Fast Automatic Skinning Transformations

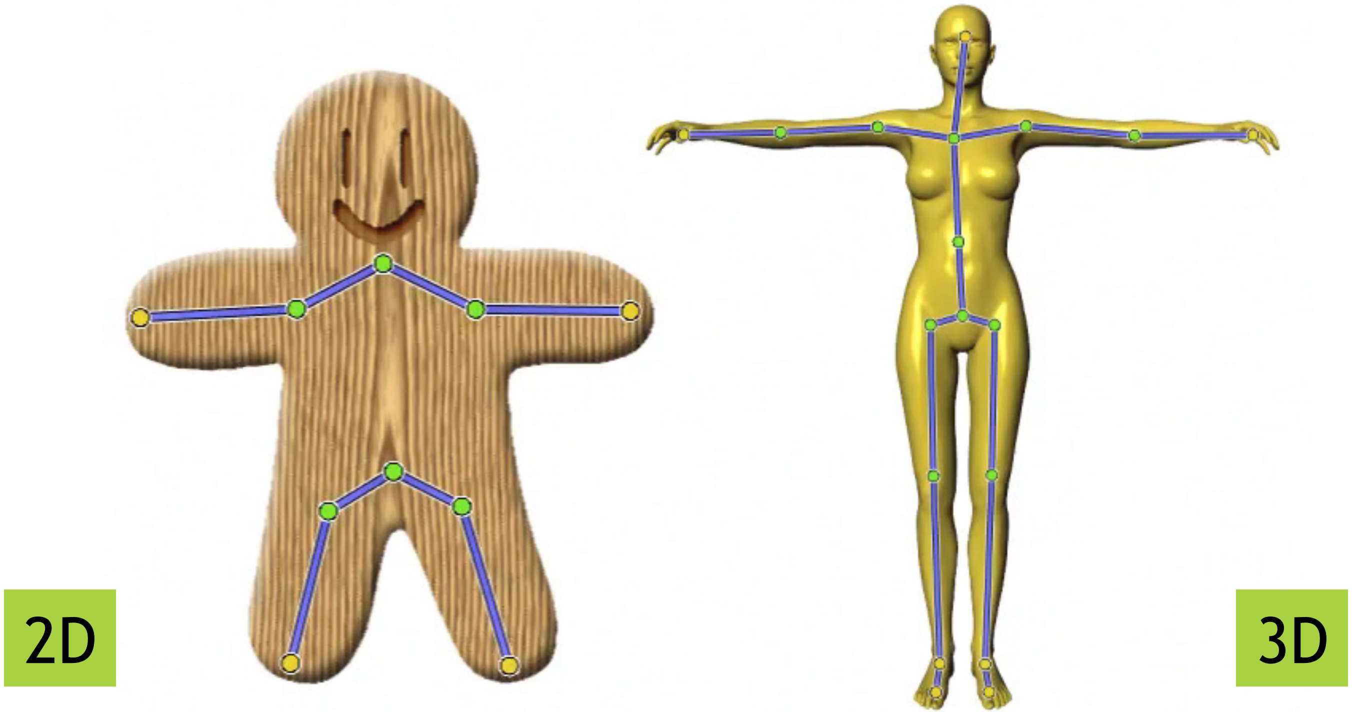
Alec Jacobson — ETH Zurich
Ilya Baran — Disney Research Zurich
Ladislav Kavan — ETH Zurich
Jovan Popović — Adobe Systems, Inc.
Olga Sorkine — ETH Zurich

igl INTERACTIVE GEOMETRY LAB

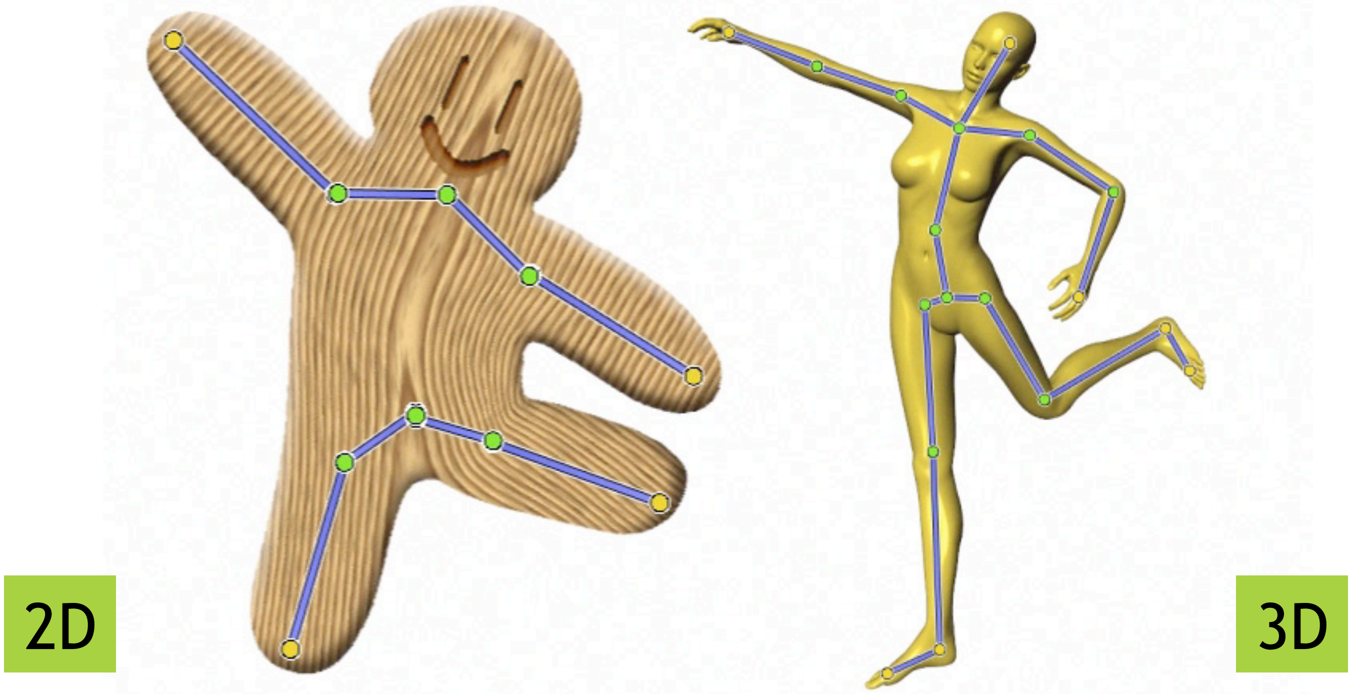
August 8, 2012

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Real-time performance critical for interactive design and animation

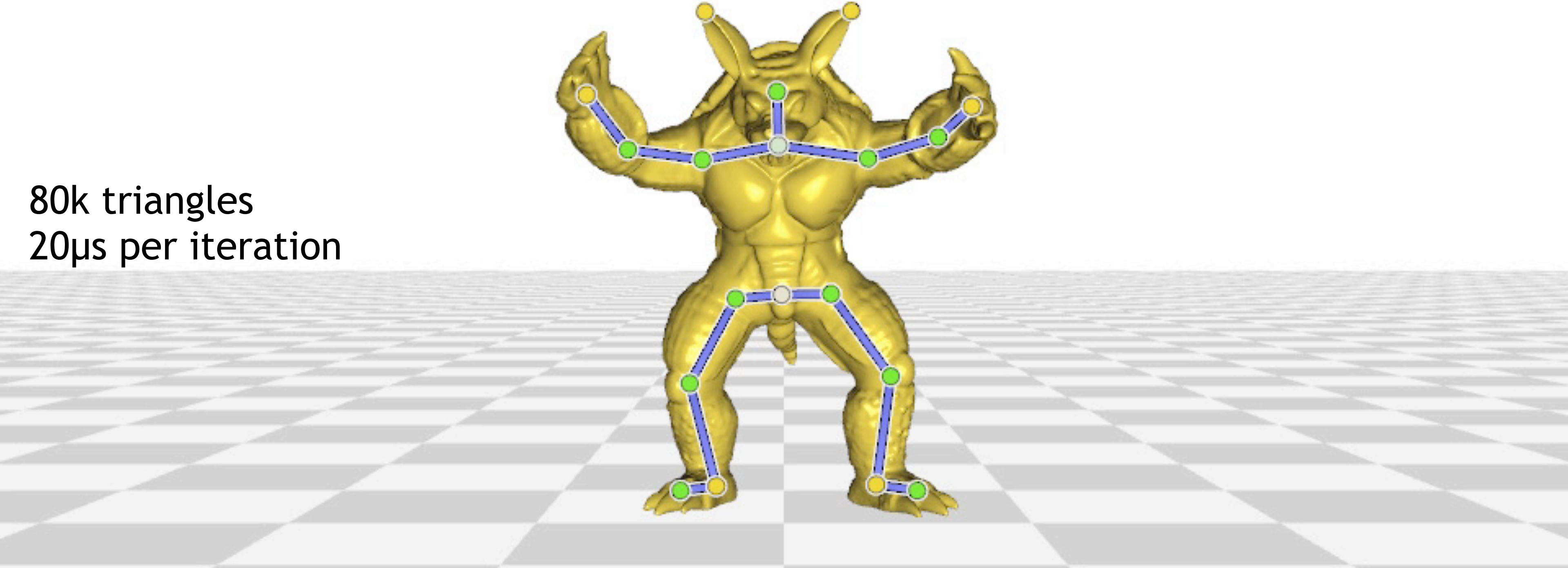


Real-time performance critical for interactive design and animation

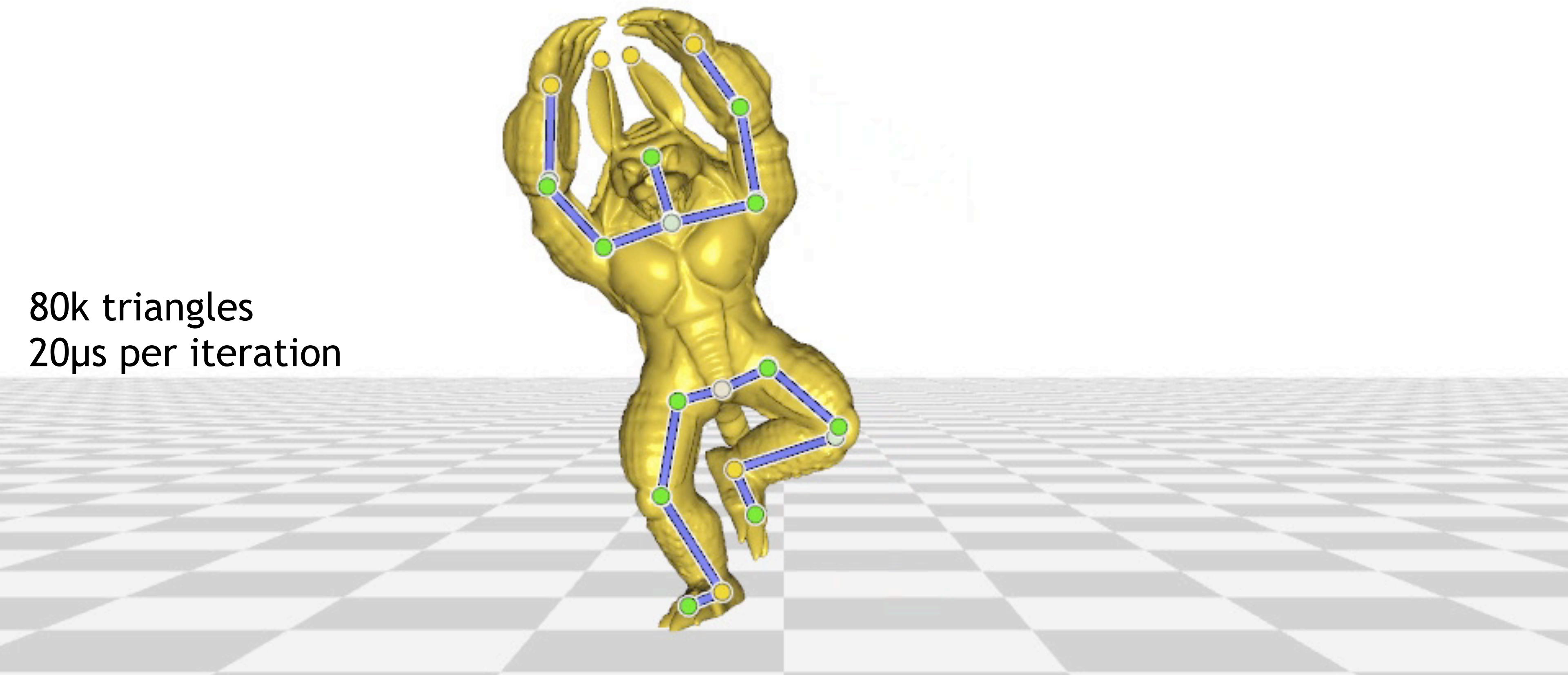


We want speeds measured in microseconds

80k triangles
20μs per iteration

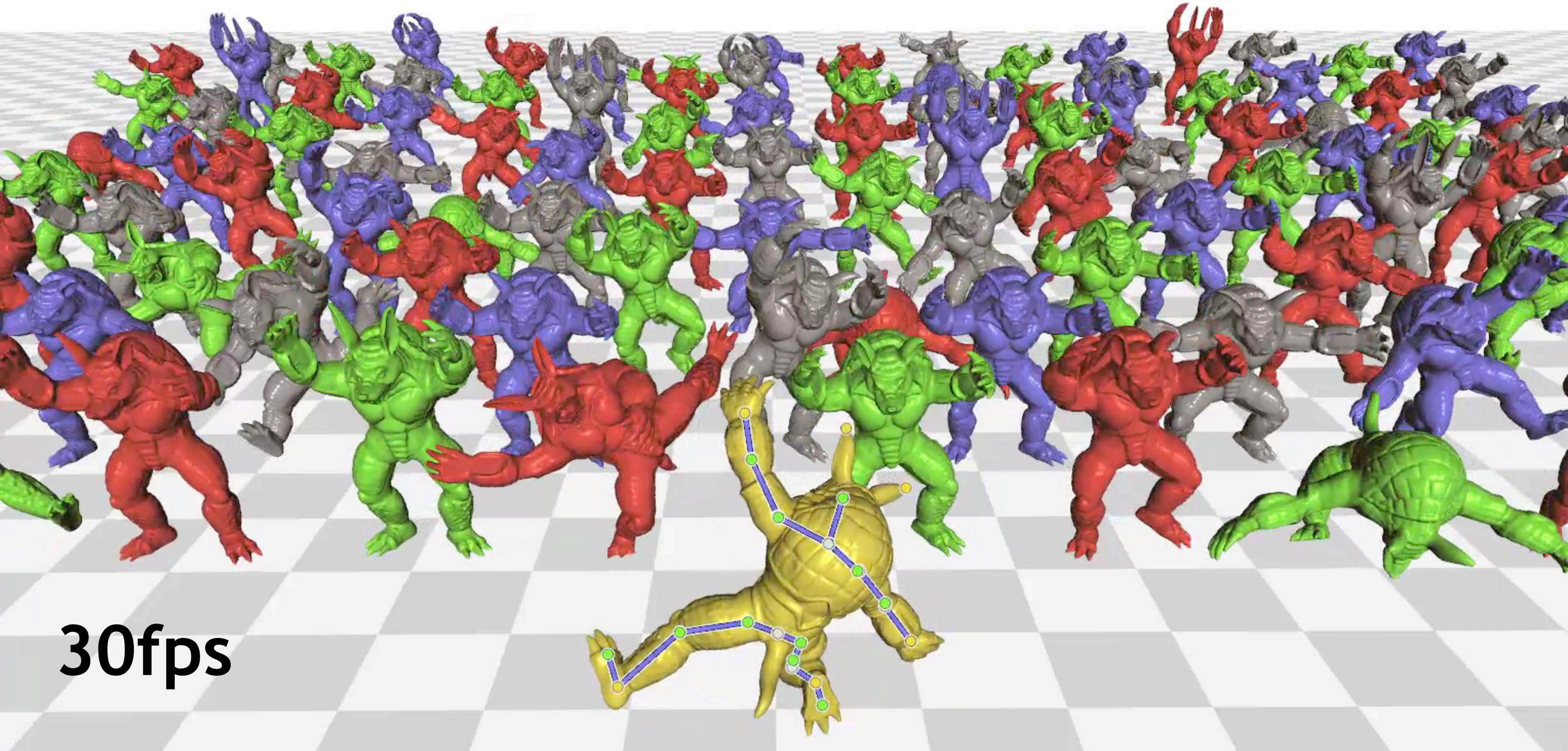


We want speeds measured in microseconds

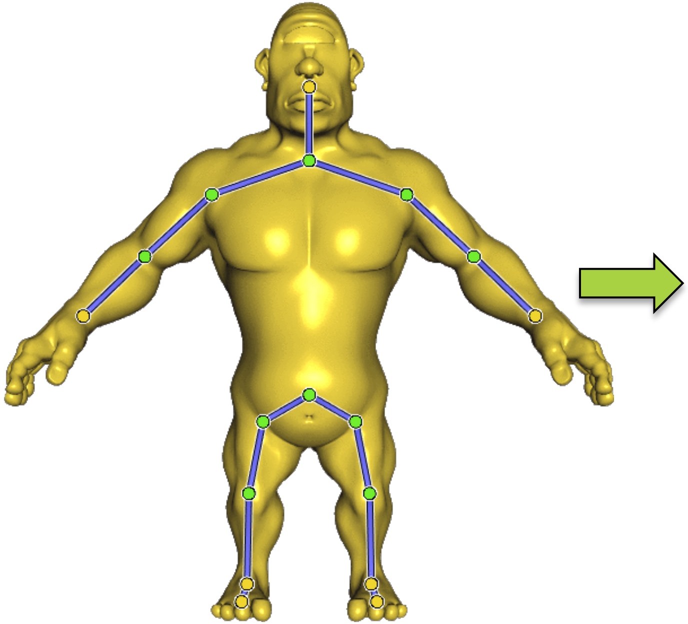


80k triangles
20μs per iteration

This means speed comparable to rendering

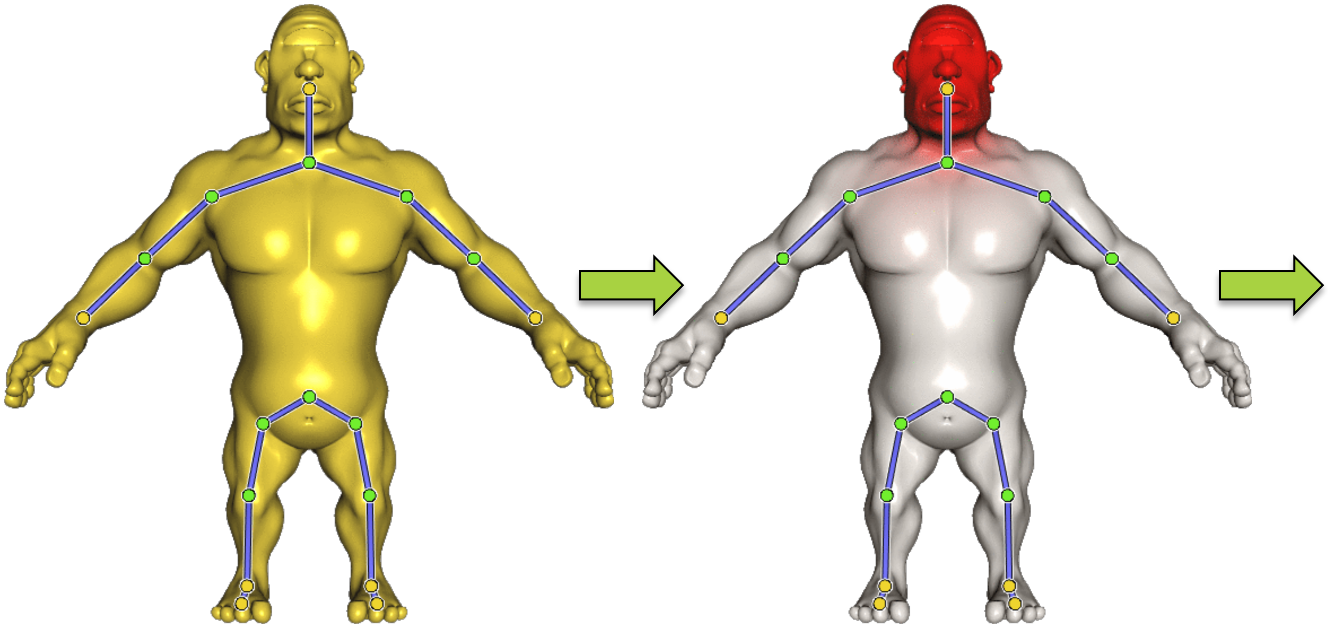


Linear Blend Skinning preferred for real-time performance



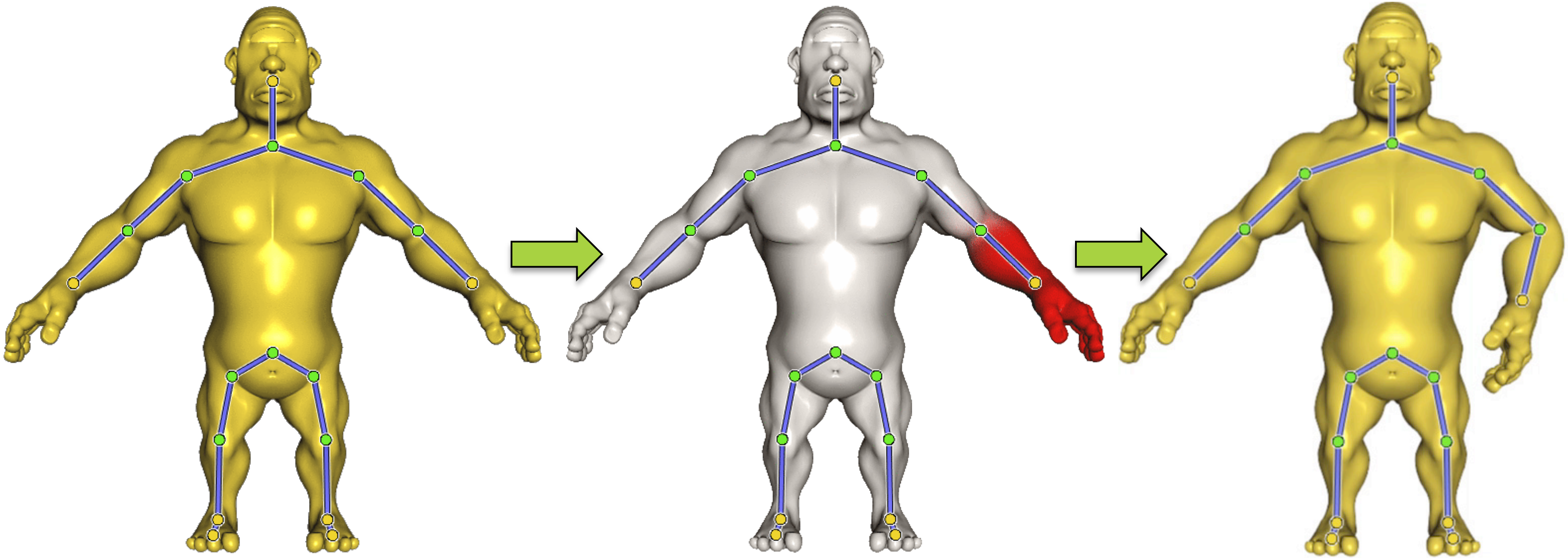
place skeleton in shape

Linear Blend Skinning preferred for real-time performance



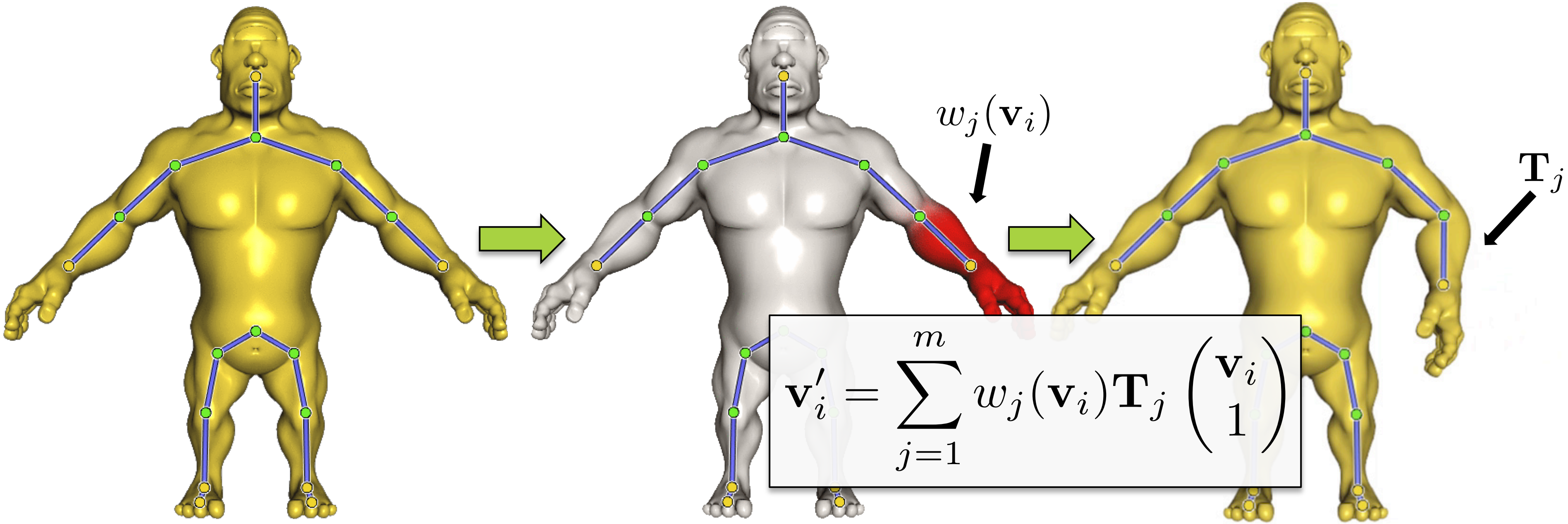
place skeleton in shape   compute/paint weights

Linear Blend Skinning preferred for real-time performance



place skeleton in shape   compute/paint weights   deform bones

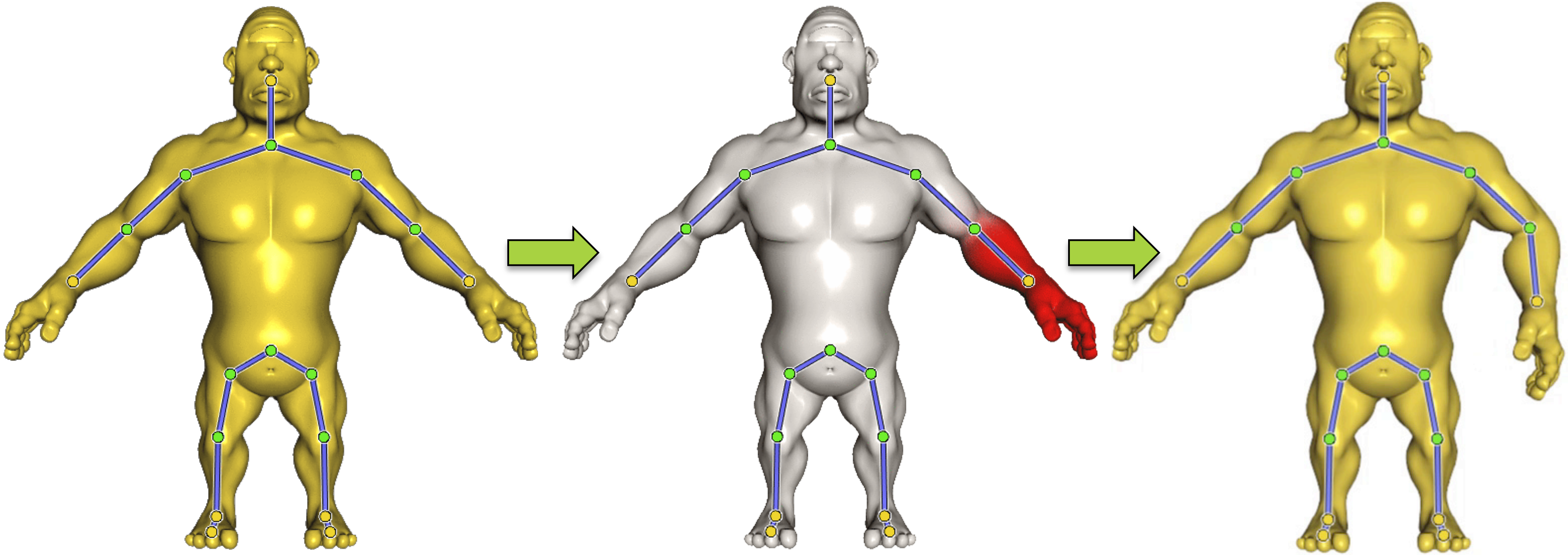
Linear Blend Skinning preferred for real-time performance



$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$

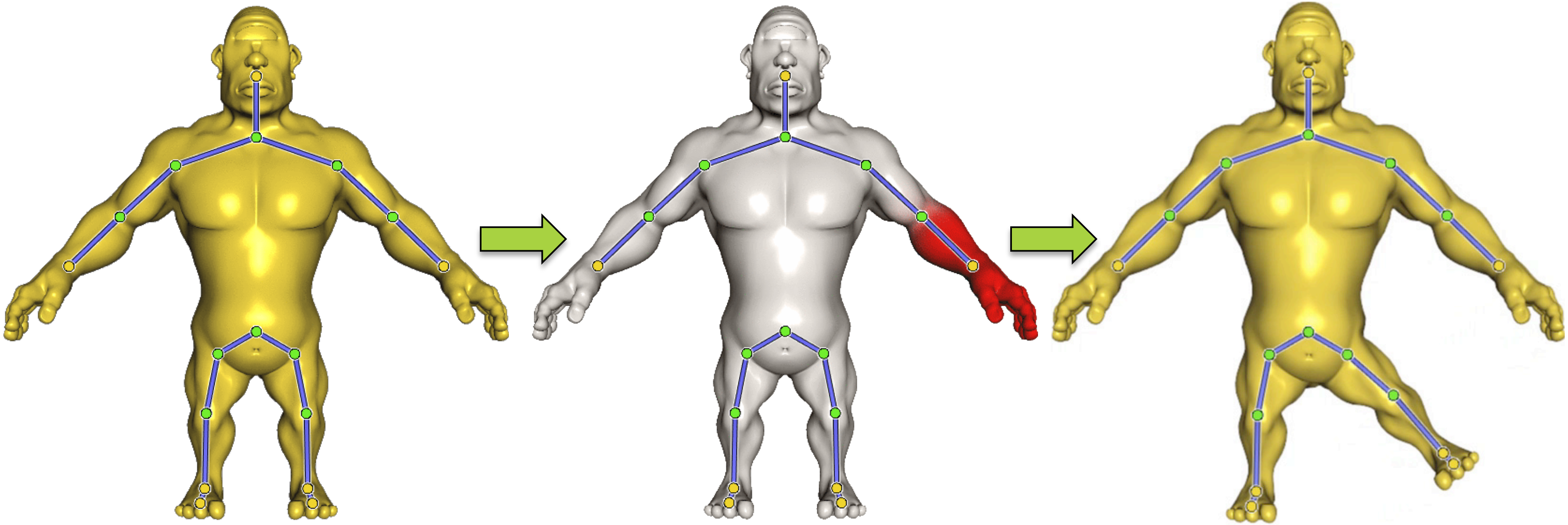
place skeleton in shape — compute/paint weights — deform bones

Linear Blend Skinning preferred for real-time performance



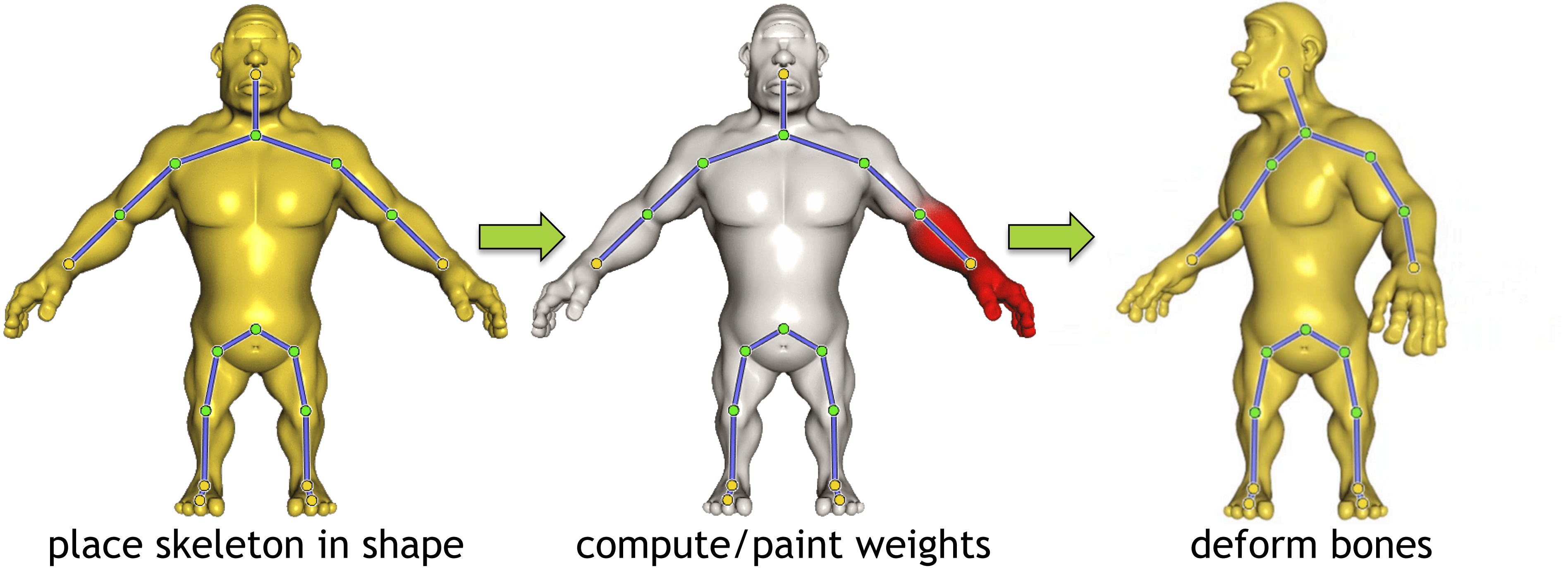
place skeleton in shape compute/paint weights deform bones

Linear Blend Skinning preferred for real-time performance

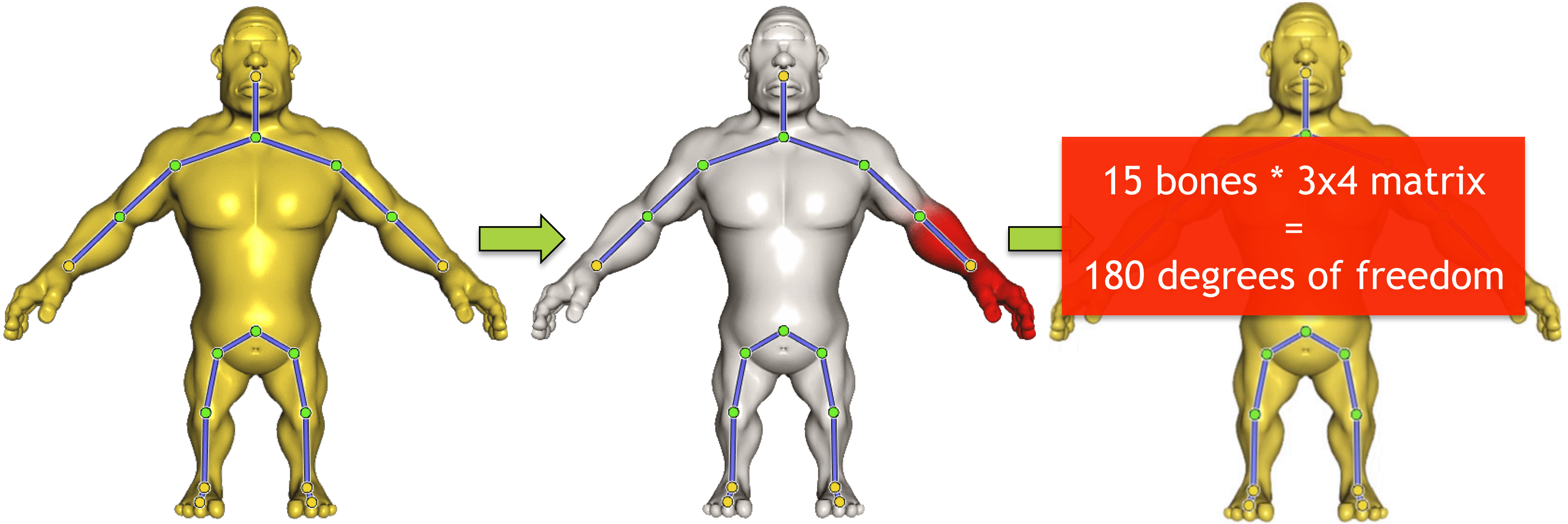


place skeleton in shape | compute/paint weights | deform bones

Linear Blend Skinning preferred for real-time performance



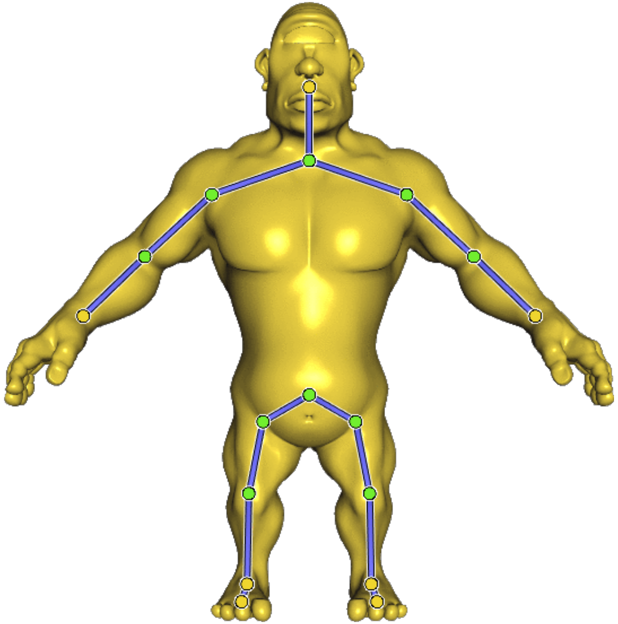
Linear Blend Skinning preferred for real-time performance



place skeleton in shape | compute/paint weights | deform bones

LBS generalizes to different handle types

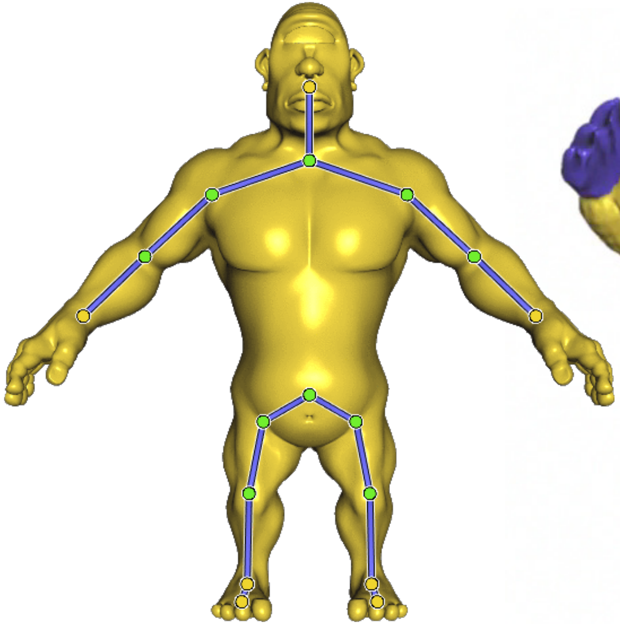
$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix}\mathbf{v}_i \\ 1\end{pmatrix}$$



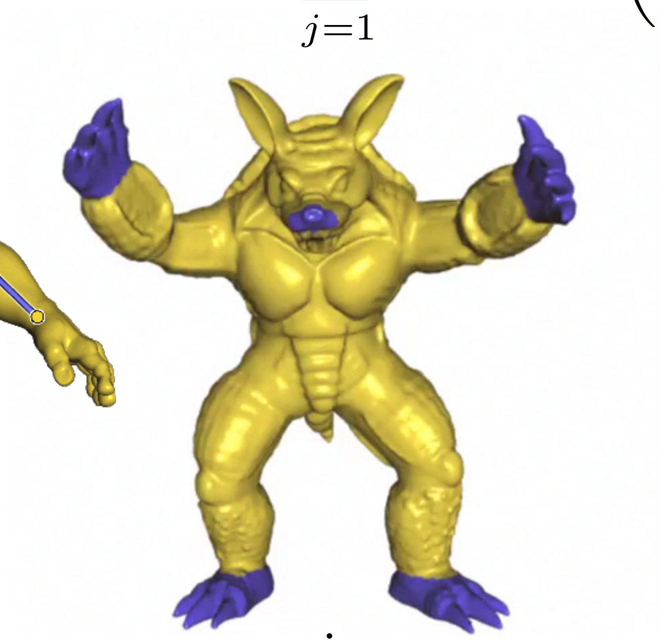
skeletons

LBS generalizes to different handle types

$$\mathbf{v}_i' = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$



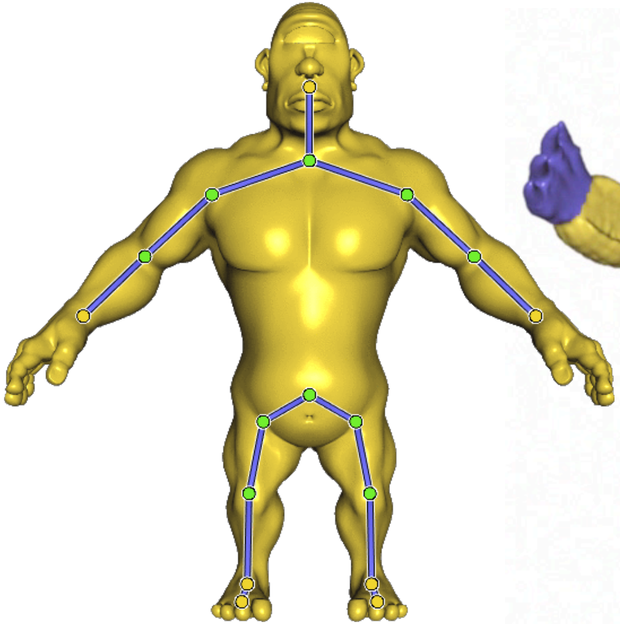
skeletons



regions

LBS generalizes to different handle types

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$

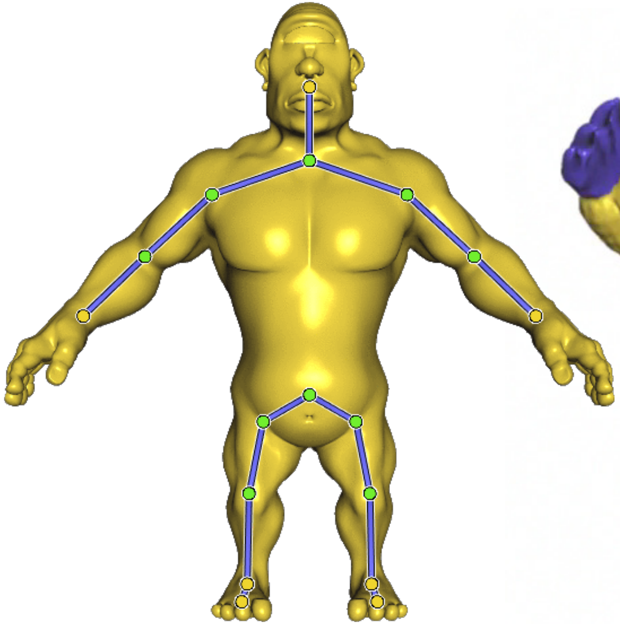


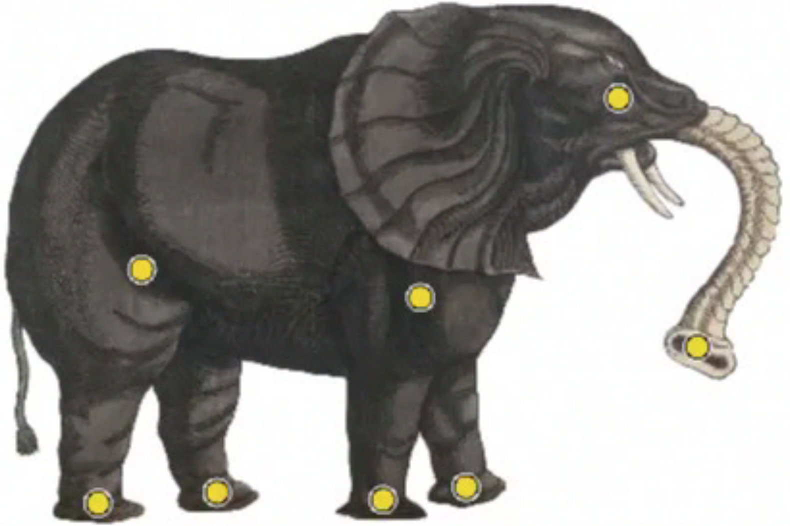
skeletons

regions

LBS generalizes to different handle types

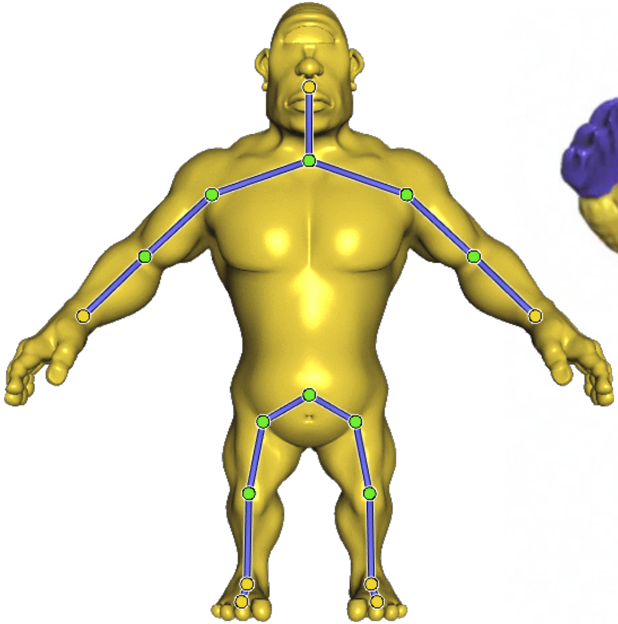
$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix}\mathbf{v}_i \\ 1\end{pmatrix}$$





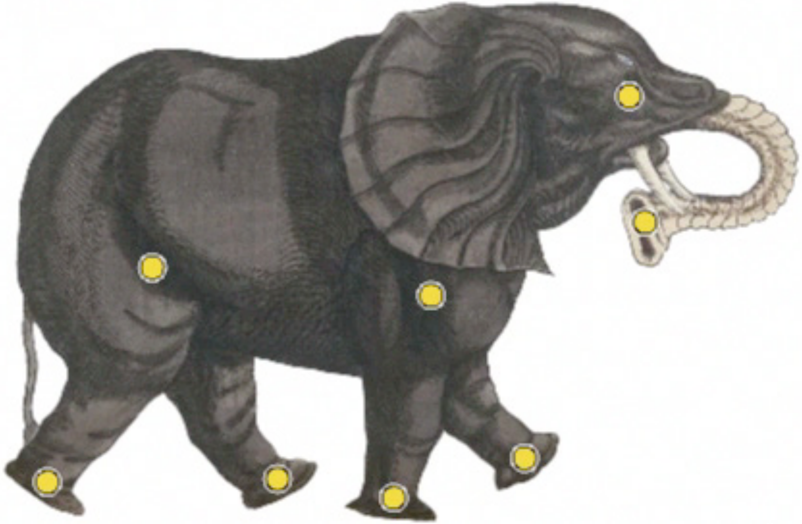
LBS generalizes to different handle types

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$



skeletons

regions



points

User specifies subset of parameters, optimize to find remaining ones

Full optimization

$$\underset{\mathbf{V}'}{\arg\min}\ E(\mathbf{V}')$$

Mesh vertex positions

User specifies subset of parameters, optimize to find remaining ones

Full optimization

$$\underset{\mathbf{V}'}{\arg\min}\ E(\mathbf{V}')$$

Reduced model

$$\mathbf{v}_i' = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix}\mathbf{v}_i \\ 1\end{pmatrix}$$

Skinning degrees of freedom

User specifies subset of parameters, optimize to find remaining ones

Full optimization

$$\arg\min_{\mathbf{V}'} E(\mathbf{V}')$$

Reduced model

$$\mathbf{v}_i' = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$

Matrix form

$$\mathbf{V}' = \mathbf{M}\mathbf{T}$$

User specifies subset of parameters, optimize to find remaining ones

Full optimization
$$\underset{\mathbf{V}'}{\arg\min}\ E(\mathbf{V}')$$

Reduced model
$$\mathbf{v}_i' = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix}\mathbf{v}_i \\ 1\end{pmatrix}$$

Matrix form
$$\mathbf{V}' = \mathbf{M}\mathbf{T}$$

Reduced optimization
$$\underset{\mathbf{T}}{\arg\min}\ E(\mathbf{M}\mathbf{T})$$

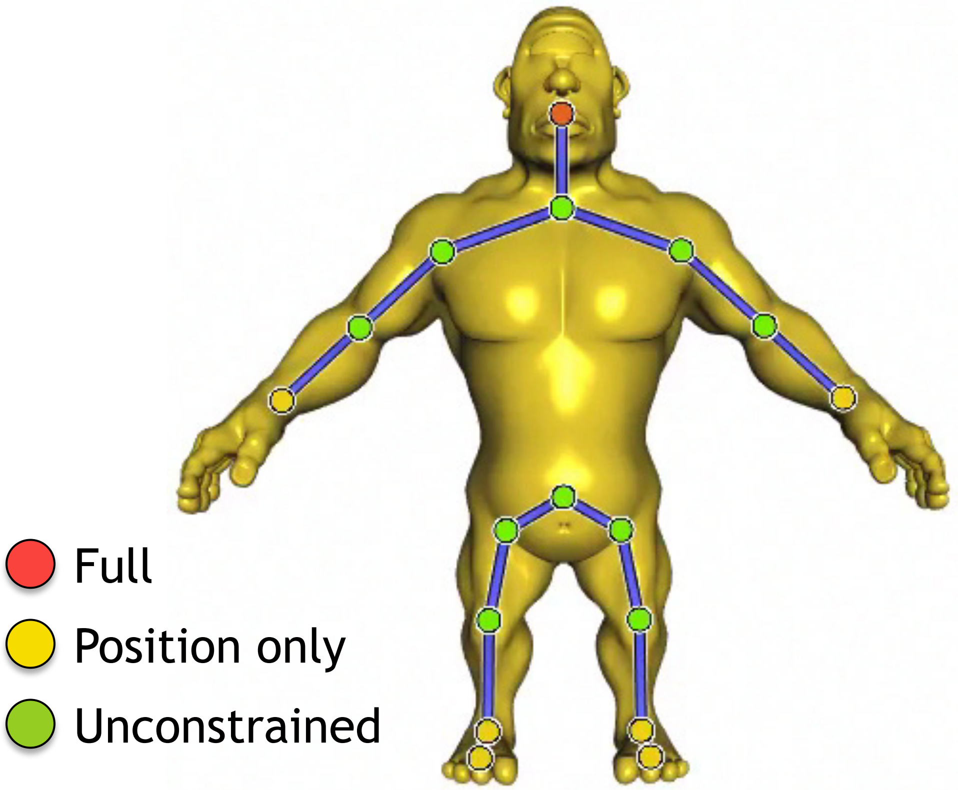
Enforce user constraints as linear equalities

Reduced optimization

$$\underset{\mathbf{T}}{\arg\min}\ E(\mathbf{MT})$$

User constraints

$$\underbrace{\begin{bmatrix} \mathbf{I}_{\text{full}} \\ \mathbf{M}_{\text{pos}} \end{bmatrix}}_{\mathbf{M}_{\text{eq}}} \mathbf{T} = \underbrace{\begin{bmatrix} \mathbf{T}_{\text{full}} \\ \mathbf{P}_{\text{pos}} \end{bmatrix}}_{\mathbf{P}_{\text{eq}}}$$



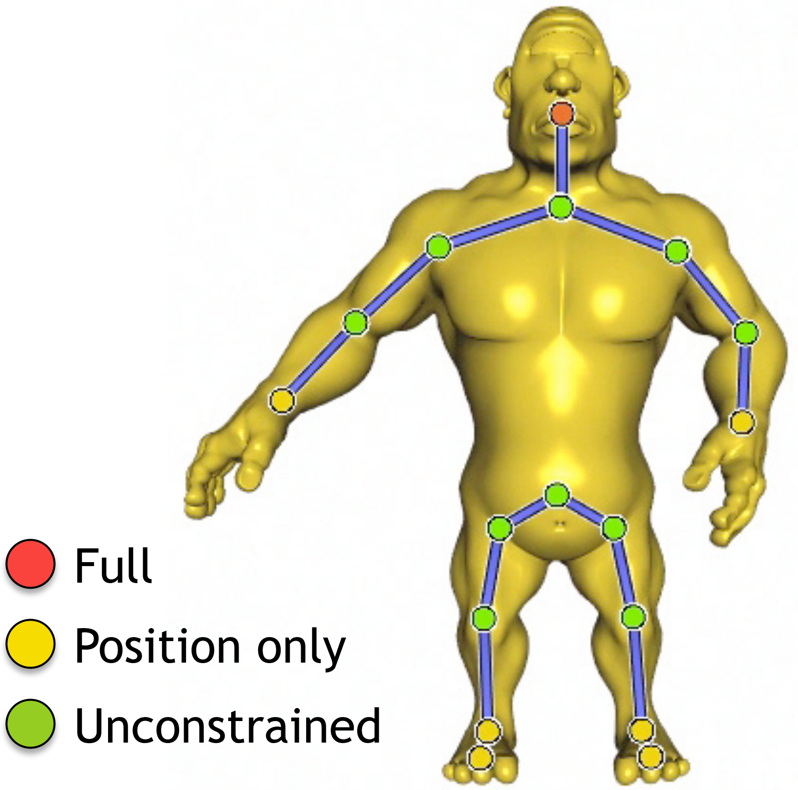
Enforce user constraints as linear equalities

Reduced optimization

$$\arg\min_{\mathbf{T}} \; E(\mathbf{MT})$$

User constraints

$$\underbrace{\begin{bmatrix}\mathbf{I}_{\text{full}}\\ \mathbf{M}_{\text{pos}}\end{bmatrix}}_{\mathbf{M}_{\text{eq}}}\mathbf{T} = \underbrace{\begin{bmatrix}\mathbf{T}_{\text{full}}\\ \mathbf{P}_{\text{pos}}\end{bmatrix}}_{\mathbf{P}_{\text{eq}}}$$



Enforce user constraints as linear equalities

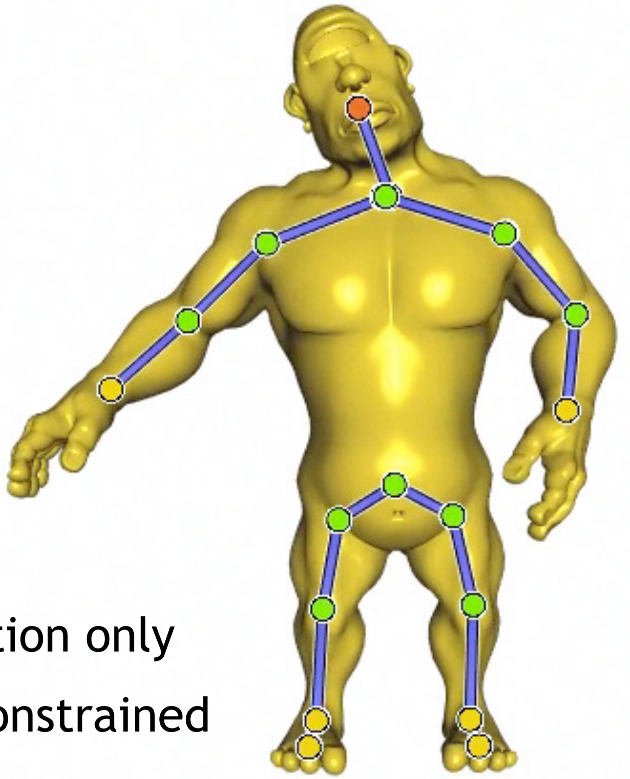
Reduced optimization

$$\arg\min_{\mathbf{T}} \; E(\mathbf{M}\mathbf{T})$$

User constraints

$$\underbrace{\begin{bmatrix} \mathbf{I}_{\text{full}} \\ \mathbf{M}_{\text{pos}} \end{bmatrix}}_{\mathbf{M}_{\text{eq}}} \mathbf{T} = \underbrace{\begin{bmatrix} \mathbf{T}_{\text{full}} \\ \mathbf{P}_{\text{pos}} \end{bmatrix}}_{\mathbf{P}_{\text{eq}}}$$

- Full
- Position only
- Unconstrained

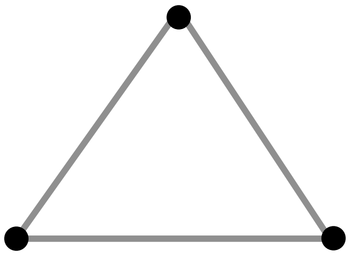


We reduce any *as-rigid-as-possible* energy

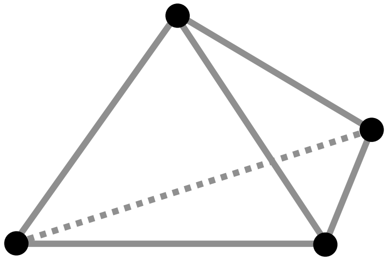
Full energies
$$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2} \sum_{k=1}^{r} \sum_{(i,j) \in \mathcal{E}_k} c_{ijk} \| (\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k (\mathbf{v}_i - \mathbf{v}_j) \|^2$$

We reduce any *as-rigid-as-possible* energy

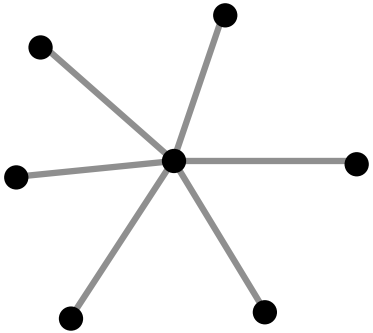
Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2} \sum_{k=1}^{r} \sum_{(i,j)\in\mathcal{E}_k} c_{ijk} \|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$



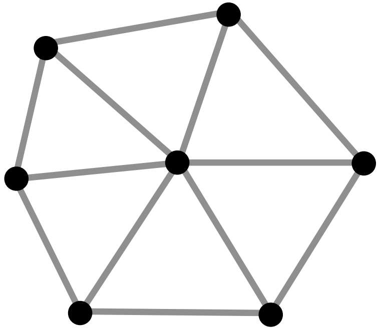
triangles
Liu et al. 08



tetrahedra
Chao et al. 10



"spokes"
Sorkine & Alexa 07



"spokes and rims"
Chao et al. 10

We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$

Local/Global optimization

Global step: Fix $\mathbf{R}$, minimize with respect to $\mathbf{V}'$

Local step: Fix $\mathbf{V}'$, minimize with respect to $\mathbf{R}$

We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2} \sum_{k=1}^{r} \sum_{(i,j) \in \mathcal{E}_k} c_{ijk} \| (\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k (\mathbf{v}_i - \mathbf{v}_j) \|^2$$

Local/Global optimization

precompute ↓

Global step: large, sparse linear solve $\mathbf{V}' = \mathbf{A}^{-1}\mathbf{b}$

Local step: Fix $\mathbf{V}'$, minimize with respect to $\mathbf{R}$

We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$

Local/Global optimization

Global step: large, sparse linear solve $\mathbf{V}' = \mathbf{A}^{-1}\mathbf{b}$

Local step: 3x3 SVD for each rotation in $\mathbf{R}$

We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2} \sum_{k=1}^{r} \sum_{(i,j)\in\mathcal{E}_k} c_{ijk} \| (\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j) \|^2$$

Local/Global optimization

precompute

Global step: small, dense linear solve $\mathbf{T} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$

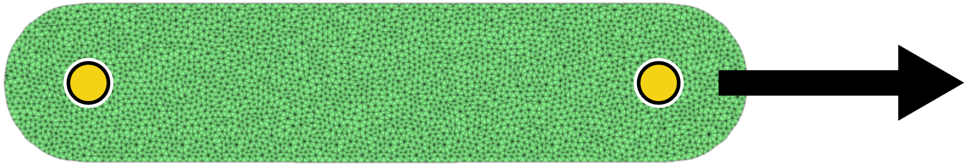
Local step: 3x3 SVD for each rotation in $\mathbf{R}$

Substitute

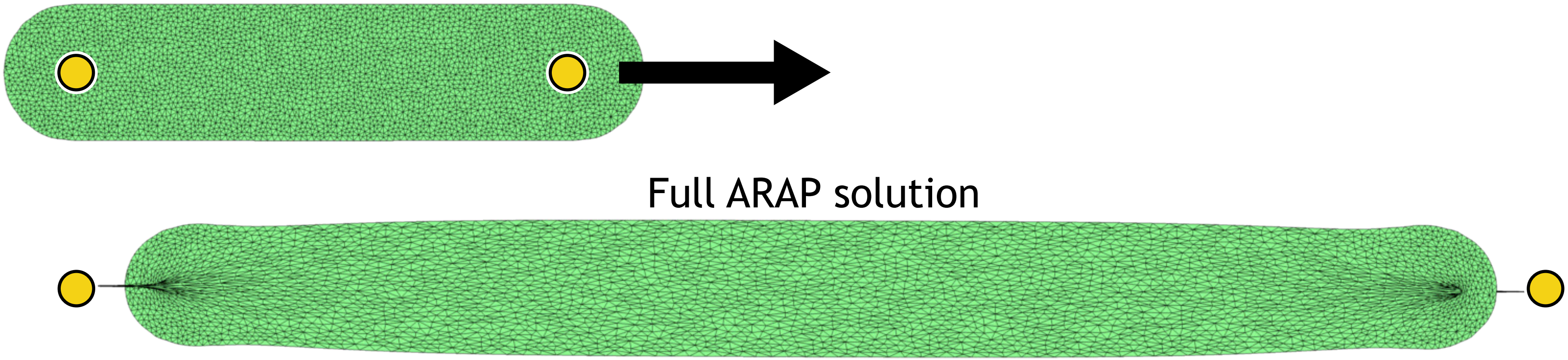
$$\mathbf{V}' = \mathbf{M}\mathbf{T}$$

Similar to:
[Huang et al. 06]
[Der et al. 06]
[Au et al. 07]
[Hildebrandt et al. 12]

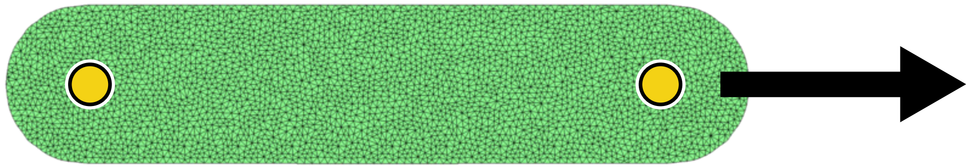
Direct reduction of elastic energies brings speed up and regularization…



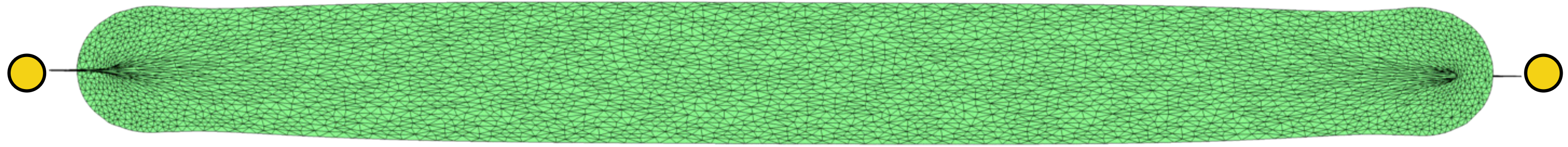
Direct reduction of elastic energies brings speed up and regularization…



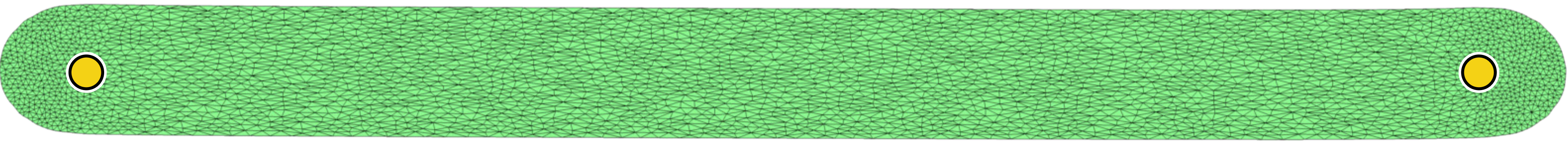
Direct reduction of elastic energies brings speed up and regularization...



Full ARAP solution



Our smooth subspace solution $\mathbf{V}' = \mathbf{M}\mathbf{T}$



We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2} \sum_{k=1}^{r} \sum_{(i,j)\in\mathcal{E}_k} c_{ijk} \|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$

Local/Global optimization

Global step: small, dense linear solve $\mathbf{T} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$

Local step: 3x3 SVD for each rotation in $\mathbf{R}$

But #rotations ~ full mesh discretization

Substitute

$$\mathbf{V}' = \mathbf{MT}$$

We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2} \sum_{k=1}^{r} \sum_{(i,j) \in \mathcal{E}_k} c_{ijk} \|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$

Local/Global optimization

Global step: small, dense linear solve $\mathbf{T} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$

Local step: 3x3 SVD for each rotation in $\mathbf{R}$

Substitute

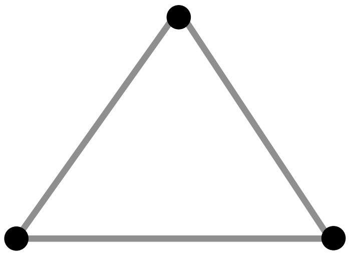
$$\mathbf{V}' = \mathbf{M}\mathbf{T}$$

Cluster

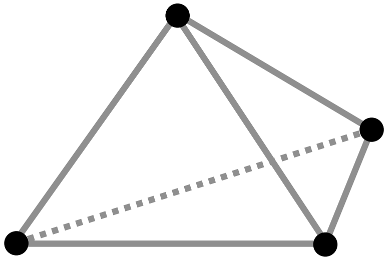
$$\mathcal{E}_k$$

Rotation evaluations may be reduced by clustering in *weight space*

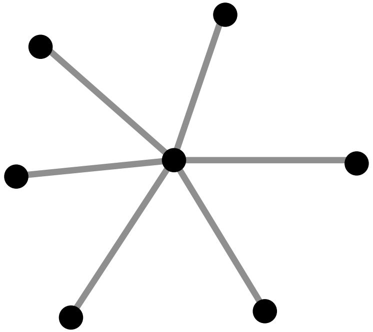
Full energies $$E(\mathbf{V}',\mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$



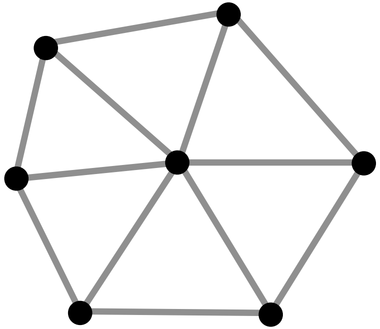
triangles
Liu et al. 08



tetrahedra
Chao et al. 10



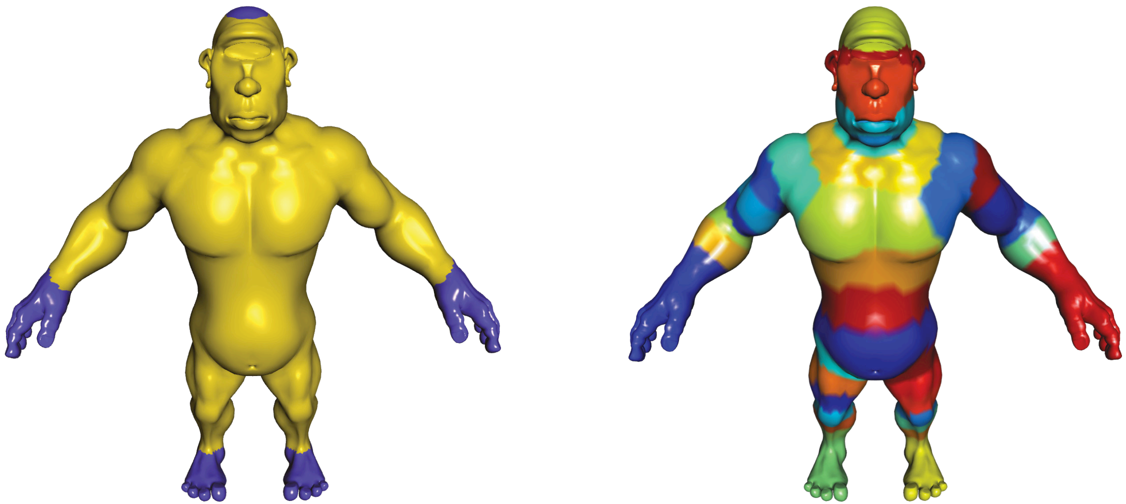
"spokes"
Sorkine & Alexa 07



"spokes and rims"
Chao et al. 10

Rotation evaluations may be reduced by k-means clustering in *weight space*

Full energies $$E(\mathbf{V}',\mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i-\mathbf{v}'_j)-\mathbf{R}_k(\mathbf{v}_i-\mathbf{v}_j)\|^2$$

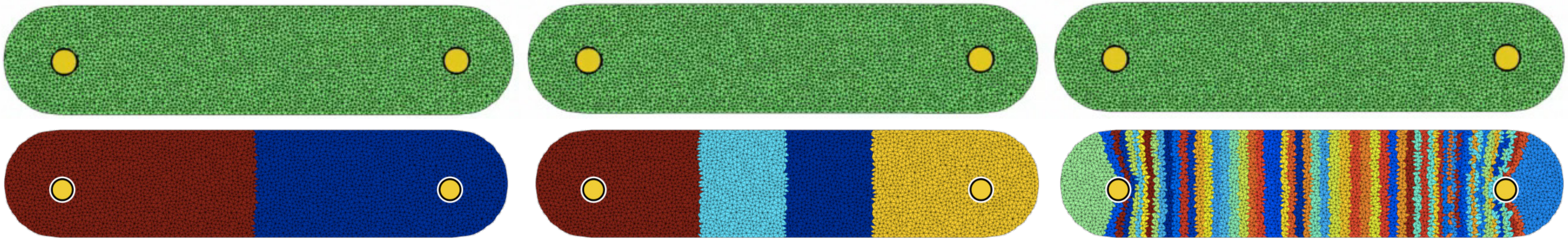


weight space

$$\mathbf{x}_j = \begin{bmatrix} w_1(\mathbf{v}_j) \\ w_2(\mathbf{v}_j) \\ \vdots \\ w_m(\mathbf{v}_j) \end{bmatrix}$$

Rotation evaluations may be reduced by clustering in *weight space*

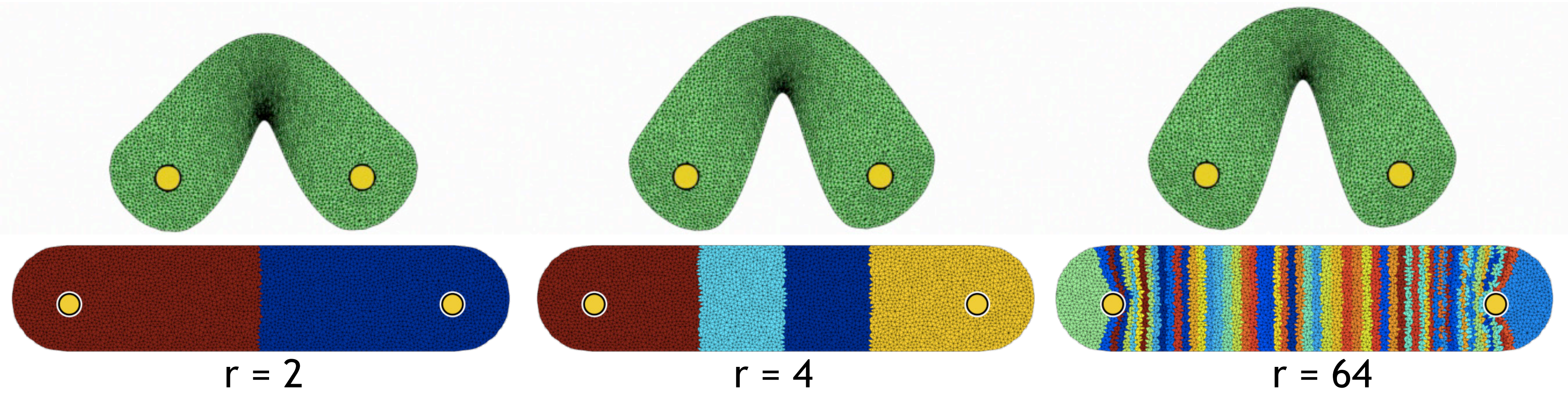
Full energies
$$E(\mathbf{V}',\mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i-\mathbf{v}'_j)-\mathbf{R}_k(\mathbf{v}_i-\mathbf{v}_j)\|^2$$



r = 2 r = 4 r = 64

Rotation evaluations may be reduced by clustering in *weight space*

Full energies $$E(\mathbf{V}',\mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i-\mathbf{v}'_j)-\mathbf{R}_k(\mathbf{v}_i-\mathbf{v}_j)\|^2$$



We reduce any *as-rigid-as-possible* energy

Full energies $$E(\mathbf{V}', \mathbf{R}) = \frac{1}{2}\sum_{k=1}^{r}\sum_{(i,j)\in\mathcal{E}_k} c_{ijk}\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_k(\mathbf{v}_i - \mathbf{v}_j)\|^2$$

Local/Global optimization

Global step: small, dense linear solve $\mathbf{T} = \tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}}$

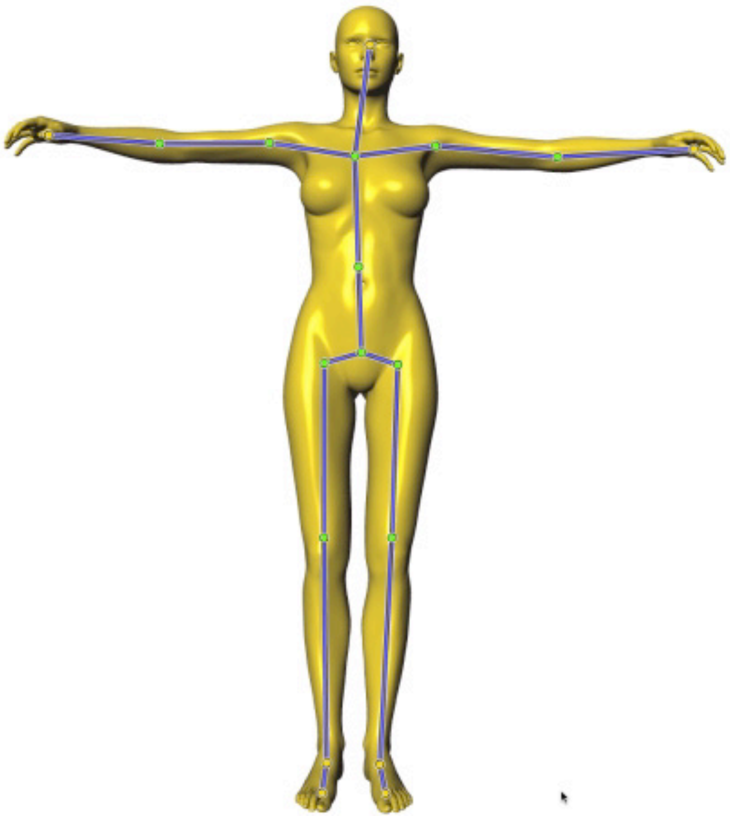
Local step: 3x3 SVD for each rotation in $\mathbf{R}$

#rotations ~ #T,
independent of full mesh resolution

Substitute
$$\mathbf{V}' = \mathbf{M}\mathbf{T}$$

Cluster
$$\mathcal{E}_k$$

Real-time automatic degrees of freedom



Real-time automatic degrees of freedom

With more and more user constraints we fall back to standard skinning

With more and more user constraints we fall back to standard skinning

With more and more user constraints we fall back to standard skinning

With more and more user constraints we fall back to standard skinning

Extra weights would expand subspace…

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix} \mathbf{v}_i \\ 1 \end{pmatrix}$$

$$\mathbf{V}' = \mathbf{MT}$$

Extra weights would expand subspace…

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix}\mathbf{v}_i\\1\end{pmatrix} + \sum_{k=1}^{m_{\text{extra}}} w_k(\mathbf{v}_i)\mathbf{T}_k \begin{pmatrix}\mathbf{v}_i\\1\end{pmatrix}$$

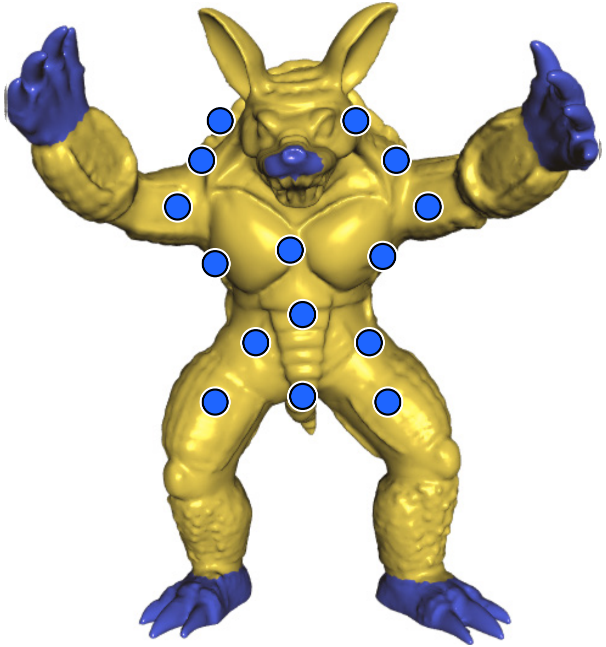
$$\mathbf{V}' = \mathbf{M}\mathbf{T}$$

Extra weights would expand subspace…

$$\mathbf{v}_i' = \sum_{j=1}^{m} w_j(\mathbf{v}_i)\mathbf{T}_j \begin{pmatrix}\mathbf{v}_i \\ 1\end{pmatrix} + \sum_{k=1}^{m_{\text{extra}}} w_k(\mathbf{v}_i)\mathbf{T}_k \begin{pmatrix}\mathbf{v}_i \\ 1\end{pmatrix}$$

$$\mathbf{V}' = \mathbf{M}\mathbf{T} + \mathbf{M}_{\text{extra}}\mathbf{T}_{\text{extra}}$$

Overlapping b-spline “bumps” in weight space



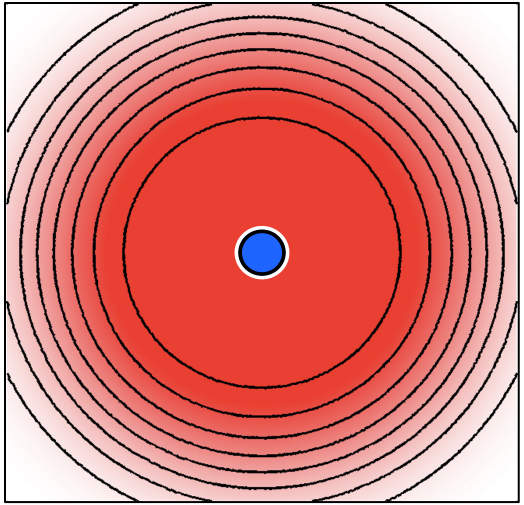
farthest point sampling

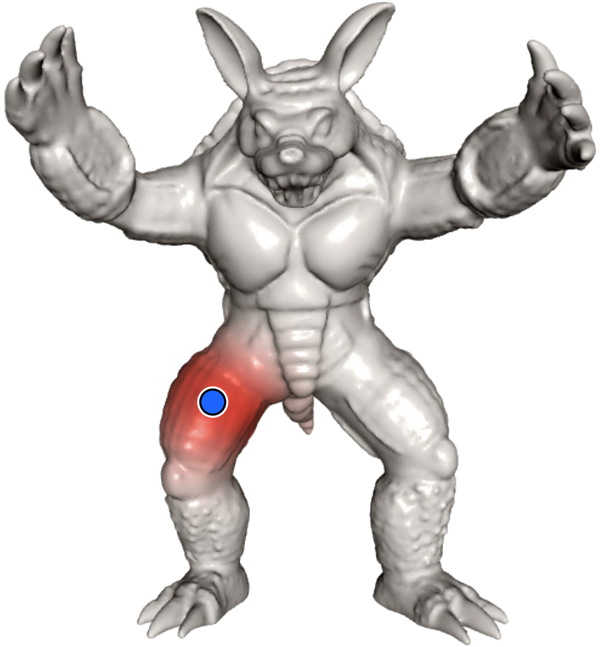
weight space

$$\mathbf{x}_j = \begin{bmatrix} w_1(\mathbf{v}_j) \\ w_2(\mathbf{v}_j) \\ \vdots \\ w_m(\mathbf{v}_j) \end{bmatrix}$$

Overlapping b-spline "bumps" in weight space

in weight space





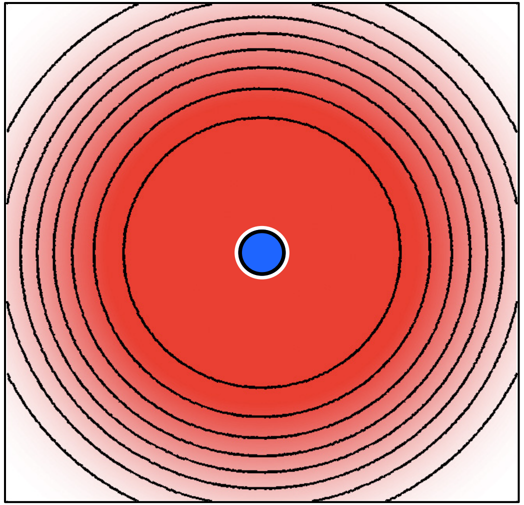
weight space

$$\mathbf{x}_j = \begin{bmatrix} w_1(\mathbf{v}_j) \\ w_2(\mathbf{v}_j) \\ \vdots \\ w_m(\mathbf{v}_j) \end{bmatrix}$$

b-spline basis parameterized by distance in weight space

Overlapping b-spline "bumps" in weight space

in weight space



weight space

$$\mathbf{x}_j = \begin{bmatrix} w_1(\mathbf{v}_j) \\ w_2(\mathbf{v}_j) \\ \vdots \\ w_m(\mathbf{v}_j) \end{bmatrix}$$

b-spline basis parameterized by distance in weight space

Extra weights expand deformation subspace

no extra weights

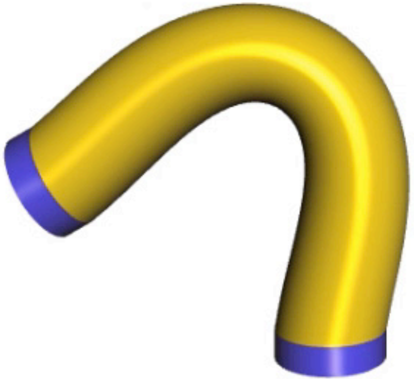
15 extra weights

Extra weights expand deformation subspace

no extra weights

15 extra weights

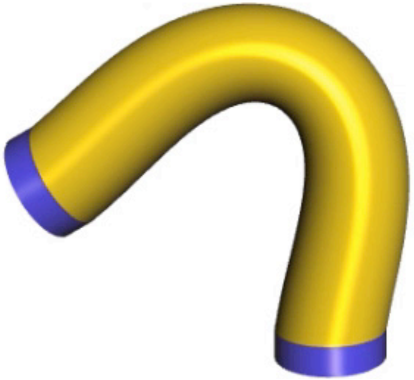
Subspace now rich enough for fast variational modeling



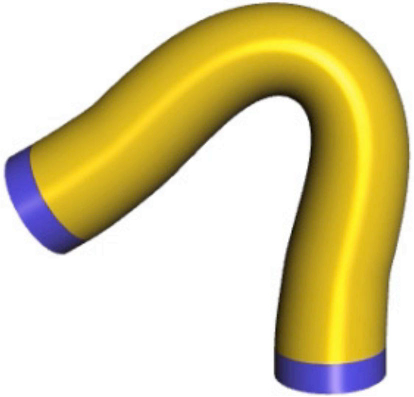
Full non-linear optimization
[Botsch et al. 2006]

Our reduced method

Subspace now rich enough for fast variational modeling

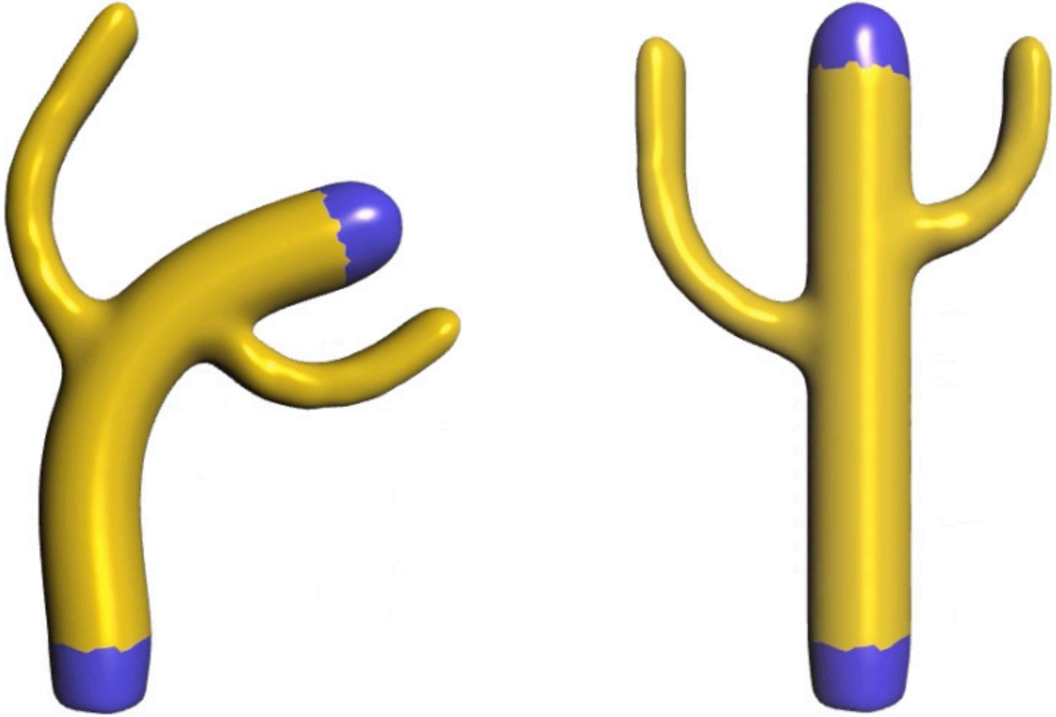


Full non-linear optimization
[Botsch et al. 2006]



Our reduced method

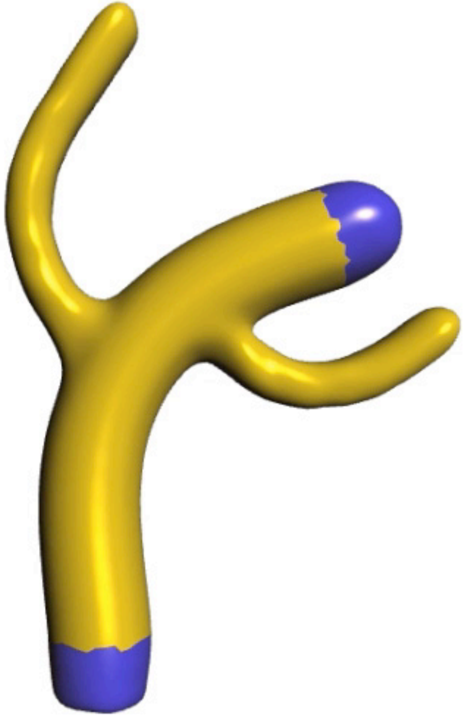
Subspace now rich enough for fast variational modeling



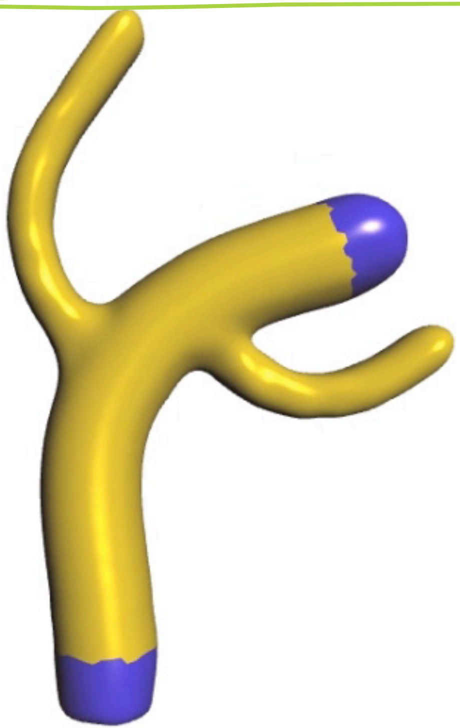
Full non-linear optimization [Botsch et al. 2006]

Our reduced method

Subspace now rich enough for fast variational modeling

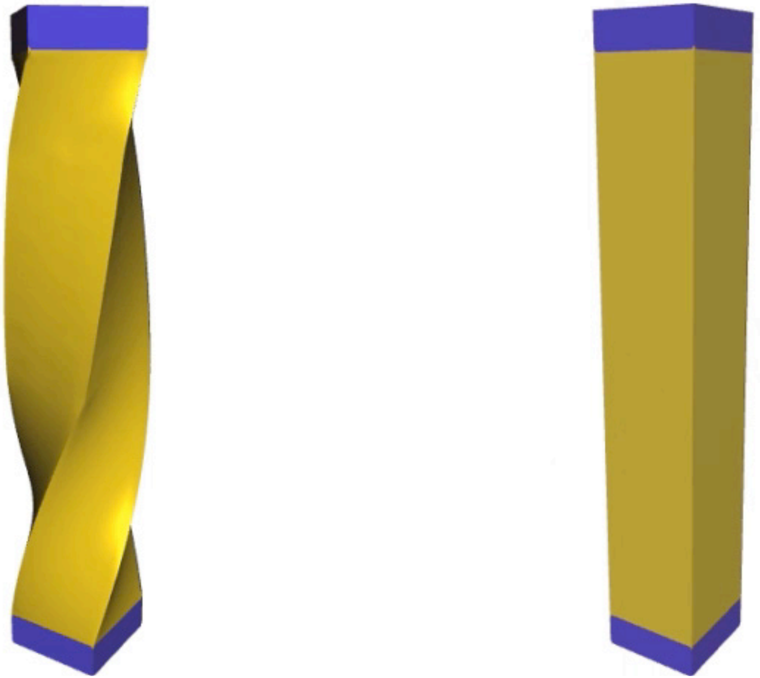


Full non-linear optimization
[Botsch et al. 2006]



Our reduced method

Subspace now rich enough for fast variational modeling



Full non-linear optimization [Botsch et al. 2006]

Our reduced method

Subspace now rich enough for fast variational modeling

Full non-linear optimization
[Botsch et al. 2006]

Our reduced method

Final algorithm is simple and FAST

Precomputation per shape+rig	*For a 50K triangle mesh:*
- Compute any additional weights	*12 seconds*
- Construct, prefactor system matrices	*2.7 seconds*

Final algorithm is simple and FAST

Precomputation per shape+rig	*For a 50K triangle mesh:*
- Compute any additional weights	*12 seconds*
- Construct, prefactor system matrices	*2.7 seconds*

Precomputation when switching constraint type

- *Re*-factor global step system	*6 milliseconds*

Final algorithm is simple and FAST

Precomputation per shape+rig *For a 50K triangle mesh:*

- Compute any additional weights *12 seconds*
- Construct, prefactor system matrices *2.7 seconds*

Precomputation when switching constraint type

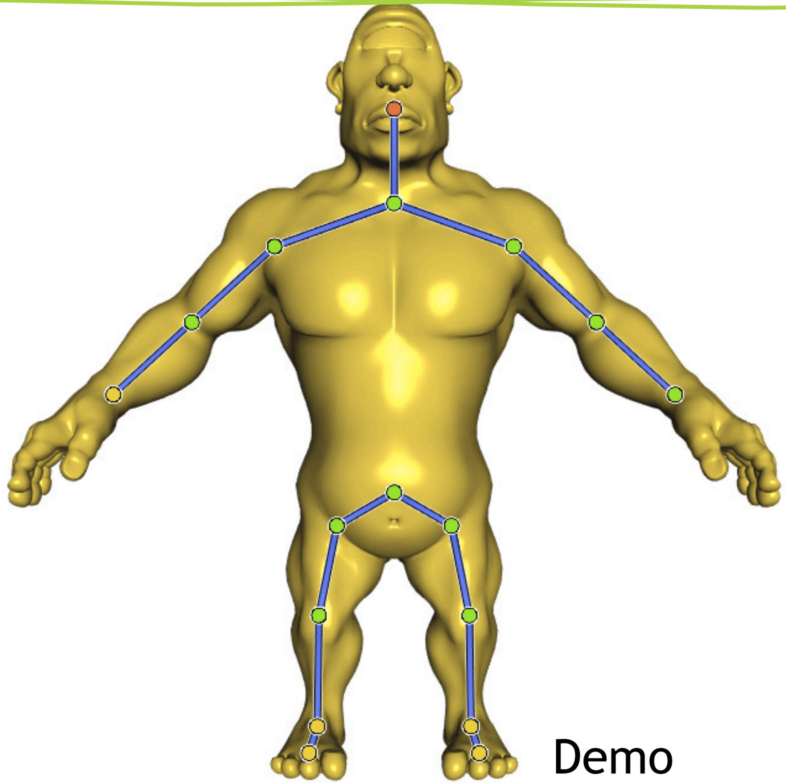
- *Re*-factor global step system *6 milliseconds*

~30 iterations *22 microseconds*

global: #weights by #weights linear solve

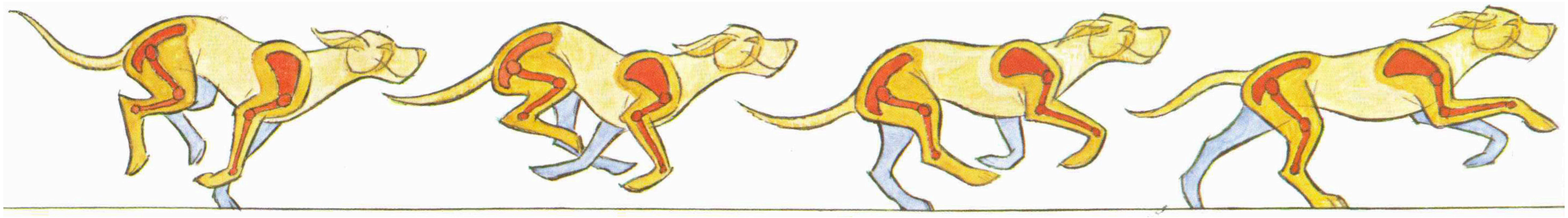
local: #rotations SVDs [McAdams et al. 2011]

Lightning FAST automatic skinning transformations



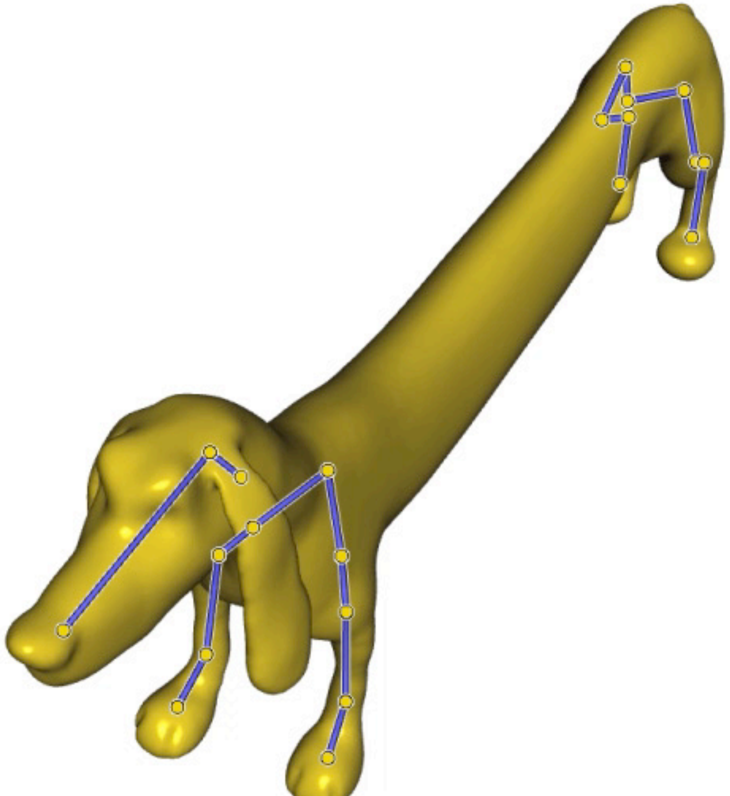
Demo

Extra weights and disjoint skeletons make flexible control easy

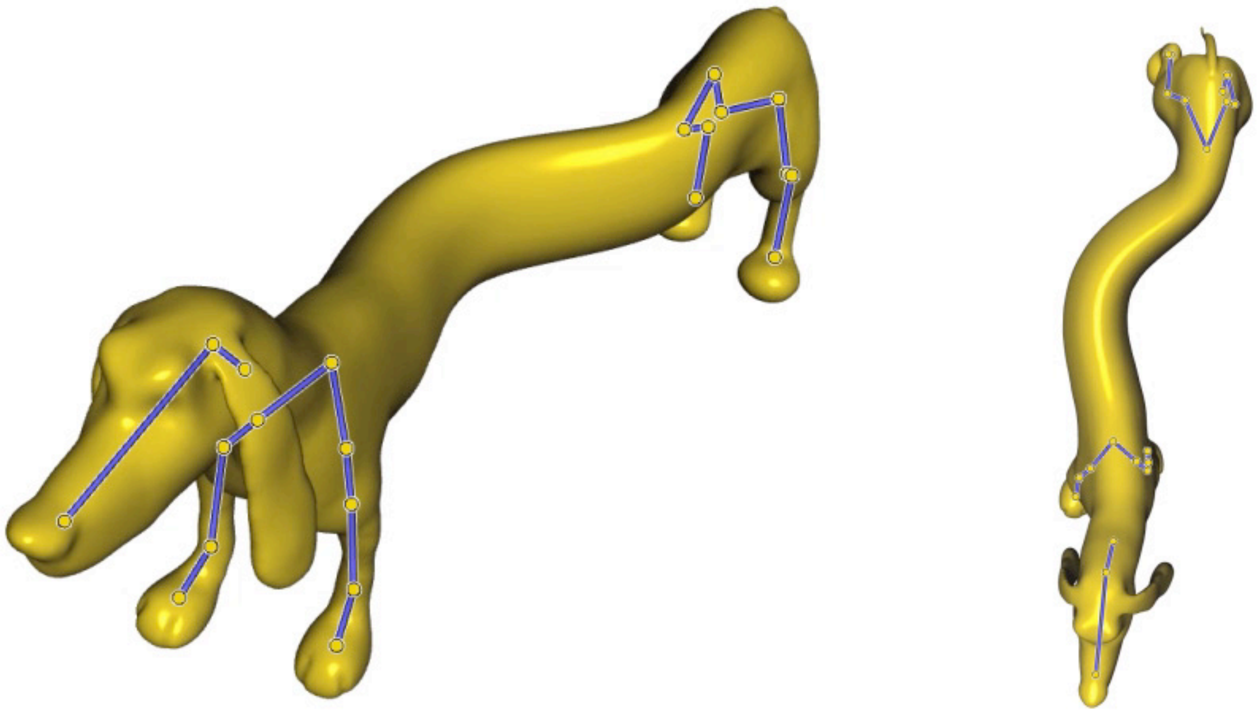


From *Cartoon Animation by Preston Blair*

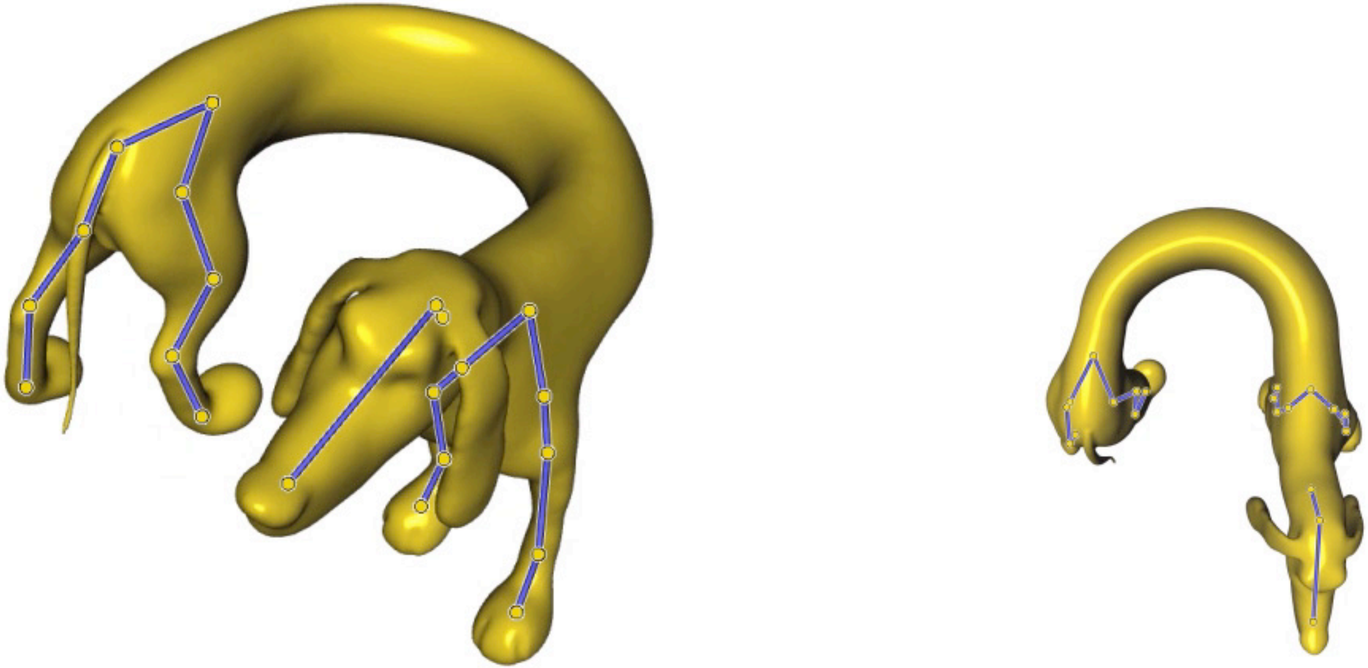
Extra weights and disjoint skeletons make flexible control easy



Extra weights and disjoint skeletons make flexible control easy



Extra weights and disjoint skeletons make flexible control easy



Our reduction preserves nature of different energies, at no extra cost

Surface ARAP

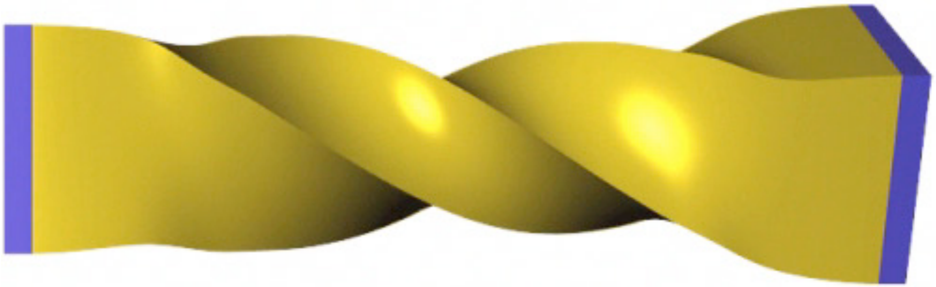
$$\mathbf{V}'_{\mathrm{surf}} = \mathbf{M}_{\mathrm{surf}} T$$

Volumetric ARAP

$$\mathbf{V}'_{\mathrm{vol}} = \mathbf{M}_{\mathrm{vol}} T$$

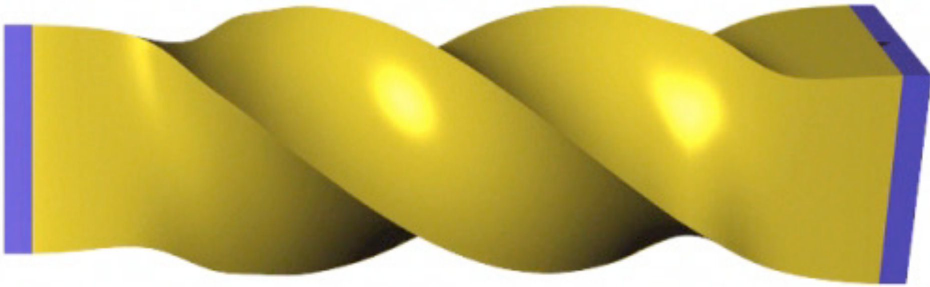
Our reduction preserves nature of different energies, at no extra cost

Surface ARAP



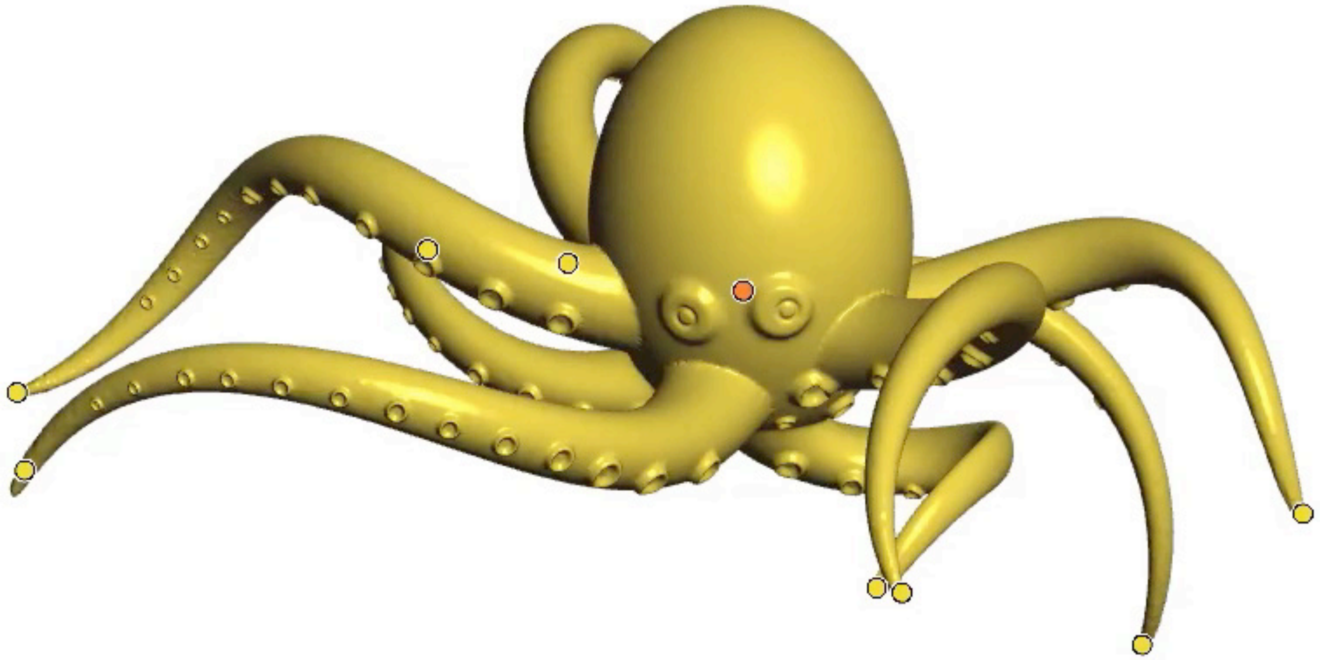
$$\mathbf{V}'_{\text{surf}} = \mathbf{M}_{\text{surf}} T$$

Volumetric ARAP

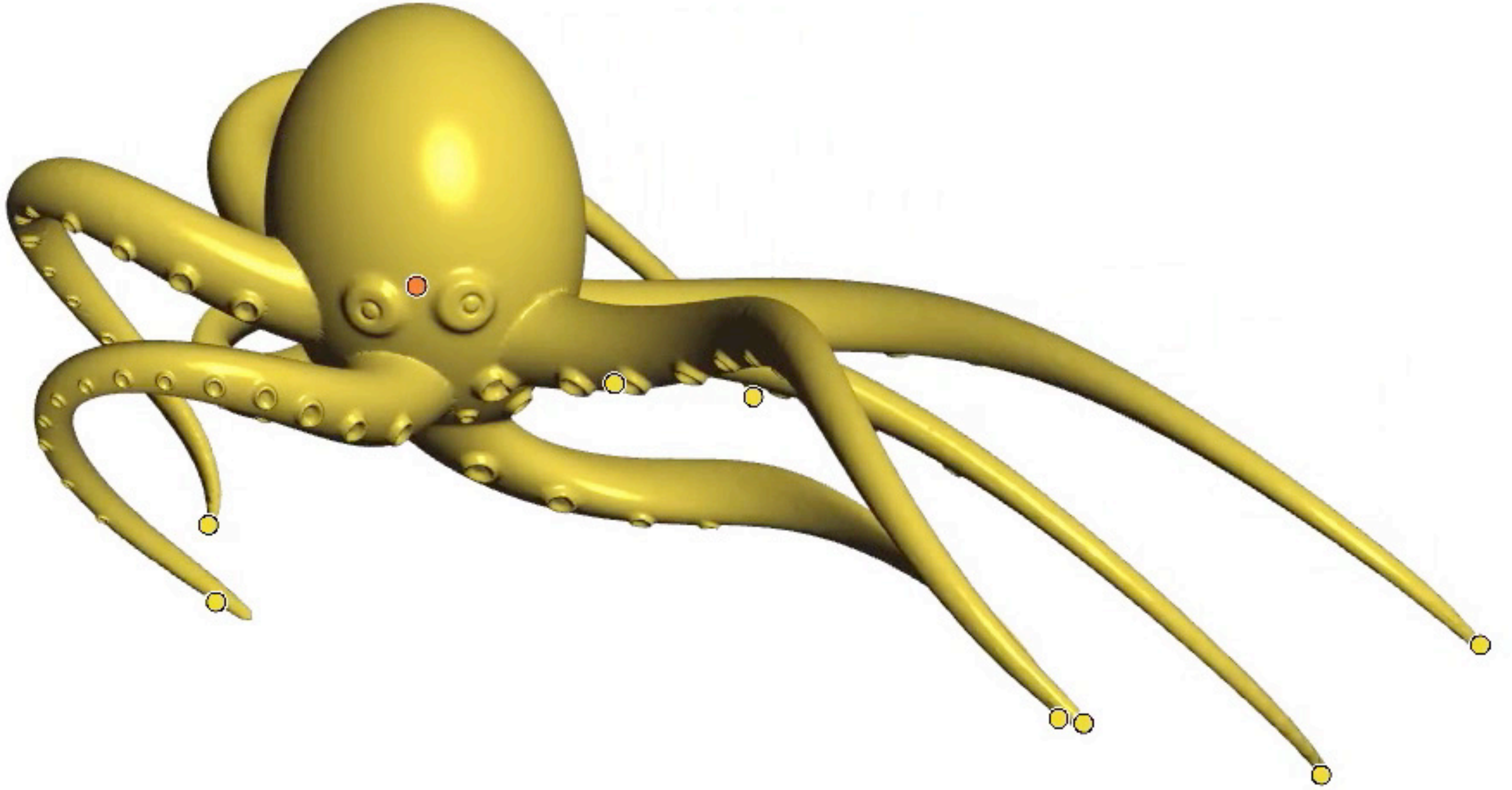


$$\mathbf{V}'_{\text{vol}} = \mathbf{M}_{\text{vol}} T$$

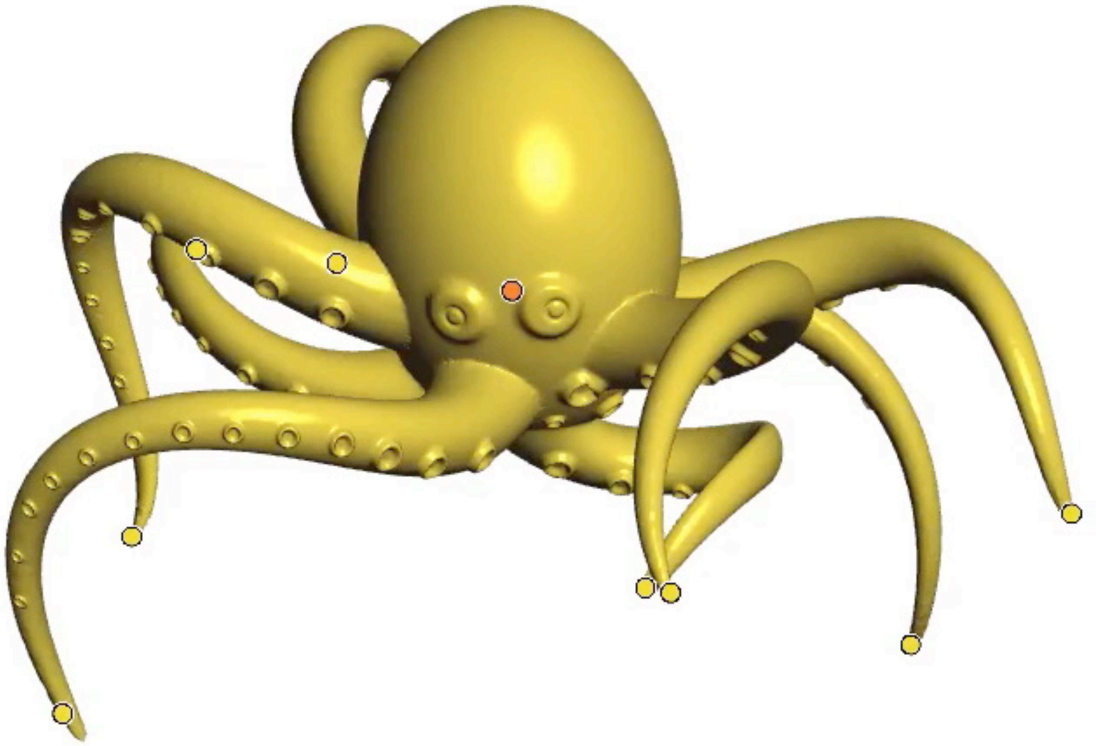
Simple drag-only interface for point handles



Simple drag-only interface for point handles



Simple drag-only interface for point handles



Skinning rig enables FAST deformation

- Substitute $\mathbf{V}' = \mathbf{M}\mathbf{T}$ to reduce DOFs

Skinning rig enables FAST deformation

- Substitute $\mathbf{V}' = \mathbf{M}\mathbf{T}$ to reduce DOFs
- Cluster rotations to reduce energy eval.

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Skinning rig enables FAST deformation

- Substitute $\mathbf{V}' = \mathbf{M}\mathbf{T}$ to reduce DOFs
- Cluster rotations to reduce energy eval.
- Additional weights to expand subspace

Skinning rig enables FAST deformation

- Substitute $\mathbf{V}' = \mathbf{M}\mathbf{T}$ to reduce DOFs
- Cluster rotations to reduce energy eval.
- Additional weights to expand subspace

Each innovation takes advantage of input skinning rig

Future work and discussion

- Alternative additional weights: sparsity?
- Joint limits, balance, etc.

Acknowledgements

We are grateful to Peter Schröder, Emily Whiting, and Maurizio Nitti.

We thank Eftychios Sifakis for his open source fast 3×3 SVD code.

This work was supported in part by an SNF award 200021_137879 and by a gift from Adobe Systems.

Fast Automatic Skinning Transformations

http://igl.ethz.ch/projects/fast

Alec Jacobson (jacobson@inf.ethz.ch),
Ilya Baran, Ladislav Kavan, Jovan Popović, Olga Sorkine