Hyperspectral Inverse Skinning

Songrun Liu George Mason University
Jianchao Tan George Mason University
Zhigang Deng University of Houston
Yotam Gingold George Mason University

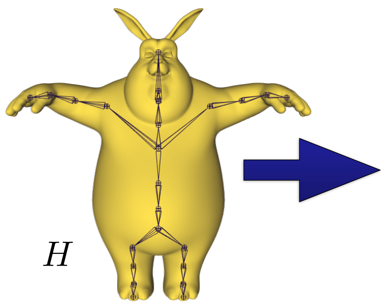
George Mason University
CraGL
Creativity and Graphics Lab

Linear Blend Skinning (LBS)

$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

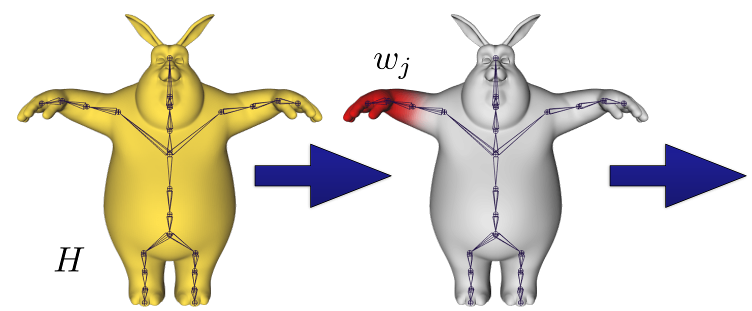
This presentation is about Linear Blend Skinning. LBS is an animation technique. We choose a set of handles H, such as the bones of a character. The handles are chosen by the designer to be convenient for whatever deformation they want to perform.

Linear Blend Skinning (LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

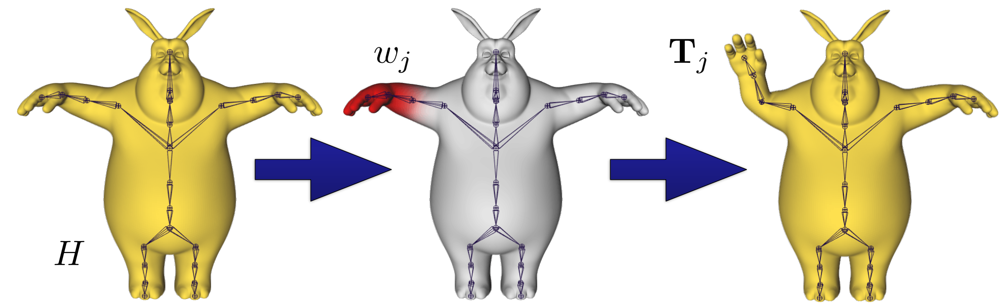
Linear Blend Skinning (LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

Each handle has a pointwise per-vertex weight. These weights are fixed for the entire animation. Different vertices have different weights.

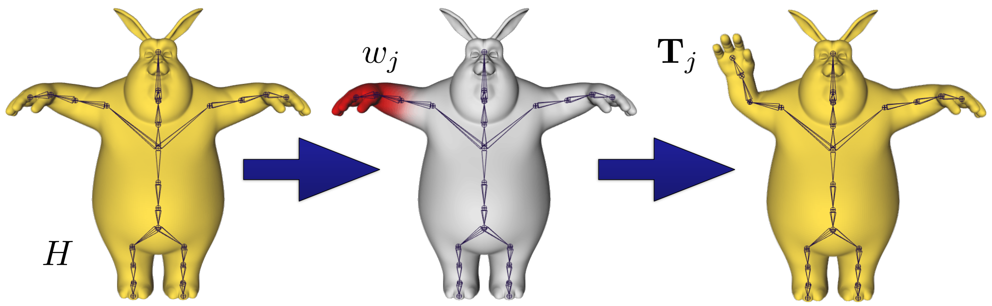
Linear Blend Skinning (LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

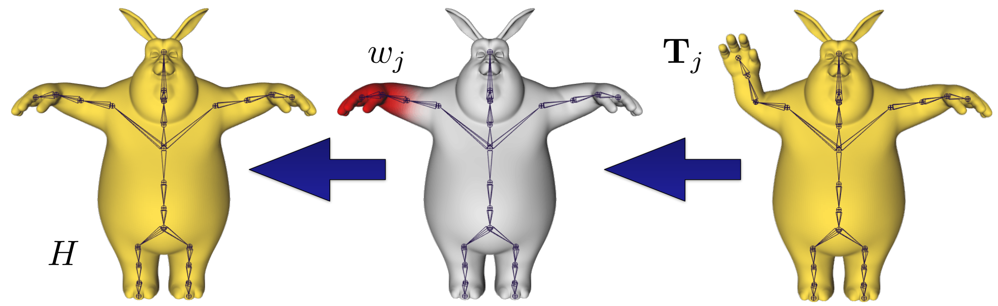
The designer adjusts the transformation matrix associated with each handle, and the shape moves.
LBS is standard in many parts of computer graphics, particularly for real-time animation.

Linear Blend Skinning (LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

Inverse Linear Blend Skinning



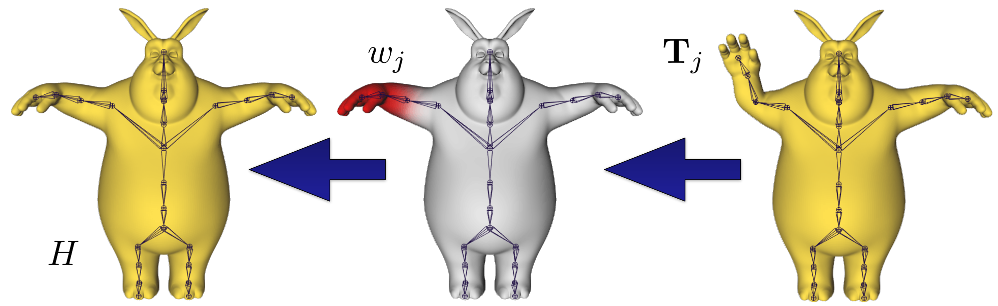
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

?

5

In this work, we consider the inverse problem. Given an animation, what should the handles and weights be?
<click> Formally, we can express this in a least squares sense.
<click> We're not the first ones to look at this problem, but we have a fresh approach.

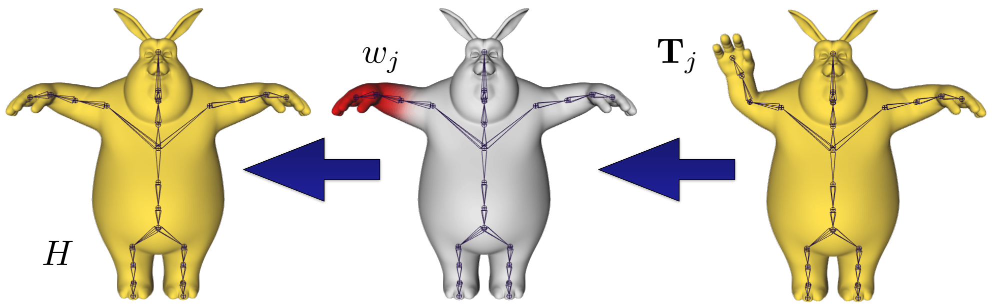
Inverse Linear Blend Skinning



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v})\mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

?

Inverse Linear Blend Skinning

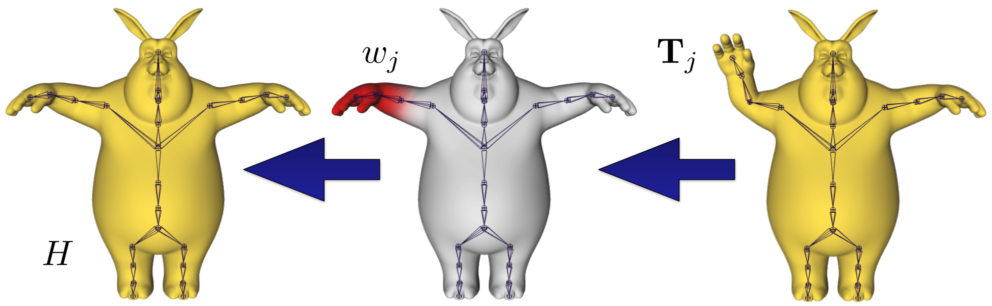


$$\min_{w,R,\mathbf{t},\mathbf{v}} \sum_{p=1}^{\#\text{poses}} \sum_{i=1}^{n} \left\| \mathbf{v}'_{p,i} - \sum_{j=1}^{h} w_{i,j} T_{p,j} \mathbf{v}_i \right\|^2$$

subject to:

$$w_{i,j} \geq 0 \qquad \text{and} \qquad \sum_{j=1}^{h} w_{i,j} = 1$$

Inverse Linear Blend Skinning



$$\min_{w,R,\mathbf{t},\mathbf{v}} \sum_{p=1}^{\#\text{poses}} \sum_{i=1}^{n} \left\| \mathbf{v}'_{p,i} - \sum_{j=1}^{h} w_{i,j} T_{p,j} \mathbf{v}_i \right\|^2$$

subject to:

$$w_{i,j} \geq 0 \quad \text{and} \quad \sum_{j=1}^{h} w_{i,j} = 1$$

Previous Work:

[James and Twigg 2005]
[Schaefer and Yuksel 2007]
[De Aguiar et al. 2008]
[Hasler et al. 2010]
[Kavan et al. 2010]
[Le and Deng 2012, 2013, 2014]

Inverse LBS is a problem in high-dimensions

6

Our observation is that …
…
Weights sum to 1 and are non-negative

Inverse LBS is a problem in high-dimensions

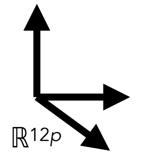
- Transformation matrices are affine: $\mathbb{R}^{12}$

Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$

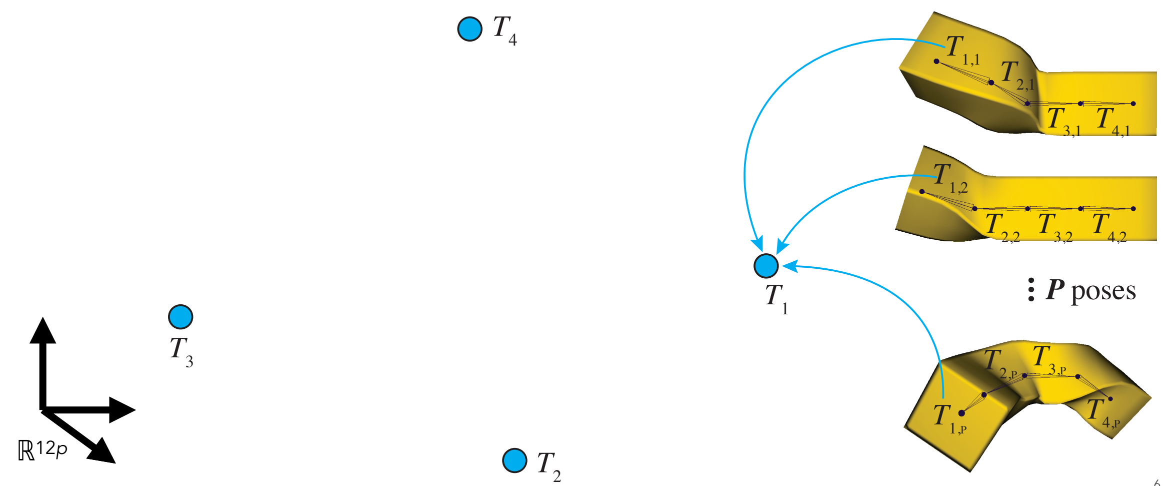
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$



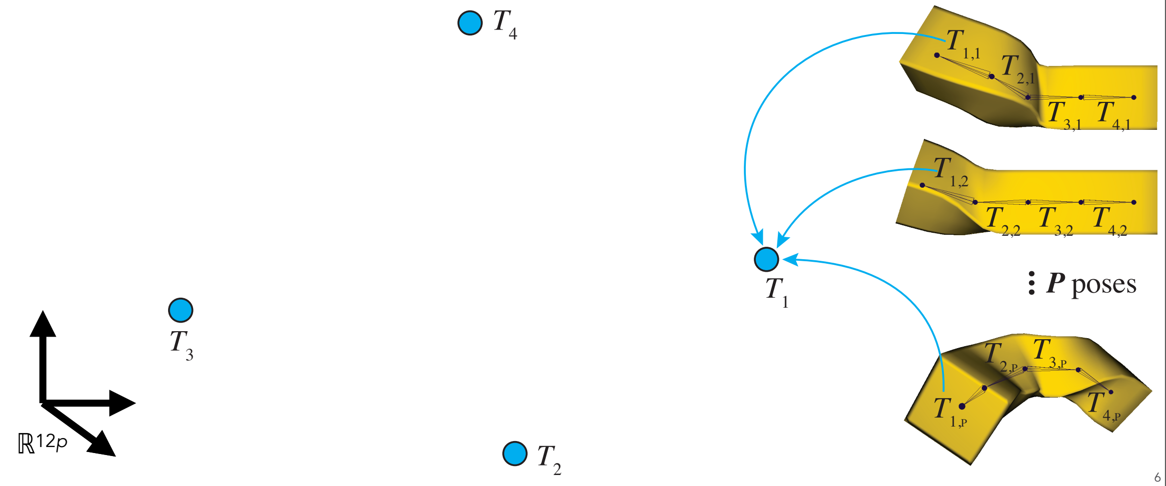
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$



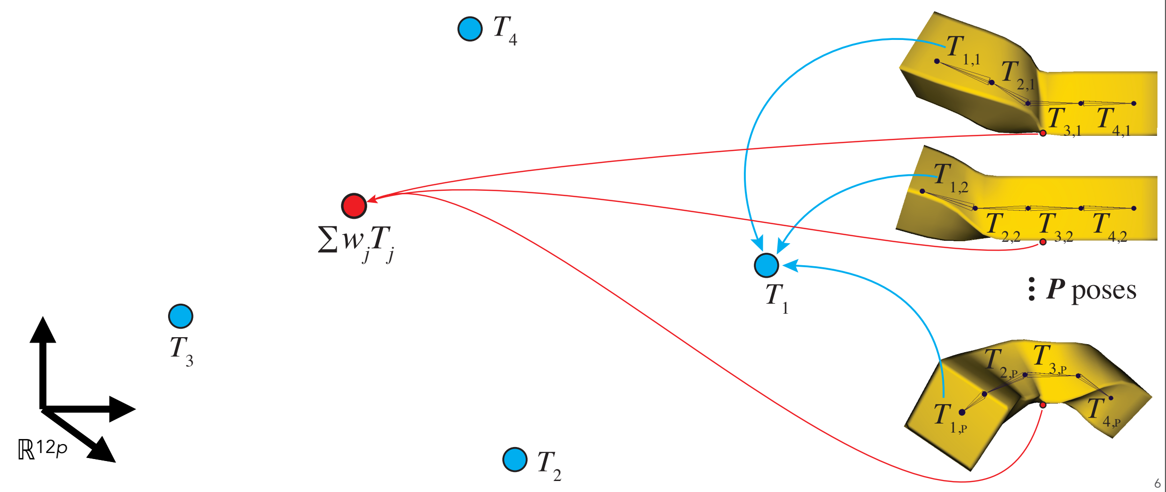
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$
- LBS takes weighted averages of these transformations



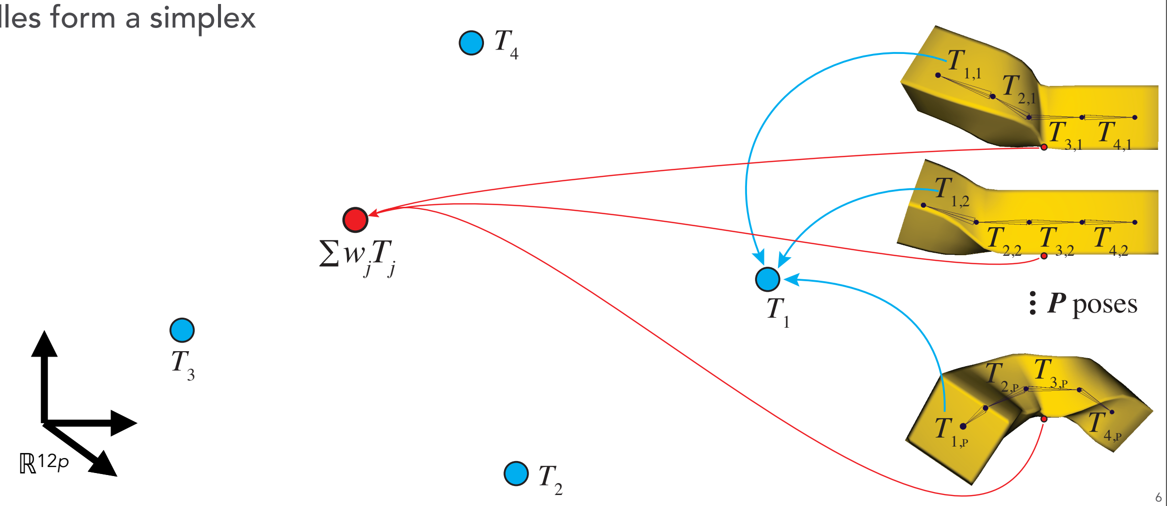
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$
- LBS takes weighted averages of these transformations



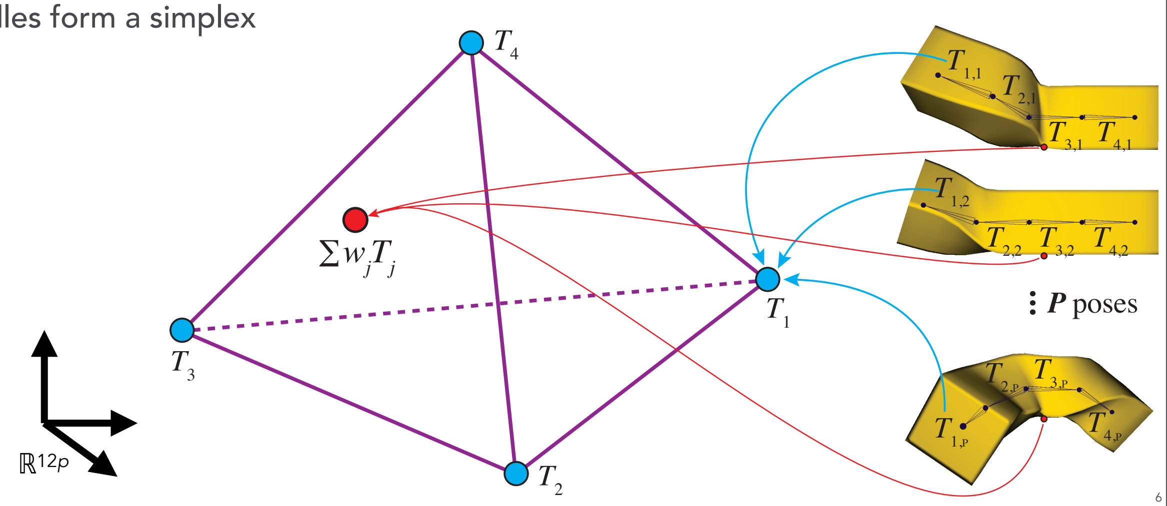
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$
- LBS takes weighted averages of these transformations
- The handles form a simplex



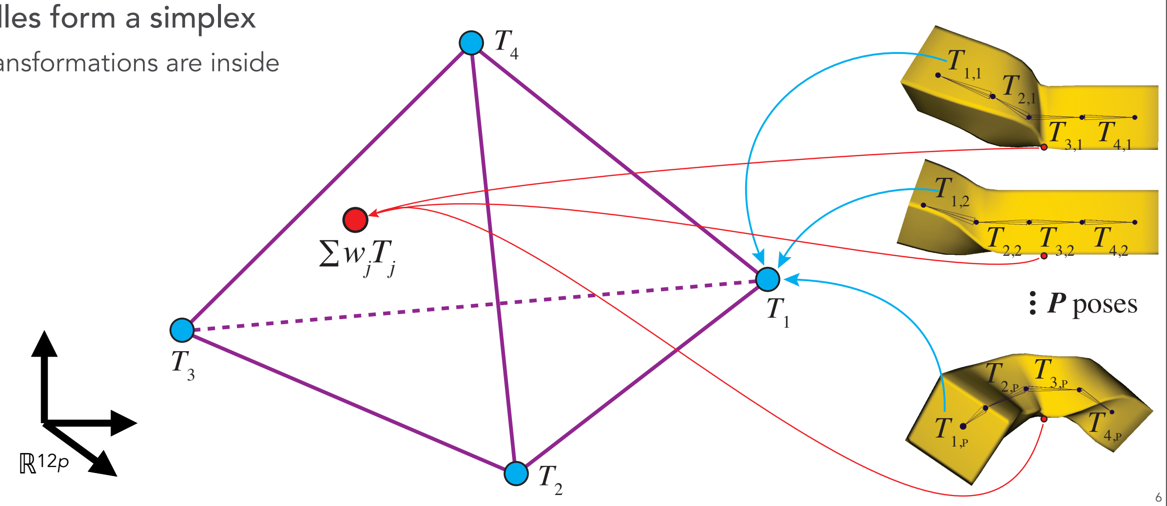
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$
- LBS takes weighted averages of these transformations
- The handles form a simplex



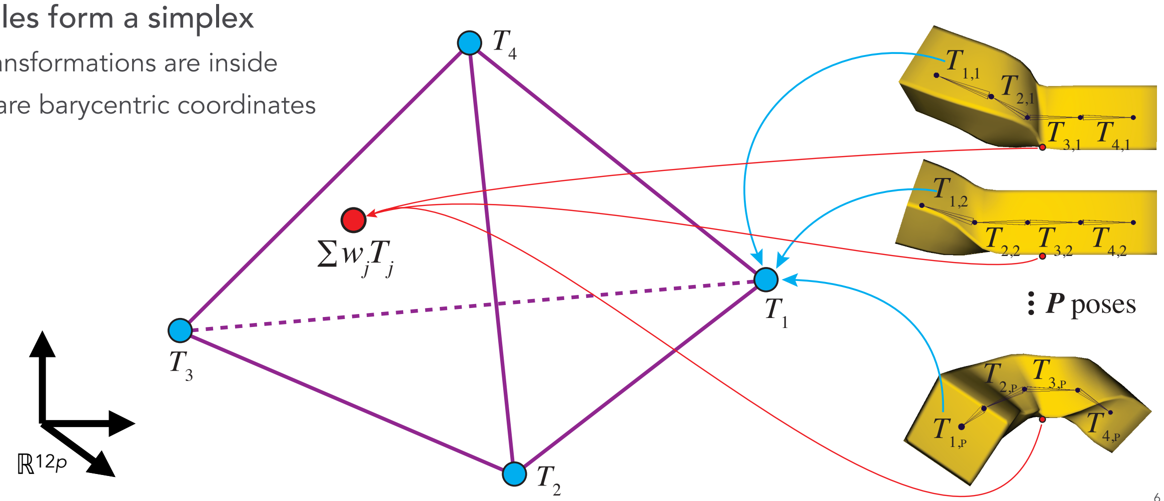
Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$
- LBS takes weighted averages of these transformations
- The handles form a simplex
  - Vertex transformations are inside



Inverse LBS is a problem in high-dimensions

- Transformation matrices are affine: $\mathbb{R}^{12}$
- Handles have transformations across all animation frames or poses: $\mathbb{R}^{12p}$
- LBS takes weighted averages of these transformations
- The handles form a simplex
  - Vertex transformations are inside
  - Weights are barycentric coordinates



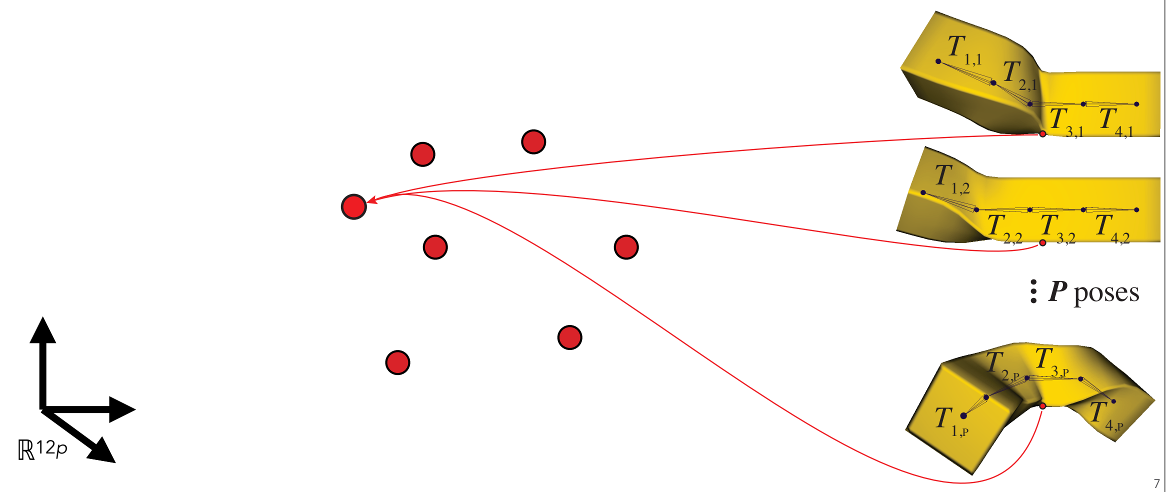
Our Approach

The LBS reconstruction error is entirely determined by the flat-flat distance. Any enclosing simplex has the same error. Smaller simplex means sparser weights.

? We don't need to worry about points on the handle flat. We will find them in Step 2.

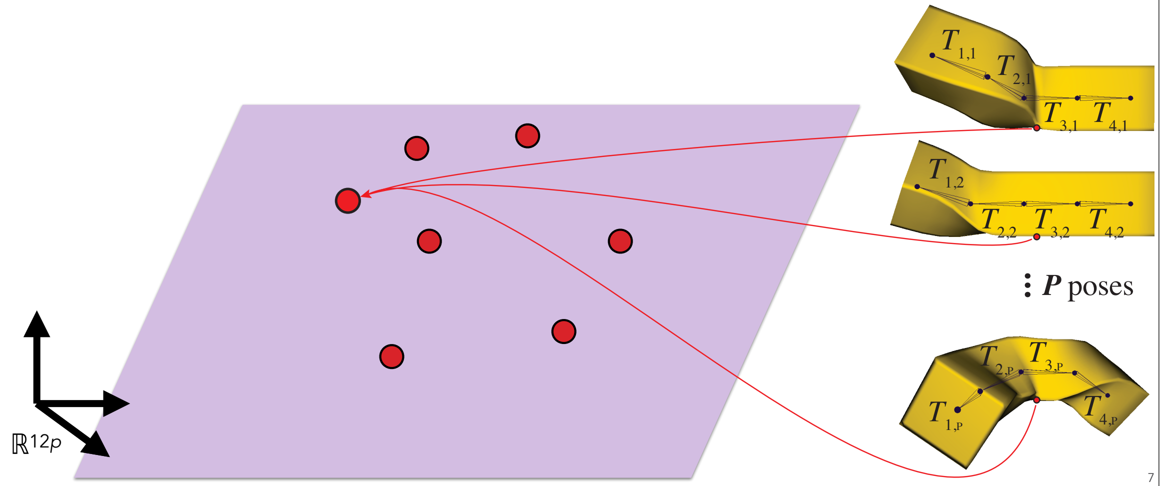
Our Approach

- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$



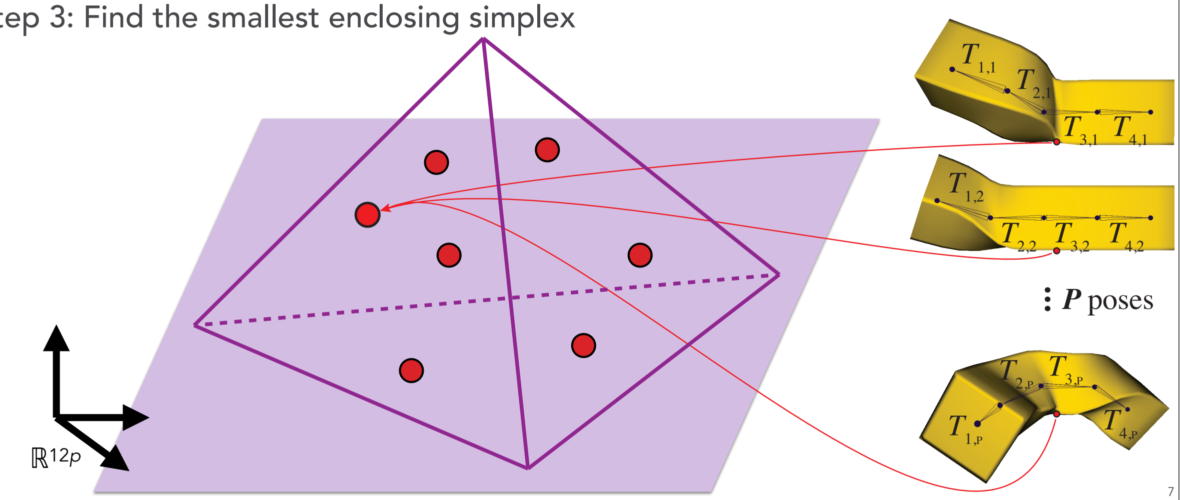
Our Approach

- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$
- Step 2: Estimate a #*handles*-dimensional subspace for the vertices



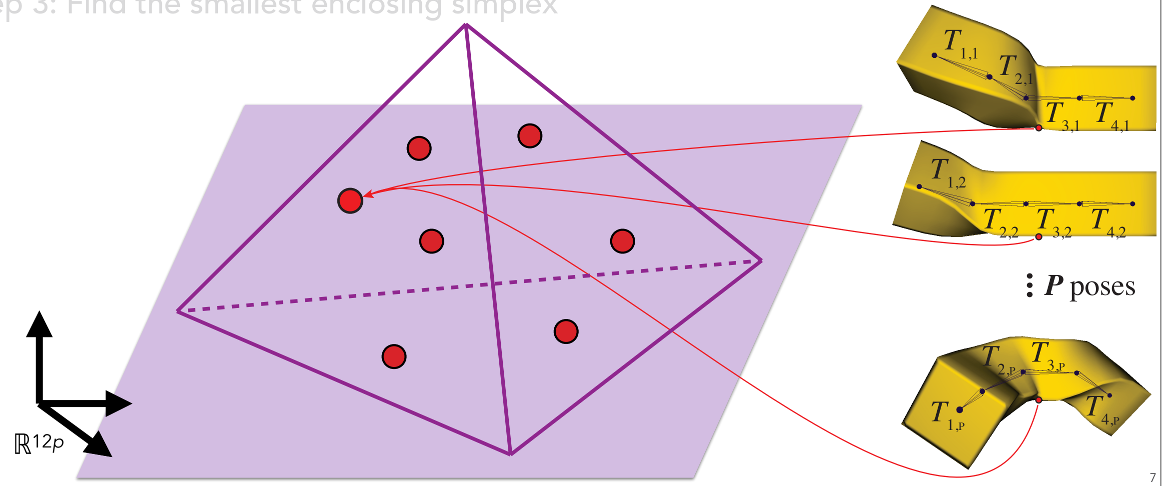
Our Approach

- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$
- Step 2: Estimate a #*handles*-dimensional subspace for the vertices
- Step 3: Find the smallest enclosing simplex



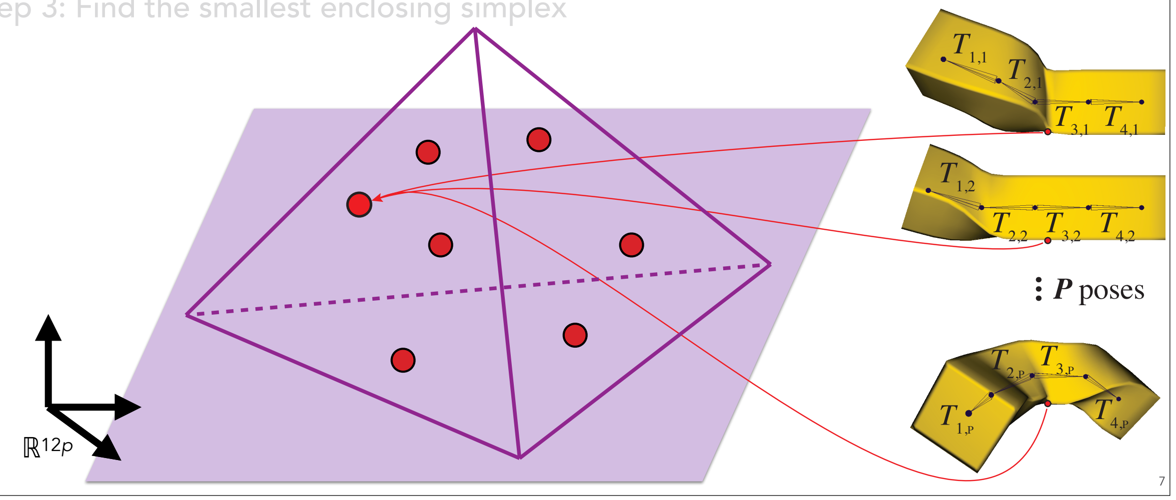
Our Approach

- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$
- Step 2: Estimate a #*handles*-dimensional subspace for the vertices
- Step 3: Find the smallest enclosing simplex



Our Approach

- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$
- Step 2: Estimate a #*handles*-dimensional subspace for the vertices
- Step 3: Find the smallest enclosing simplex

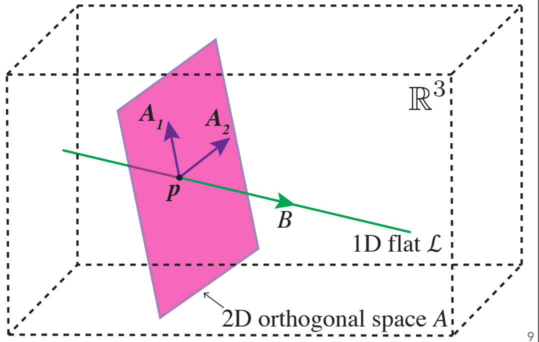


Step 1: Estimate vertex positions in $\mathbb{R}^{12p}$

- For each pose, we know the vertex's rest and deformed position. This constrains possible handle transformations to an affine subspace or *flat* in $\mathbb{R}^{9p}$

$$\bar{V}_i \mathbf{x} = \begin{bmatrix} \mathbf{v}'_{1,i} \\ \vdots \\ \mathbf{v}'_{p,i} \end{bmatrix} = \mathbf{v}'_i$$

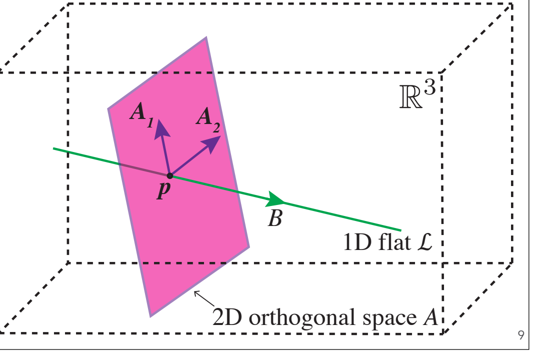
Flats…



* In 2D or 3D, lines or planes (respectively) almost always intersect. That's because they have dimension one less than the ambient space. In general, flats don't intersect, just like lines rarely intersect in 3D.
* columns of B span directions parallel to the flat, z is the vector of parameters, p is a point on the flat
* the columns of F are points in the flat, the parameters w sum to 1
* the rows of A are orthogonal directions to the flat

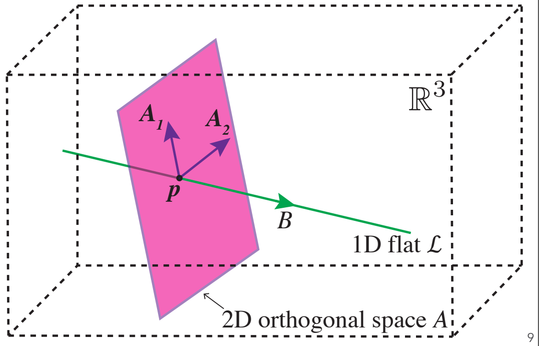
Flats…

- … generalize a line or plane (a linear subspace offset from the origin) to higher dimensions



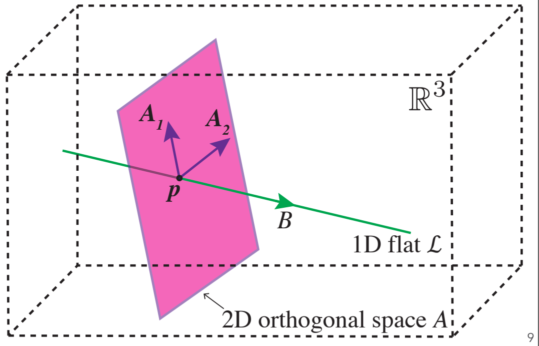
Flats…

- … generalize a line or plane (a linear subspace offset from the origin) to higher dimensions
- … can be defined explicitly: $\mathscr{L} = \{\mathbf{p} + B\mathbf{z}\}$



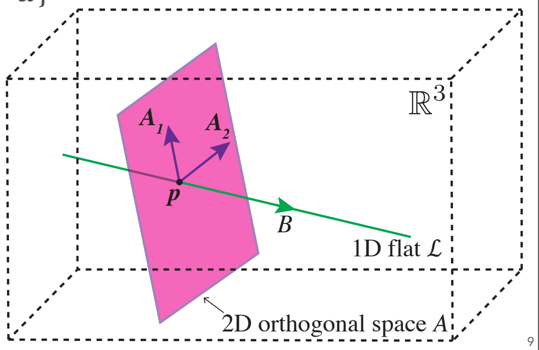
Flats…

- … generalize a line or plane (a linear subspace offset from the origin) to higher dimensions
- … can be defined explicitly: $\mathscr{L} = \{\mathbf{p} + B\mathbf{z}\}$
- … can be defined as weighted average: $\mathscr{L} = \{F\mathbf{w}\}$



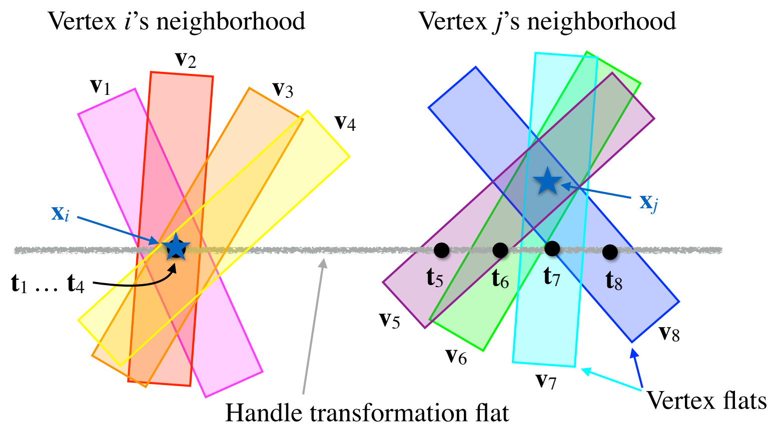
Flats…

- … generalize a line or plane (a linear subspace offset from the origin) to higher dimensions
- … can be defined explicitly: $\mathscr{L} = \{\mathbf{p} + B\mathbf{z}\}$
- … can be defined as weighted average: $\mathscr{L} = \{F\mathbf{w}\}$
- … can be defined implicitly: $\mathscr{L} = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a}\}$



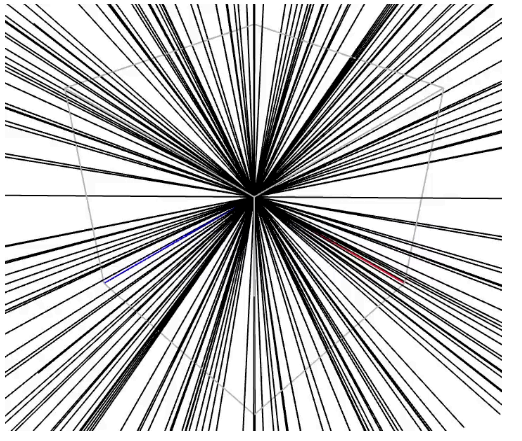
Step 2: Estimate a *handle* subspace close to the vertices

- We want a (#*handles-1*)-dimensional flat that intersects or is as close as possible to all individual vertices' flats.



If there's a handle flat that intersects all vertex flats, then there's a zero-error solution to inverse skinning. Minimizing the distance minimizes the error.

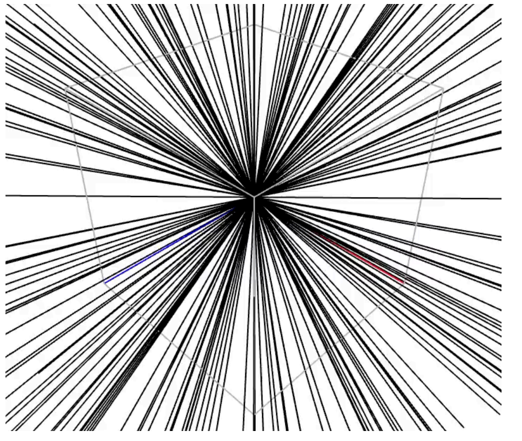
The Closest Flat Problem is Hard



TODO: Cube edges

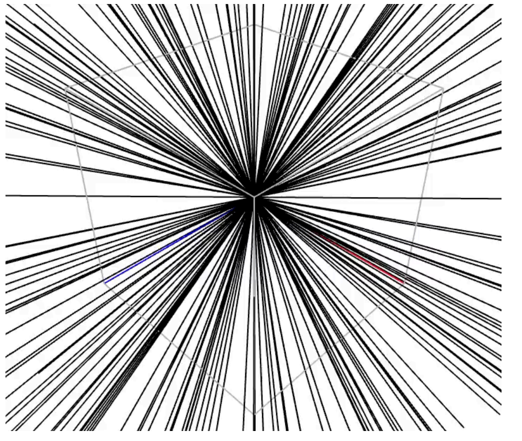
The Closest Flat Problem is Hard

- It's not convex. How hard is it?



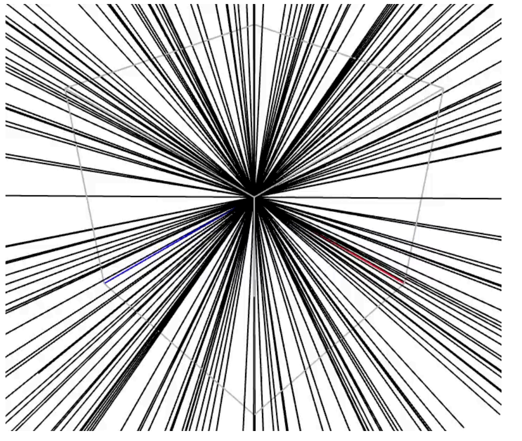
The Closest Flat Problem is Hard

- It's not convex. How hard is it?
- Generate random 3D lines that intersect a known line. Can we recover the known line from a random initial guess?



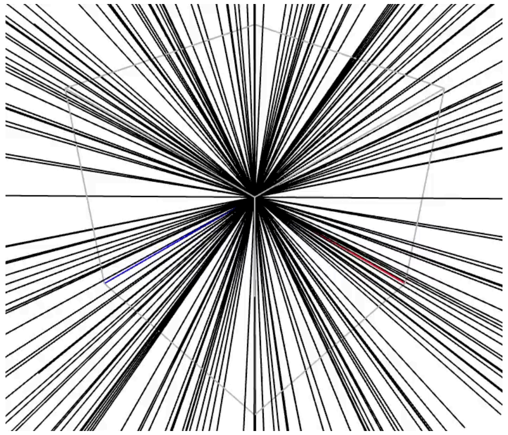
The Closest Flat Problem is Hard

- It's not convex. How hard is it?
- Generate random 3D lines that intersect a known line. Can we recover the known line from a random initial guess?
- In 3D, the closest line to a set of lines.



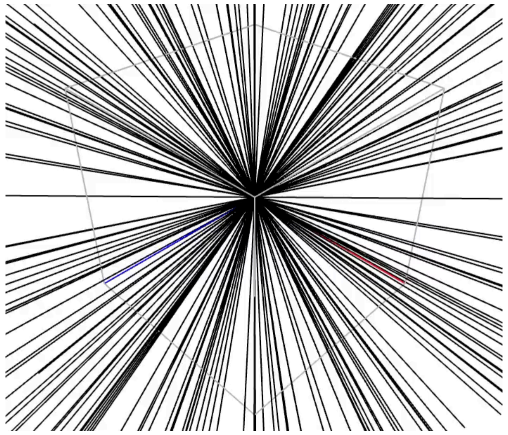
The Closest Flat Problem is Hard

- It's not convex. How hard is it?
- Generate random 3D lines that intersect a known line. Can we recover the known line from a random initial guess?
- In 3D, the closest line to a set of lines.
    - **Closest line** optimization as seen from a camera looking along the ground truth line: (the ground truth line looks like a point at the origin)



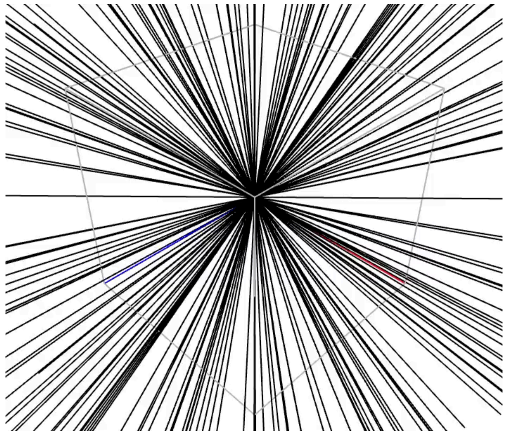
The Closest Flat Problem is Hard

- It's not convex. How hard is it?
- Generate random 3D lines that intersect a known line. Can we recover the known line from a random initial guess?
- In 3D, the closest line to a set of lines.
  - **Closest line** optimization as seen from a camera looking along the ground truth line: (the ground truth line looks like a point at the origin)



The Closest Flat Problem is Hard

- It's not convex. How hard is it?
- Generate random 3D lines that intersect a known line. Can we recover the known line from a random initial guess?
- In 3D, the closest line to a set of lines.
  - **Closest line** optimization as seen from a camera looking along the ground truth line: (the ground truth line looks like a point at the origin)
  - **Success!**



The Closest Flat Problem is Hard

TODO: Cube edges

The Closest Flat Problem is Hard

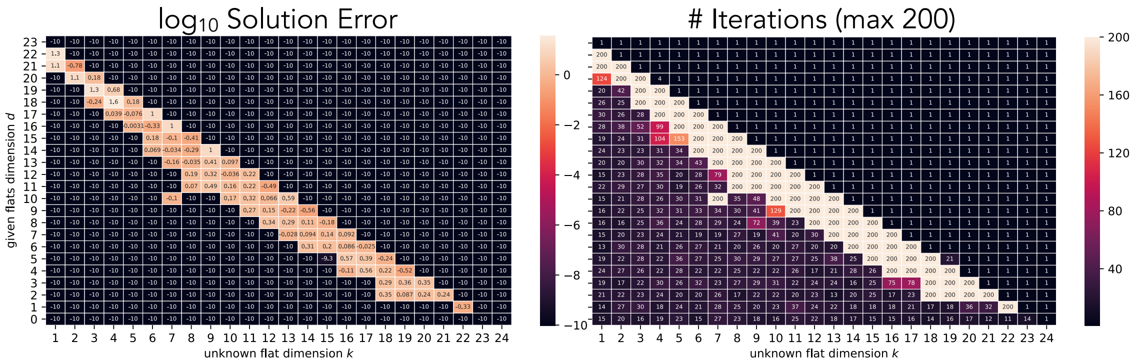
- An experiment in $\mathbb{R}^{24}$

The Closest Flat Problem is Hard

- An experiment in $\mathbb{R}^{24}$
- Generate random *d*-dimensional flats that intersect a known *k*-dimensional flat. Can we recover the *k*-dimensional flat from a random initial guess?

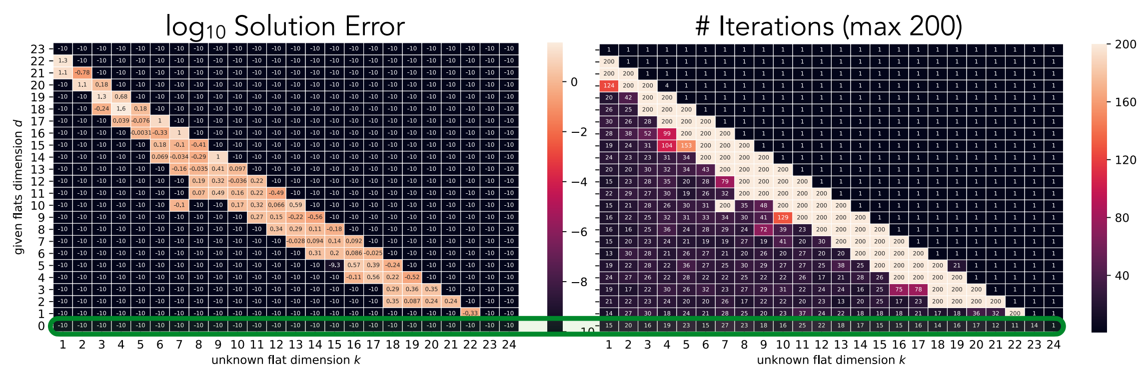
The Closest Flat Problem is Hard

- An experiment in $\mathbb{R}^{24}$
- Generate random *d*-dimensional flats that intersect a known *k*-dimensional flat. Can we recover the *k*-dimensional flat from a random initial guess?



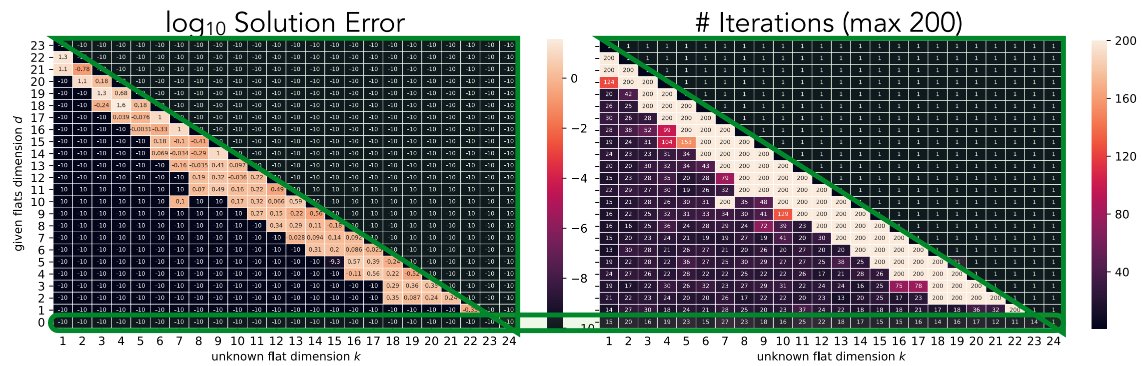
The Closest Flat Problem is Hard

- An experiment in $\mathbb{R}^{24}$
- Generate random *d*-dimensional flats that intersect a known *k*-dimensional flat. Can we recover the *k*-dimensional flat from a random initial guess?
  - When *d*=0, the given flats are points. It's a simple least squares problem



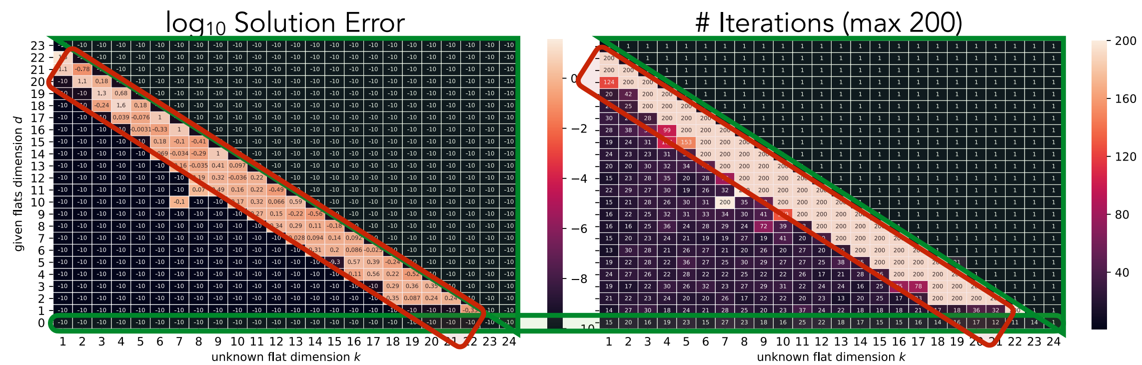
The Closest Flat Problem is Hard

- An experiment in $\mathbb{R}^{24}$
- Generate random *d*-dimensional flats that intersect a known *k*-dimensional flat. Can we recover the *k*-dimensional flat from a random initial guess?
  - When d=0, the given flats are points. It's a simple least squares problem
  - When $d+k \geq 24$, it's trivial. A random initial guess almost surely intersects all flats.



The Closest Flat Problem is Hard

- An experiment in $\mathbb{R}^{24}$
- Generate random *d*-dimensional flats that intersect a known *k*-dimensional flat. Can we recover the *k*-dimensional flat from a random initial guess?
  - When *d*=0, the given flats are points. It's a simple least squares problem
  - When *d*+*k*≥24, it's trivial. A random initial guess almost surely intersects all flats.
  - When *d*+*k*<24, there is a difficult zone as *d*+*k* approach 24.



The Closest Flat Problem is Hard

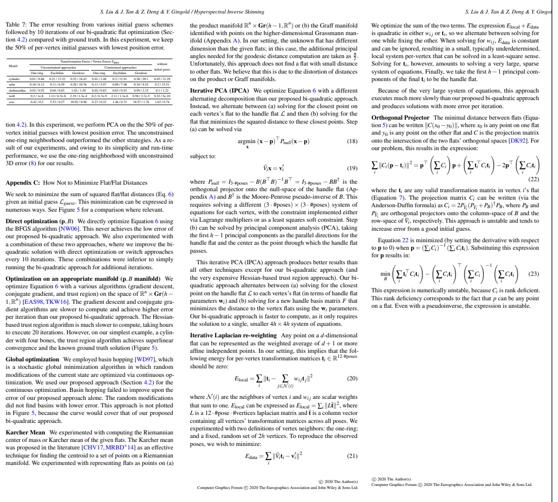
TODO: Cube edges

The Closest Flat Problem is Hard

- Tried many possibilities
  - direct gradient and Hessian-based optimization for an explicit representation of the flat
  - optimization on the Graff manifold
  - gradient-based optimization of projection matrices
  - global optimization via basin hopping
  - Karcher mean
  - alternating optimization strategies

The Closest Flat Problem is Hard

- Tried many possibilities
  - direct gradient and Hessian-based optimization for an explicit representation of the flat
  - optimization on the Graff manifold
  - gradient-based optimization of projection matrices
  - global optimization via basin hopping
  - Karcher mean
  - alternating optimization strategies
- See our Appendix “How Not to Minimize Flat/Flat Distances”



Minimizing Flat/Flat Distance: Initial Guess

Minimizing Flat/Flat Distance: Initial Guess

- Find an $\mathbb{R}^{12p}$ point $\mathbf{x}_i$ for each vertex

Minimizing Flat/Flat Distance: Initial Guess

- Find an $\mathbb{R}^{12p}$ point $\mathbf{x}_i$ for each vertex
    - If the vertex $\mathbf{v}_i$ and its one-ring move rigidly, there is a unique solution. If not, there is a least-squares solution…

Minimizing Flat/Flat Distance: Initial Guess

- Find an $\mathbb{R}^{12p}$ point $\mathbf{x}_i$ for each vertex
  - If the vertex $\mathbf{v}_i$ and its one-ring move rigidly, there is a unique solution. If not, there is a least-squares solution…
    - …measuring error in $\mathbb{R}^{12p}$:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j \in \{i\} \cup \mathcal{N}(i)} \left\| \frac{1}{\|\mathbf{v}_j\|^2} \bar{V}_j^\top \bar{V}_j (\mathbf{x} - \mathbf{t}_j) \right\|^2$$

Minimizing Flat/Flat Distance: Initial Guess

- Find an $\mathbb{R}^{12p}$ point $\mathbf{x}_i$ for each vertex
  - If the vertex $\mathbf{v}_i$ and its one-ring move rigidly, there is a unique solution. If not, there is a least-squares solution…
    - …measuring error in $\mathbb{R}^{12p}$:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j \in \{i\} \cup \mathcal{N}(i)} \left\| \frac{1}{\|\mathbf{v}_j\|^2} \bar{V}_j^\top \bar{V}_j (\mathbf{x} - \mathbf{t}_j) \right\|^2$$

    - …measuring error in 3D:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j \in \{i\} \cup \mathcal{N}(i)} \|\bar{V}_j \mathbf{x} - \mathbf{v}'_j\|^2$$

Minimizing Flat/Flat Distance: Initial Guess

- Find an $\mathbb{R}^{12p}$ point $\mathbf{x}_i$ for each vertex
  - If the vertex $\mathbf{v}_i$ and its one-ring move rigidly, there is a unique solution. If not, there is a least-squares solution…
    - …measuring error in $\mathbb{R}^{12p}$:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j \in \{i\} \cup \mathcal{N}(i)} \left\| \frac{1}{\|\mathbf{v}_j\|^2} \bar{V}_j^\top \bar{V}_j (\mathbf{x} - \mathbf{t}_j) \right\|^2$$

    - …measuring error in 3D:

$$\mathbf{x}_i = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j \in \{i\} \cup \mathcal{N}(i)} \|\bar{V}_j \mathbf{x} - \mathbf{v}'_j\|^2$$

- PCA on the 12p-dimensional points gives us an initial guess for the flat.

Minimizing Flat/Flat Distance: Optimization

- We use an explicit expression for a flat: $$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}'_i\|^2$$

  subject to: $\sum \mathbf{w}_i = 1$

Minimizing Flat/Flat Distance: Optimization

- We use an explicit expression for a flat: $$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}'_i\|^2$$
  subject to: $\sum \mathbf{w}_i = 1$
- Quadratic in F, w_i, and even $\bar{V}_i$.

Minimizing Flat/Flat Distance: Optimization

- We use an explicit expression for a flat: $$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}'_i\|^2$$ subject to: $\sum \mathbf{w}_i = 1$
- Quadratic in F, w_i, and even $\bar{V}_i$.
- Alternates between local and global steps:

Minimizing Flat/Flat Distance: Optimization

- We use an explicit expression for a flat: $$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}'_i\|^2$$ subject to: $\sum \mathbf{w}_i = 1$
- Quadratic in F, w_i, and even $\bar{V}_i$.
- Alternates between local and global steps:
  - Local steps: $\mathbf{w}_i$ are independent

Minimizing Flat/Flat Distance: Optimization

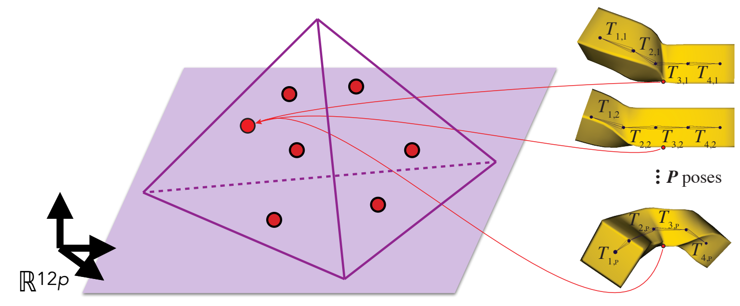
- We use an explicit expression for a flat: $$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}'_i\|^2$$ subject to: $\sum \mathbf{w}_i = 1$
- Quadratic in F, w_i, and even $\bar{V}_i$.
- Alternates between local and global steps:
  - Local steps: $\mathbf{w}_i$ are independent
  - Global step: minimizing F entails solving a linear matrix equation: $\sum_i \left(I_{3 \cdot \#\text{poses}} \otimes (\mathbf{v}_i \mathbf{v}_i^\top) \right) F \left(\mathbf{w}_i \mathbf{w}_i^\top \right) = - \sum_i \bar{V}_i \mathbf{v}_i^\top \mathbf{w}_i$

Minimizing Flat/Flat Distance: Optimization

- We use an explicit expression for a flat: $$\min_F \sum_i \|\bar{V}_i F \mathbf{w}_i - \mathbf{v}_i'\|^2$$ subject to: $\sum \mathbf{w}_i = 1$
- Quadratic in F, w_i, and even $\bar{V}_i$.
- Alternates between local and global steps:
  - Local steps: $\mathbf{w}_i$ are independent
  - Global step: minimizing F entails solving a linear matrix equation: $$\sum_i \left(I_{3 \cdot \#\text{poses}} \otimes (\mathbf{v}_i \mathbf{v}_i^\top)\right) F \left(\mathbf{w}_i \mathbf{w}_i^\top\right) = -\sum_i \bar{V}_i \mathbf{v}_i^\top \mathbf{w}_i$$
    - This reduces to a $4h \times 4h$ system of equations

Minimizing Flat/Flat Distance: Optimization

- Let's visualize optimization steps.
- A cylinder with 4 handles. The handle simplex is a tetrahedron. The handle flat is 3D. Let's visualize the closest points on the flat to the cylinder vertices.



The orientation is arbitrary, so we minimize unnecessary rotation via a Procrustes transformation.

Minimizing Flat/Flat Distance: Optimization

The orientation is arbitrary, so we minimize unnecessary rotation via a Procrustes transformation.

Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles

Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles

⇒ The handle simplex is a tetrahedron

Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles

  ⇒ The handle simplex is a tetrahedron

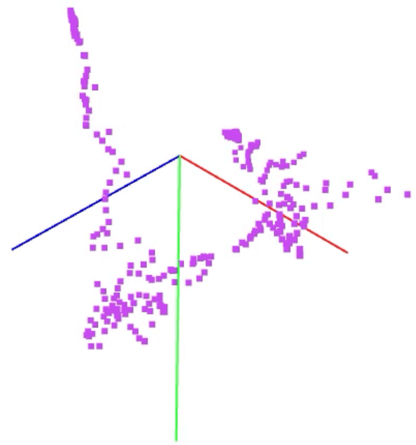
  ⇒ The handle flat is 3D

Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles
  ⇒ The handle simplex is a tetrahedron
  ⇒ The handle flat is 3D
- Visualizing vertex transformations $\in \mathbb{R}^{12p}$ as points projected onto the handle flat:

Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles
  ⇒ The handle simplex is a tetrahedron
  ⇒ The handle flat is 3D
- Visualizing vertex transformations $\in \mathbb{R}^{12p}$ as points projected onto the handle flat:

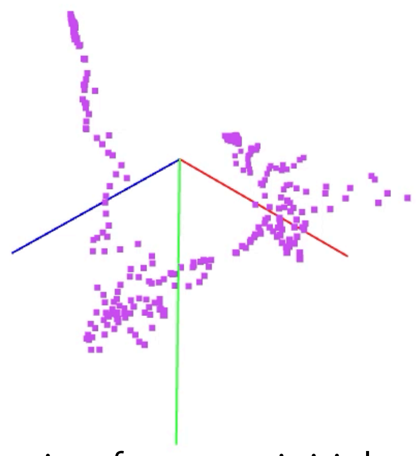


Optimization from our initial guess
(slow motion, converges in ~10 iterations)

The orientation is arbitrary, so we minimize unnecessary rotation via a Procrustes transformation.

Minimizing Flat/Flat Distance: Optimization

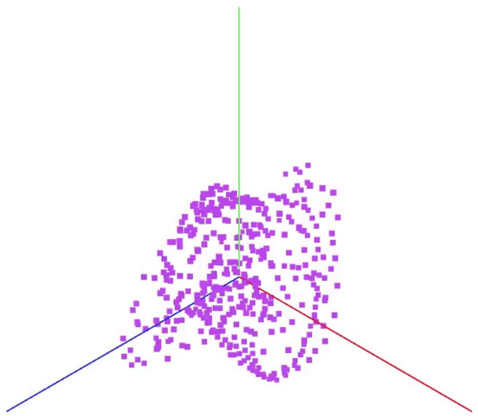
- A cylinder with 4 handles
  ⇒ The handle simplex is a tetrahedron
  ⇒ The handle flat is 3D
- Visualizing vertex transformations $\in \mathbb{R}^{12p}$ as points projected onto the handle flat:



Optimization from our initial guess
(slow motion, converges in ~10 iterations)

Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles
  ⇒ The handle simplex is a tetrahedron
  ⇒ The handle flat is 3D
- Visualizing vertex transformations $\in \mathbb{R}^{12p}$ as points projected onto the handle flat:

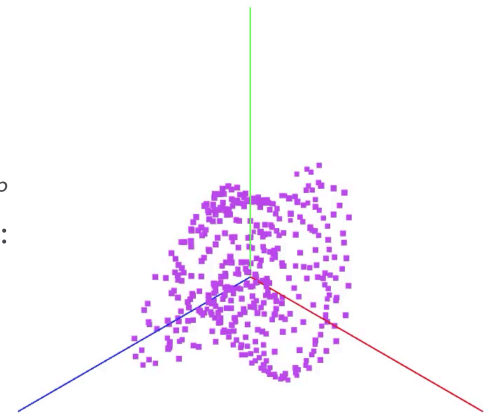


Optimization from a random initial guess
(>10,000 iterations)

The orientation is arbitrary, so we minimize unnecessary rotation via a Procrustes transformation.

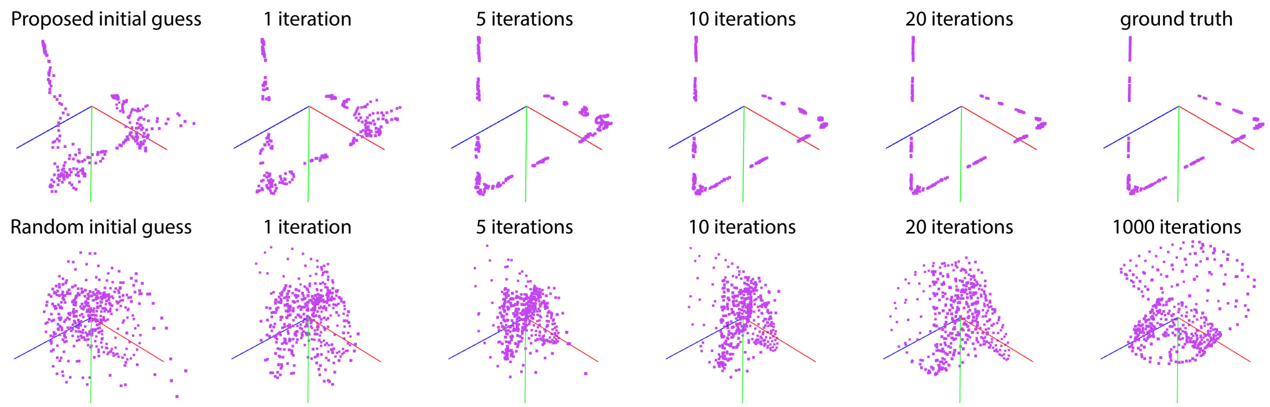
Minimizing Flat/Flat Distance: Optimization

- A cylinder with 4 handles
  ⇒ The handle simplex is a tetrahedron
  ⇒ The handle flat is 3D
- Visualizing vertex transformations $\in \mathbb{R}^{12p}$ as points projected onto the handle flat:



Optimization from a random initial guess
(>10,000 iterations)

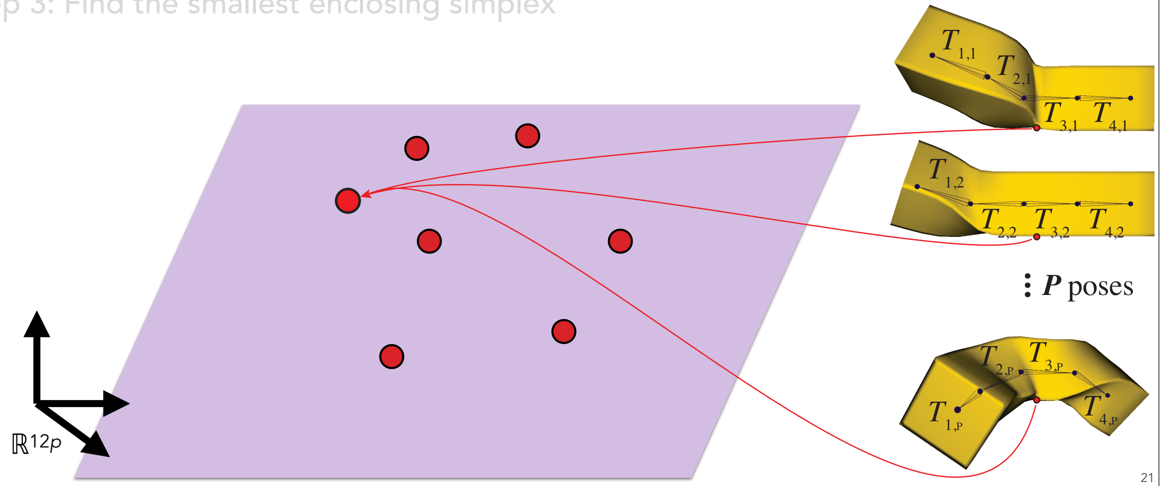
Minimizing Flat/Flat Distance: Optimization



Here they are side-by-side

Our Approach

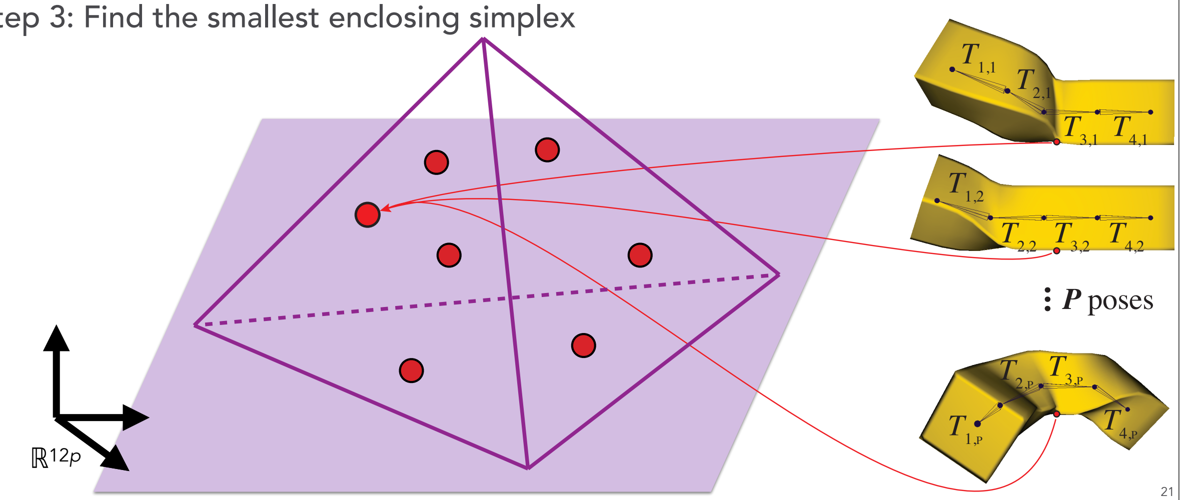
- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$
- Step 2: Estimate a #*handles*-dimensional subspace for the vertices
- Step 3: Find the smallest enclosing simplex



Optimization gave us a flat. We project all vertices into this flat. The error should be small. The LBS reconstruction error is entirely determined by this projection distance.
<click> All that's left is finding handles which enclose the projected vertices. This is the minimum volume enclosing simplex problem from Hyperspectral Imaging! Any enclosing simplex has the same error. Smaller simplex means sparser weights.

Our Approach

- Step 1: Estimate vertex transformations in $\mathbb{R}^{12p}$
- Step 2: Estimate a #*handles*-dimensional subspace for the vertices
- Step 3: Find the smallest enclosing simplex



Hyperspectral Image Unmixing

[European Union, Copernicus Sentinel-2 imagery]

end members are our handles. abundances are our weights.

Hyperspectral Image Unmixing

- Satellites capture high-dimensional data from far away

[European Union, Copernicus Sentinel-2 imagery]

Hyperspectral Image Unmixing

- Satellites capture high-dimensional data from far away
- Pixels contain mixtures of objects

[European Union, Copernicus Sentinel-2 imagery]

Hyperspectral Image Unmixing

- Satellites capture high-dimensional data from far away
- Pixels contain mixtures of objects
- What are the objects (*endmembers*)?

[European Union, Copernicus Sentinel-2 imagery]

Hyperspectral Image Unmixing

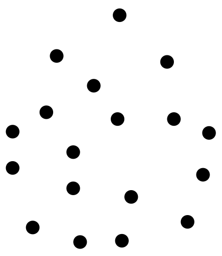
- Satellites capture high-dimensional data from far away
- Pixels contain mixtures of objects
- What are the objects (*endmembers*)?
- What mixture is in a pixel (*abundances*)?

[European Union, Copernicus Sentinel-2 imagery]

Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

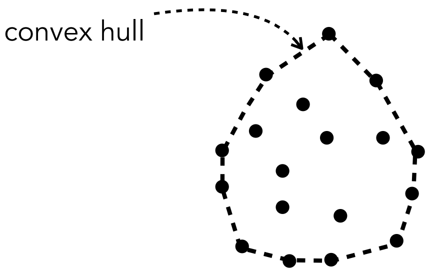
- Given points in high dimensions, perform PCA and then find the MVES



Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

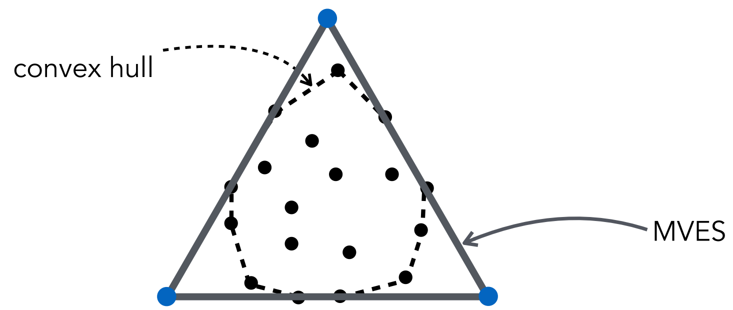
- Given points in high dimensions, perform PCA and then find the MVES



Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

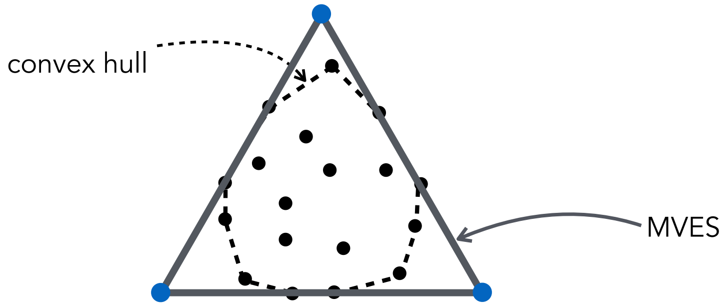
- Given points in high dimensions, perform PCA and then find the MVES



Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

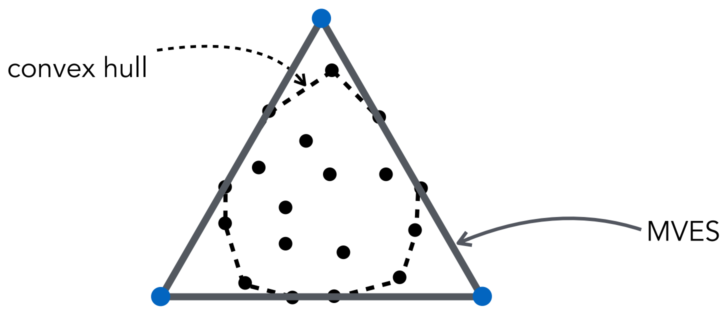
- Given points in high dimensions, perform PCA and then find the MVES
- **State of the art:** [Chan et al. 2009, Bioucas-Dias 2009, Ambikapathi et al. 2010, Agathos et al. 2014, Lin et al. 2016]



Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

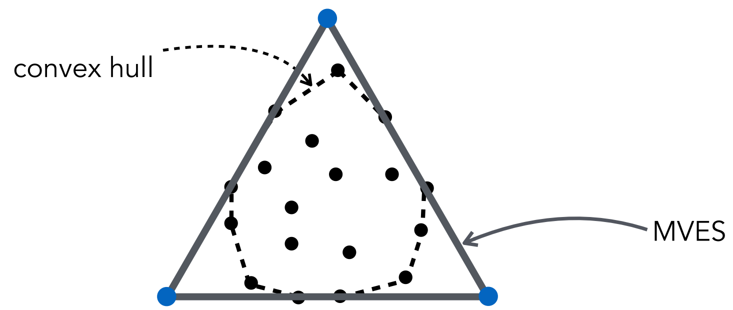
- Given points in high dimensions, perform PCA and then find the MVES
- **State of the art:** [Chan et al. 2009, Bioucas-Dias 2009, Ambikapathi et al. 2010, Agathos et al. 2014, Lin et al. 2016]
- In theory, should be difficult [Hendrix et al. 2013] but works well in papers



Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

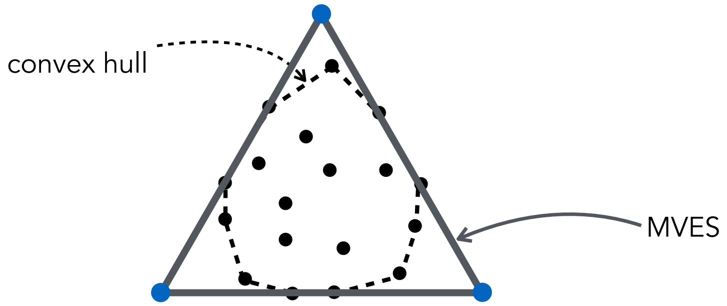
- Given points in high dimensions, perform PCA and then find the MVES
- **State of the art:** [Chan et al. 2009, Bioucas-Dias 2009, Ambikapathi et al. 2010, Agathos et al. 2014, Lin et al. 2016]
- In theory, should be difficult [Hendrix et al. 2013] but works well in papers
- In theory, gets easier the higher the dimension [Lin et al. 2015, Fu et al. 2015]



Minimum Volume Enclosing Simplex (MVES)

[Craig 1994]

- Given points in high dimensions, perform PCA and then find the MVES
- **State of the art:** [Chan et al. 2009, Bioucas-Dias 2009, Ambikapathi et al. 2010, Agathos et al. 2014, Lin et al. 2016]
- In theory, should be difficult [Hendrix et al. 2013] but works well in papers
- In theory, gets easier the higher the dimension [Lin et al. 2015, Fu et al. 2015]
- Related to non-negative matrix factorization [Arora et al. 2012]



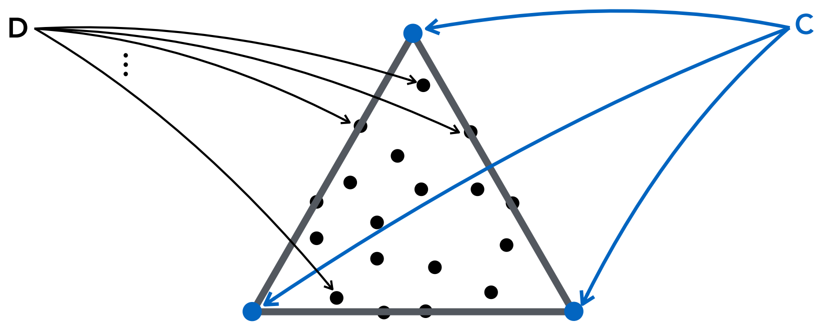
Minimum Volume Enclosing Simplex (MVES)

- Formally: $$\min_{C} |\det(C)|$$

subject to:

$$C^{-1}D \geq 0$$ (weights ≥ 0)

$$C_{h,i} = 1, \quad \forall i \in [1, h]$$ (homogeneous coordinates)



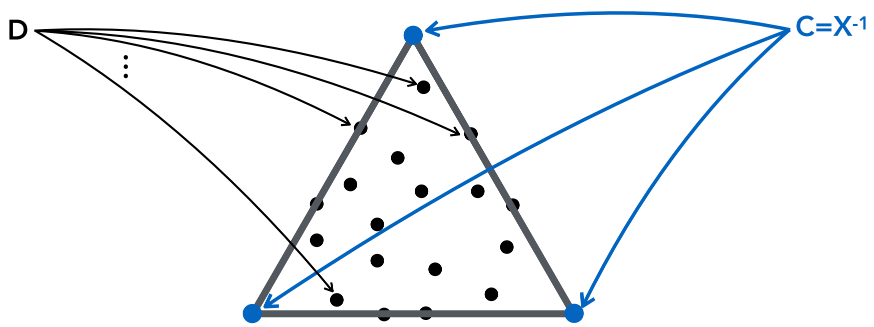
Minimum Volume Enclosing Simplex (MVES)

- Formally: $\min(-\log\det(X))$

  subject to:

  $XD \geq 0$ (weights ≥ 0)

  $X\mathbf{1}_h = [0,0,0,\ldots,1]^T$ (homogeneous coordinates)



This version is equivalent but avoids inverses and numerical blow-up. We follow a recent approach.

Minimum Volume Enclosing Simplex (MVES)

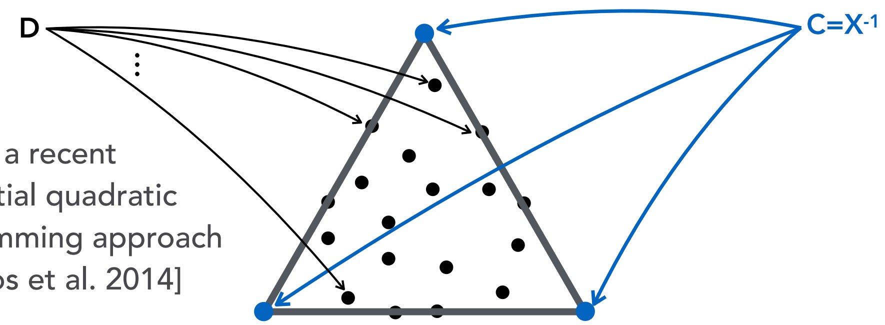
- Formally: $\min(-\log\det(X))$

  subject to:

  $XD \geq 0$ (weights ≥ 0)

  $X\mathbf{1}_h = [0, 0, 0, \ldots, 1]^T$ (homogeneous coordinates)

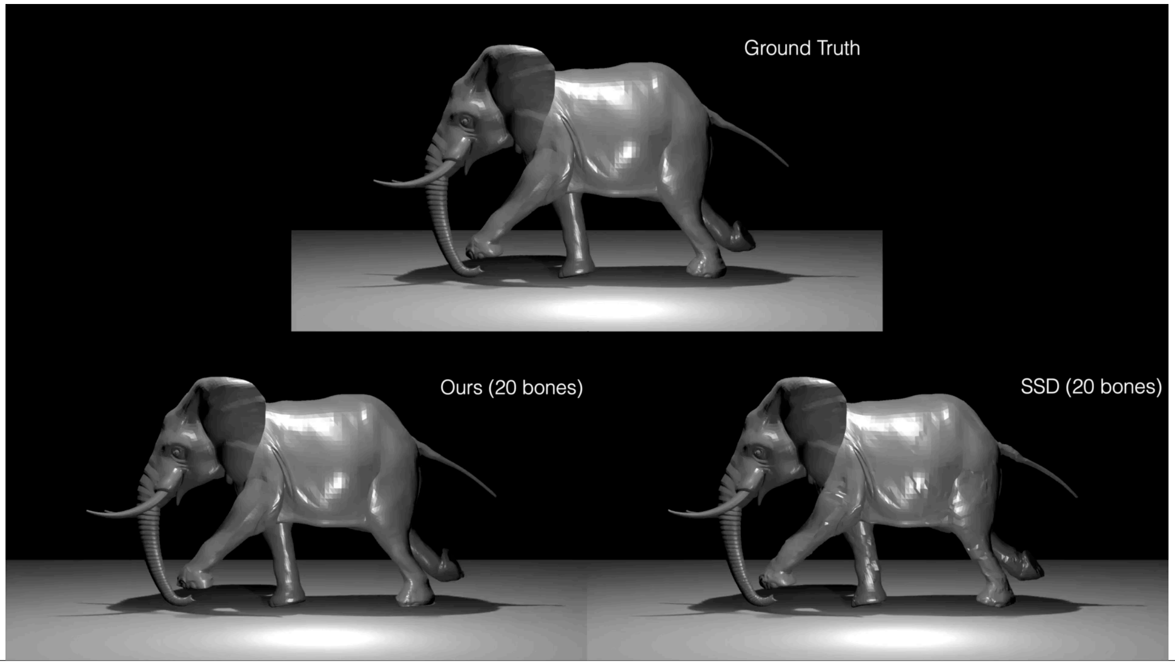
- We use a recent sequential quadratic programming approach [Agathos et al. 2014]



Results

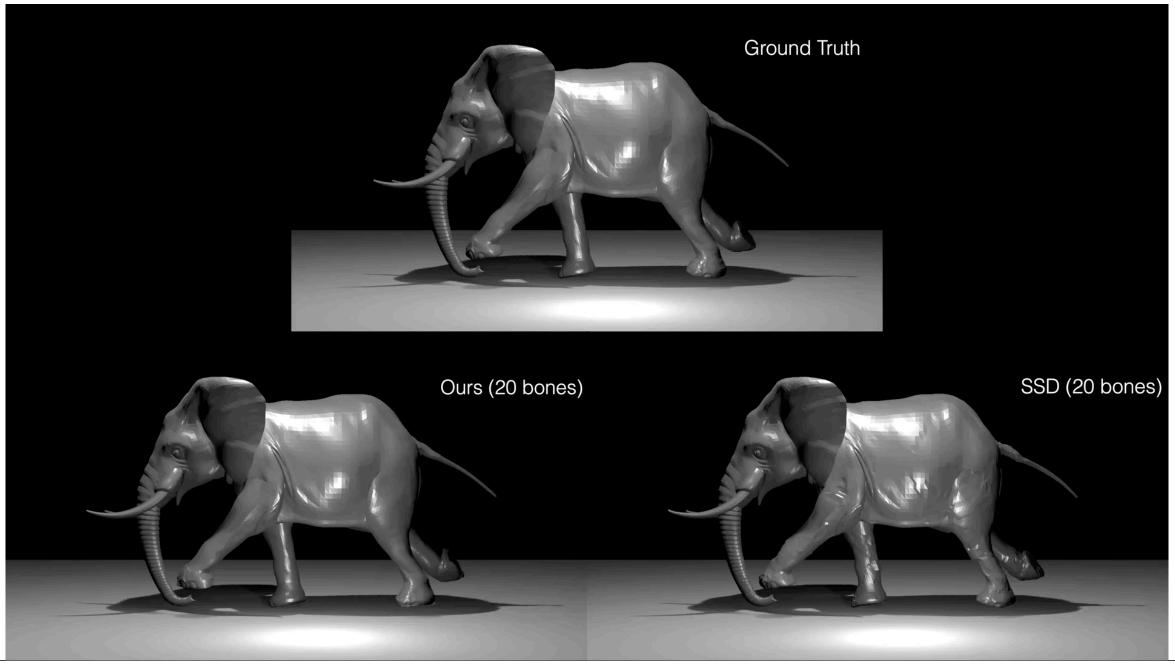
Let's see some results

Comparison to SSDR [Le and Deng 2012]



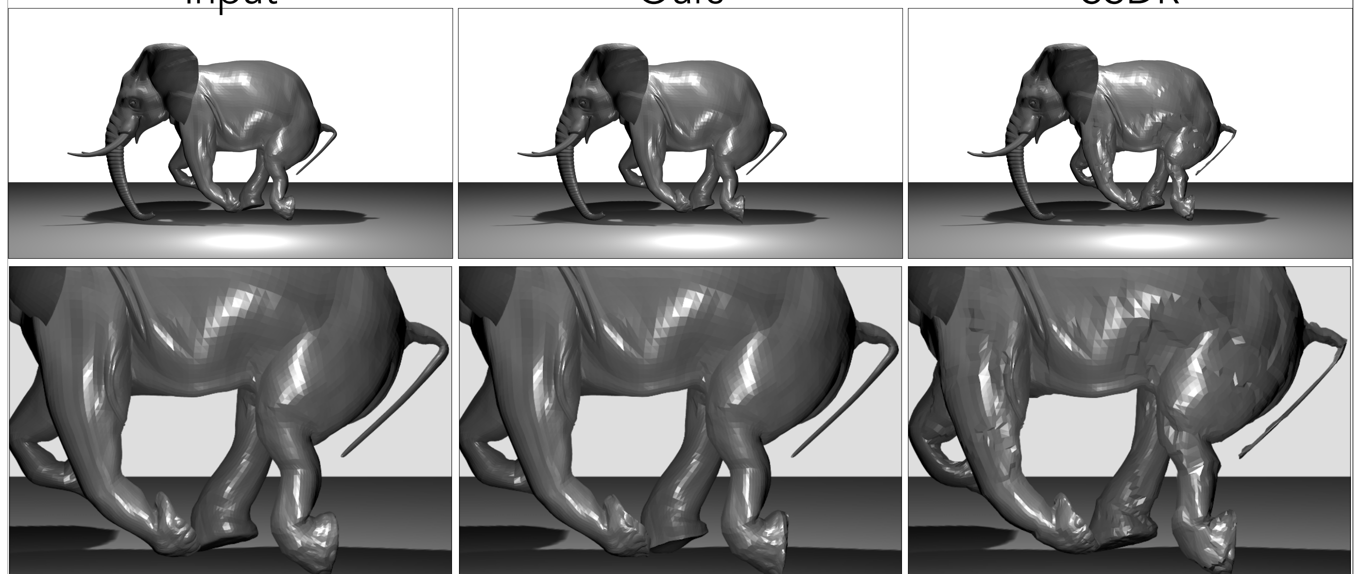
Our approach is faster and has lower error compared to the SSDR (Smooth Skinning Decomposition with Rigid Bones) technique of Le and Deng.

Comparison to SSDR [Le and Deng 2012]



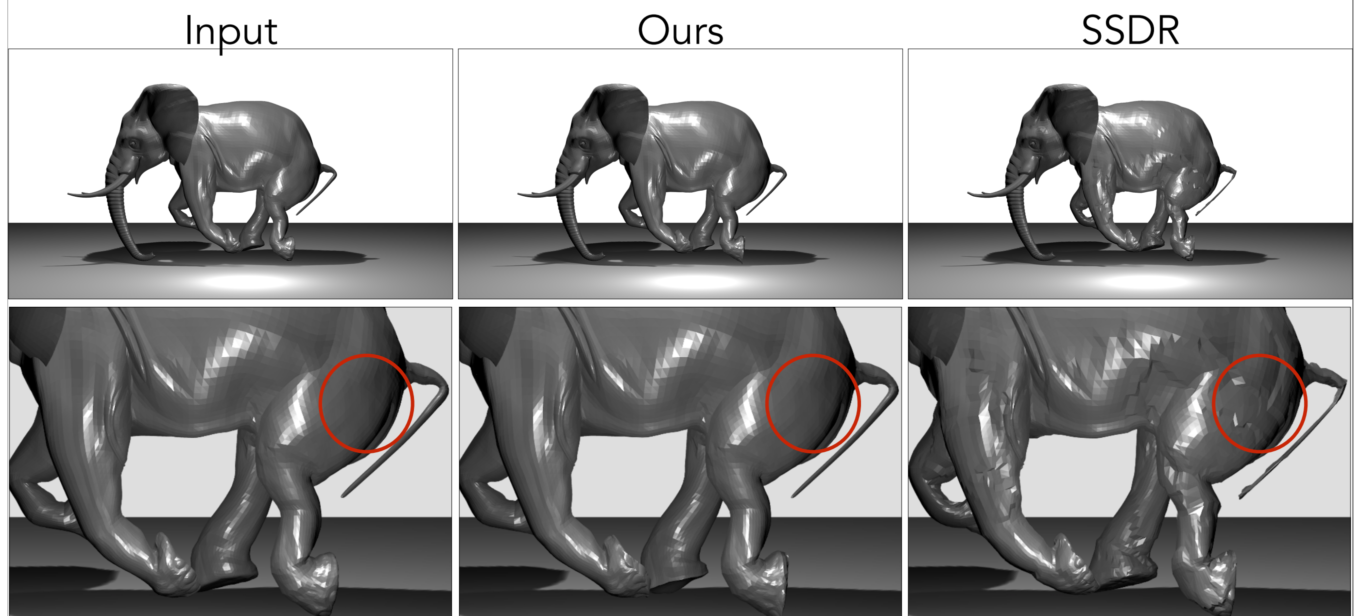
Comparison to SSDR [Le and Deng 2012]

Input | Ours | SSDR

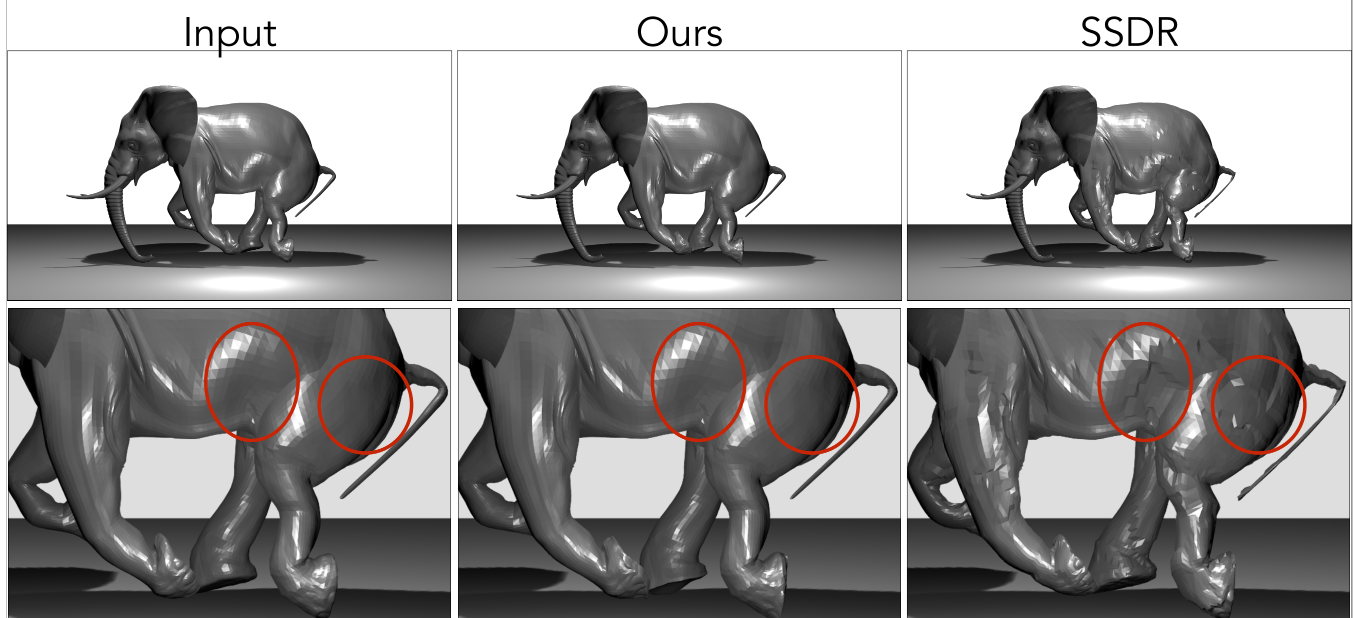


Here is a close-up. We use flat-shading to emphasize the surface quality.

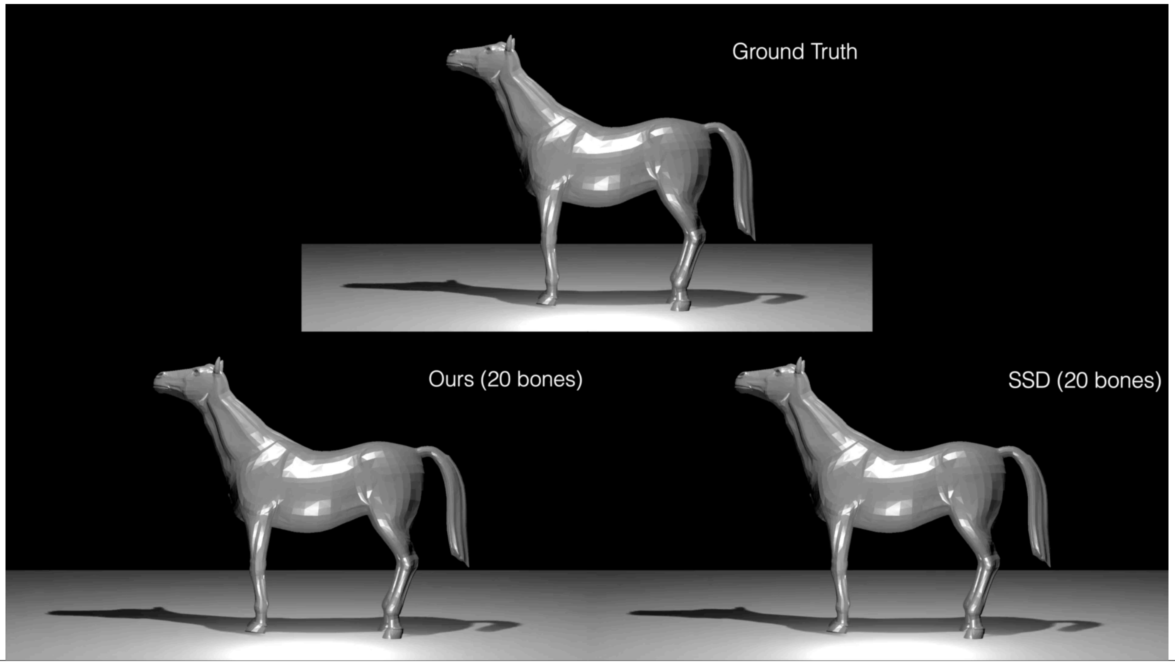
Comparison to SSDR [Le and Deng 2012]



Comparison to SSDR [Le and Deng 2012]

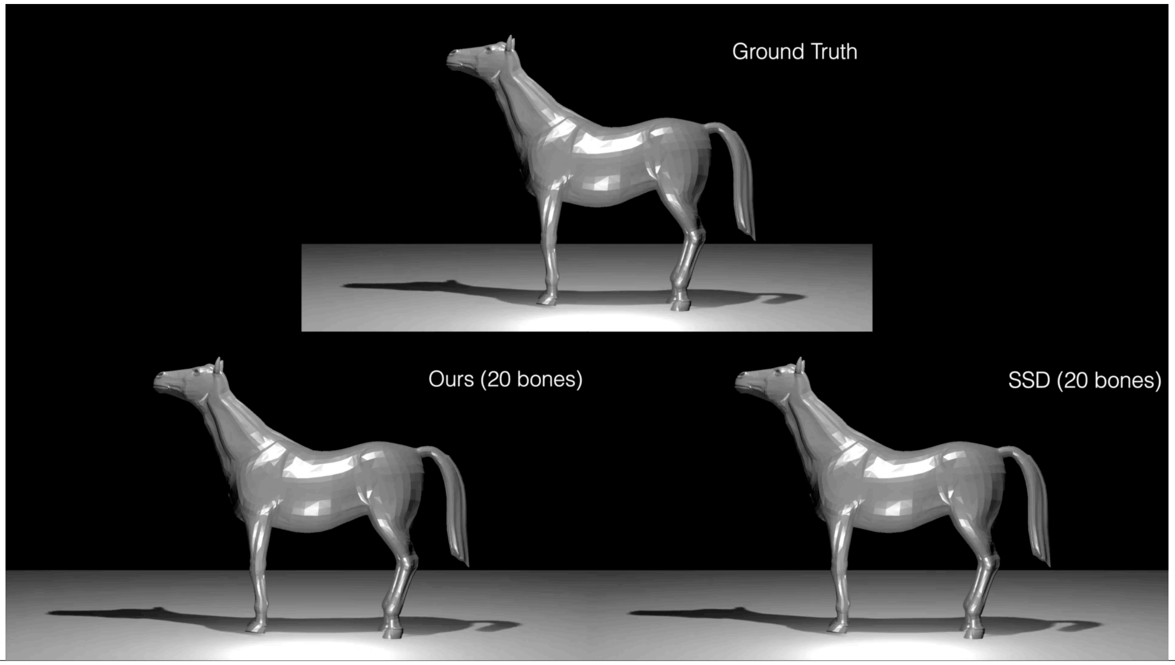


Comparison to SSDR [Le and Deng 2012]



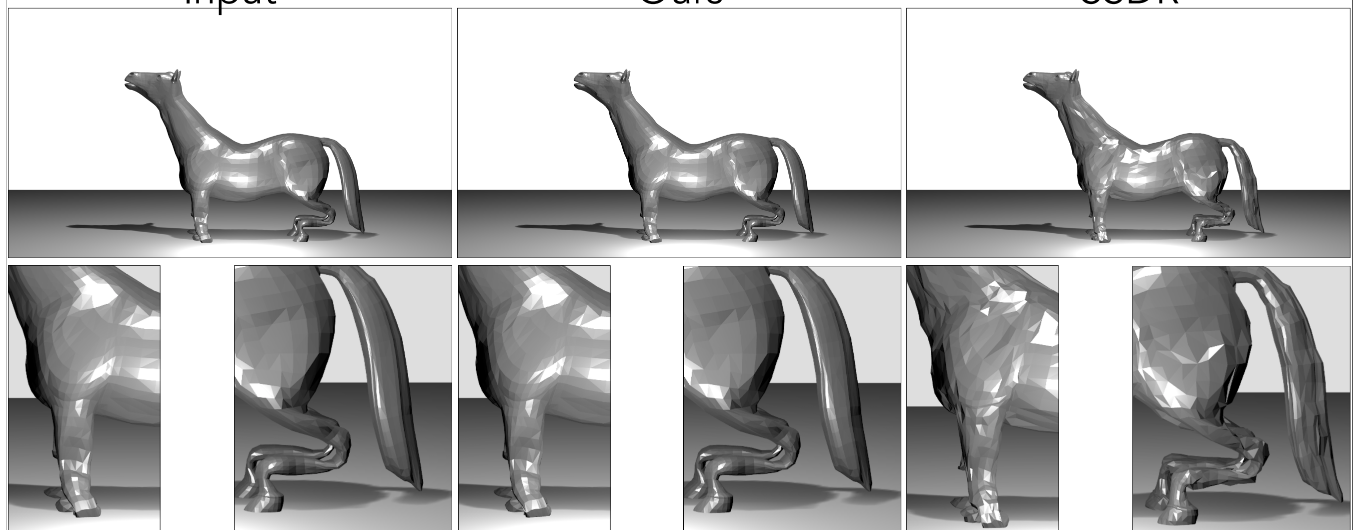
Here is another example. The horse behaves very non-rigidly.

Comparison to SSDR [Le and Deng 2012]



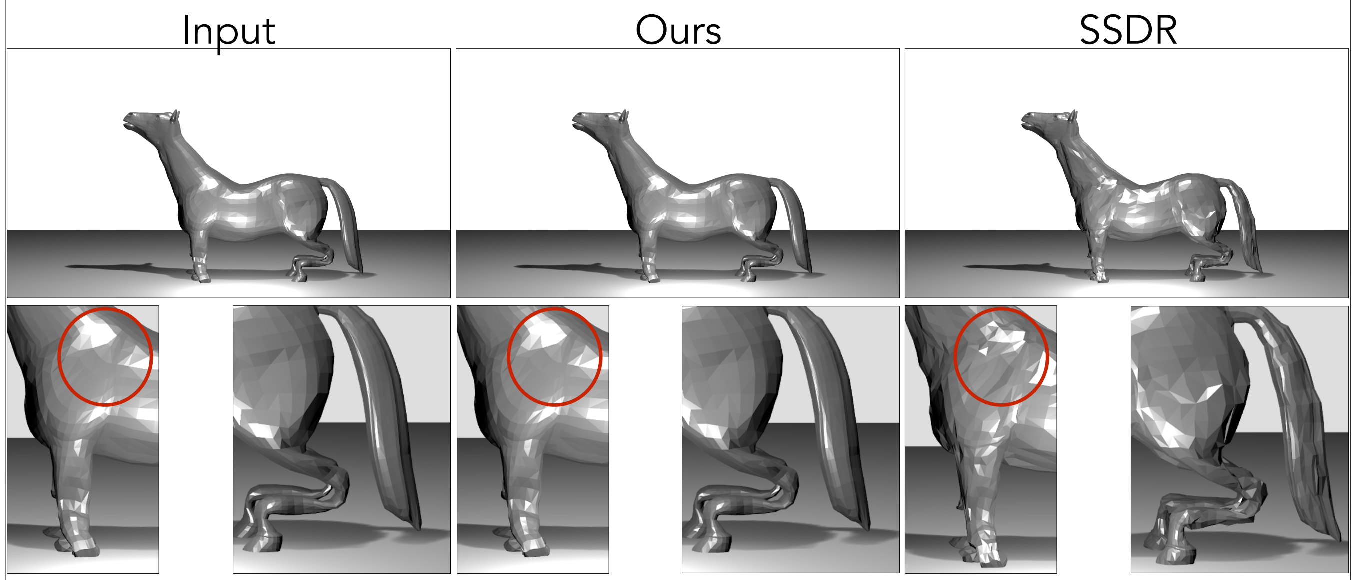
Comparison to SSDR [Le and Deng 2012]

Input | Ours | SSDR



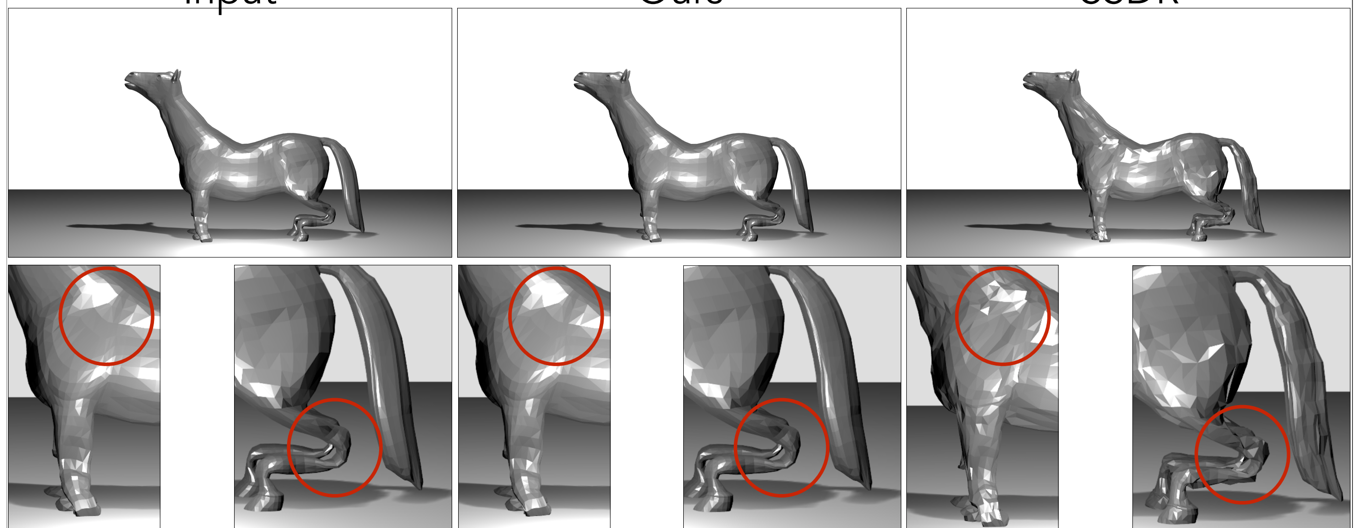
This example is particularly challenging for SSDR, since SSDR maintains transformation rigidity.

Comparison to SSDR [Le and Deng 2012]



Comparison to SSDR [Le and Deng 2012]

Input | Ours | SSDR



Comparison to Kavan et al. [2010]

Dataset	# vertices	# poses	# bones	Approx. error E_{RMS}		Execution time (minutes)	
				Kavan et al.	Ours	Kavan et al.	Ours
crane	10002	175	40	1.4	0.73	0.36	2.66
elasticCow	2904	204	18	3.6	3.23	0.08	1.16
elephant	42321	48	25	1.4	0.46	0.37	3.49
horse	8431	48	30	1.3	0.35	0.07	0.67
samba	9971	175	30	1.5	0.86	0.26	2.1

Compared to Kavan et al [2010], our approach has lower error. Our approach doesn't consider sparsity, which is sometimes a requirement.

Comparison to Kavan et al. [2010]

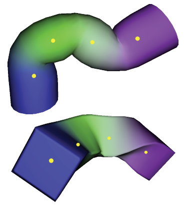
Dataset	# vertices	# poses	# bones	Approx. error E_{RMS}		Execution time (minutes)	
				Kavan et al.	Ours	Kavan et al.	Ours
crane	10002	175	40	1.4	0.73	0.36	2.66
elasticCow	2904	204	18	3.6	3.23	0.08	1.16
elephant	42321	48	25	1.4	0.46	0.37	3.49
horse	8431	48	30	1.3	0.35	0.07	0.67
samba	9971	175	30	1.5	0.86	0.26	2.1

Kavan et al's approach is highly optimized and takes advantage of their sparsity assumption.

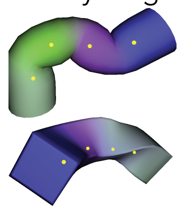
Recovering Ground Truth

- Our approach recovers ground truth for simple cases

Ground Truth

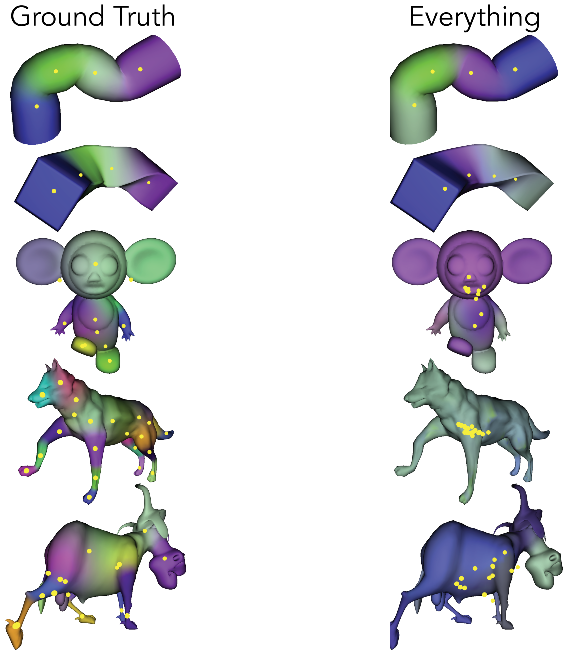


Everything



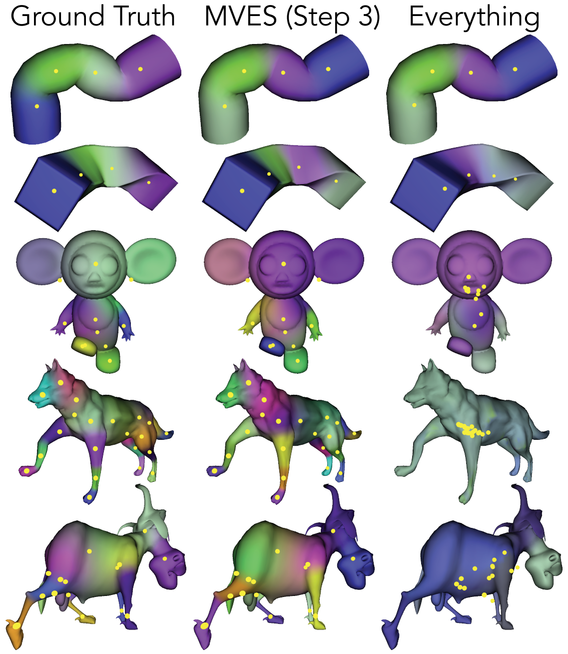
Recovering Ground Truth

- Our approach recovers ground truth for simple cases
- Always recovers vertex positions (perhaps with different handle transformations and weights)



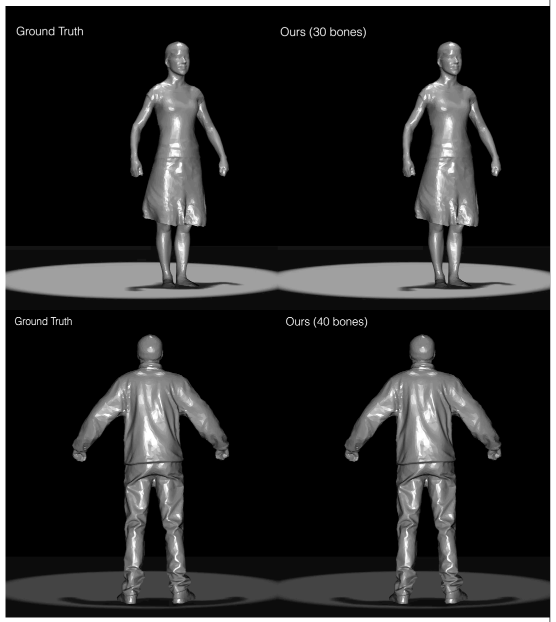
Recovering Ground Truth

- Our approach recovers ground truth for simple cases
- Always recovers vertex positions (perhaps with different handle transformations and weights)
- Given true per-vertex transformations, MVES recovers true handles and weights



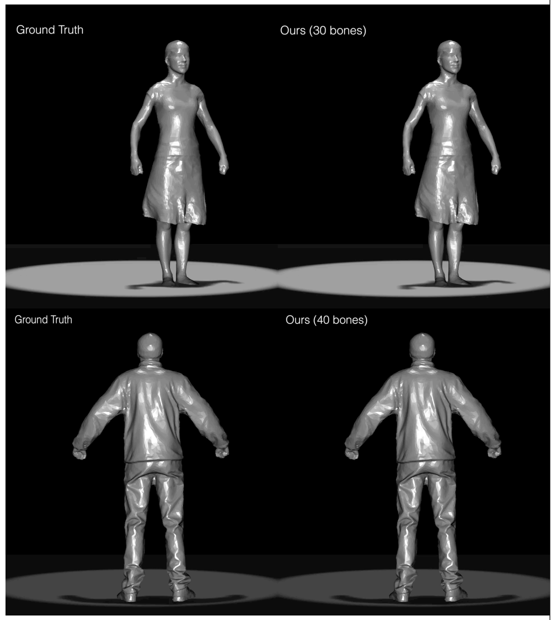
Given a known LBS rig

Mesh Animation Compression



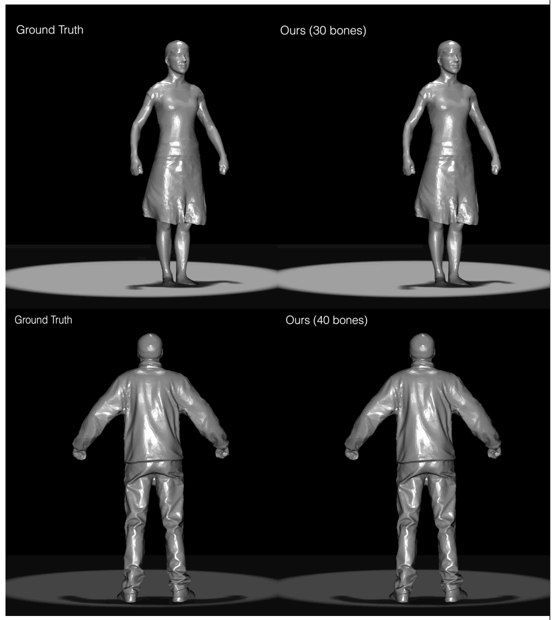
For a given bpfv, our approach has 4.6× lower error

Mesh Animation Compression



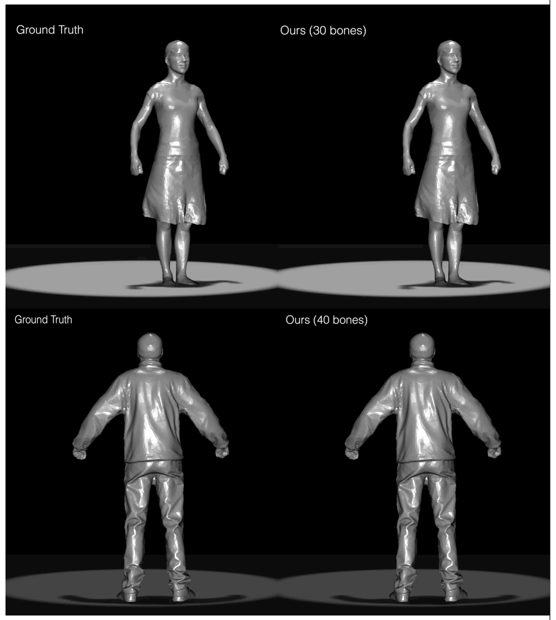
Mesh Animation Compression

- Measured in bits per vertex per frame (bpfv)



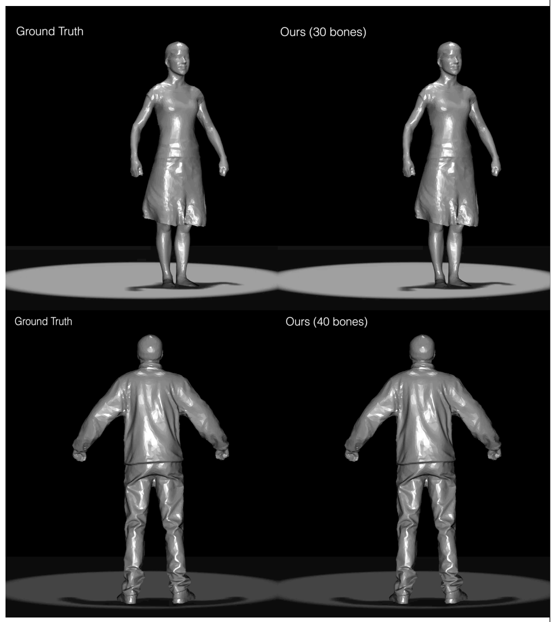
Mesh Animation Compression

- Measured in bits per vertex per frame (bpfv)
- Weights are a one-time per-vertex cost
  - $32h$ bits per vertex (h floats/vertex · 32 bits/float)



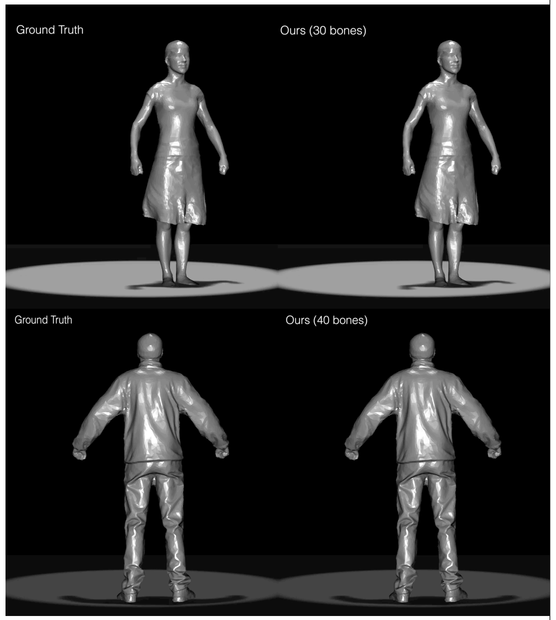
Mesh Animation Compression

- Measured in bits per vertex per frame (bpfv)
- Weights are a one-time per-vertex cost
  - 32*h* bits per vertex (h floats/vertex · 32 bits/float)
- Each frame: one affine matrix per *handle*, shared by all vertices
  - bpfv = 12*h*/#vertices · 32 bits (12 floats/handle · 32 bits/float amortized over all vertices)
  - very low incremental cost per frame

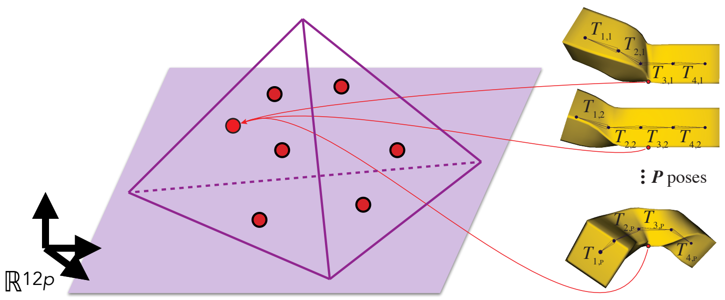


Mesh Animation Compression

- Measured in bits per vertex per frame (bpfv)
- Weights are a one-time per-vertex cost
  - 32*h* bits per vertex (h floats/vertex · 32 bits/float)
- Each frame: one affine matrix per *handle*, shared by all vertices
  - bpfv = 12*h*/#vertices · 32 bits
    (12 floats/handle · 32 bits/float amortized over all vertices)
  - very low incremental cost per frame
- 4.6× lower error than state of the art [Luo et al. 2019]

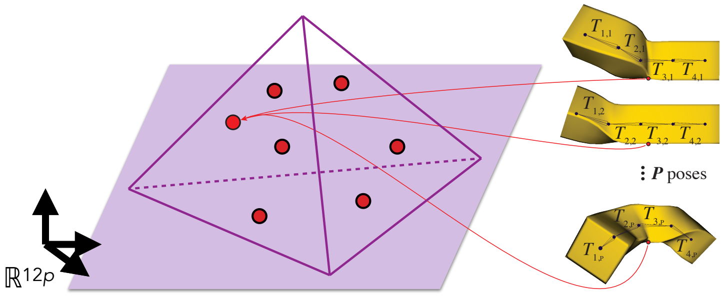


Conclusion



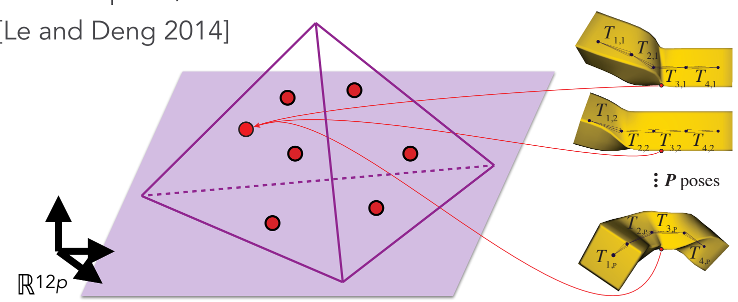
Conclusion

- Inverse Skinning is a problem in high-dimensional geometry
  - Simple expression
  - Benefits from improvements in Hyperspectral Image Unmixing
  - Benefits from improvements to the closest flat problem



Conclusion

- Inverse Skinning is a problem in high-dimensional geometry
  - Simple expression
  - Benefits from improvements in Hyperspectral Image Unmixing
  - Benefits from improvements to the closest flat problem
- Limitations
  - Transformations aren't rigid. They makes them less useful when editing.
  - No sparsity. Sometimes LBS weights aren't sparse, but this is often desirable.
  - We don't recover a bone skeleton [Le and Deng 2014]



Thank You

- Code and data: https://cragl.cs.gmu.edu/hyperskinning/

- Acknowledgements:
  - Harry Gingold, Alec Jacobson, Shahar Kovalsky, Arthur Dupre, Jie Gao, and Leonard Schulman for informative discussions
  - Kyle Falicov for help with rendering
  - Guoliang Luo for running his algorithm on our data
- Financial support
  - US NSF (IIS-1524782 & IIS-1453018)
  - Google
  - Adobe