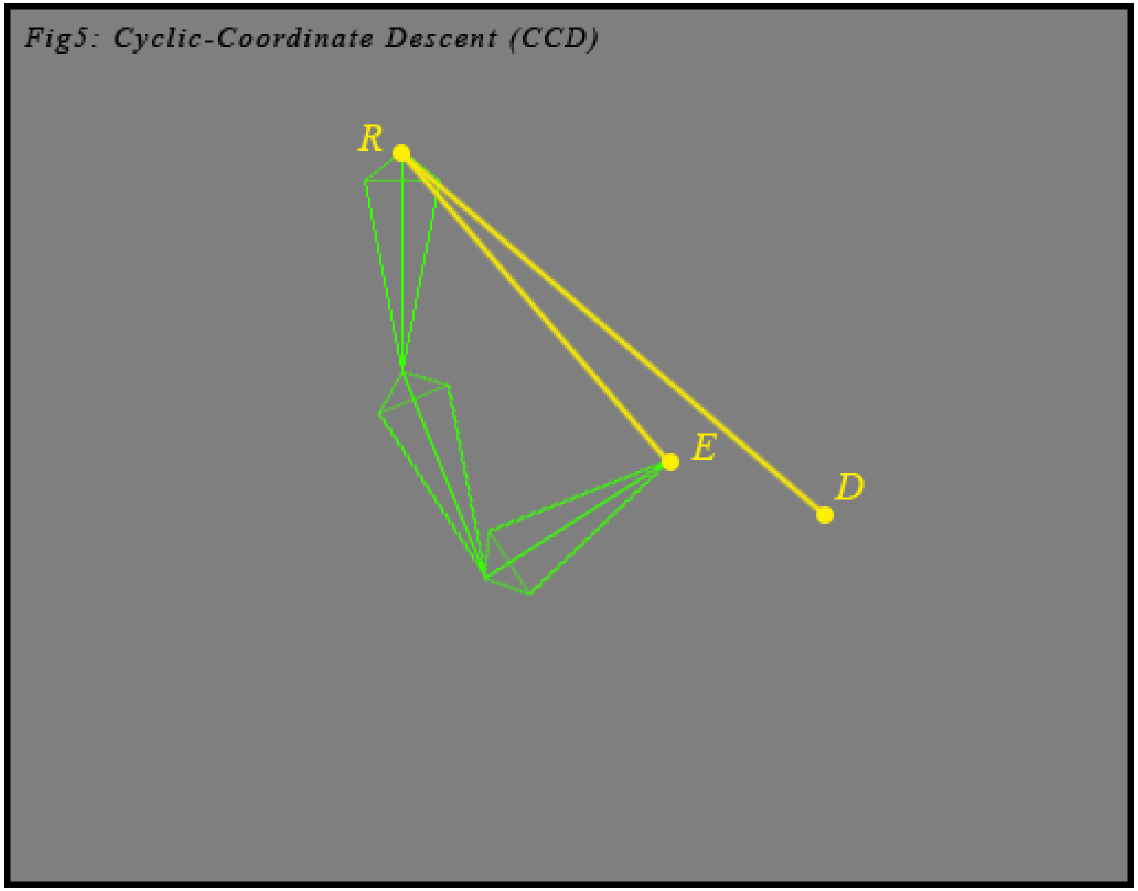
Cyclic-Coordinate Descent

The process is basically repeated until the root joint is reached. Then the process begins all over again starting with the end effector, and will continue until we are close enough to D for an acceptable solution.



One type of IK Solutions

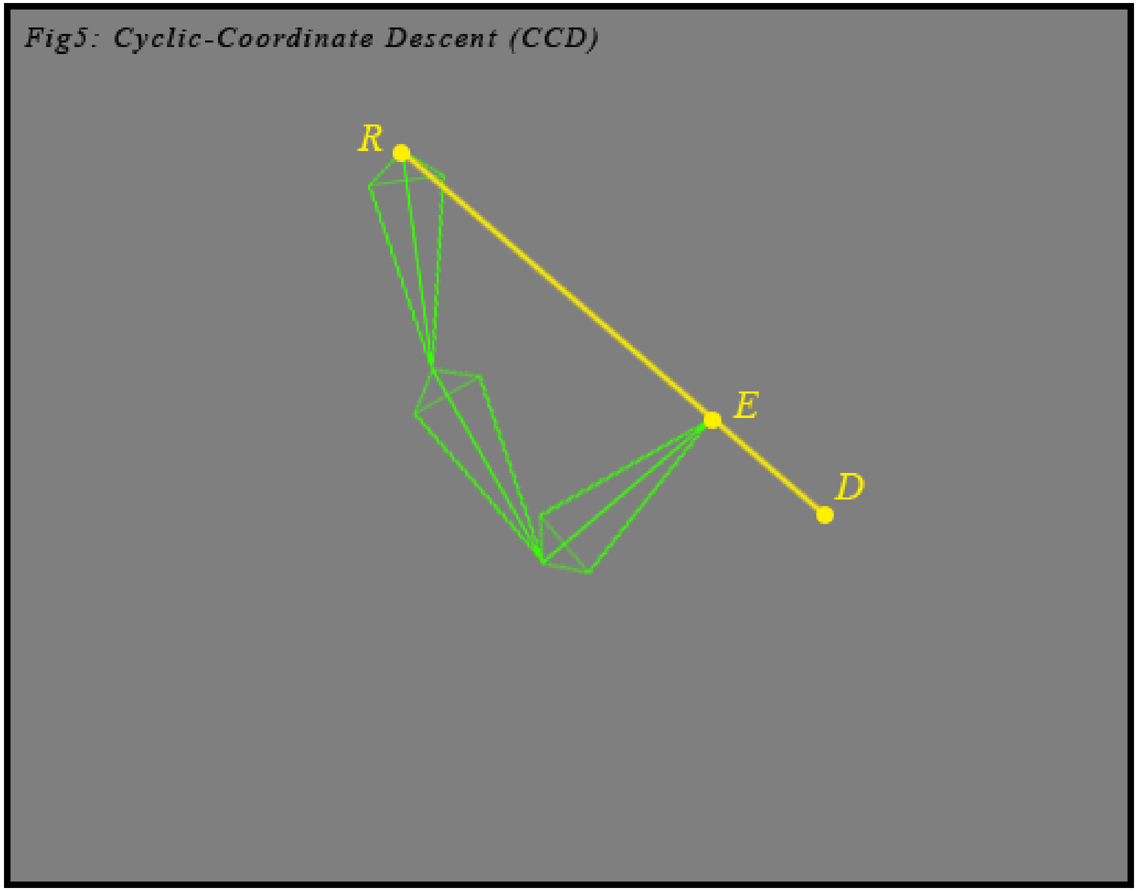
Cyclic-Coordinate Descent



One type of IK Solutions

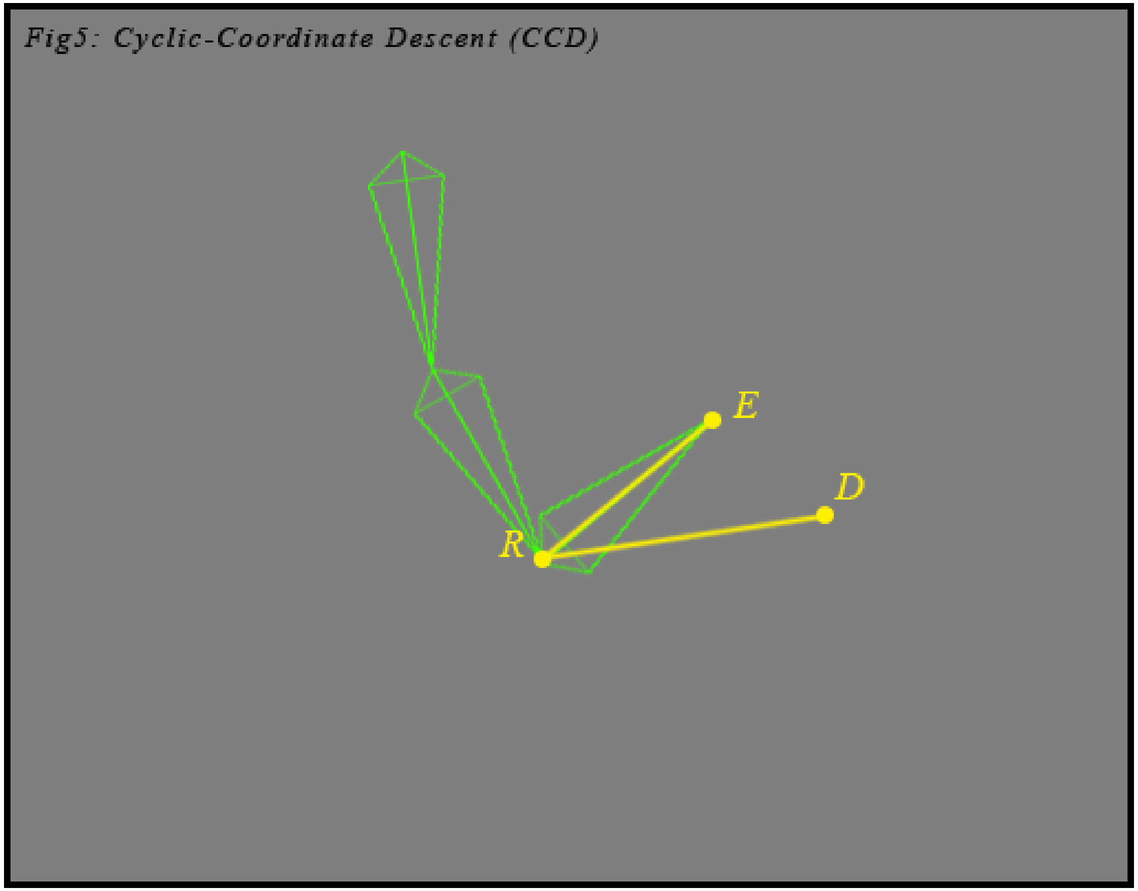
Cyclic-Coordinate Descent

We've reached the root. Repeat the process



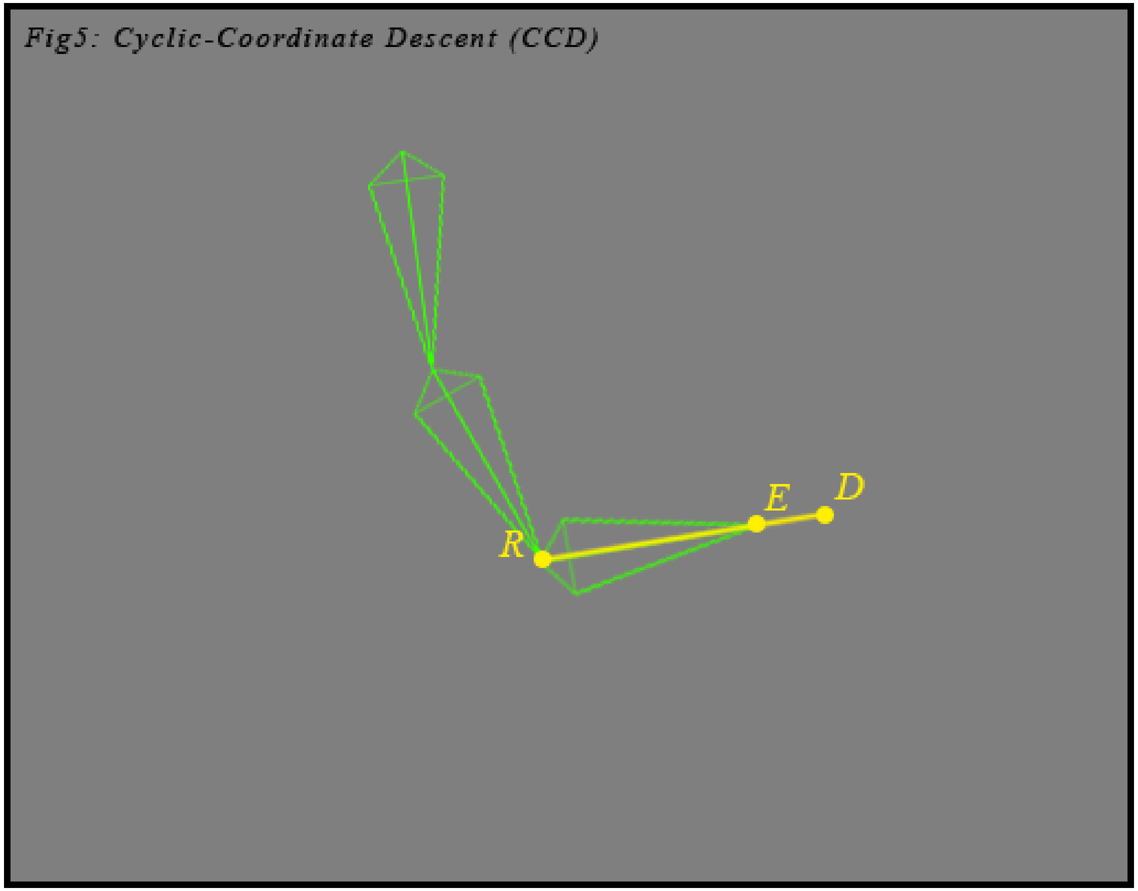
One type of IK Solutions

Cyclic-Coordinate Descent



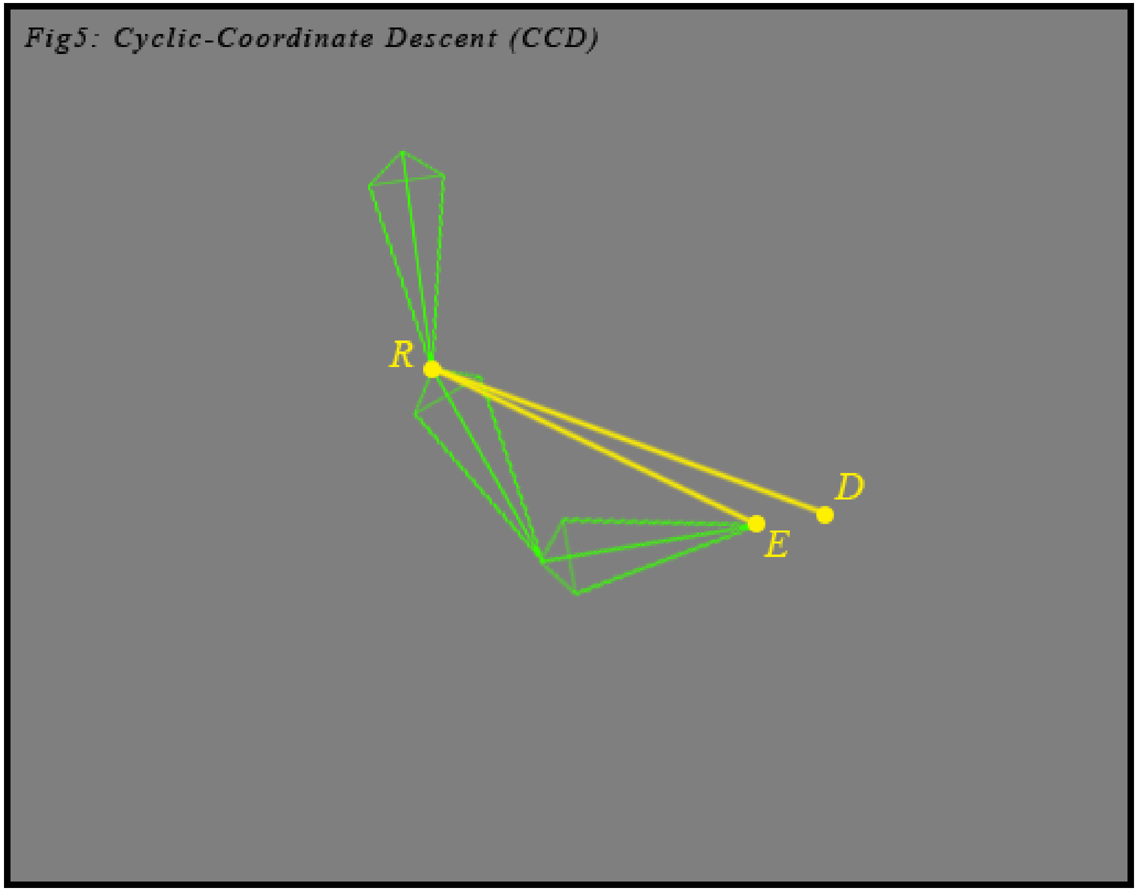
One type of IK Solutions

Cyclic-Coordinate Descent



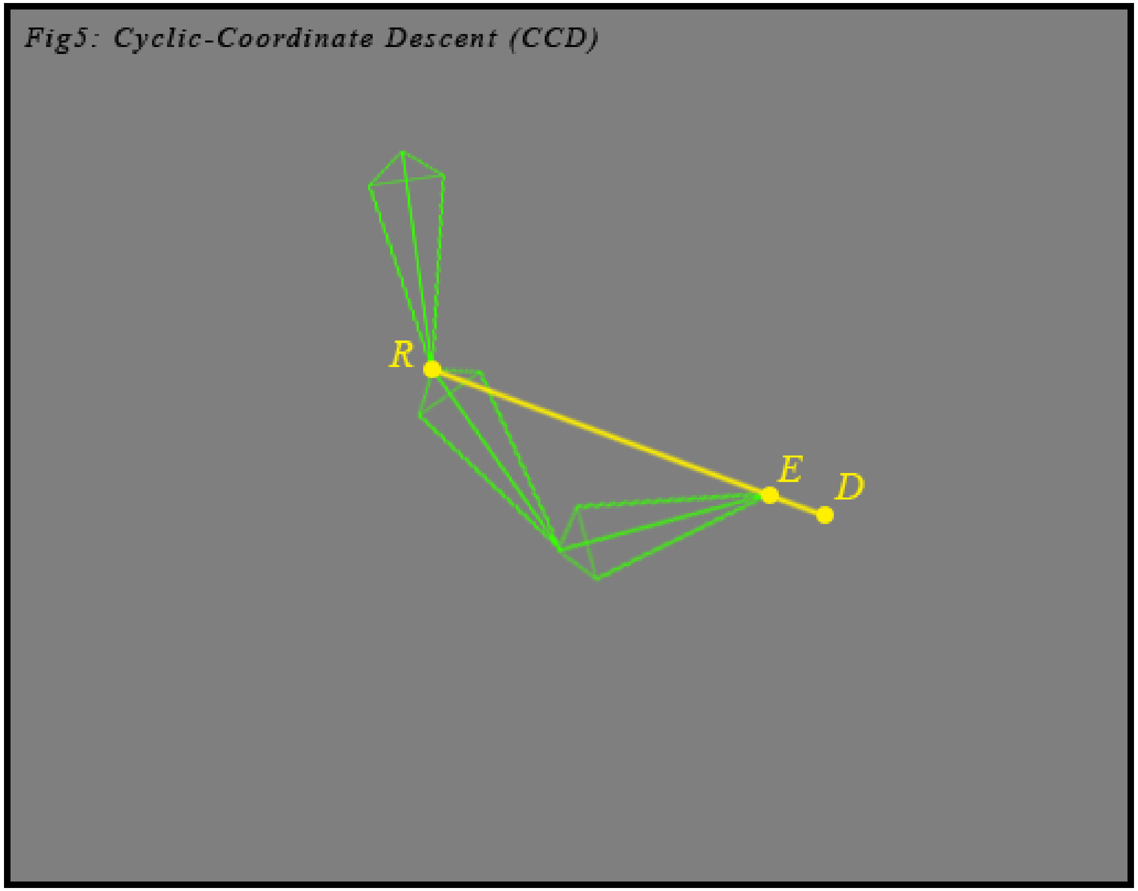
One type of IK Solutions

Cyclic-Coordinate Descent



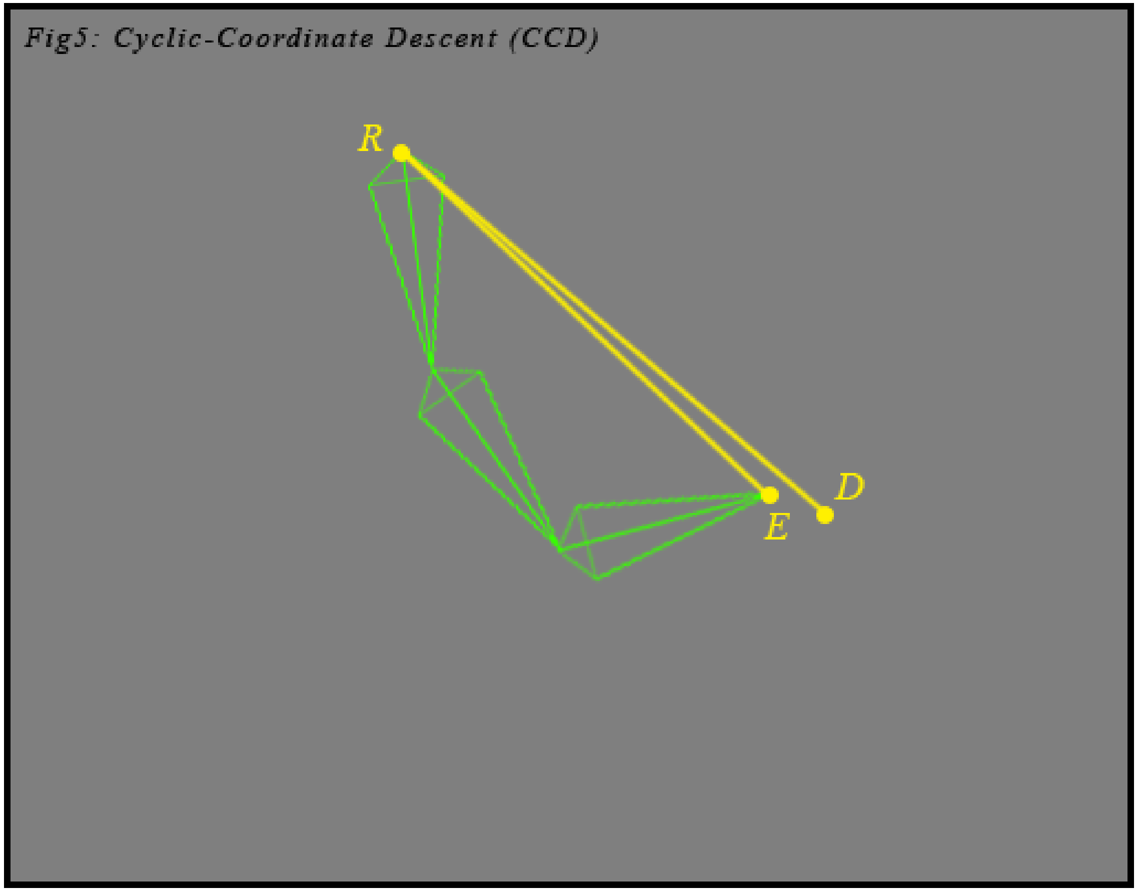
One type of IK Solutions

Cyclic-Coordinate Descent



One type of IK Solutions

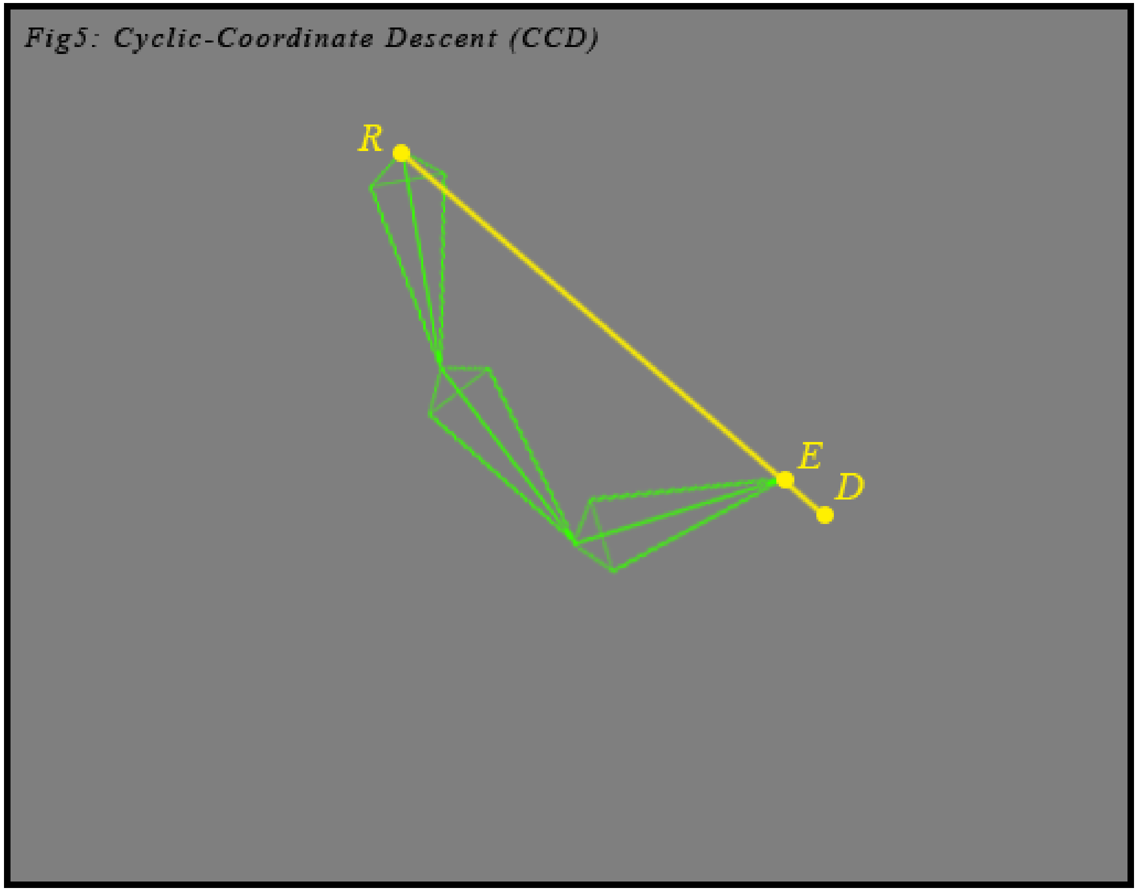
Cyclic-Coordinate Descent



One type of IK Solutions

Cyclic-Coordinate Descent

We've reached the root again. Repeat the process until solution reached.



Using IK in Game Development

Examples of CCD IK in action:

- Character Animation Demo (Softimage XSI 5.0, Blender, Maya, everywhere)

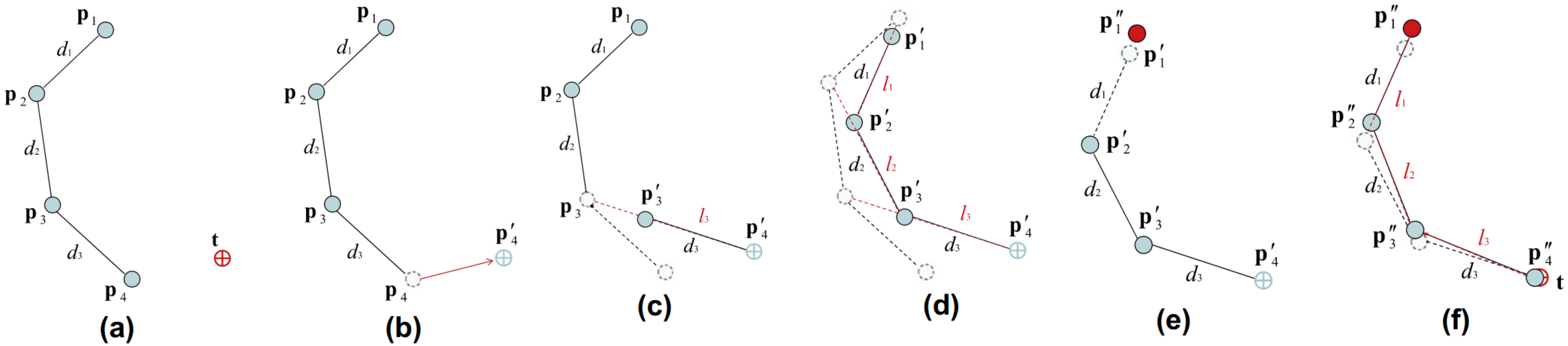
- Real-Time calculations: E3 2003 Demo Footage of Half-Life 2

Problems of CCD

- Bones optimized « one after the other »
- Might not be optimal in terms of realism
- Does not take « physics » into account (balancing the efforts of bending the various joints)
- Example of alternative methods allowing for all of that : the Jacobian method
  - Solve $f(\theta_1, \theta_2, \ldots) = (x, y, z)$
  - Update $f(\overline{\theta}) + Jf \, . \, d\overline{\theta} = (x, y, z)$

FABRIK : Forward And Backward Reach Inverse Kinematics

- Not much more complex than CCD
- Implemented in Unreal4, Unity, …
- Comparison with other methods



Conclusions

- Skeletons are adapted to character animation
- You find them in games, shape recognition, movies, ...
- The influence of the bones needs to be defined by weights
- Plenty of possibilities for the weights
- Plenty of possibilities for the blending model (LBS, DQS, …)
- Simple structure → advanced deformation mechanisms (IK)
- Plenty of open problems