Geometric Registration for Deformable Shapes

3.5 Articulated Registration

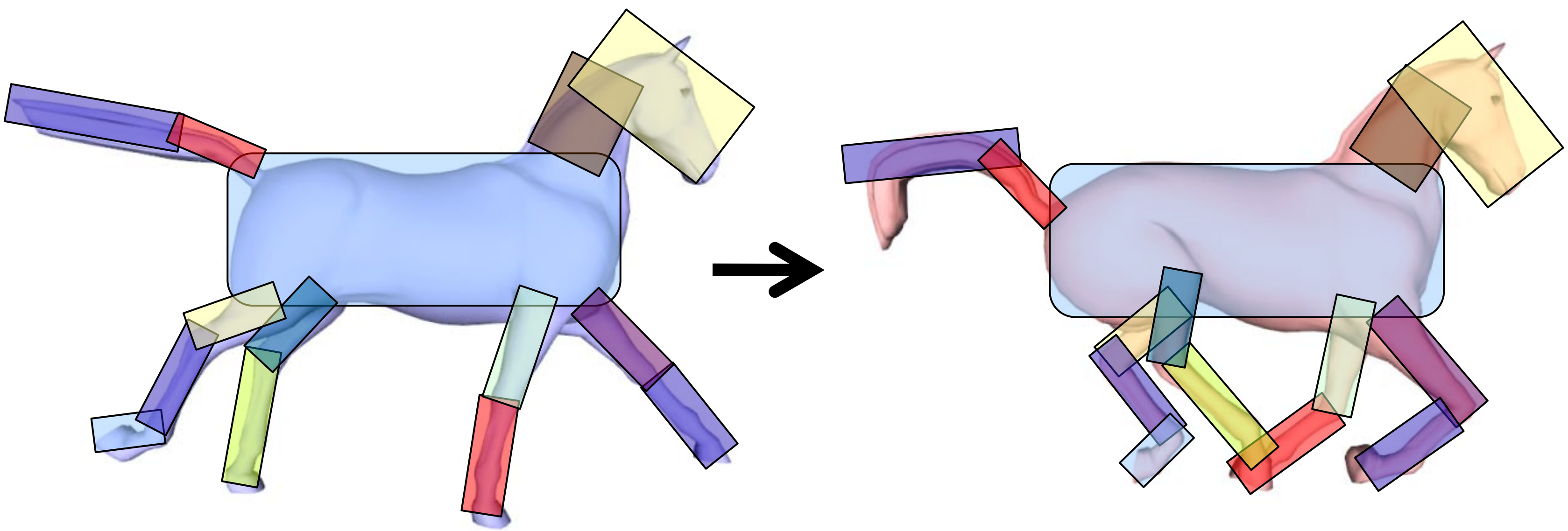
Graph cuts and piecewise-rigid registration [CZ08]
Articulated registration [CZ09]
Implementation issues and alternatives

31st Annual Conference of the
European Association for Computer Graphics
eurographics 2010

Articulated registration

Movement consists of few parts

- Material so far focused on matching individual corresp
- **Now: point groups move together**
    - Each group according to a single rigid transformation



How can we simplify the problem?

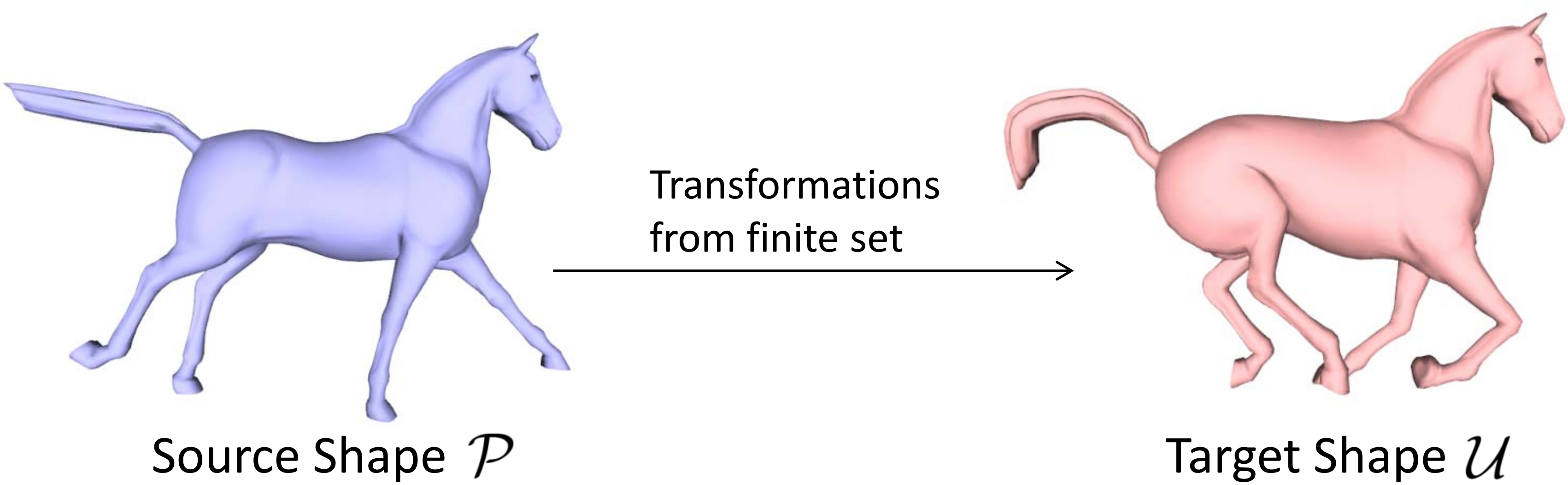
- **Before:** Optimizing individual correspondence assignment
- **Articulated:** Optimizing correspondence of groups

- Q) What are the groups?
  - Generally: don't know in advance.
  - *If we know in advance: [PG08]*

- Q) What is the motion for each group?
  - We can guess well
  - *ICP based search, feature based search*

Basic idea

- If we know the articulated movement (small set of transformations ***{T}***)
- **Reformulate optimization**
  - Find an assignment of transformations to the points that “minimizes registration error”

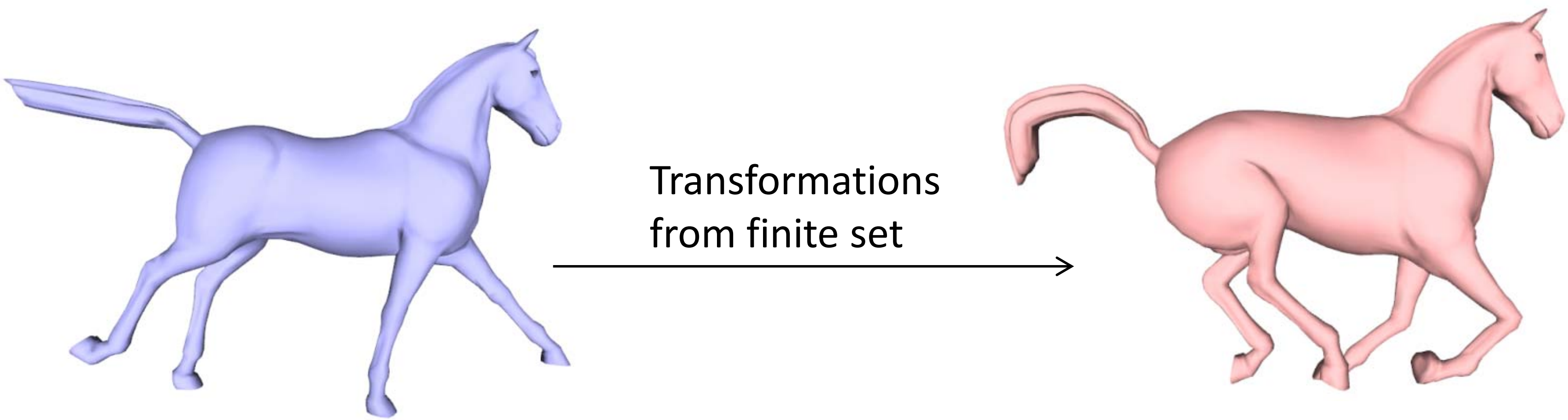


Basic idea

Find the assignment of transformations in *{T}* to points in P, that maximizes:

$$P^{(match)}(x_1,\ldots,x_n) = \prod_{i=1}^{n} P_i^{(single)} \prod_{i,j=1}^{n} P_{i,j}^{(compatible)}, x_i \in \{T\}$$

"Data" and "Smoothness" terms evaluate quality of assignment



Source Shape $\mathcal{P}$ Target Shape $\mathcal{U}$

How to find transformations?

Global search / feature matching strategy [CZ08]

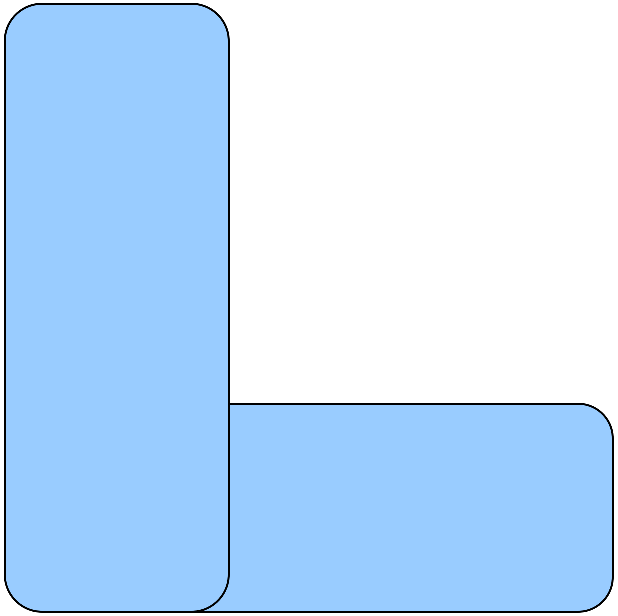
- Sample transformations in advance by feature matching
- Inspired by partial symmetry detection [MGP06]

Local search / refinement strategy [CZ09]

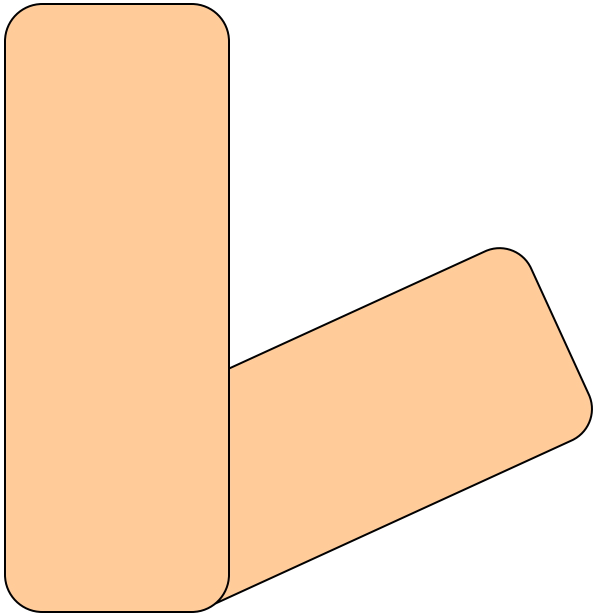
- Start with initial part labeling, keep refining transformations of each part via ICP
- Refine part labels using transformations, repeat alternation

Search by Feature Matching

Find transformations that move parts of the source to parts of the target



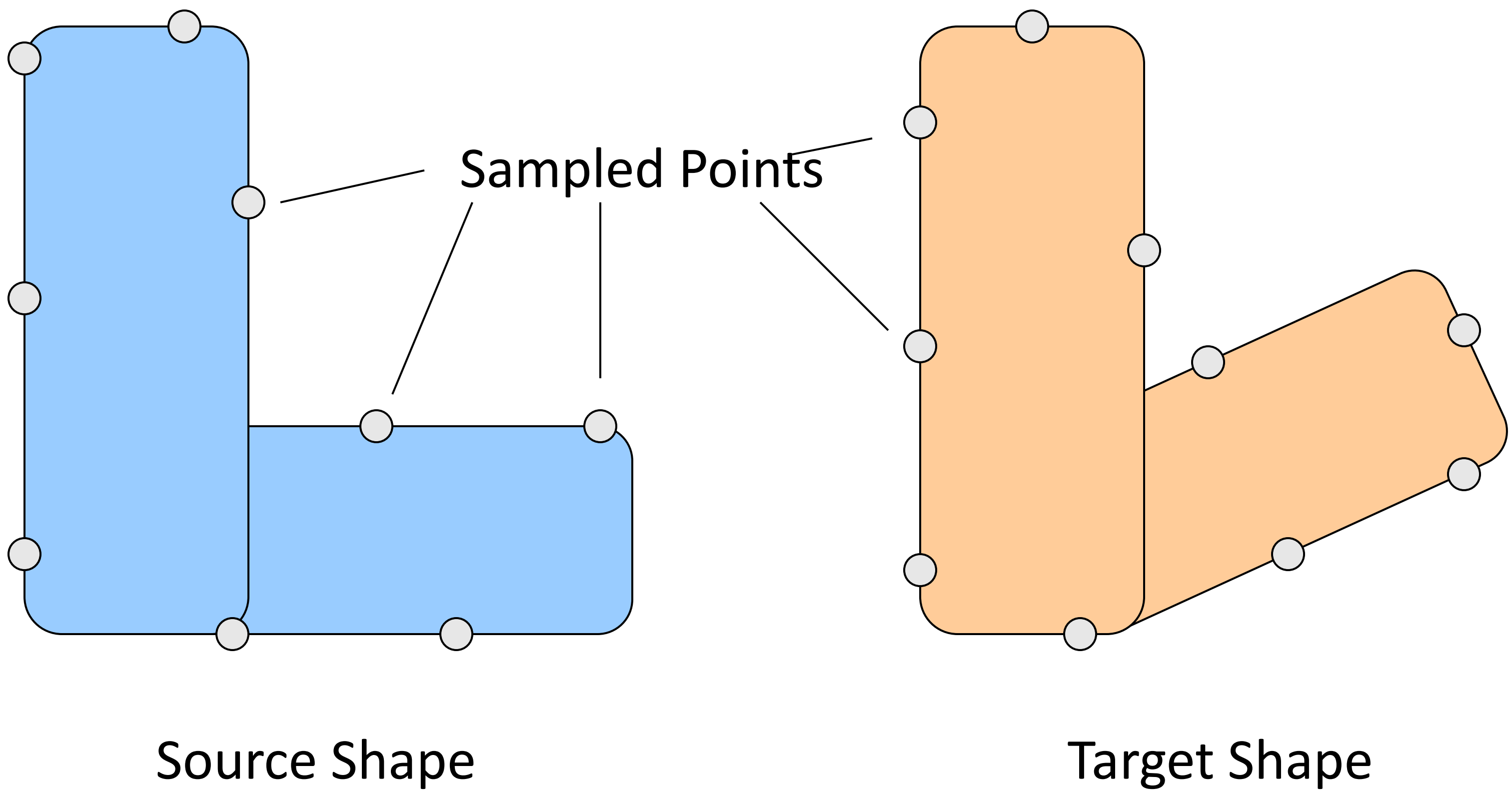
Source Shape



Target Shape

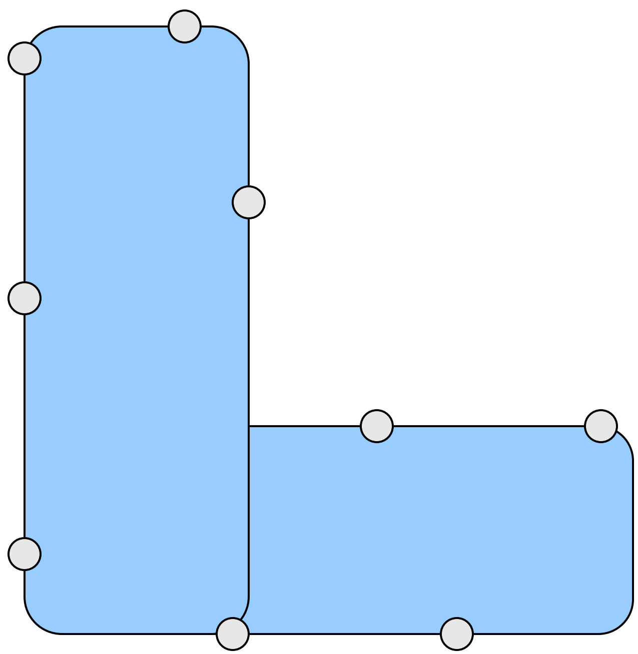
Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target

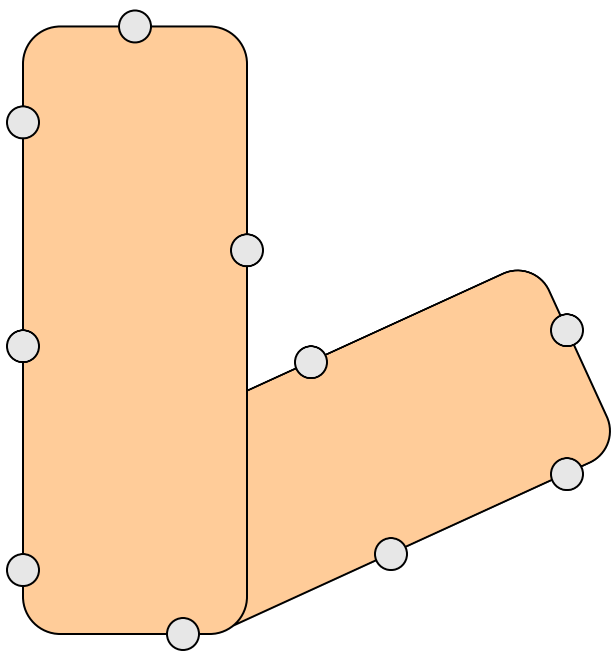


Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target



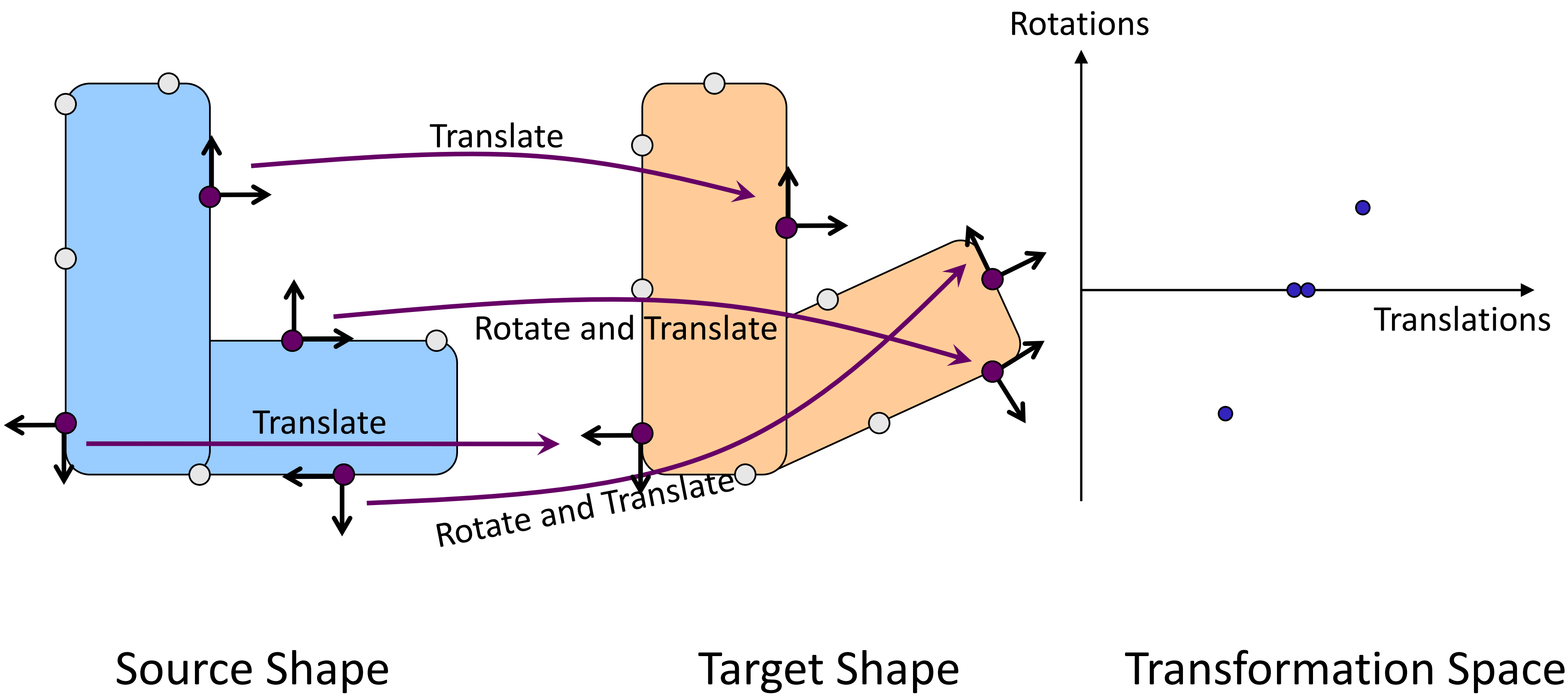
Source Shape



Target Shape

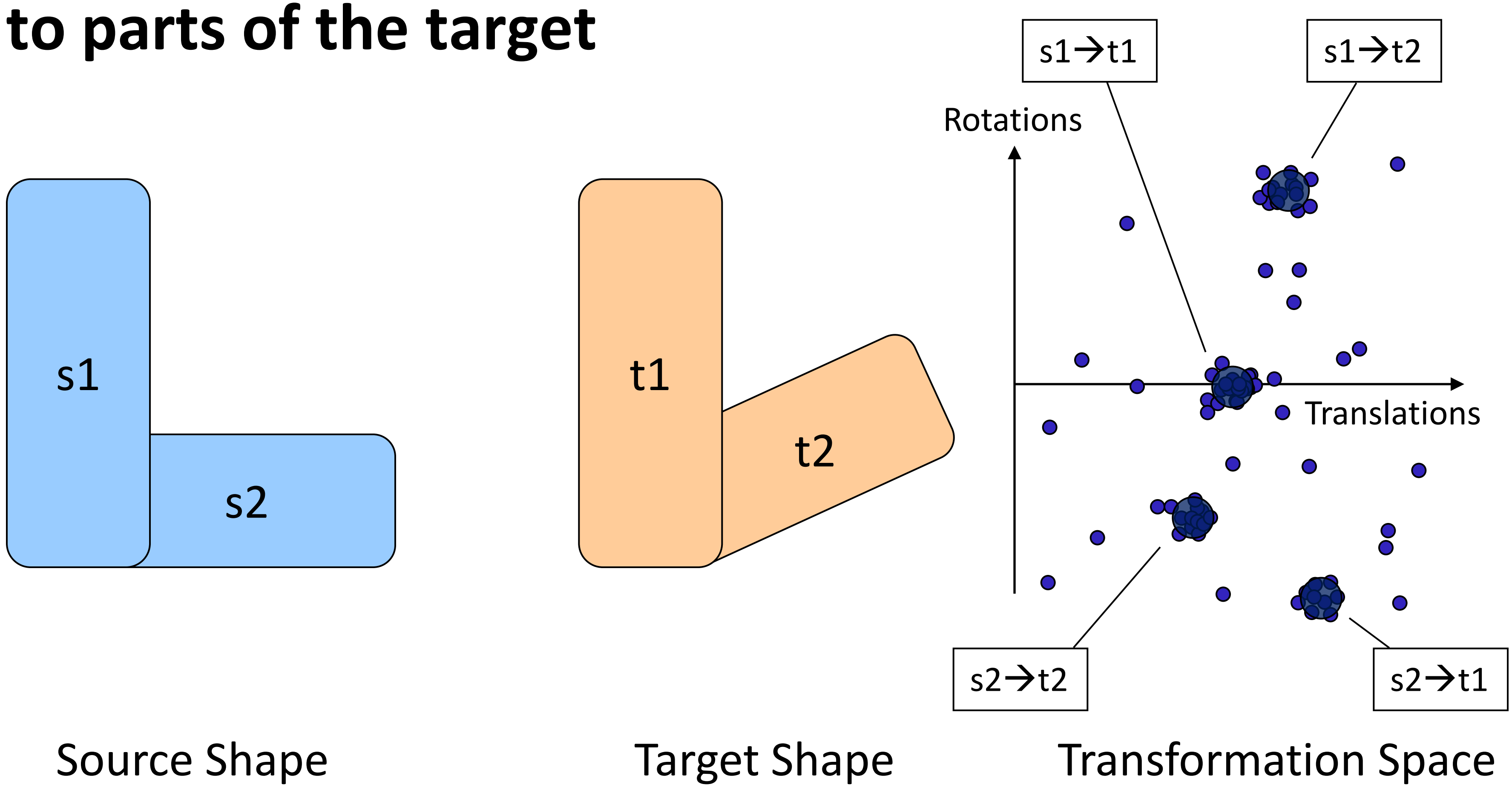
Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target



Motion Sampling Illustration

Find transformations that move parts of the source to parts of the target



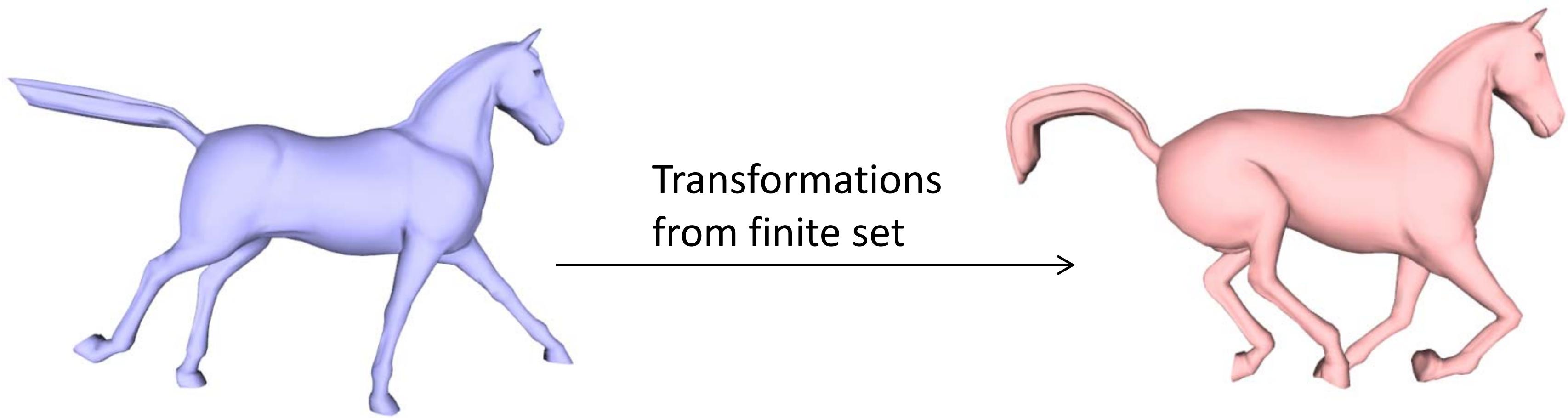
Basic idea

Find the assignment of transformations in *{T}* to points in P, that maximizes:

$$P^{(match)}(x_1,\ldots,x_n) = \boxed{\prod_{i=1}^{n} P_i^{(single)}} \boxed{\prod_{i,j=1}^{n} P_{i,j}^{(compatible)}}, x_i \in \{T\}$$

"Data" and "Smoothness" terms evaluate quality of assignment

A *discrete labelling problem* → Graph Cuts for optimization



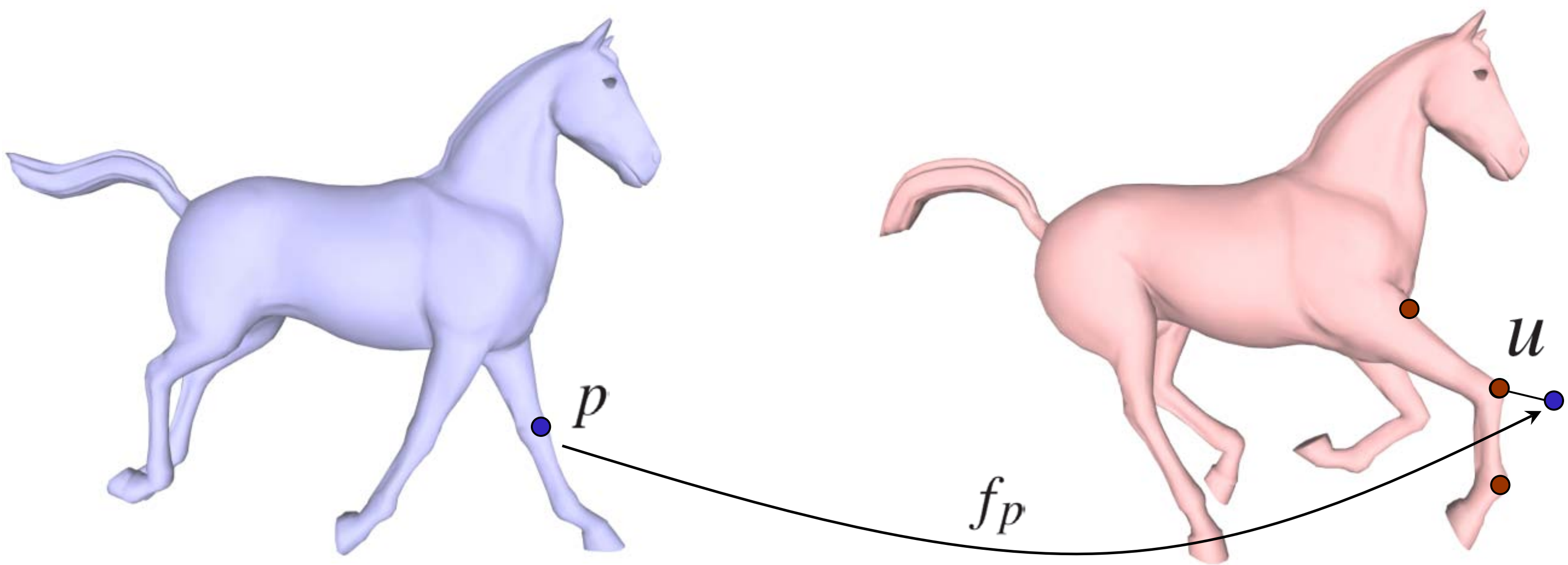
Source Shape $\mathcal{P}$ Target Shape $\mathcal{U}$

Data Term

For each mesh vertex: Move close to target

How to measure distance to target?

- Apply assigned transformation f_p for all p = $f_p(p)$
- Measure distance to closest point u in target

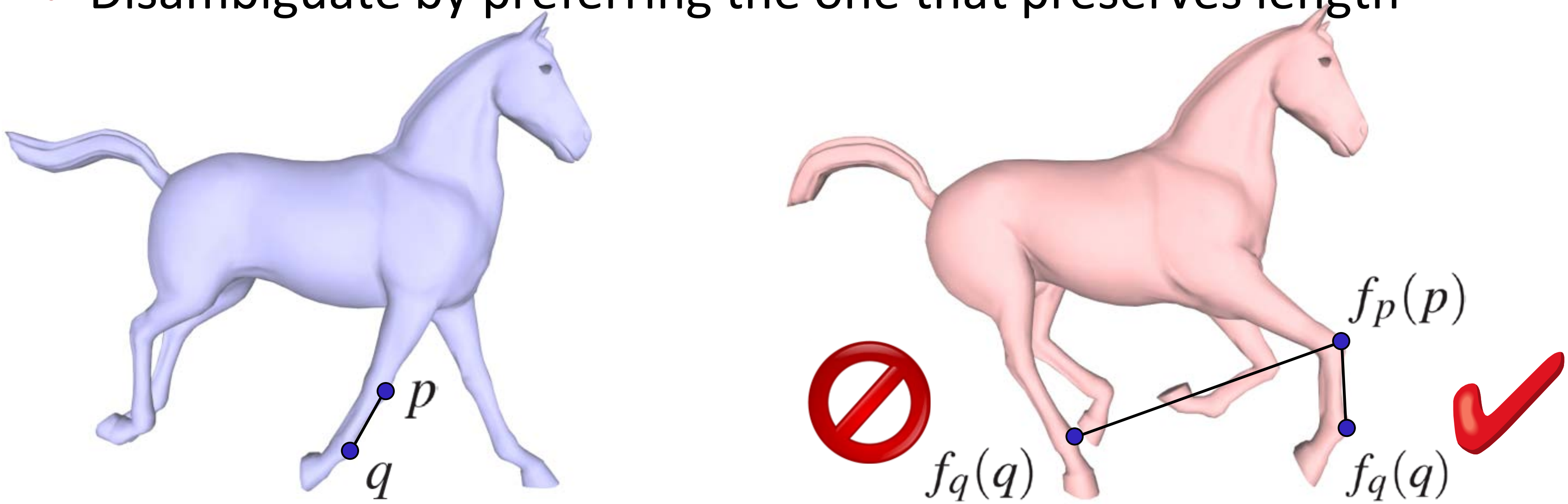


Smoothness Term

For each mesh edge: preserve length of edge

$$V(p,q,f_p,f_q) = \left| \underbrace{\|p-q\|}_{\text{Original Length}} - \underbrace{\|f_p(p)-f_q(q)\|}_{\text{Transformed Length}} \right|$$

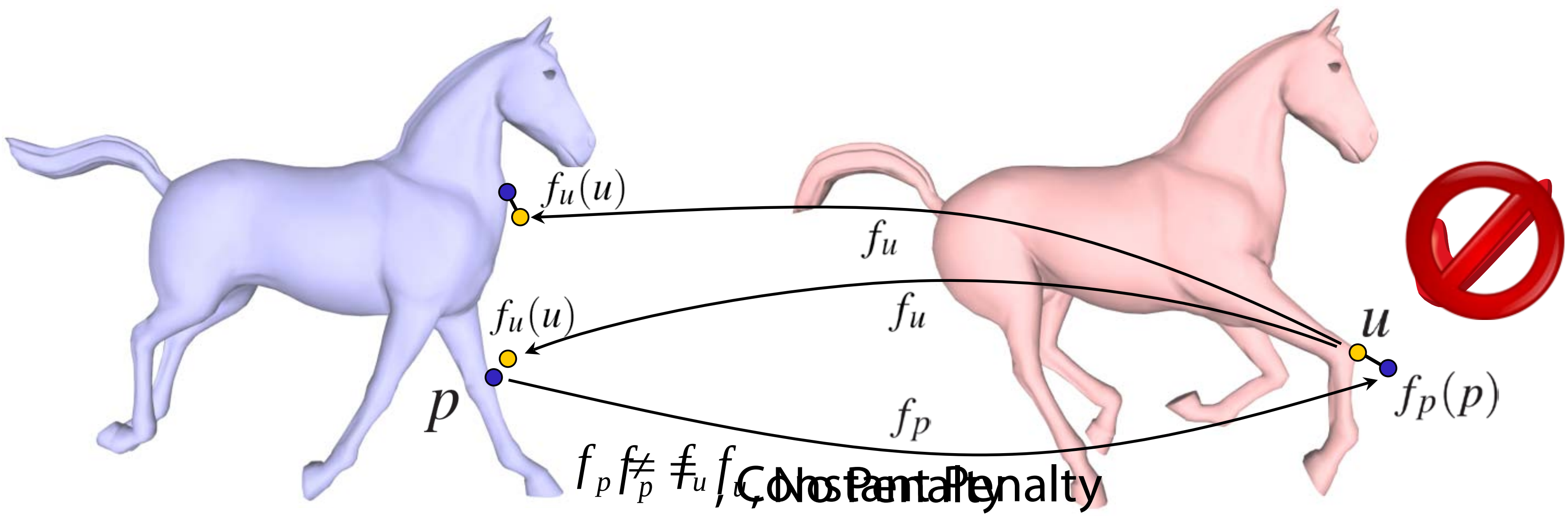
- Both versions of $f_q(q)$ moved q close to the target
- Disambiguate by preferring the one that preserves length



Symmetric Cost Function

Swapping source / target can give different results

- Optimize *{T}* assignment in both meshes
- Assign *{T}* on source vertices, *{T⁻¹}* on target vertices
- Enforce consistent assignment: penalty when $f_p \neq f_u$

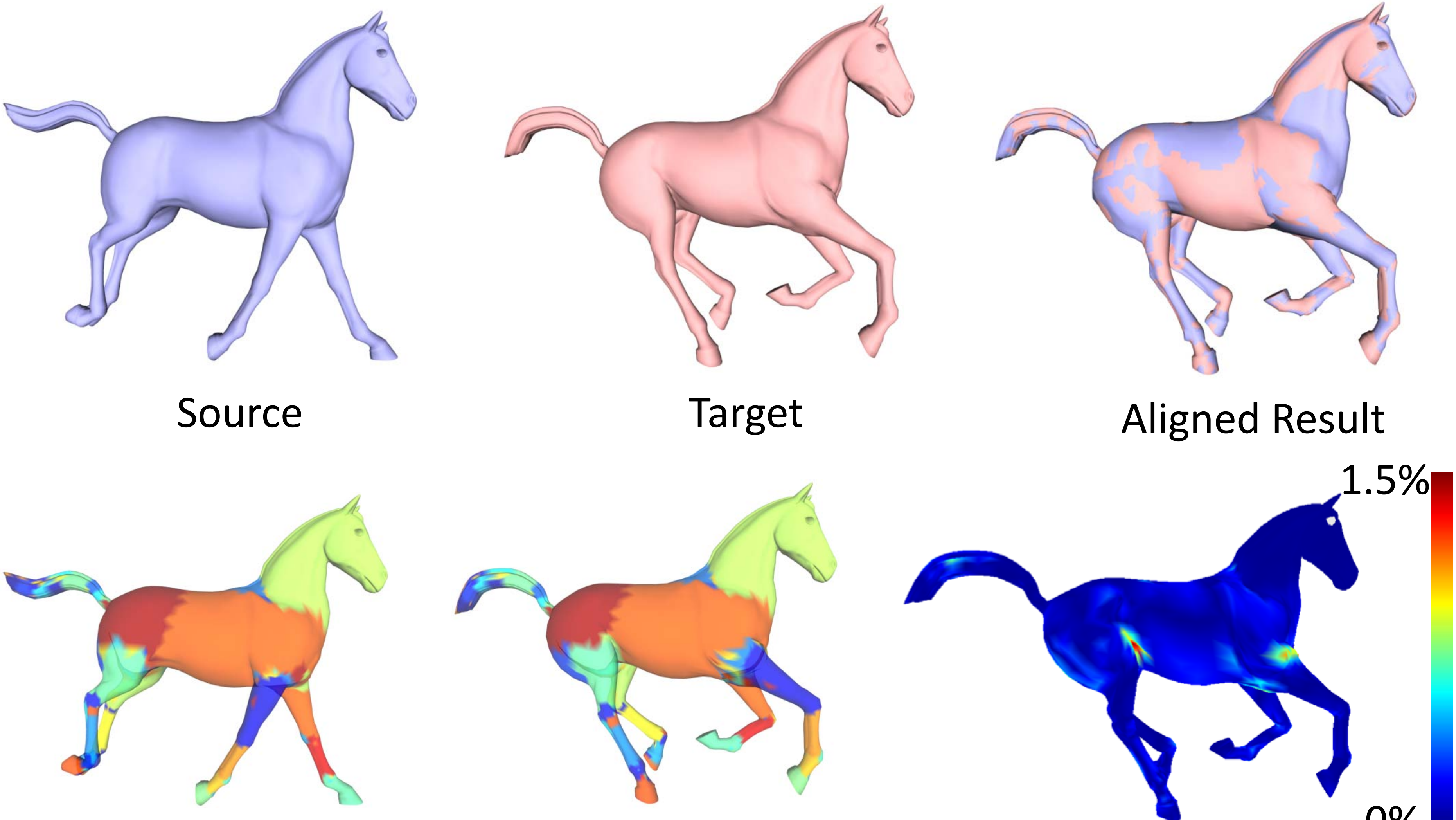


Optimization Using Graph Cuts

$$\underset{\substack{\text{Assignment from a set}\\ \text{of transformations}}}{\textbf{argmin}} \quad \text{Data}_{\textit{Source}} + \text{Smoothness}_{\textit{Source}} + \text{Data}_{\textit{Target}} + \text{Smoothness}_{\textit{Target}} + \text{Symmetric Consistency}_{\textit{Source \& Target}}$$

- Data and smoothness terms apply to both shapes
- Additional symmetric consistency term
- Weights to control relative influence of each term
- Use "graph cuts" to optimize assignment
  - [Boykov, Veksler & Zabih PAMI '01]

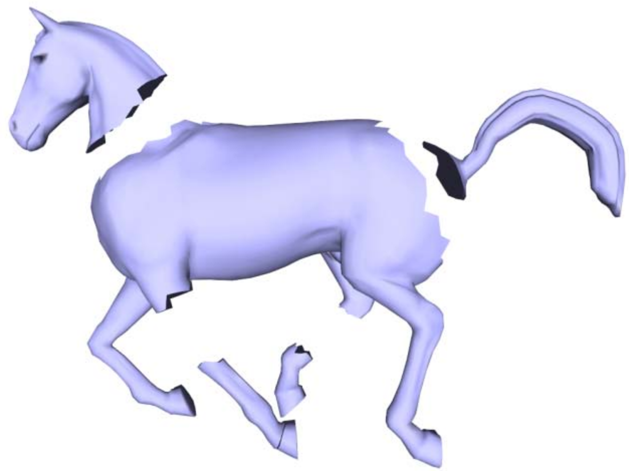
Synthetic Dataset Example



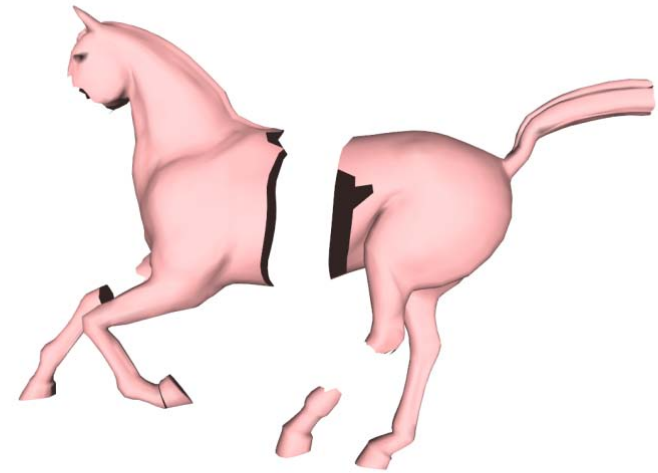
Motion Segmentation (from Graph Cuts)

Registration Error

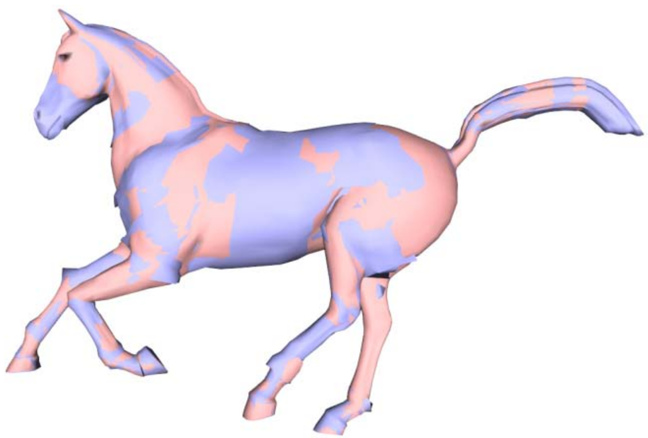
Synthetic Dataset w/ Holes



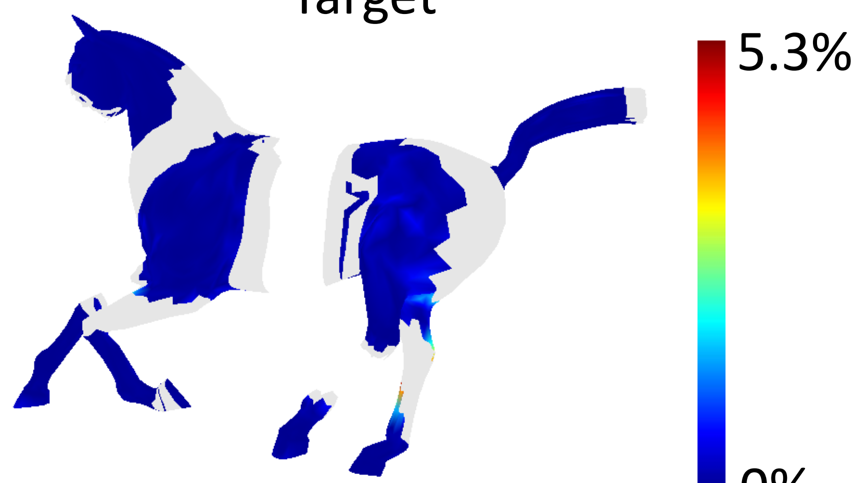
Source



Target

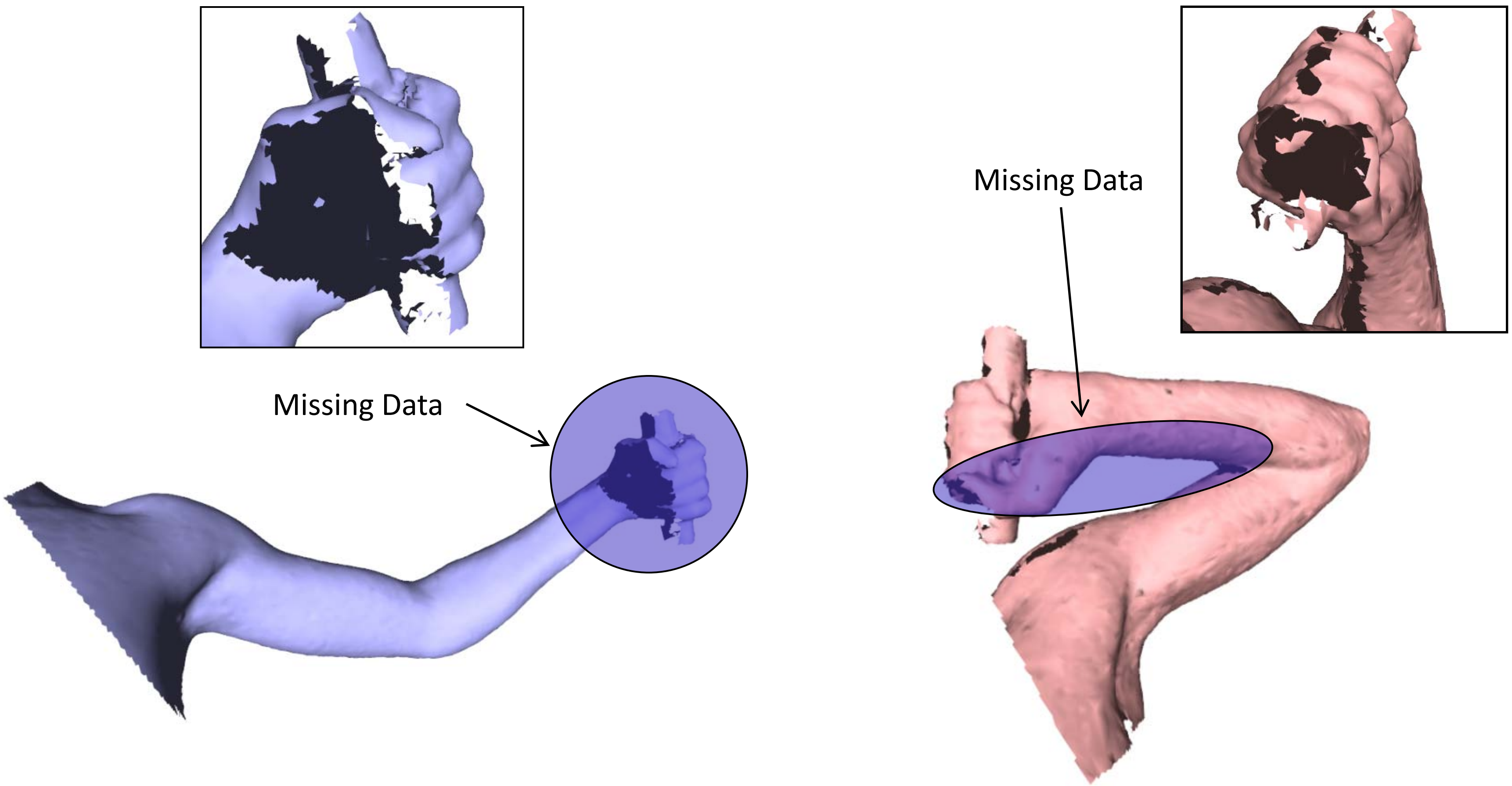


Aligned Result



Distance (from Target) to the closest point
(% bounding box diagonal)

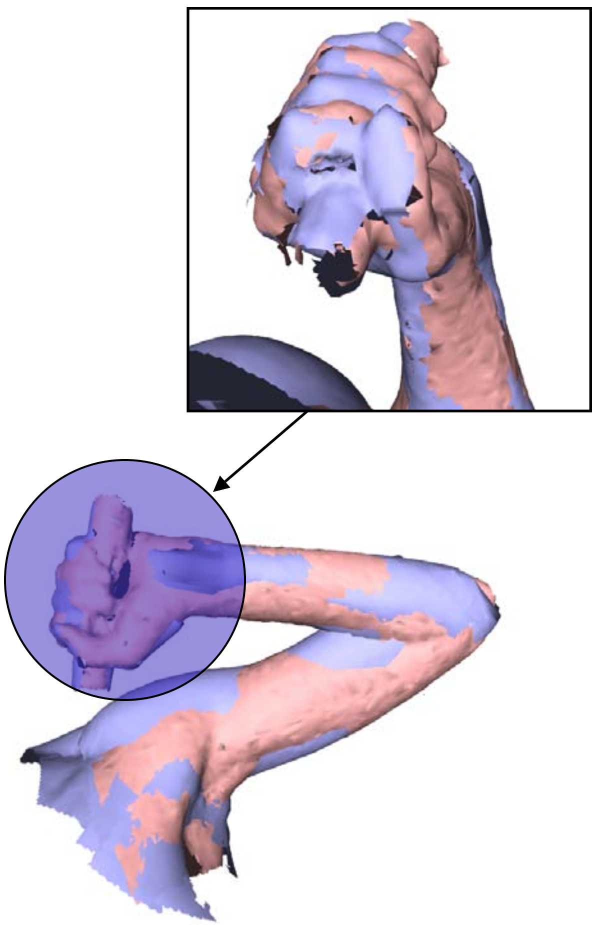
Arm Dataset Example



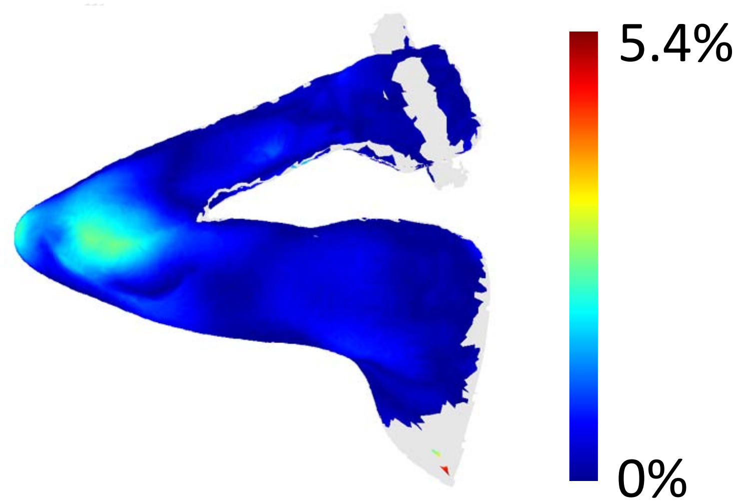
Source

Noisy Target

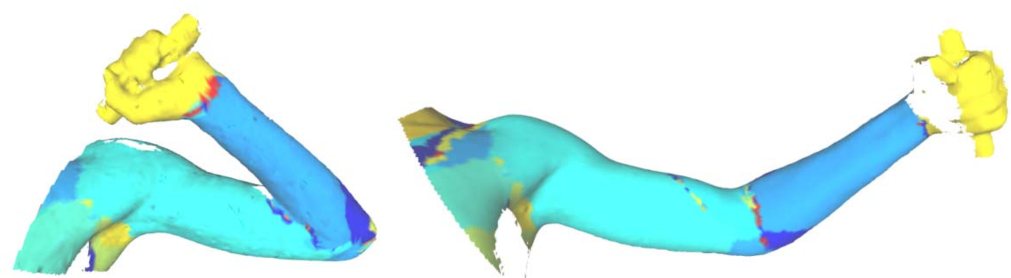
Arm Dataset Example



Aligned Result



Distance (from Target) to the closest point (% bounding box diagonal)



Motion Segmentation

Performance

Dataset	#Points	# Labels	Matching	Clustering	Pruning	Graph Cuts
Horse	8431	1500	2.1 min	3.0 sec	(skip) 1.6 sec	1.1 hr
Arm	11865	1000	55.0 sec	0.9 sec	12.4 min	1.2 hr
Hand (Front)	8339	1500	14.5 sec	0.7 sec	7.4 min	1.2 hr
Hand (Back)	6773	1500	17.3 sec	0.9 sec	9.4 min	1.6 hr

Graph cuts optimization is most time-consuming step

- Symmetric optimization doubles variable count
- Symmetric consistency term introduces many edges

Performance improved by subsampling

- Use k-nearest neighbors for connectivity

How to find transformations?

Global search / feature matching strategy

- Sample transformations in advance by feature matching
- Inspired by partial symmetry detection [MGP06]

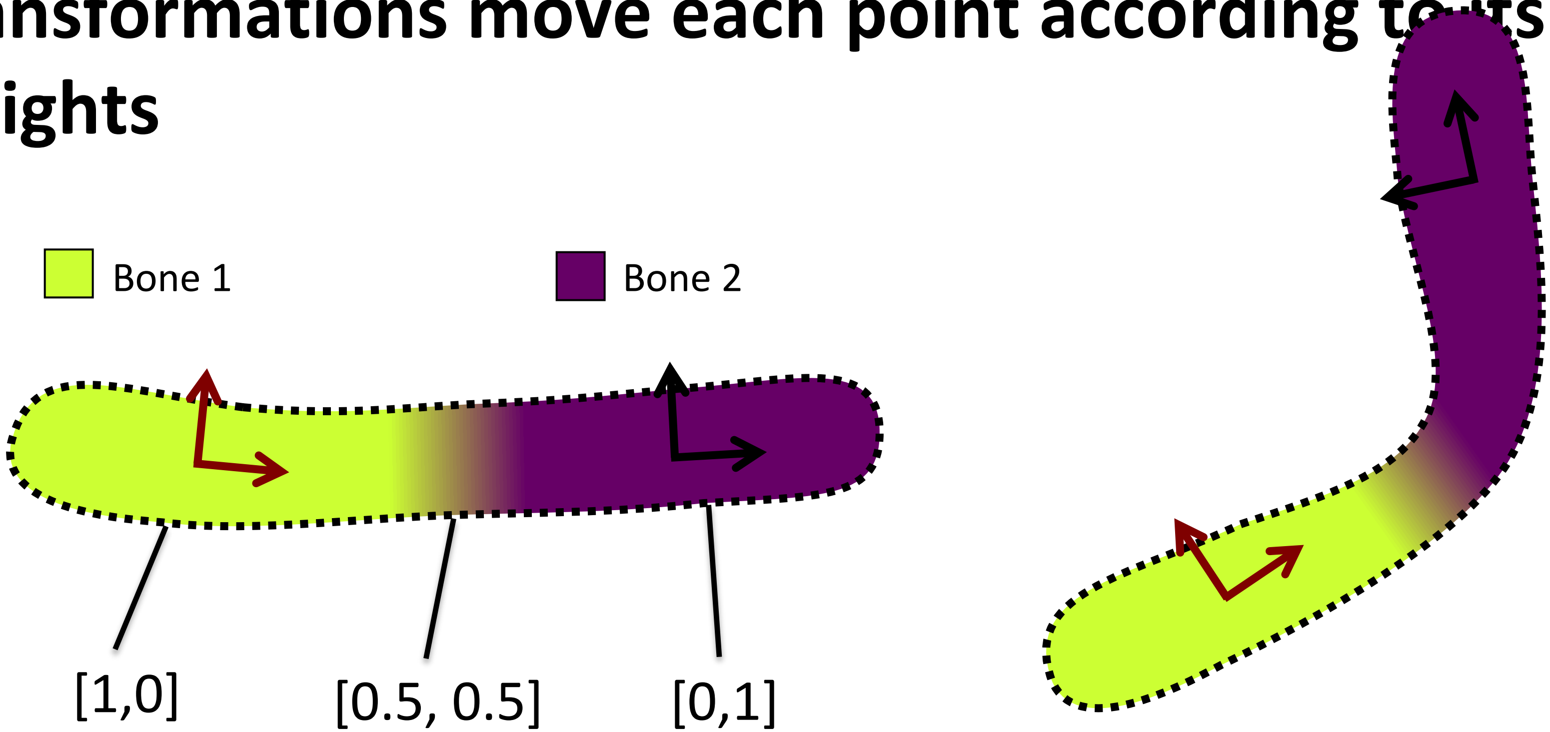
Local search / refinement strategy

- Start with initial part labeling, keep refining transformations of each part via ICP
- Iterate between transformation refinement / part assignment until convergence
- Establish relationship between parts → preserve shape connectivity & obtain deformable model

Part label representation

Part labelling: each point assigned vector of weights

Transformations move each point according to its weights

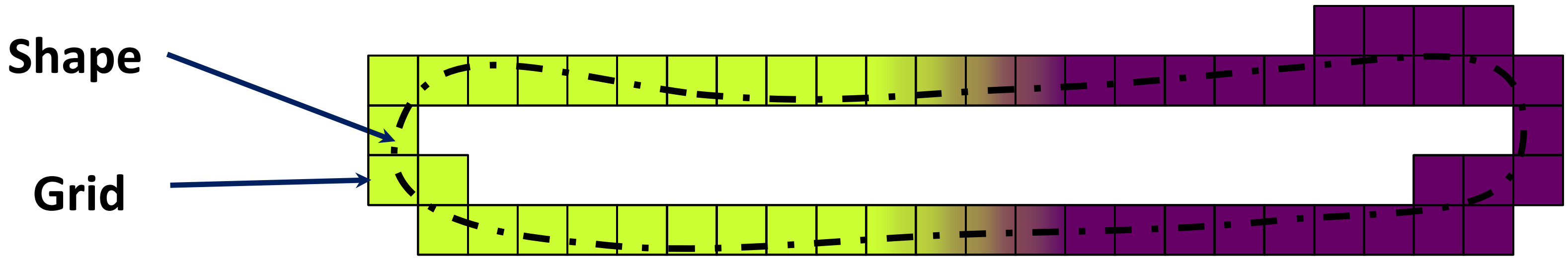


Weighted Blending Result

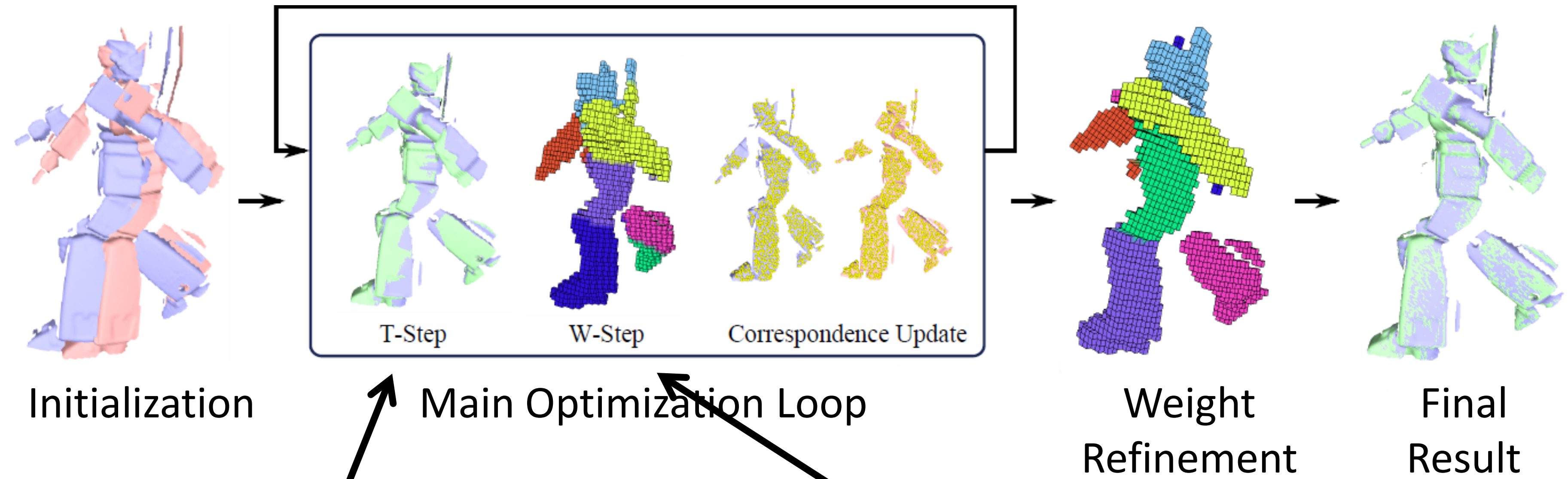
Weight Grid

Define weights on grid enclosing surface

- Covers small holes, reduces variables
- Provides regular structure for optimization
- Trilinear interpolation inside grid cells – gives weights everywhere inside the grid domain

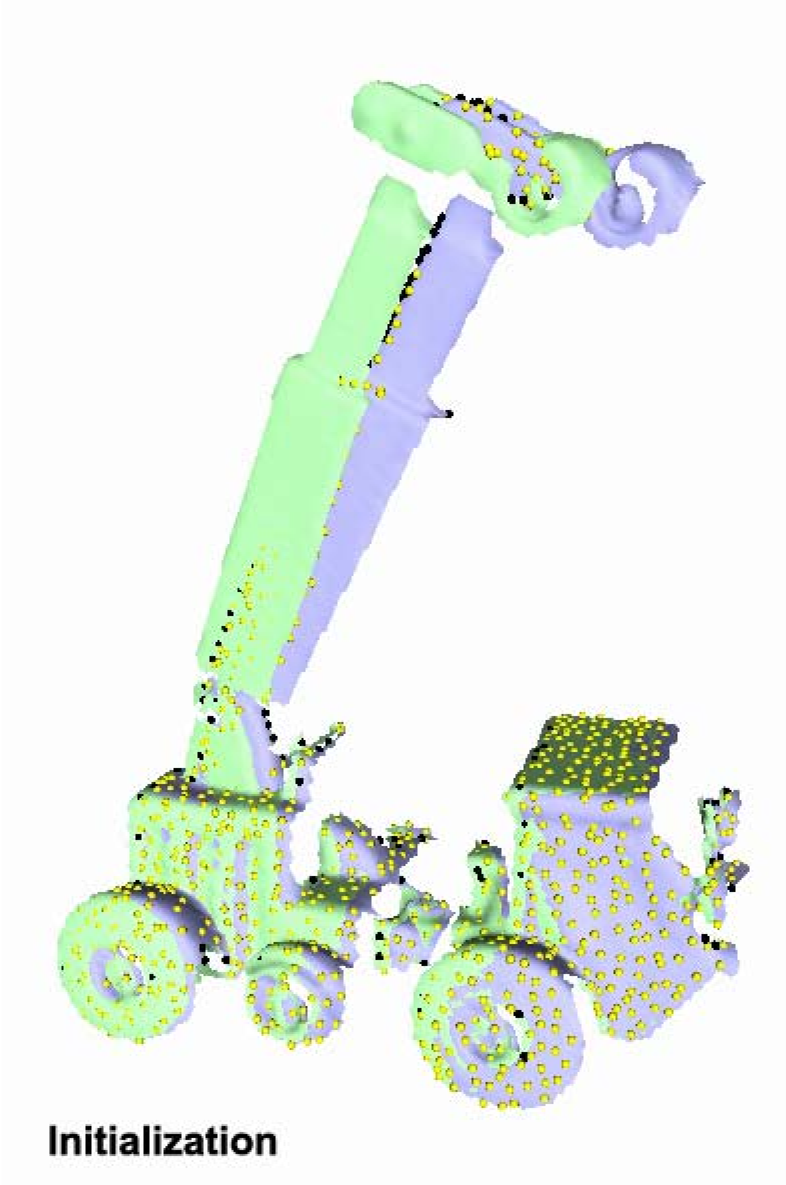


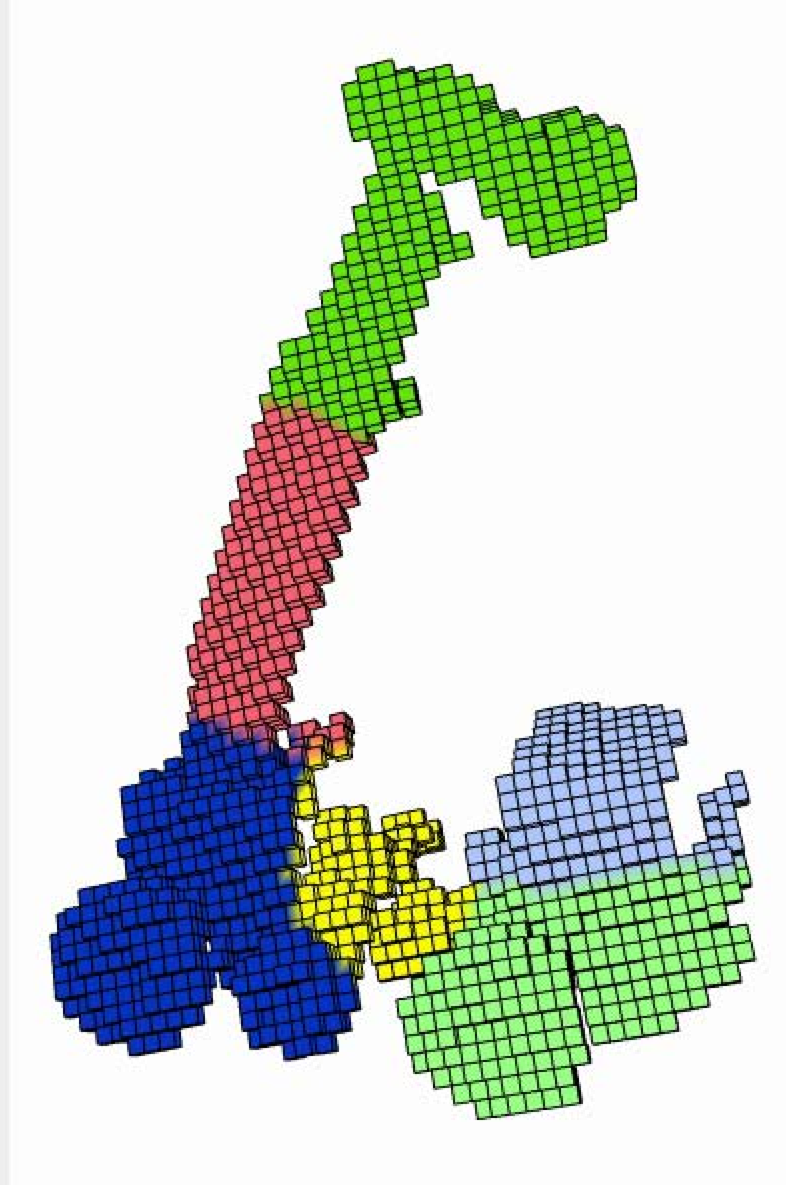
Optimization overview

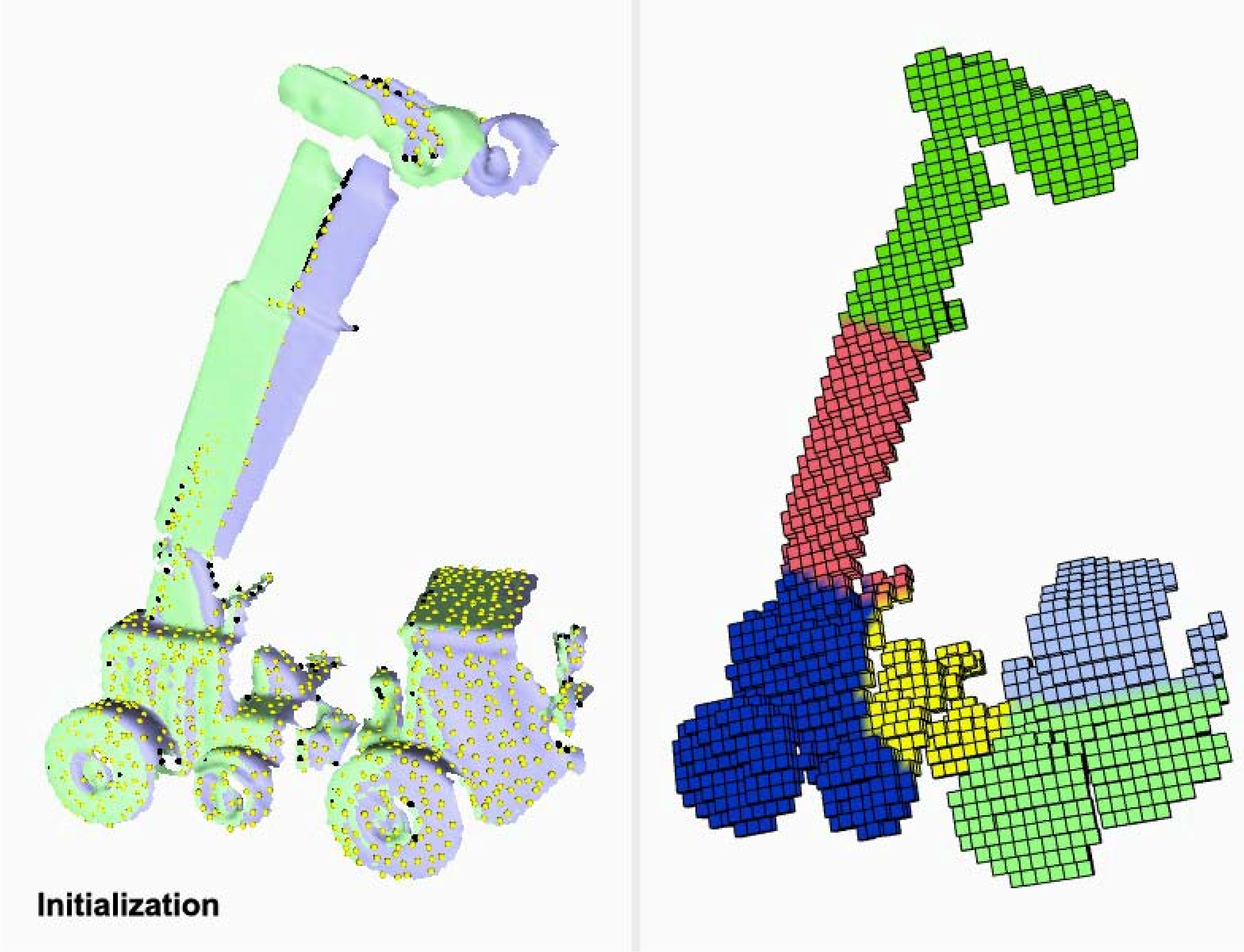


T-Step: Refine the transformations for each part

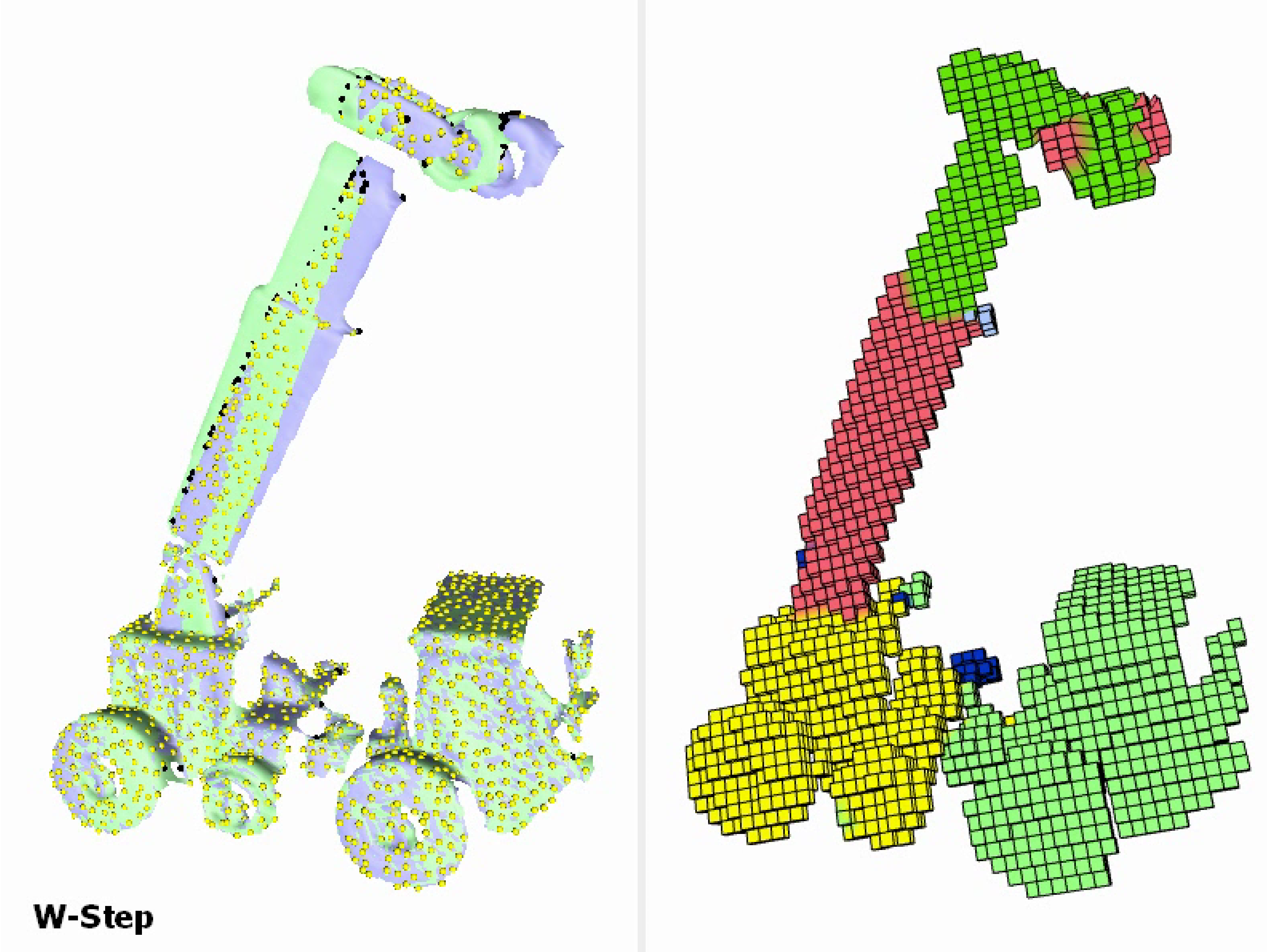
W-Step: Optimize assignment of transformations

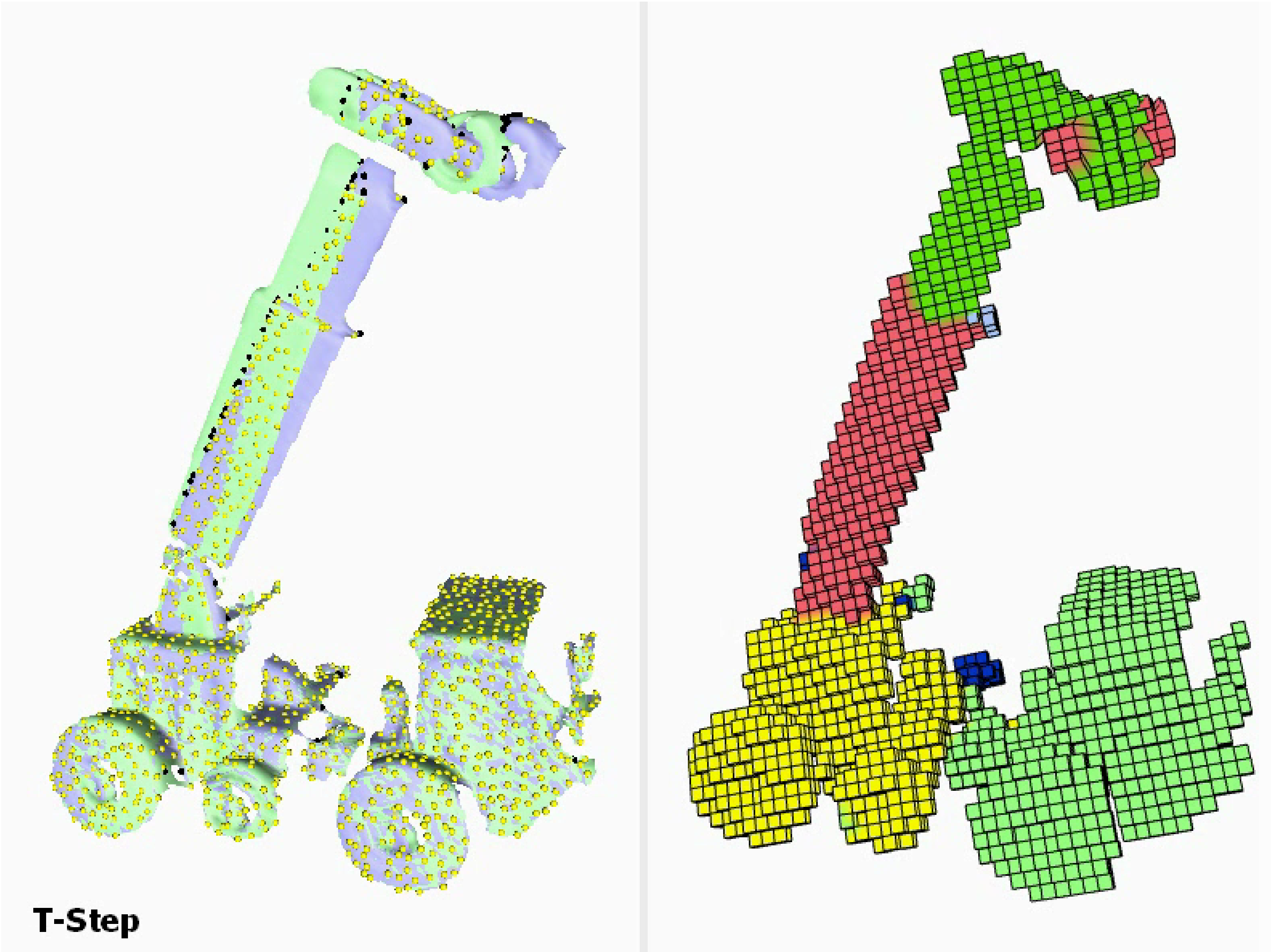

Initialization



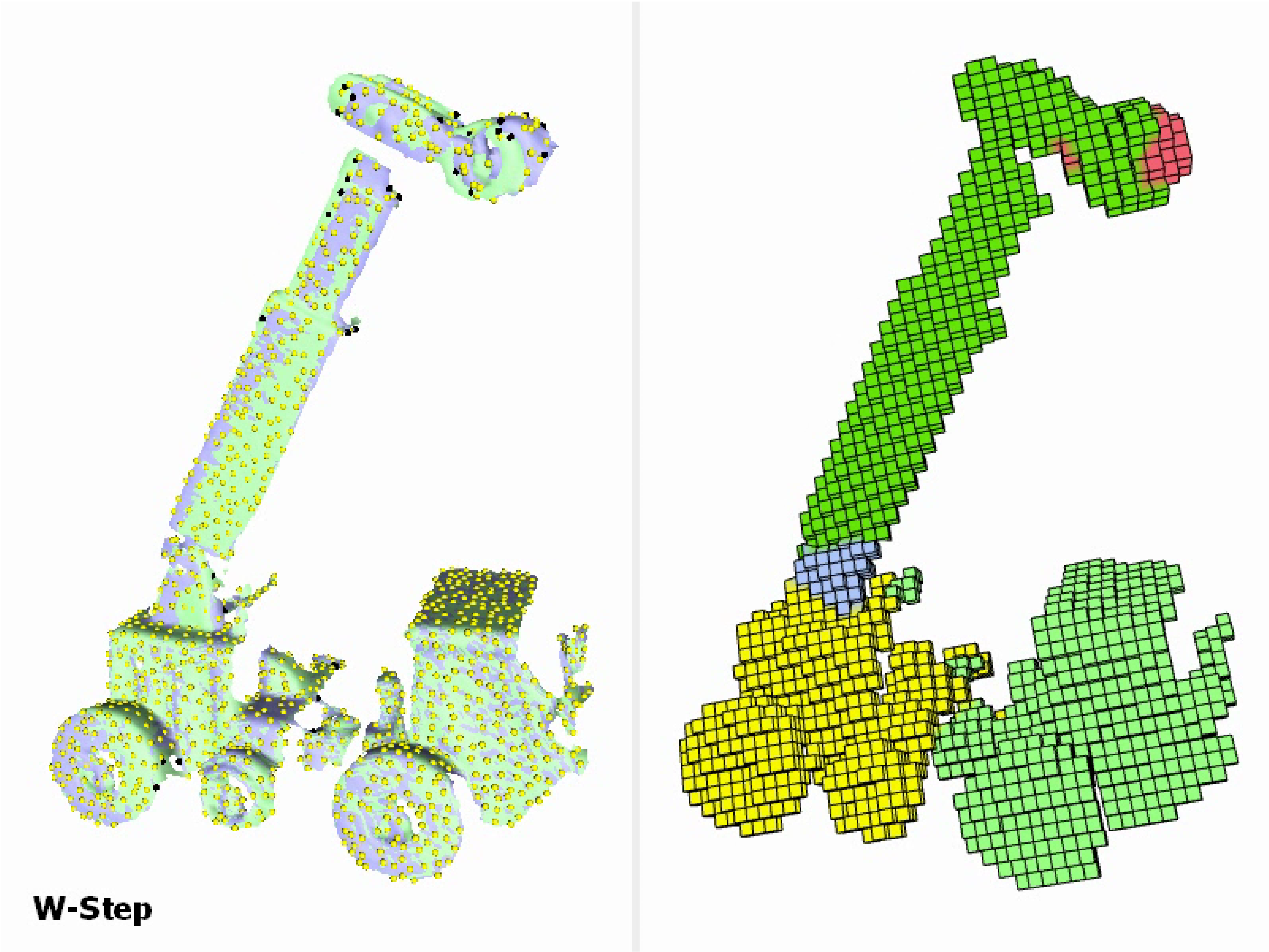


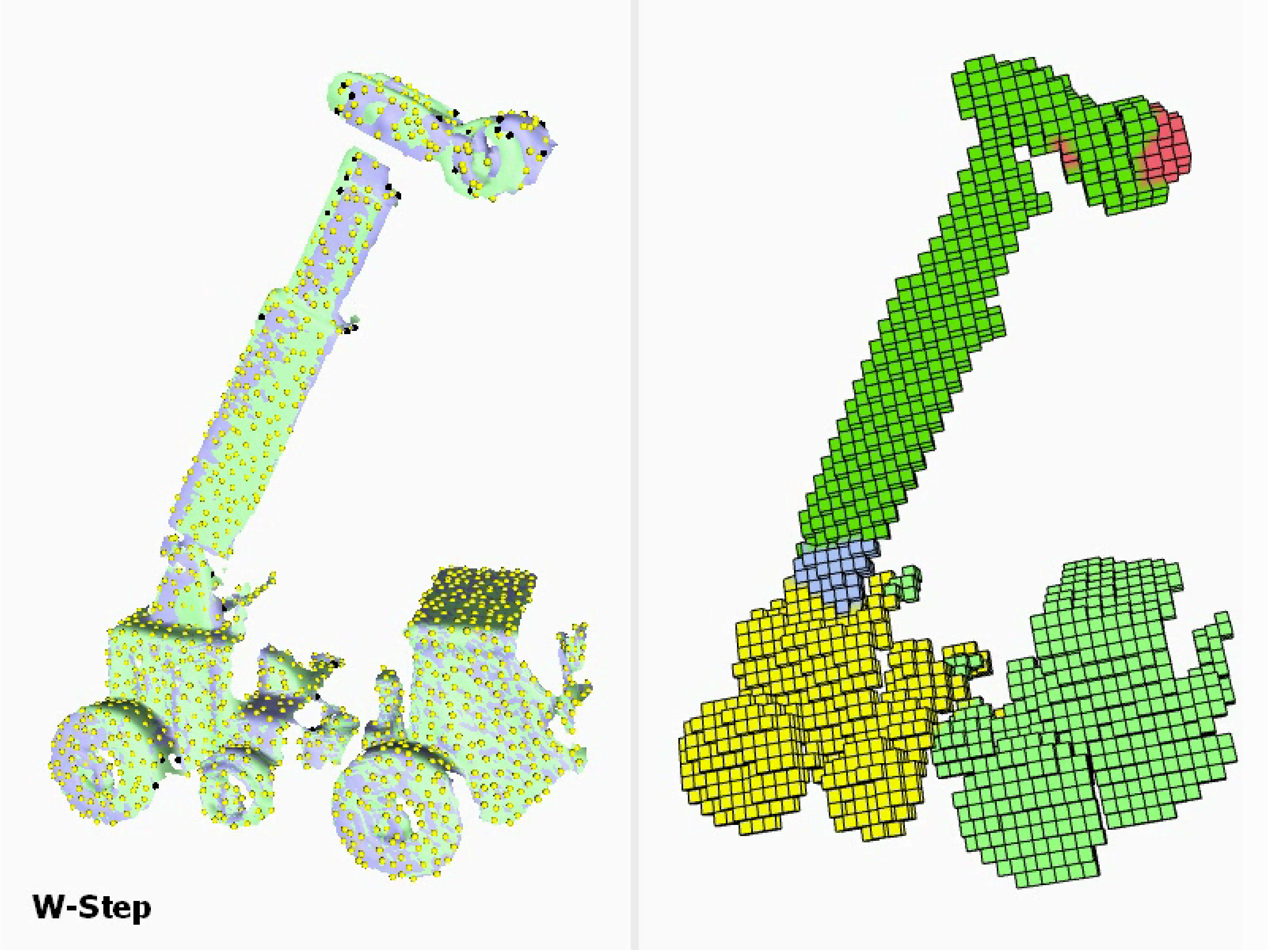
(Converged)


W-Step



(Converged)

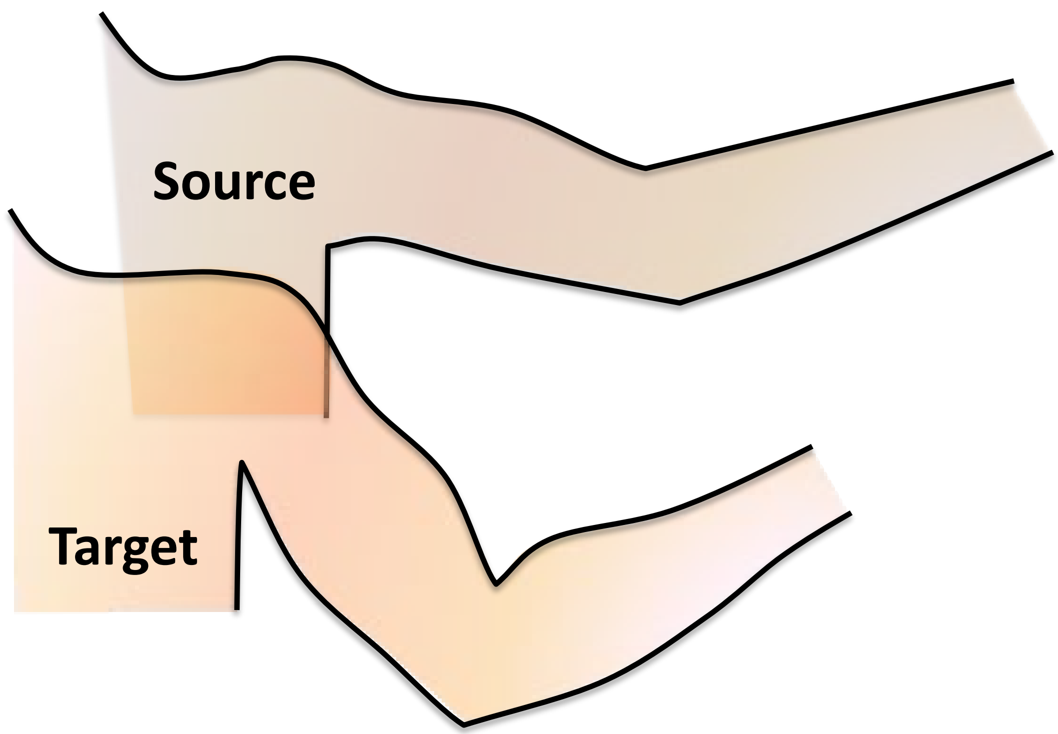

W-Step



(Finished)

T-Step: Distance Term

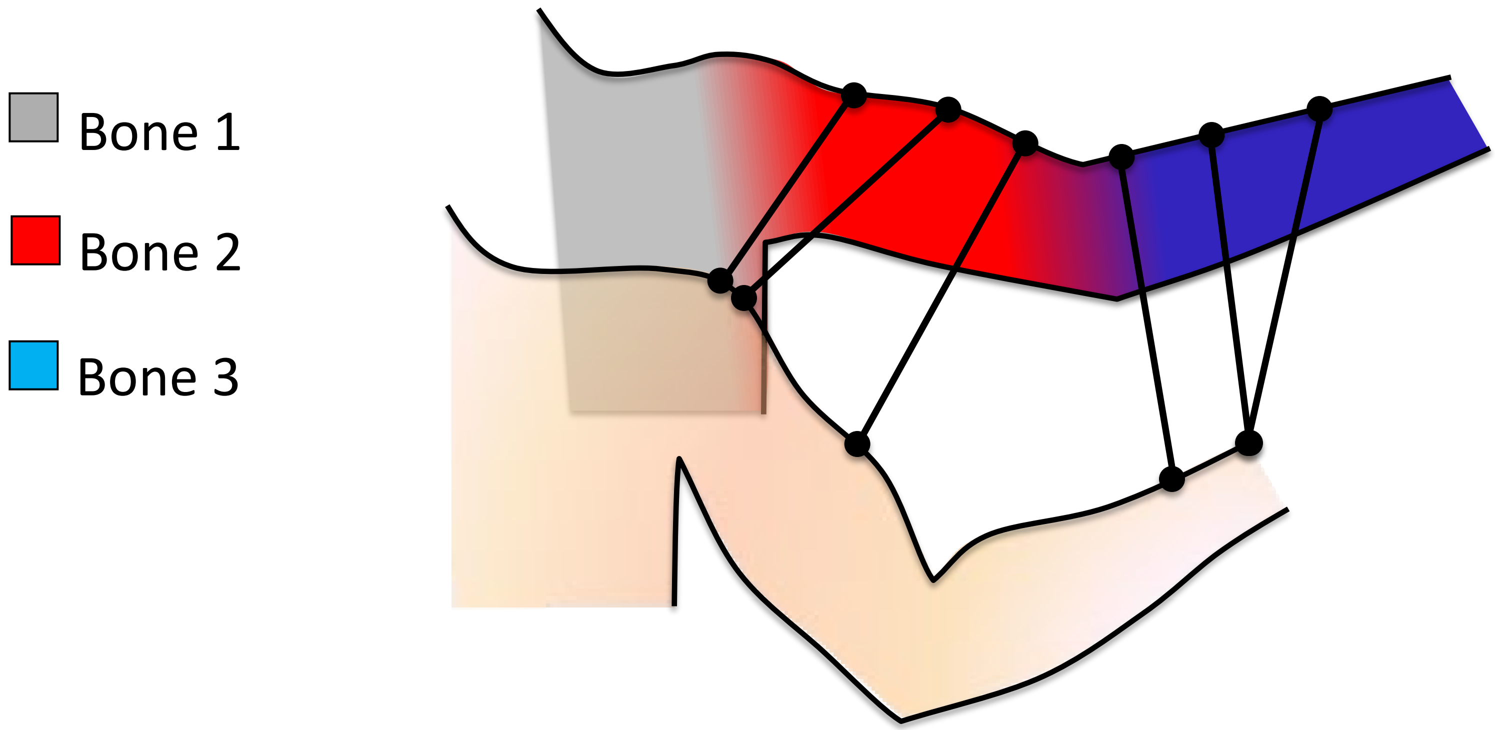
Fix weights & solve for transformations



T-Step: Distance Term

Fix weights & solve for transformations

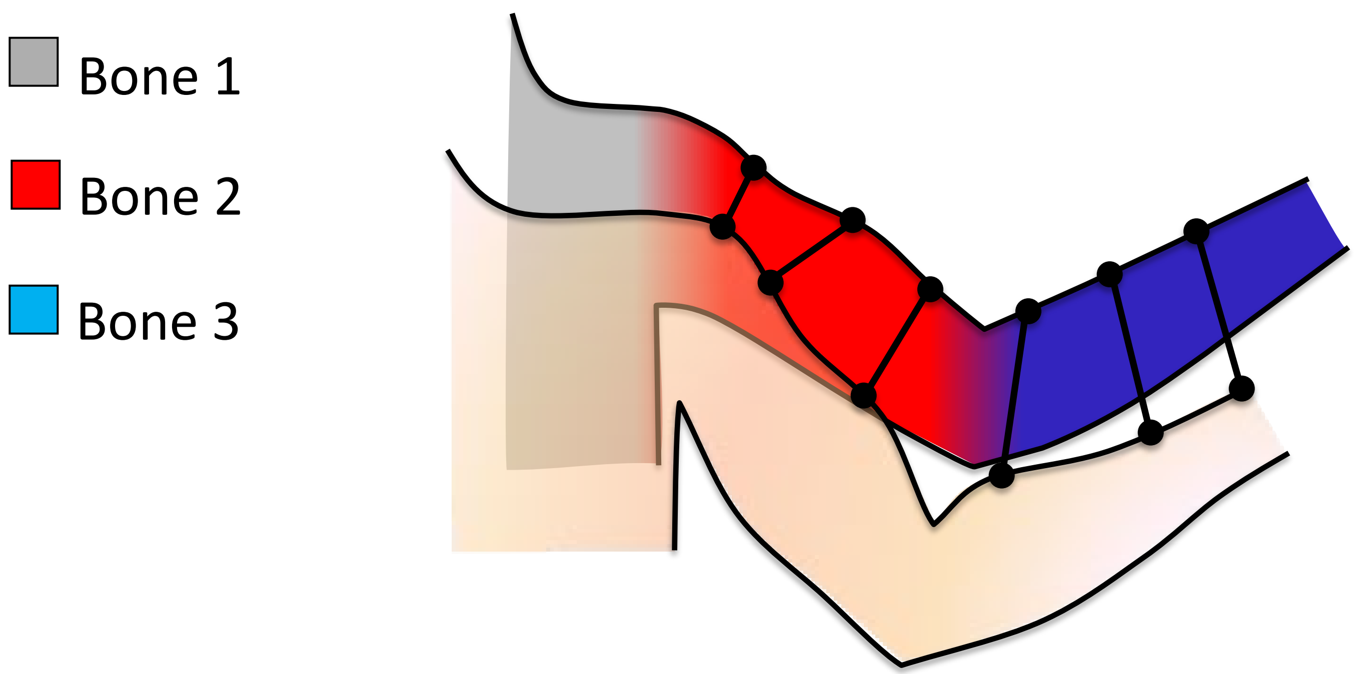
- Use closest point correspondences



T-Step: Distance Term

Fix weights & solve for transformations

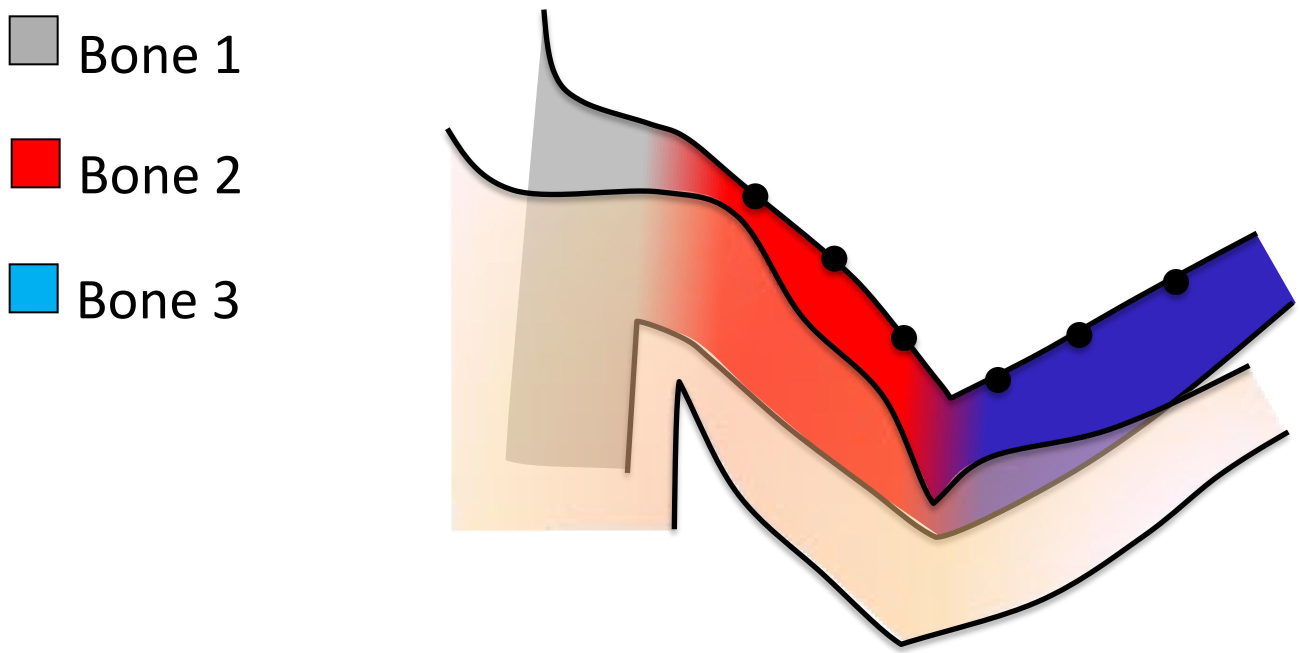
- Use closest point correspondences



T-Step: Distance Term

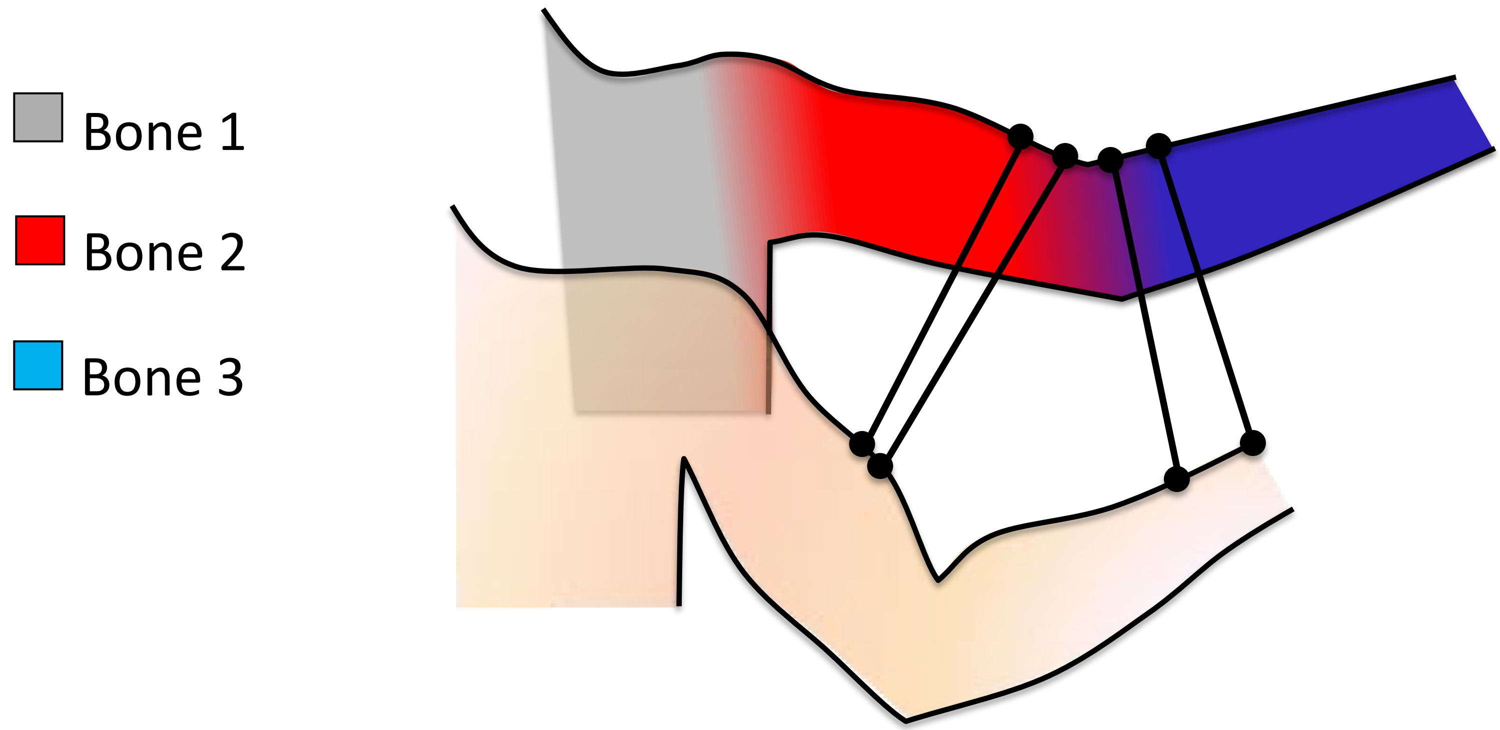
Fix weights & solve for transformations

- Use closest point correspondences
- Iterate further until convergence



T-Step: Joint Constraint Term

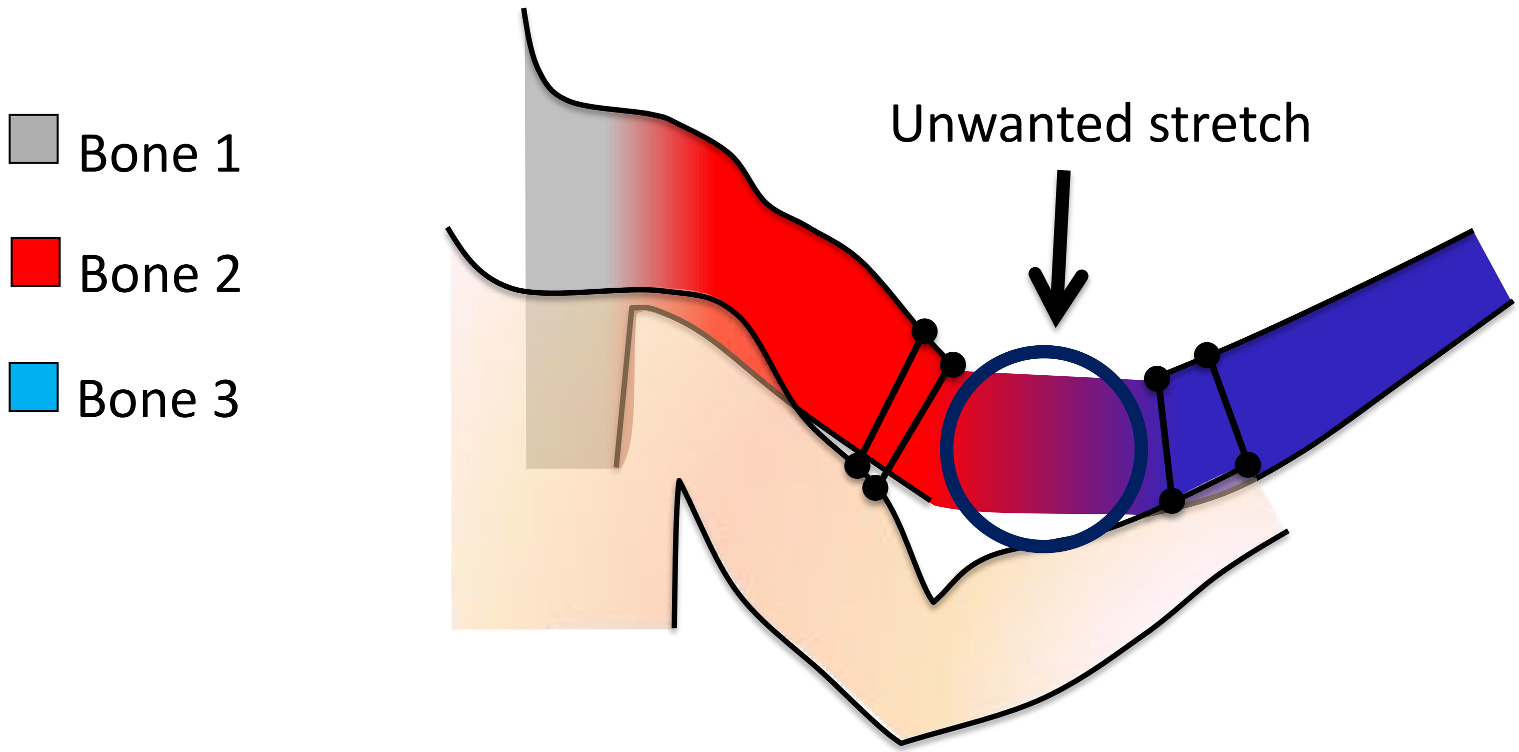
Prevent neighboring bones from separating



T-Step: Joint Constraint Term

Prevent neighboring bones from separating

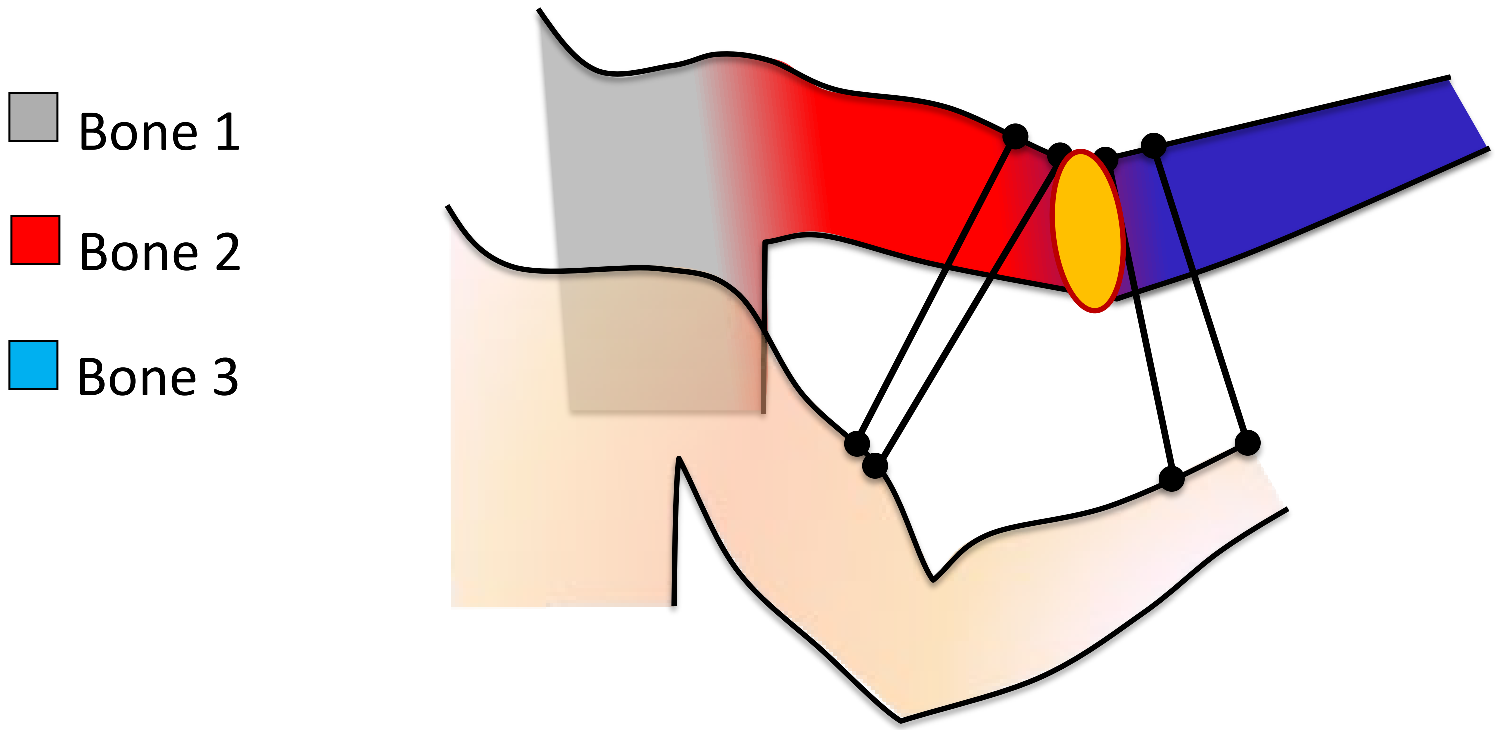
- Constrain overlapping weight regions



T-Step: Joint Constraint Term

Prevent neighboring bones from separating

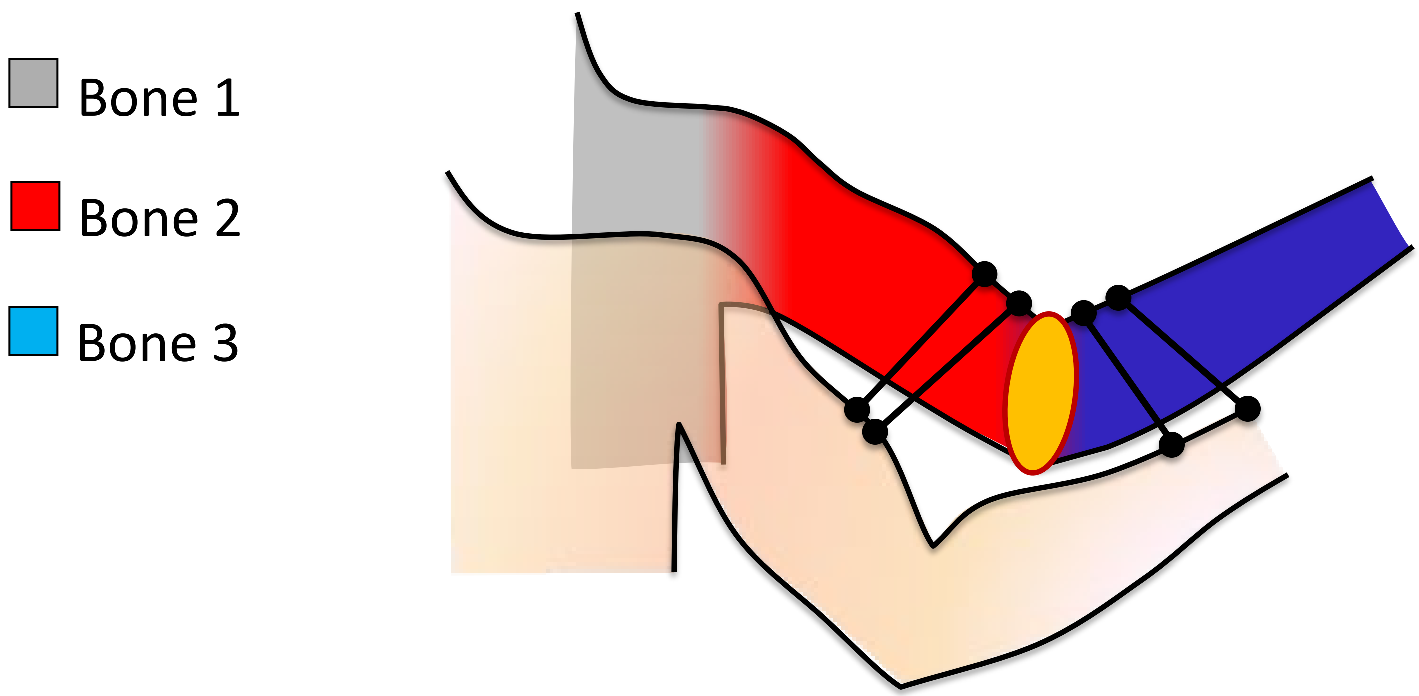
- Constrain overlapping weight regions



T-Step: Joint Constraint Term

Prevent neighboring bones from separating

- Constrain overlapping weight regions

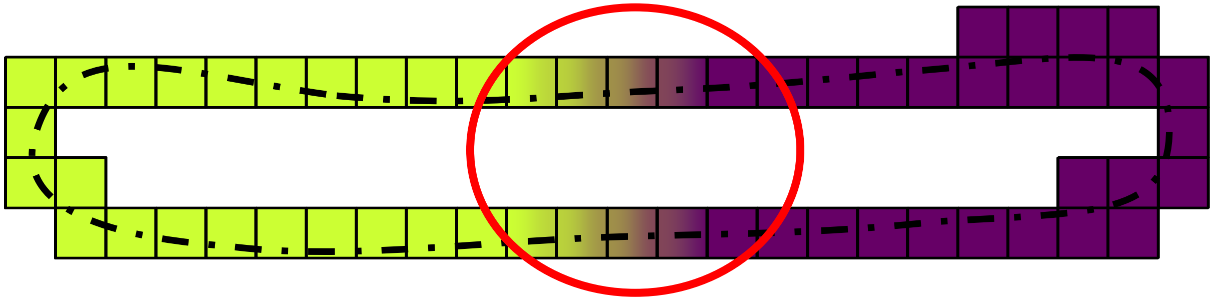


Identifying Overlapping Regions

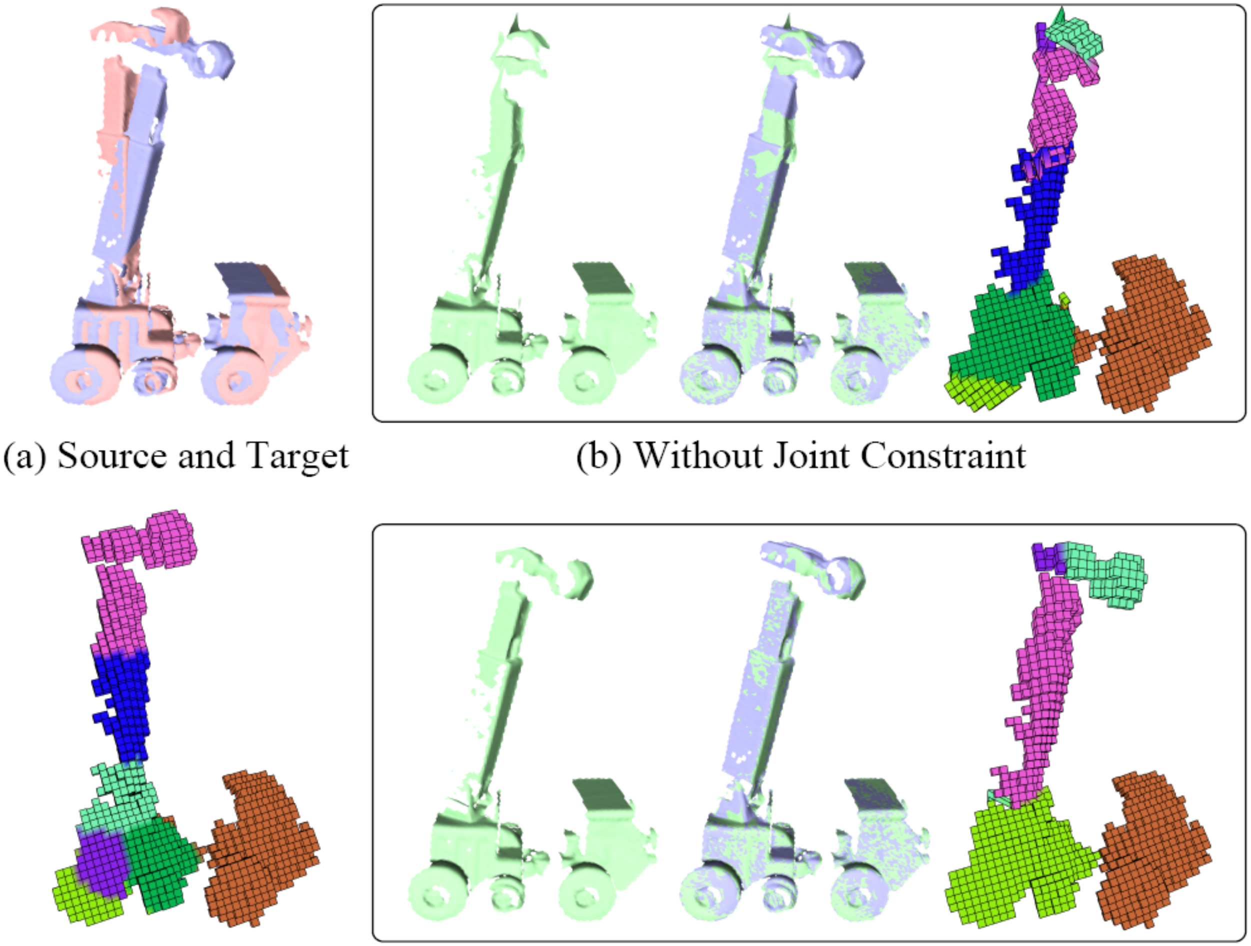
Use the weight grid to find overlap

- Identify overlap over the grid (including cell interior)
- Multiply weight components to determine overlap
- Constrain so that transformations map point to same location

$$E_{\text{joint}} = \sum_{i,j} \tau_{ij} \int_{\mathbf{x} \in \mathbb{R}^3} w_i(\mathbf{x}) w_j(\mathbf{x}) \|T_i(\mathbf{x}) - T_j(\mathbf{x})\|^2 d\mathbf{x}$$



Joint Constraint Comparison



(a) Source and Target (b) Without Joint Constraint

(c) Initial Labeling (d) With Joint Constraint

T-Step: Optimization summary

Like rigid registration

- Except multiple parts & joint constraints

Non-linear least squares optimization

- Solving for a rotation matrix
- Gauss-Newton algorithm
- Solve by iteratively linearizing solution

Few variables → Fast performance

- # variables = 6 x #bones
- Typically 5~10 bones in our examples

W-Step: Assign transformations

Assign transformations to grid

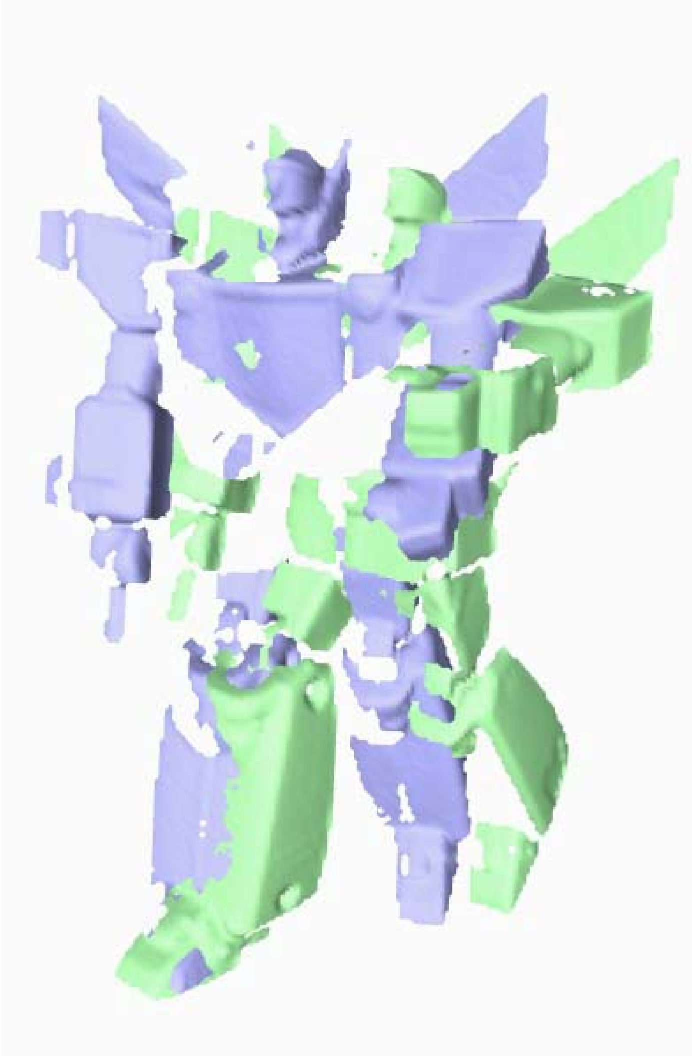
Similar distance and smoothness terms

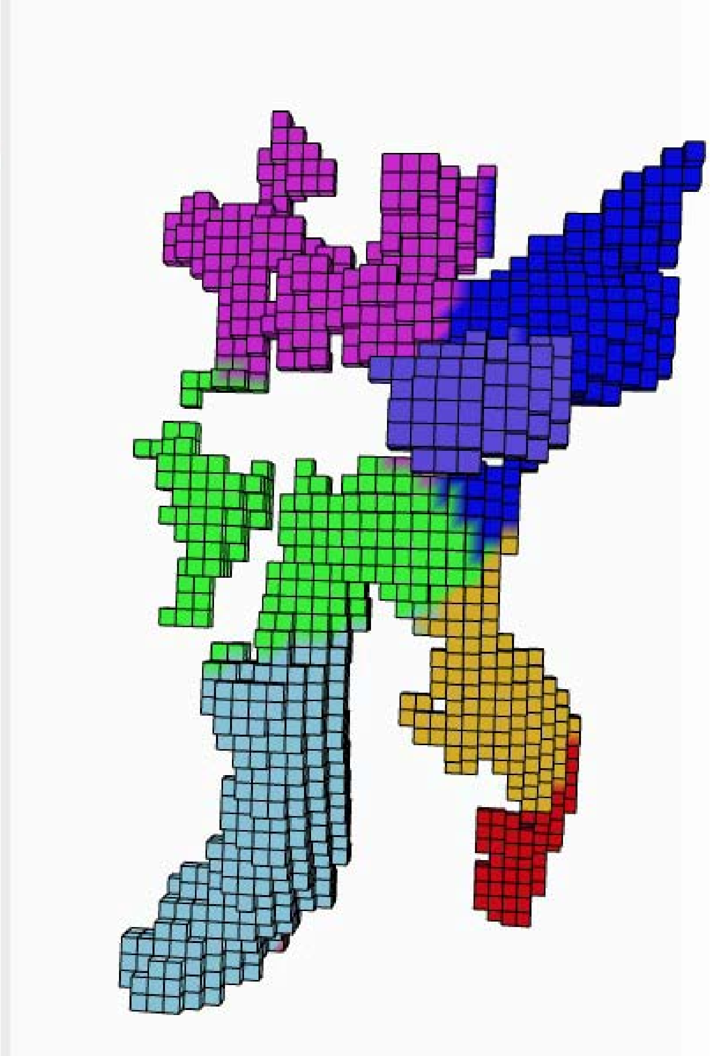
- Distance: measures alignment for a given label (same as before)
- Smoothness: penalizes different labels for adjacent cells (simpler than before)

Graph cuts for optimization → Good Performance

- Only 1000~5000 grid cells (graph nodes) & 5~10 labels
- Fast performance for graph cuts

Robot video (real-time recording)



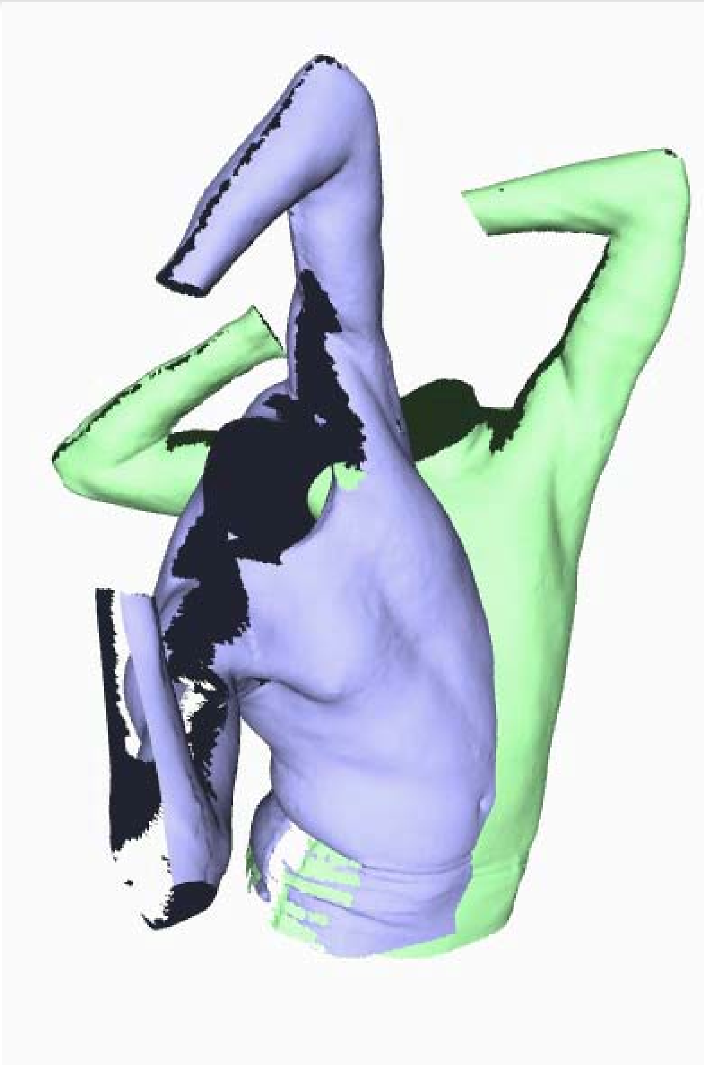


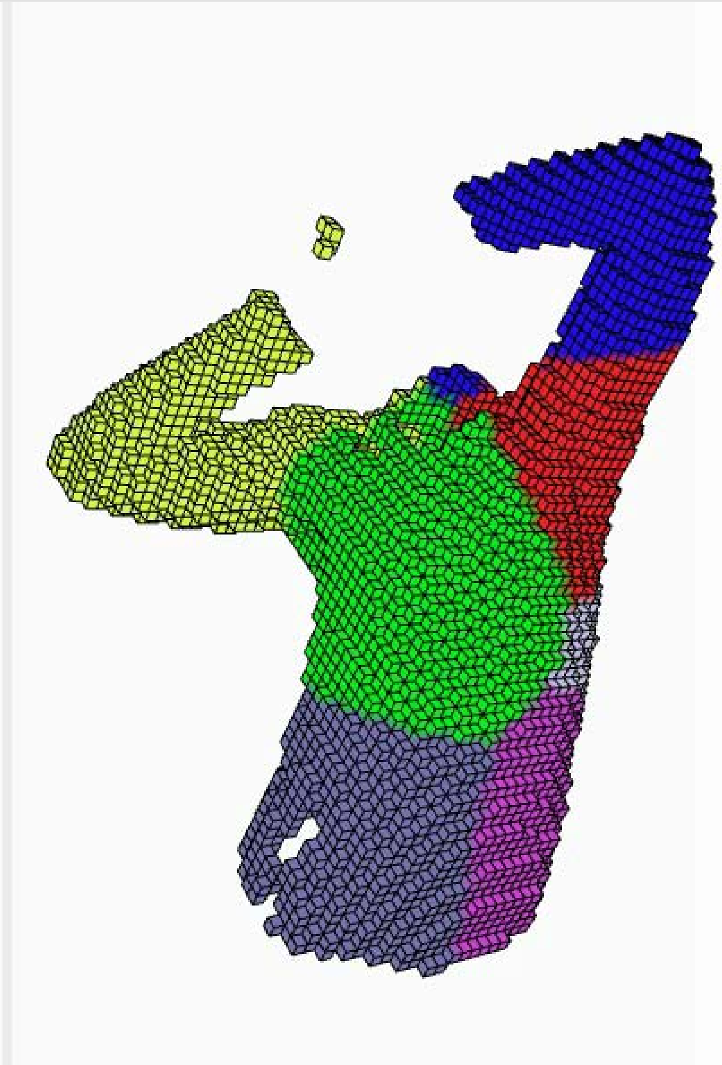
7 bones
1454 cells

Alignment Result

Solved Weights

Torso video (2x speed recording)



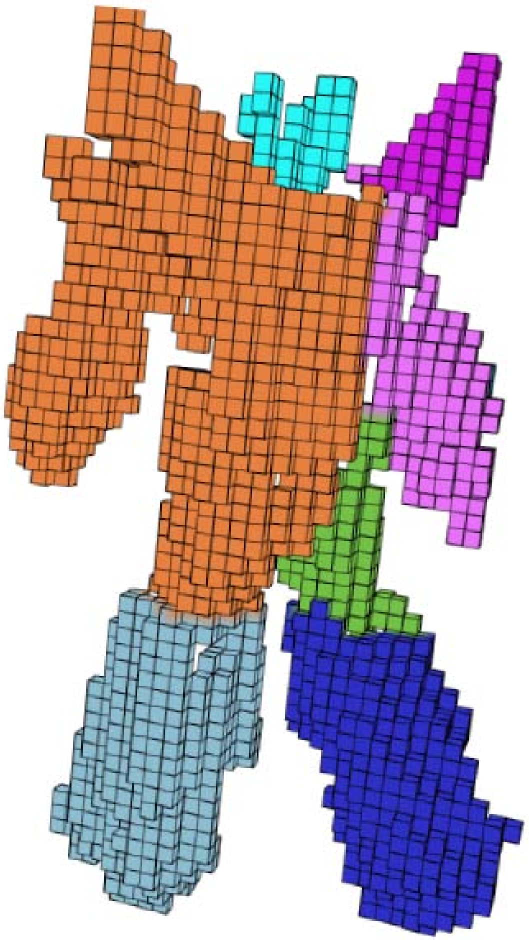


7 bones
4890 cells

Alignment Result

Solved Weights

Interactive posing (real-time recording)



Solved Weights
(7 bones, 1598 cells)

Interactive Posing Result

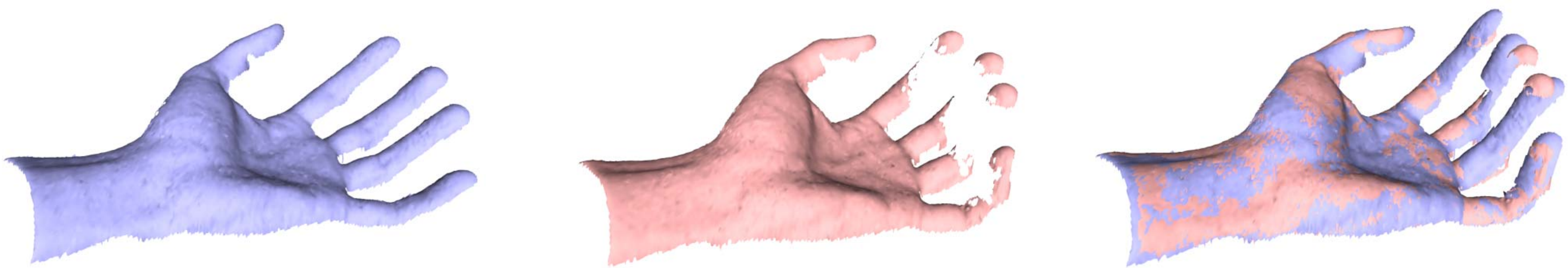
Average performance statistics

	Car	Robot	Walk	Hand
Bones	7	7	10	12
Corresp.	1200	1200	1000	1500
Vertices	5389	9377	4502	34342
Max Dist	20	40	20	30
Grid Res	60	65	50	40
Grid Cells	1107	1295	1014	814
Grid Points	2918	3366	2553	1884
Setup	0.185 sec	0.234 sec	0.136s ec	0.078 sec
RANSAC	8.089 sec	20.001 sec	5.517 sec	N/A
Align	9.945 sec	19.644 sec	23.092 sec	49.918 sec
Weight	6.135 sec	10.713 sec	10.497 sec	3.689 sec
Total Time	24.355 sec	50.591 sec	39.242 sec	53.684 sec

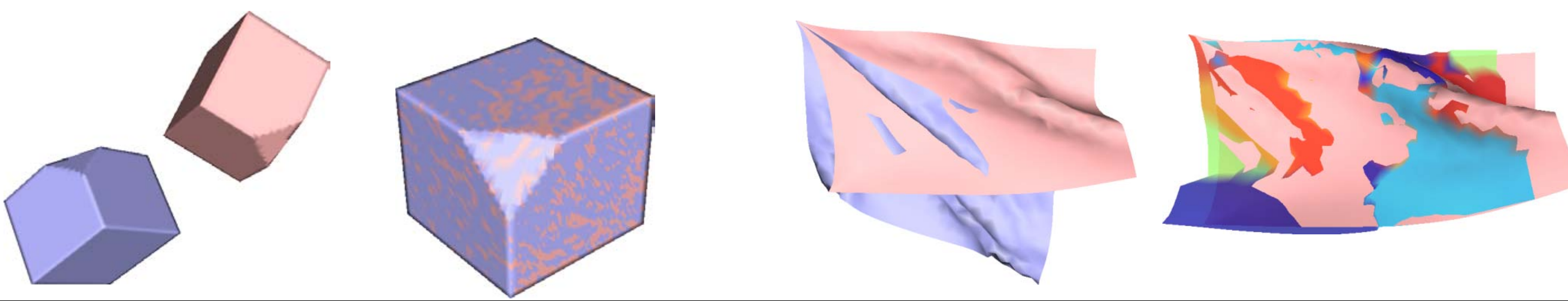
Pros/Cons

Feature matching: Insensitive to initial pose

- *May fail to sample properly when too much missing data, non-rigid motion*
- *Hard assignment of transformations*



Source Target Registration



Pros/Cons

Local Search: faster and more precise

- Faster method: no “useless” transformations corresponding to wrong feature matches
- Can refine transformations until precise alignment is achieved
- *Sensitive to initial pose*
- *Topological issues with grid*

We can combine the two methods

- Initialize with first, refine with second
- Also graph can be more robust than grid

Conclusions

We can simplify the problem for articulated shapes

- Instead of searching for corresponding points, search for an assignment of transformations
- Explicitly sample a discrete set of transformations
- Refine the transformations via local search
- Optimize the assignment using graph cuts
- No marker, template, segmentation information needed
- Robust to occlusion & missing data

Thank you for listening!