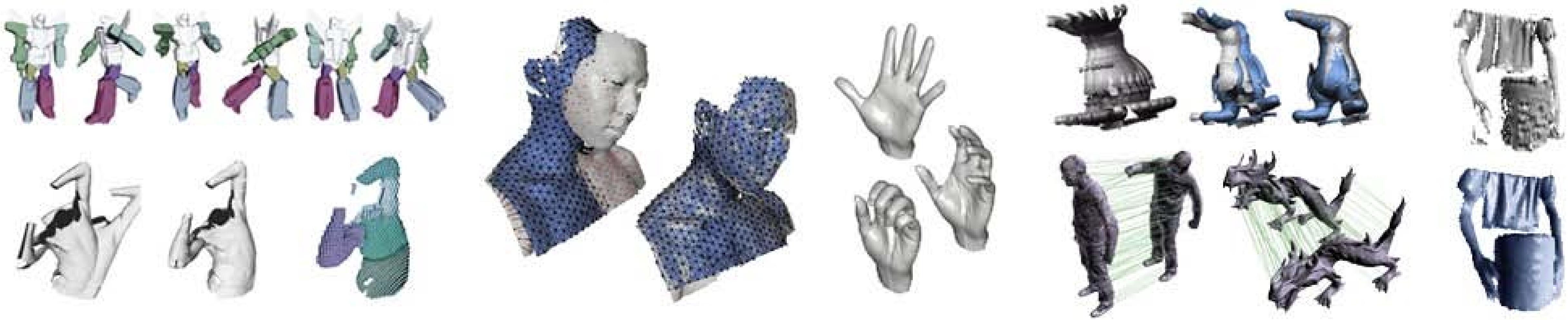
Eurographics 2012
Cagliari, Italy May 13-18
33rd ANNUAL CONFERENCE OF THE EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS



Dynamic Geometry Processing

EG 2012 Tutorial

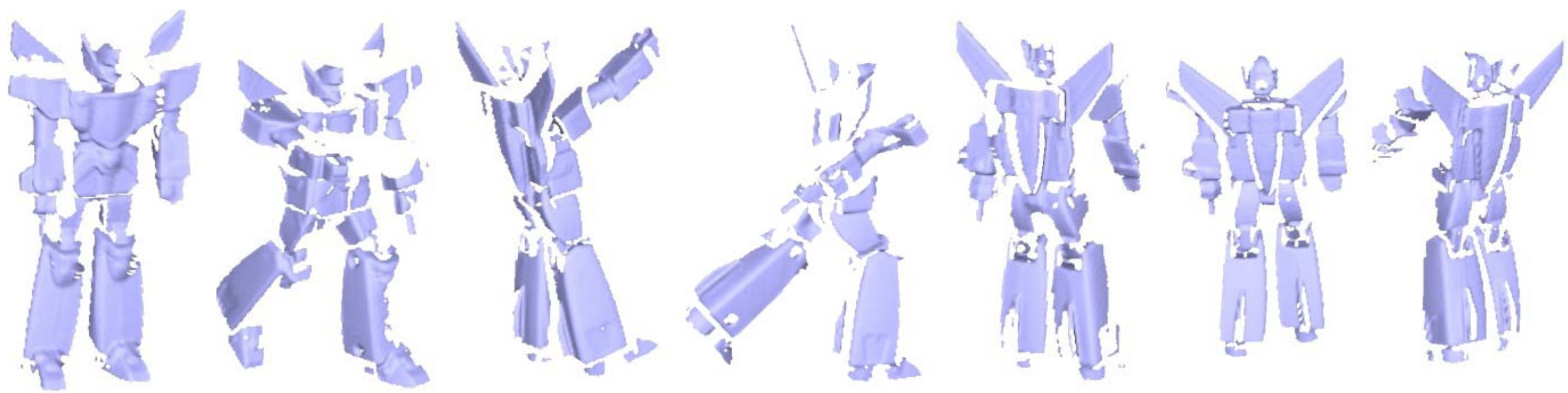
Will Chang, Hao Li, Niloy Mitra,
Mark Pauly, Michael Wand

Articulated Global Registration

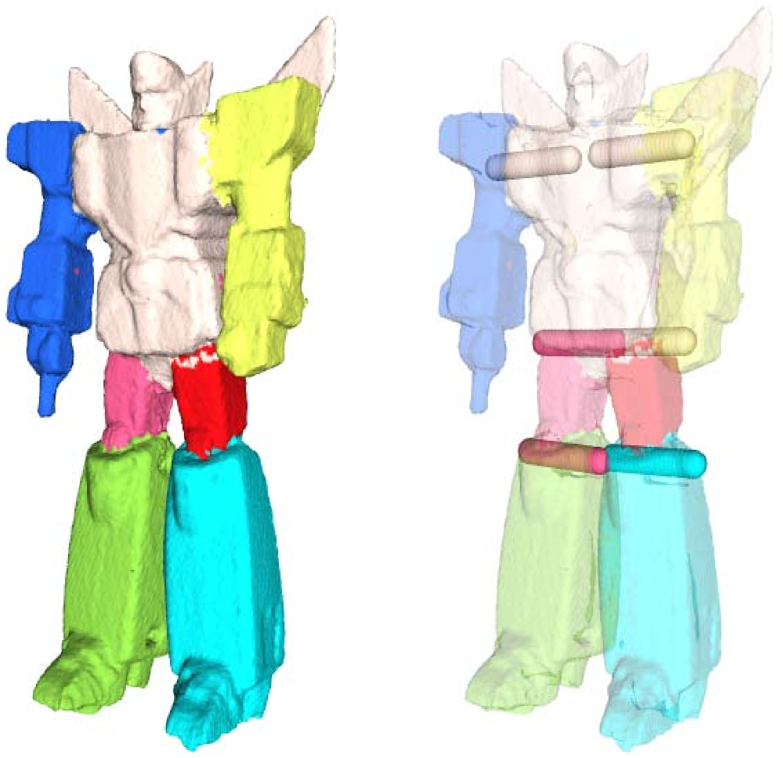
Introduction and Overview

Articulated Global Registration

- Complete models from dynamic range scans
- No template, markers, skeleton, segmentation
- Articulated models
  - Movement described by piecewise-rigid components



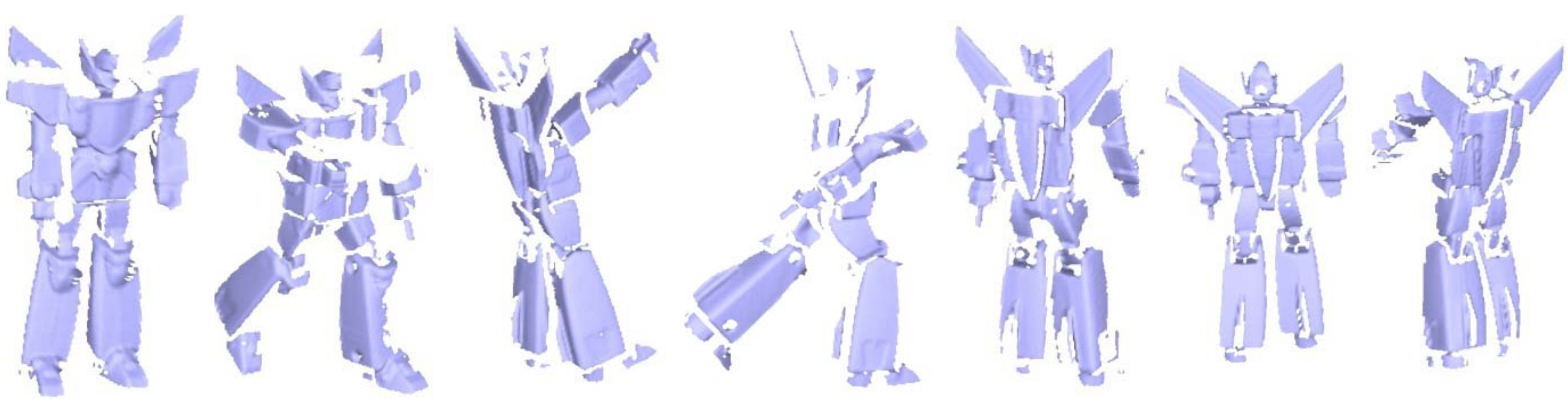
Input Range Scans



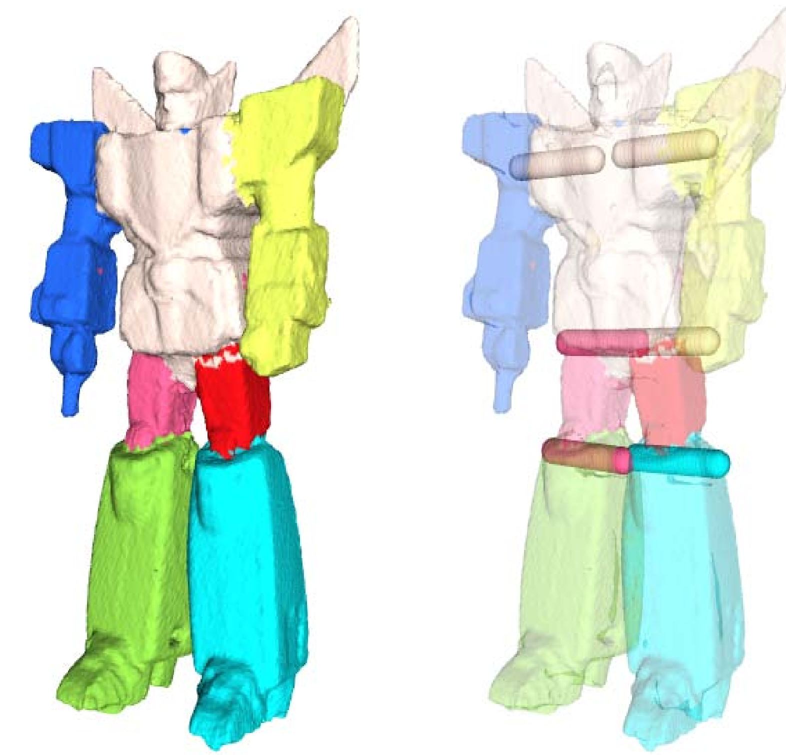
Reconstructed 3D Model

Features

- Handles large, fast motion
- Incomplete scans (holes, missing data)
- 1 or 2 simultaneous viewpoints
- Optimization is over all scans



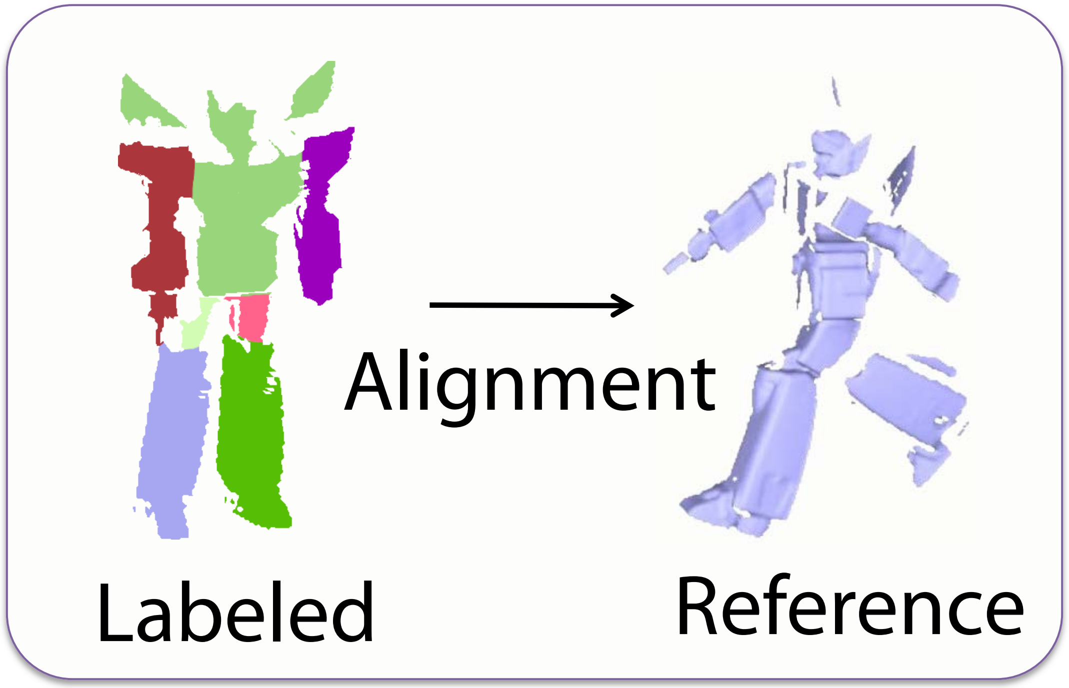
Input Range Scans



Reconstructed 3D Model

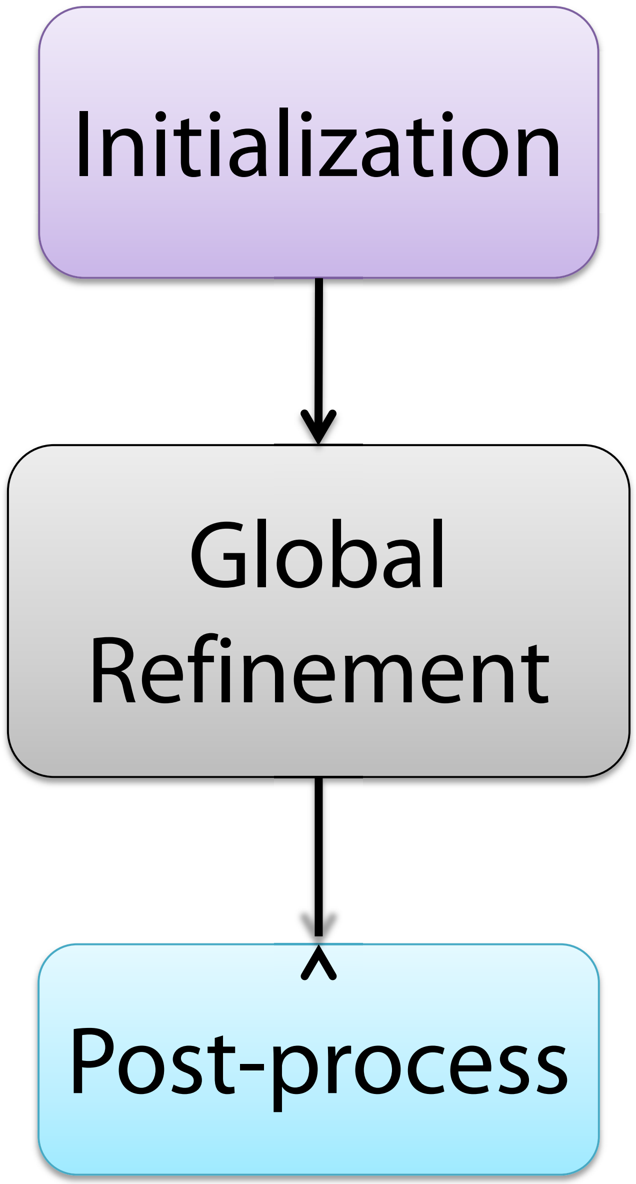
Reconstructing Articulated Models

- **For every frame, determine**
  - **Labeling** into constituent parts (per-vertex)
  - **Motion** of each part into reference pose (per-label)
- Solve simultaneously for labels, motion, joint constraints



Algorithm Overview

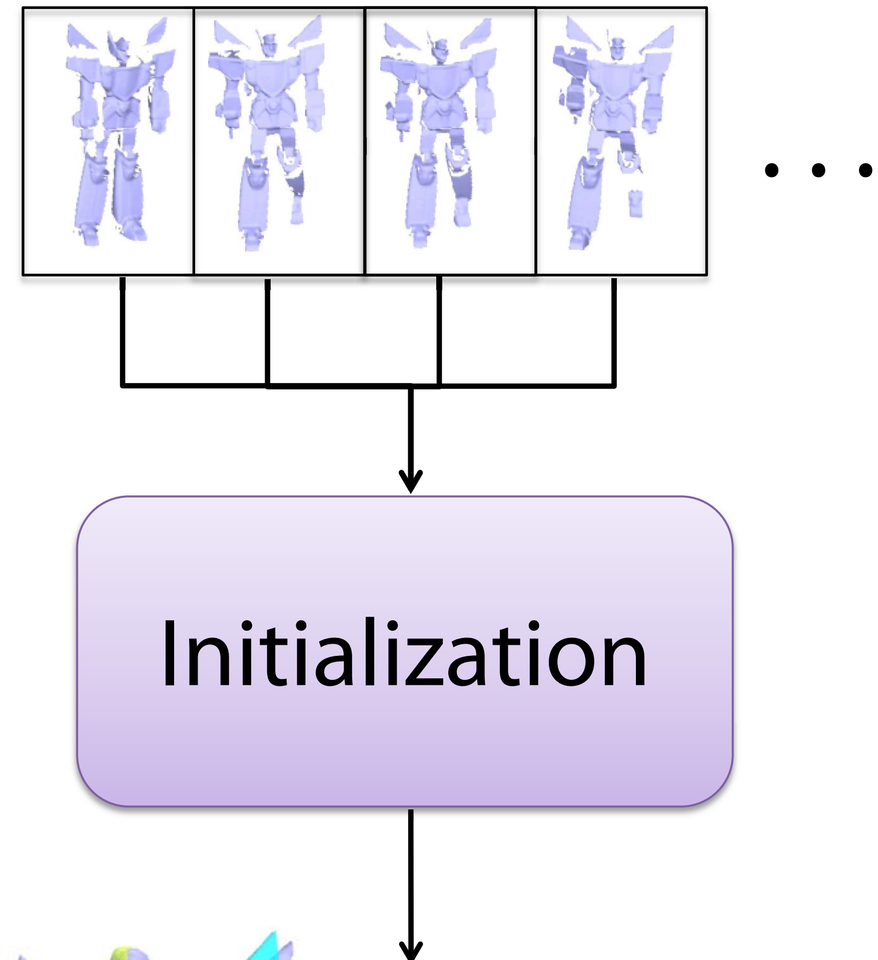
- Initialization
- Global refinement
- Post-process

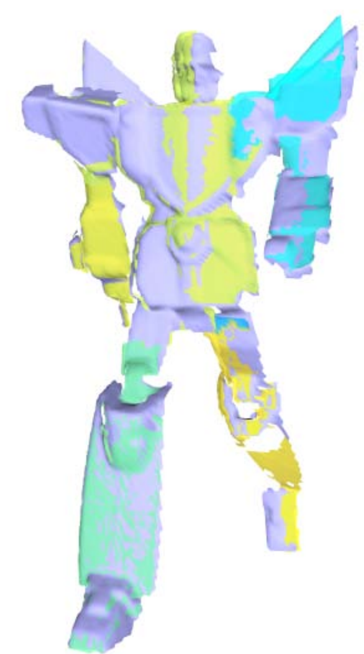


Algorithm Overview

Initialization

– Coarse pairwise registration





Labels (per-vertex)
and
Transformations
(per-label) for a
coarse registration

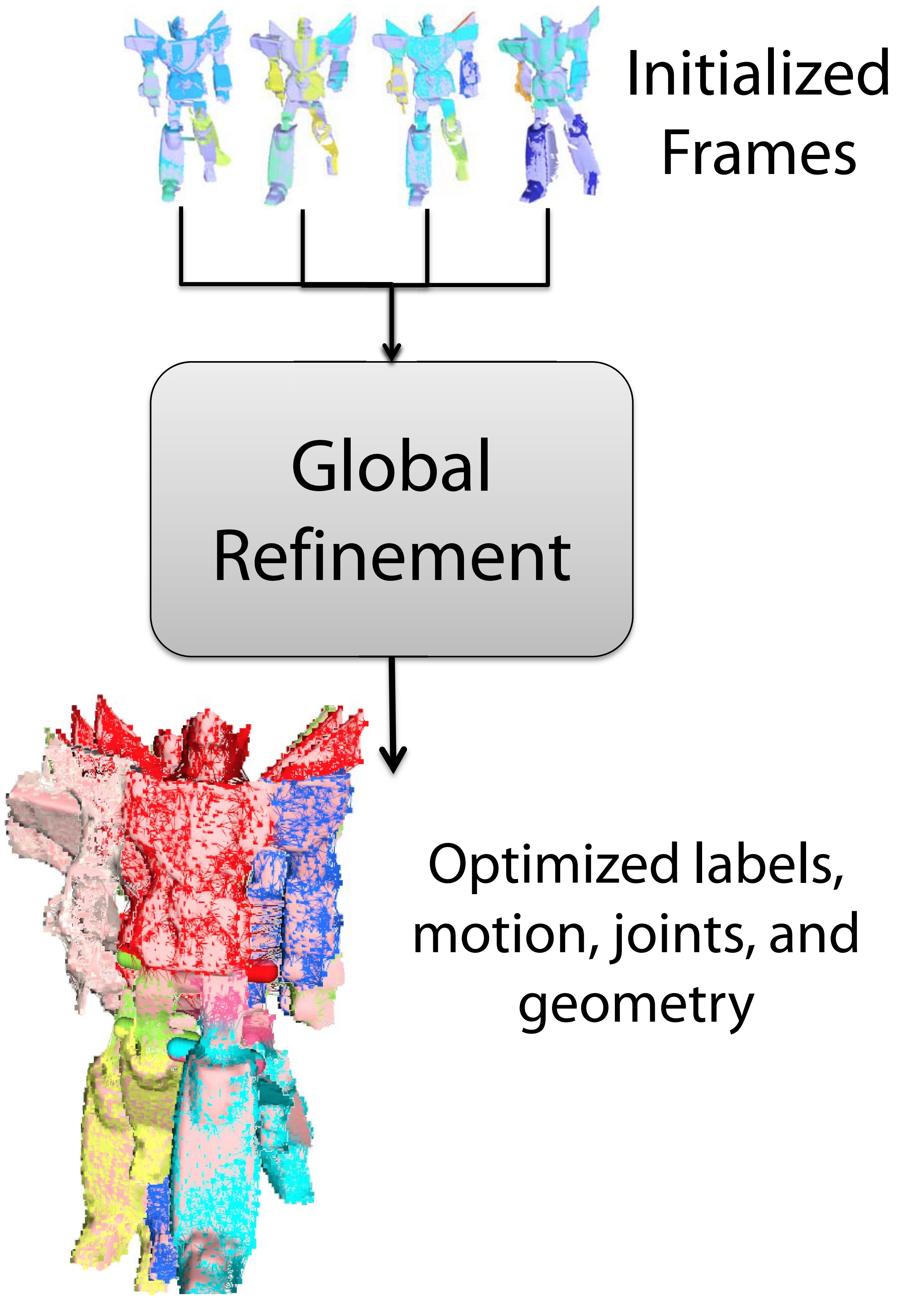
Algorithm Overview

Initialization

– Coarse pairwise registration

Global refinement

– Solve global model incorporating all frames



Algorithm Overview

Initialization

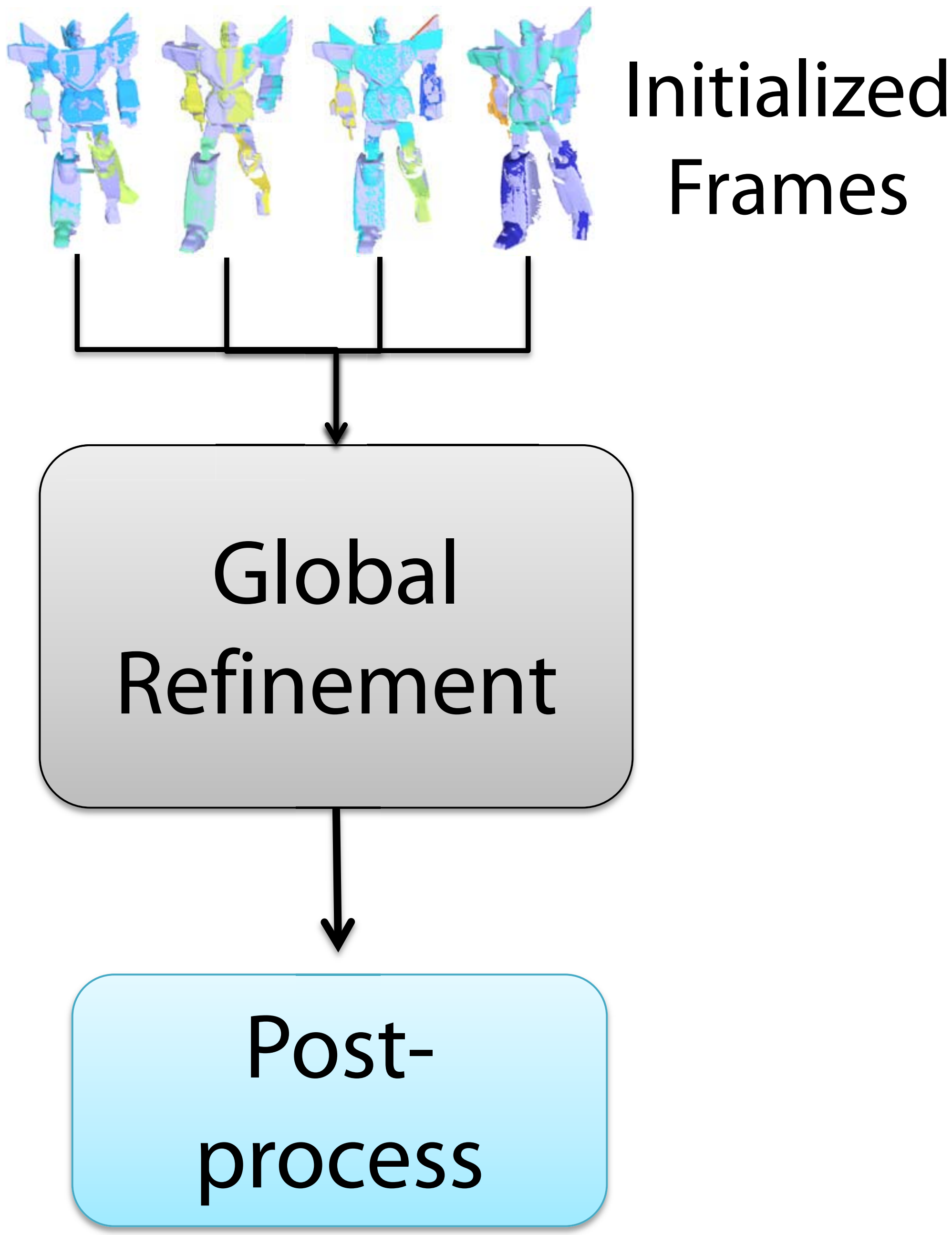
– Coarse pairwise registration

Global refinement

– Solve global model incorporating all frames

Post-process

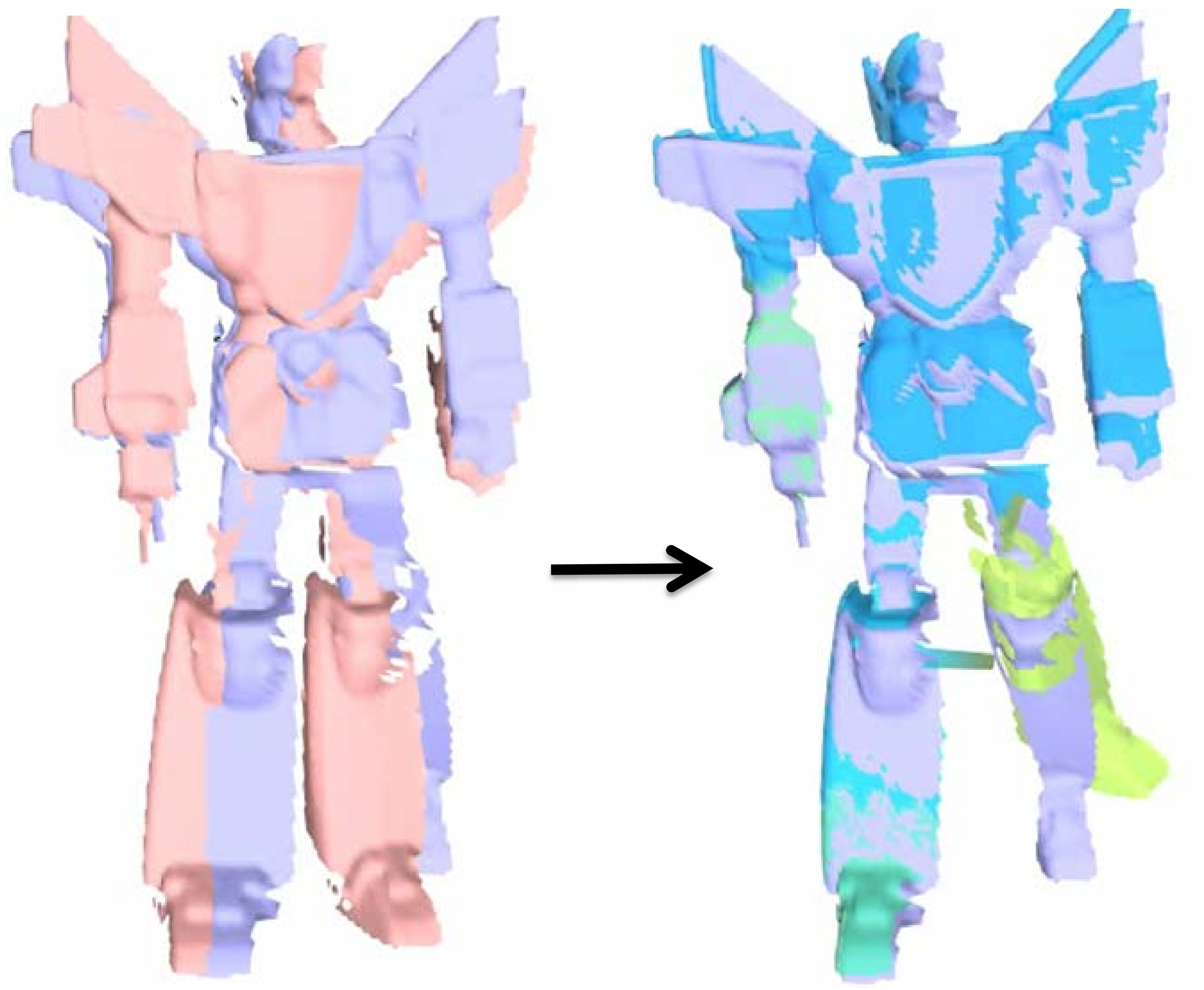
– Gather frames, reconstruct mesh



Part I: Initialization

Initialization

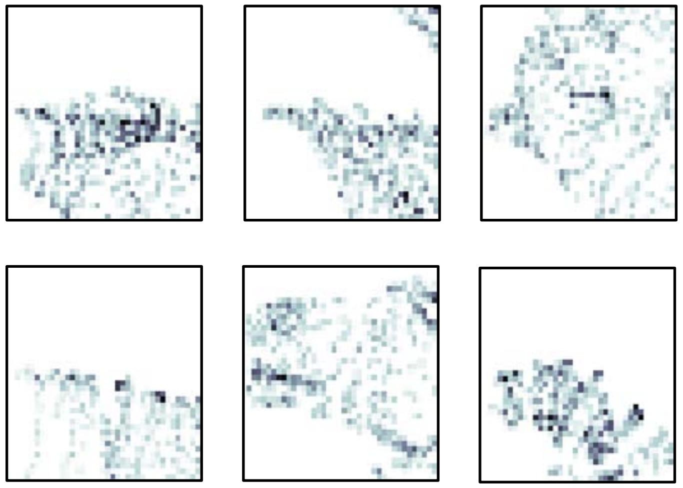
Goal: To establish initial correspondence of consecutive frames



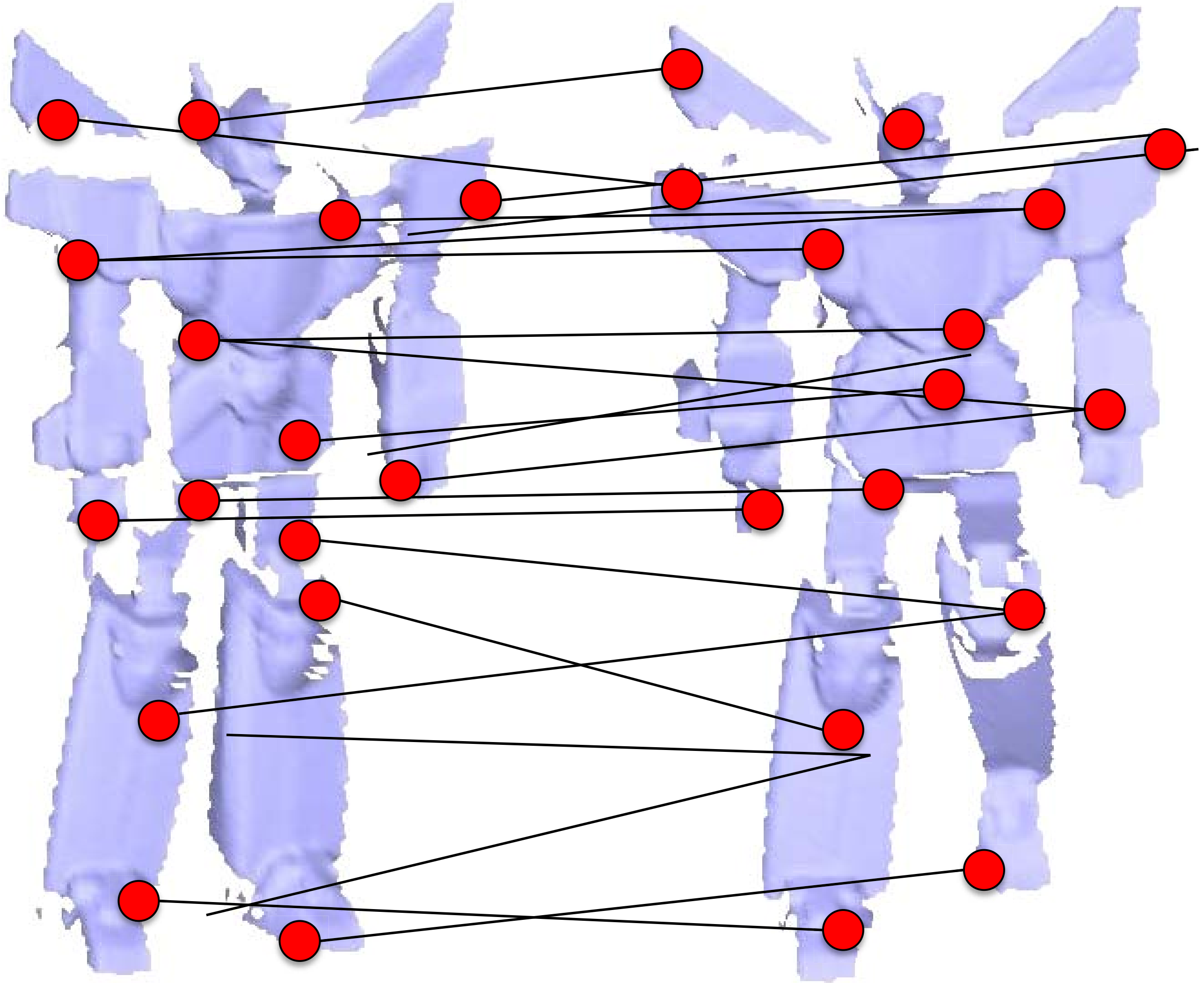
Frame ***i*** and ***i+1*** Registered Result

Initialization

1. Point correspondence using feature descriptors

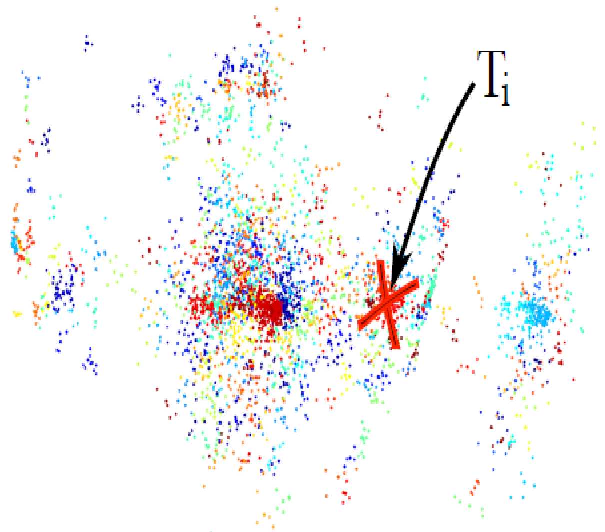


Spin Image examples

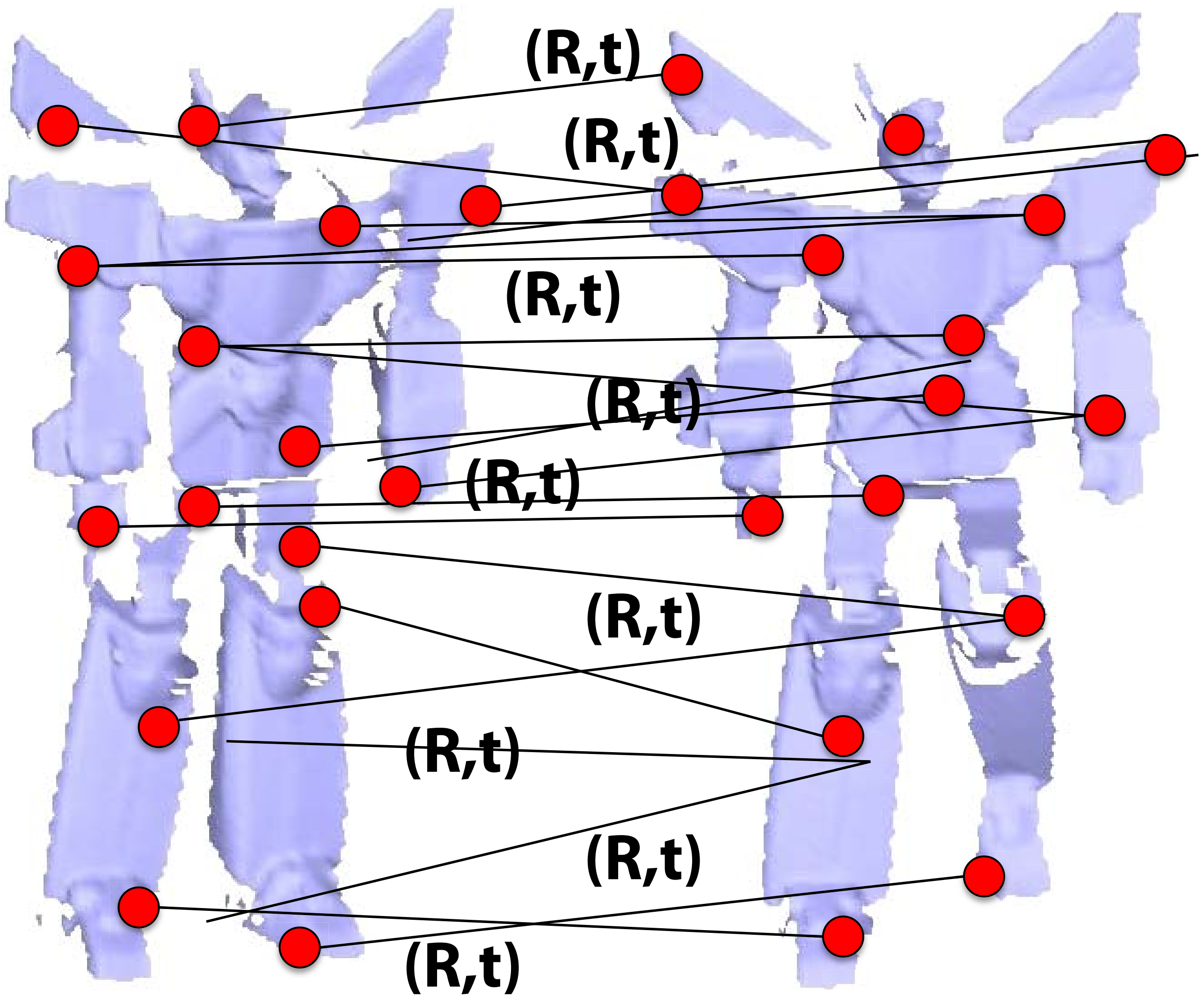


Initialization

1. Point correspondence using feature descriptors
2. Transformation (R,t) per correspondence
3. Cluster (R,t)

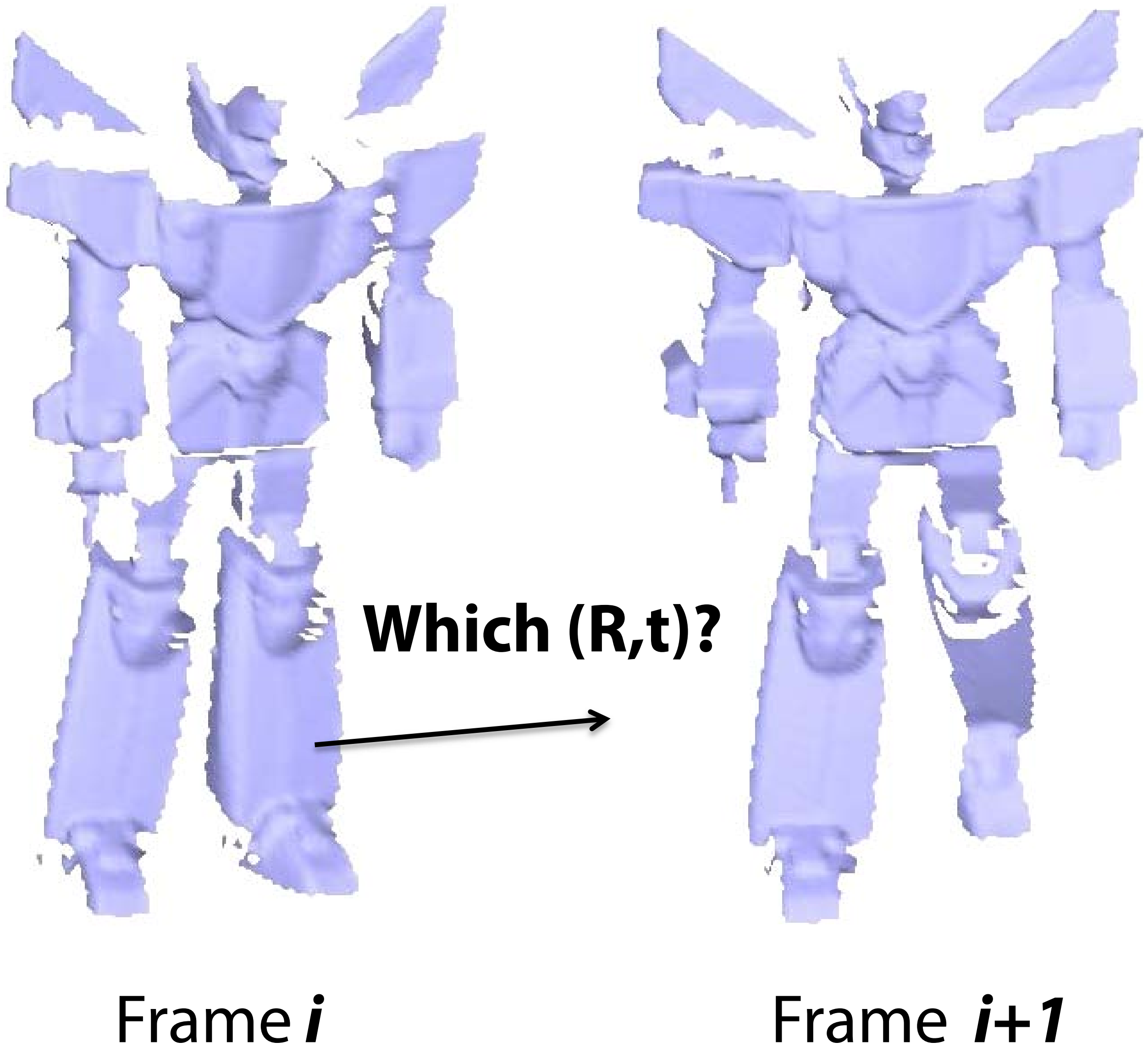


Transformation Space *se(3)*



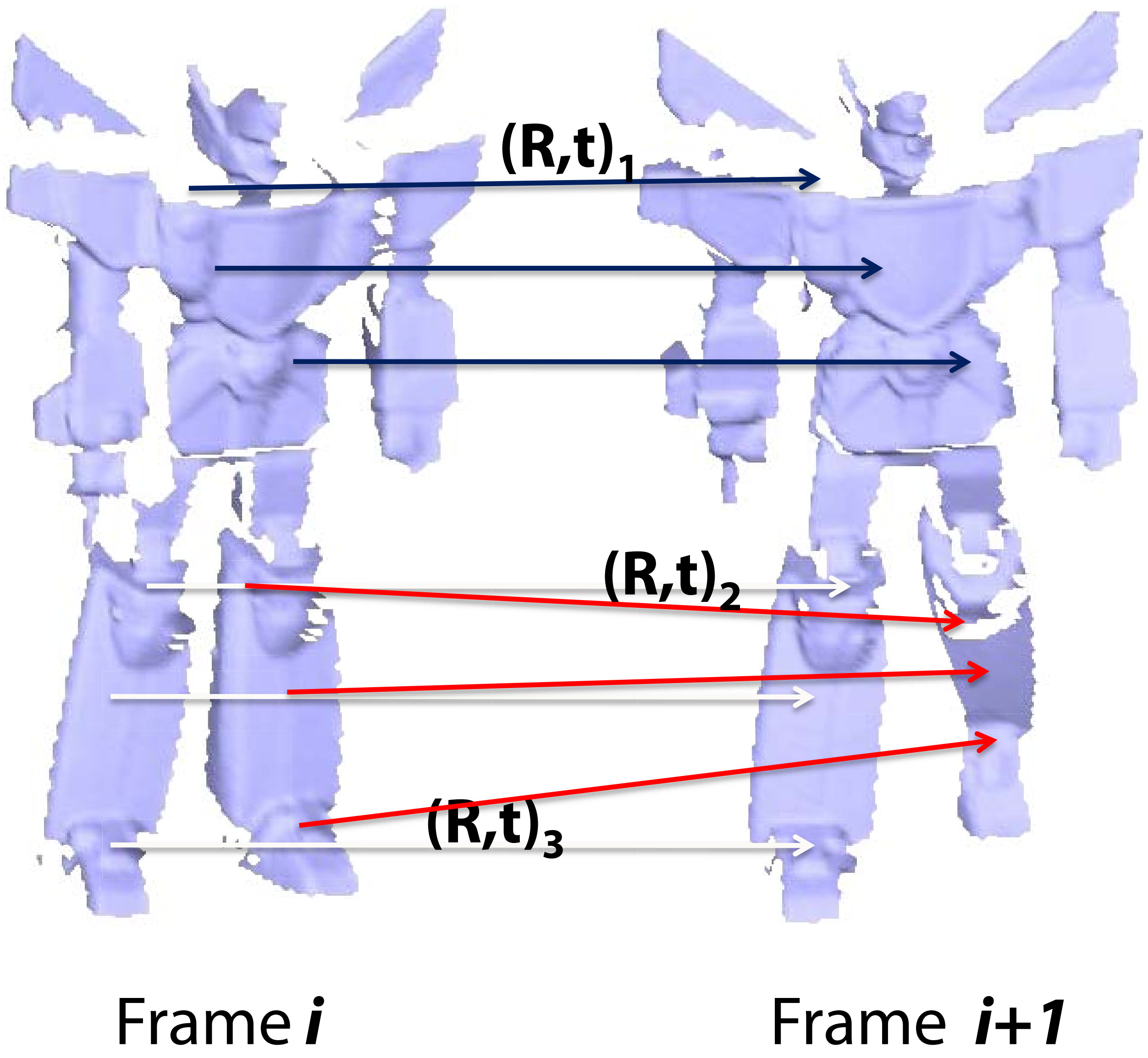
Initialization

1. Point correspondence using feature descriptors
2. Transformation (R,t) per correspondence
3. Cluster (R,t)
4. Optimize using “Graph Cuts” [Boykov et al. 2001]



Initialization

1. Point correspondence using feature descriptors
2. Transformation (R,t) per correspondence
3. Cluster (R,t)
4. Optimize using “Graph Cuts” [Boykov et al. 2001]



Initialization Result

Both Frames

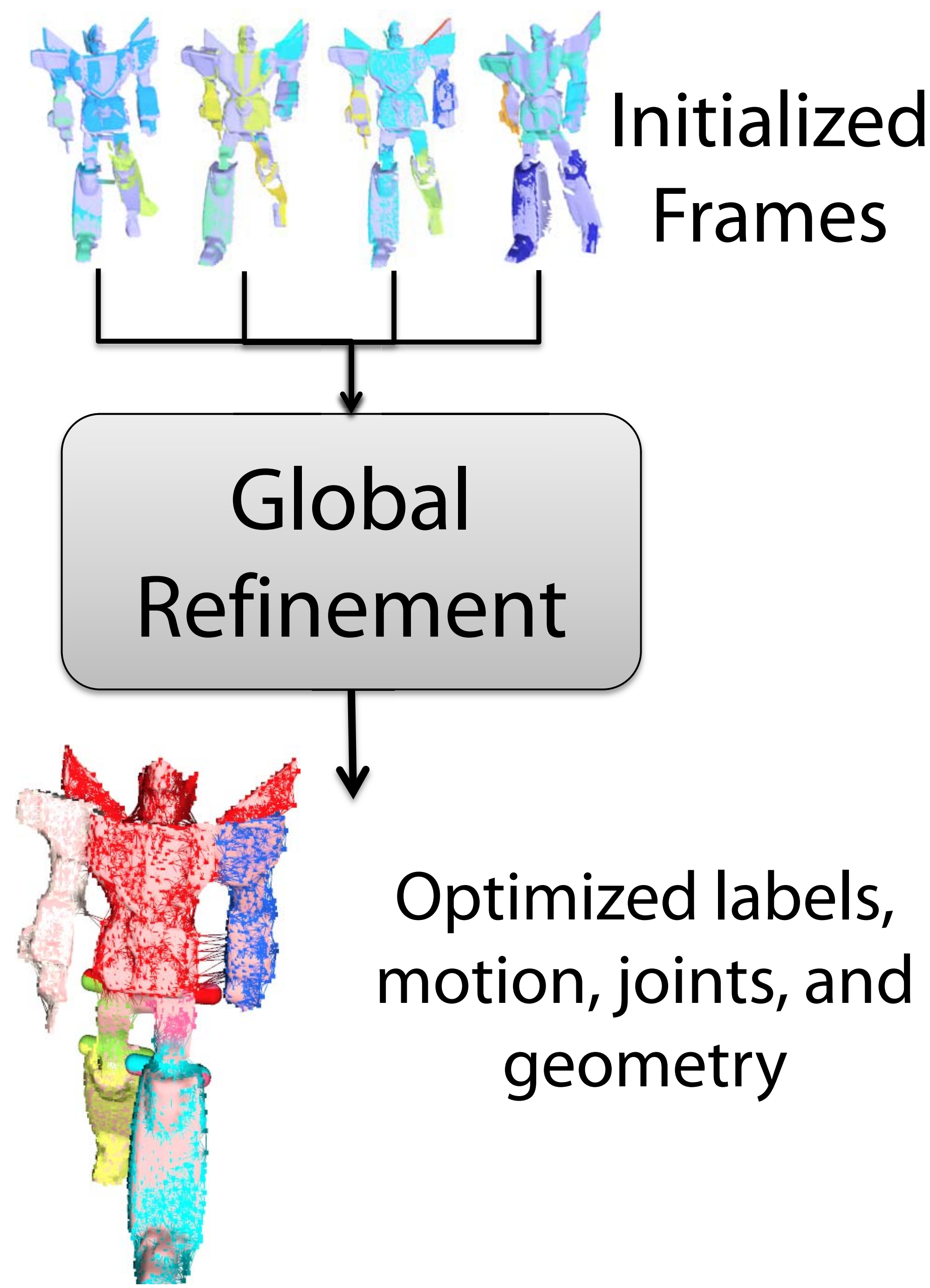
Registered Result

Part II: Global Refinement

Global Refinement

Global refinement

- Solve global model incorporating all frames



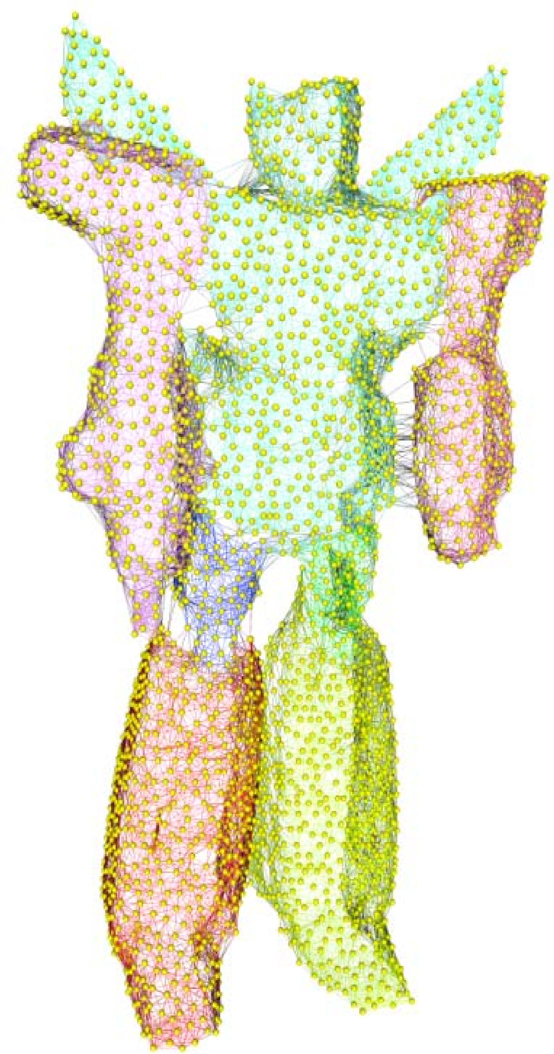
Dynamic Sample Graph (DSG)

Sparse representation

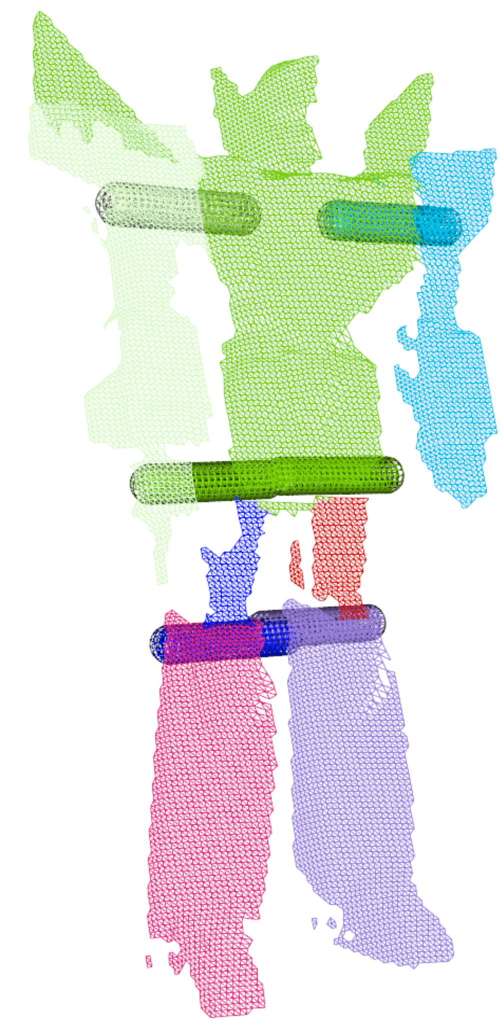
- Increases efficiency
- Joints: part connectivity

Continuously updating

- Update samples from new surface data



Dynamic Sample Graph (DSG)



Extracted Joints

Global Refinement

Fit the DSG to all scans simultaneously (Global Fit)

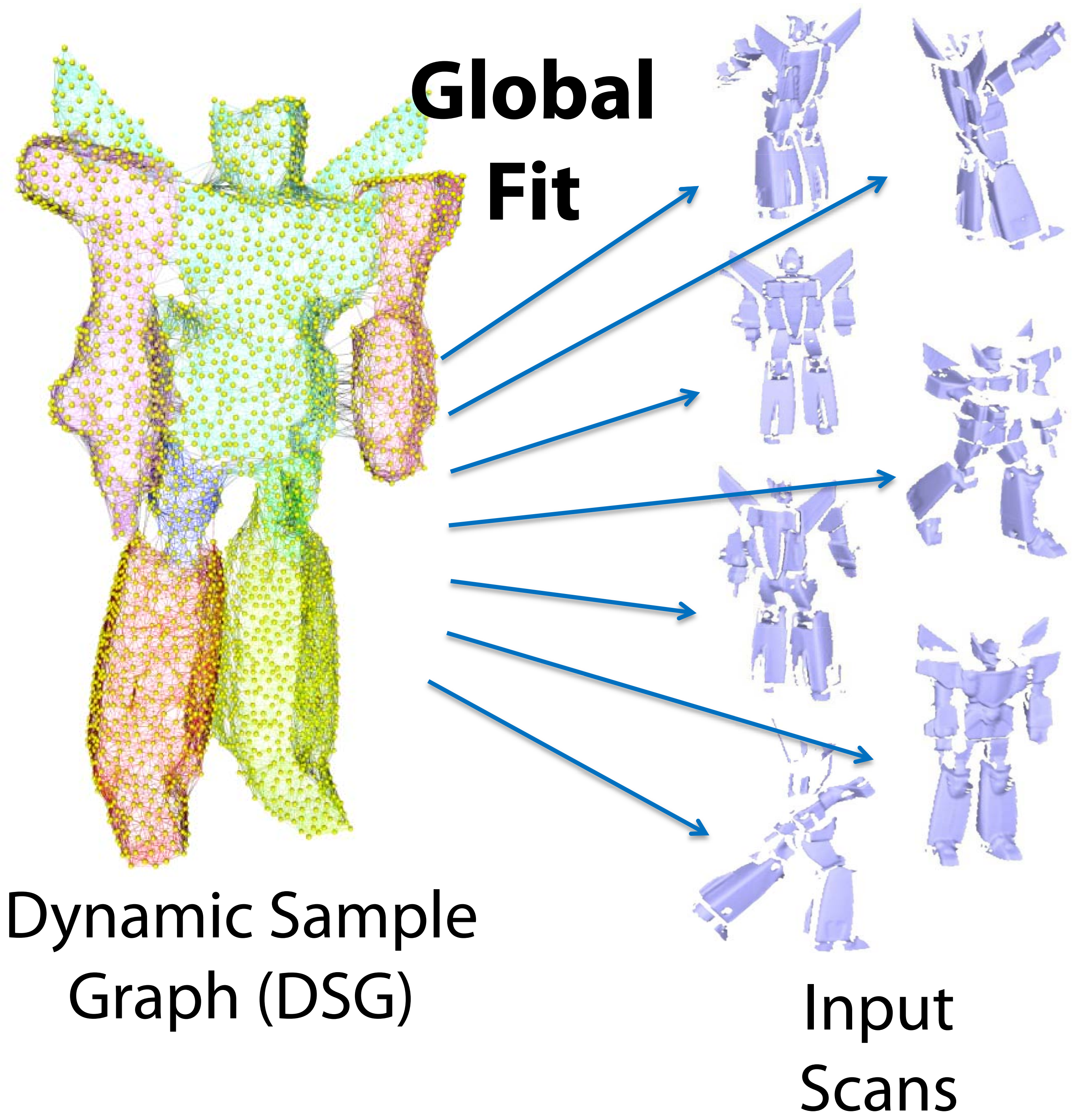
Alternating Optimization

1. Optimize Transformations
2. Optimize Labels
3. Update joint locations

Repeat until convergence

– 3-5 iterations/frame

Update samples



Transformation Optimization

- Align DSG as closely as possible to all scans
- Labels fixed
- Measure alignment using closest point distance

Before

After

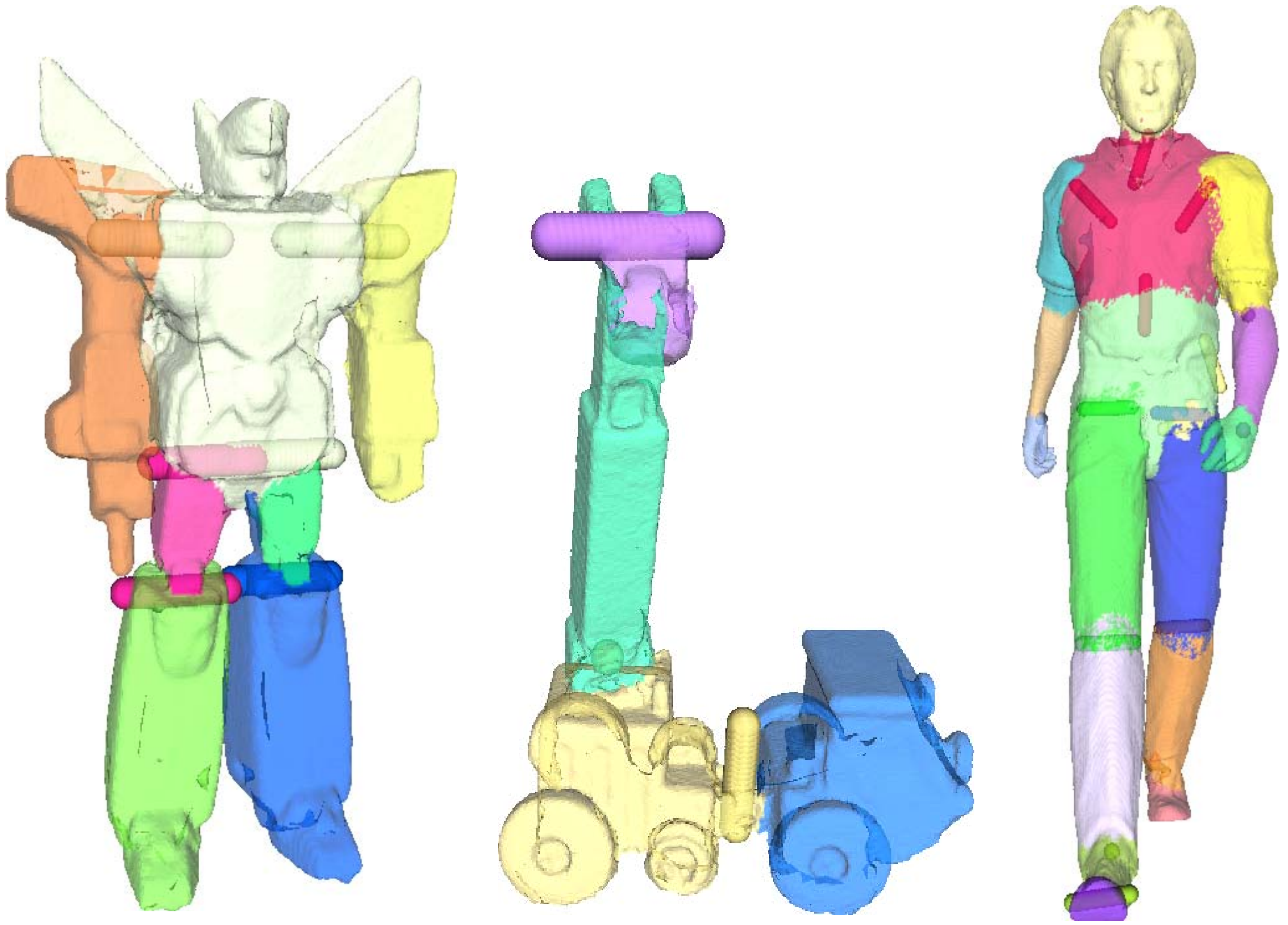
Transformation Optimization

- Multi-part, multi-frame articulated Iterative Closest Point (ICP)
    - Update closest point
    - Solve for transformation
    - Repeat until convergence
- Gauss-Newton for non-linear least squares

(Converged)

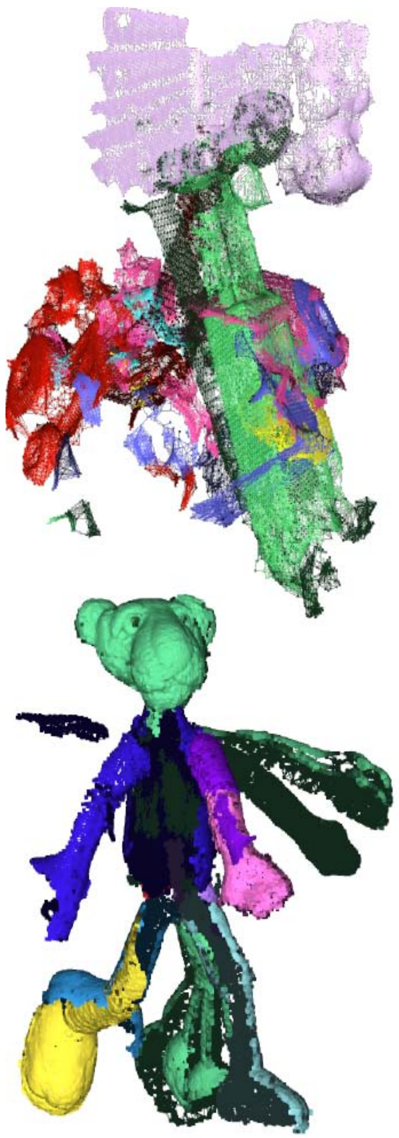
Joint Constraint

- **Prevents parts from separating**
- **Two types of joints**
  - Ball Joints (3 DOF)
  - Hinge Joints (1 DOF)
- **Derived from part boundaries & transformations solved so far**

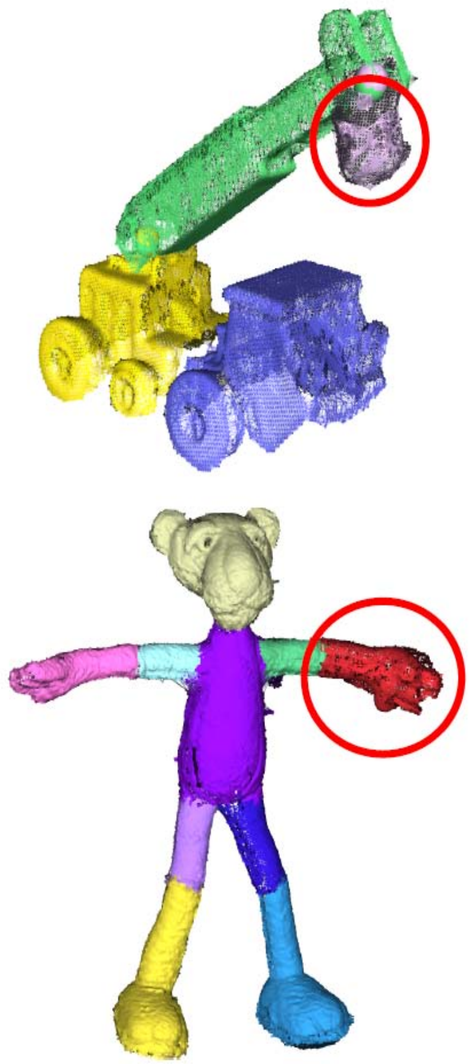


Reconstructed Joints

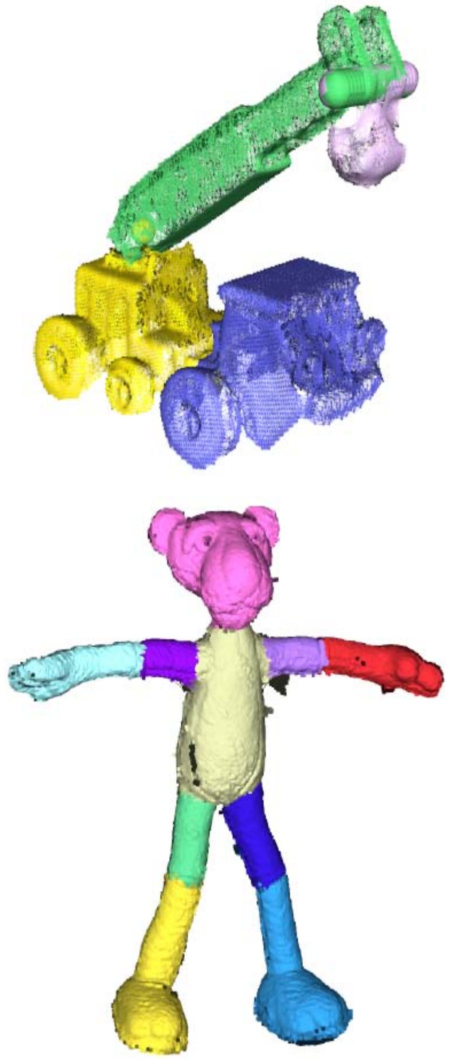
Joint Constraint



No Joints



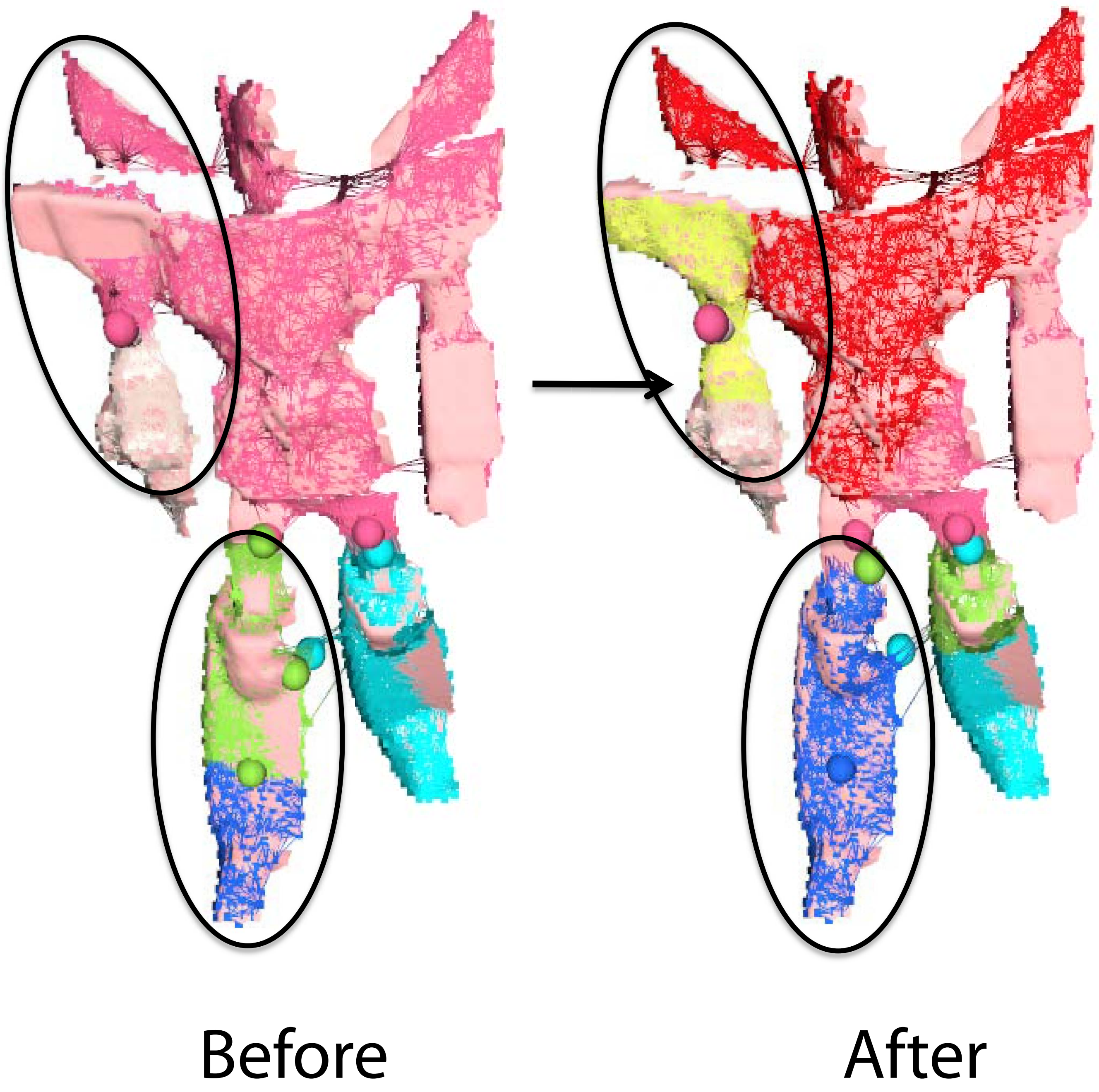
Ball Joints Only



Ball and Hinge Joints

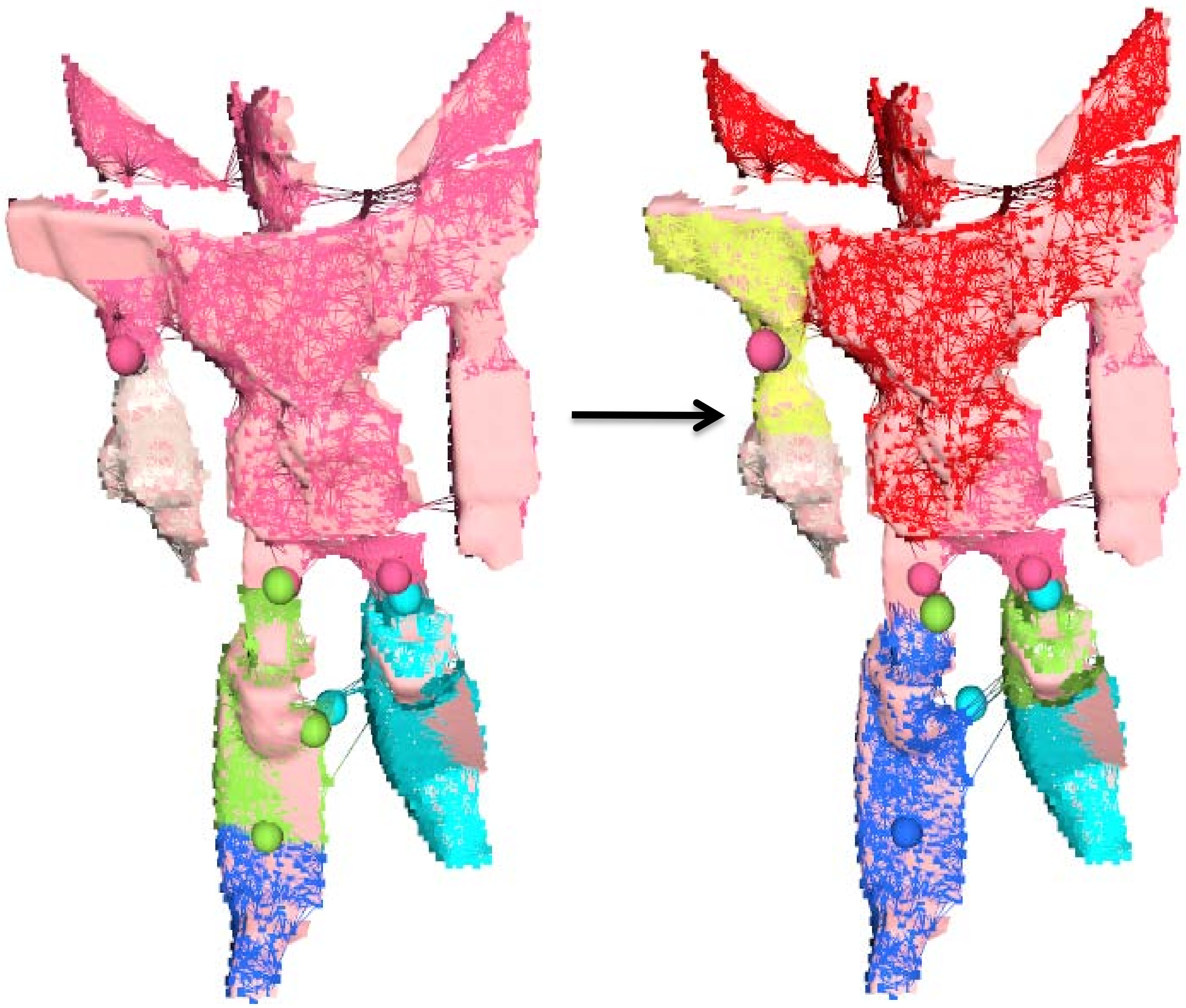
Label Optimization

- Change the labels to produce better alignment
- Transformations fixed
- Measure alignment using closest point distance



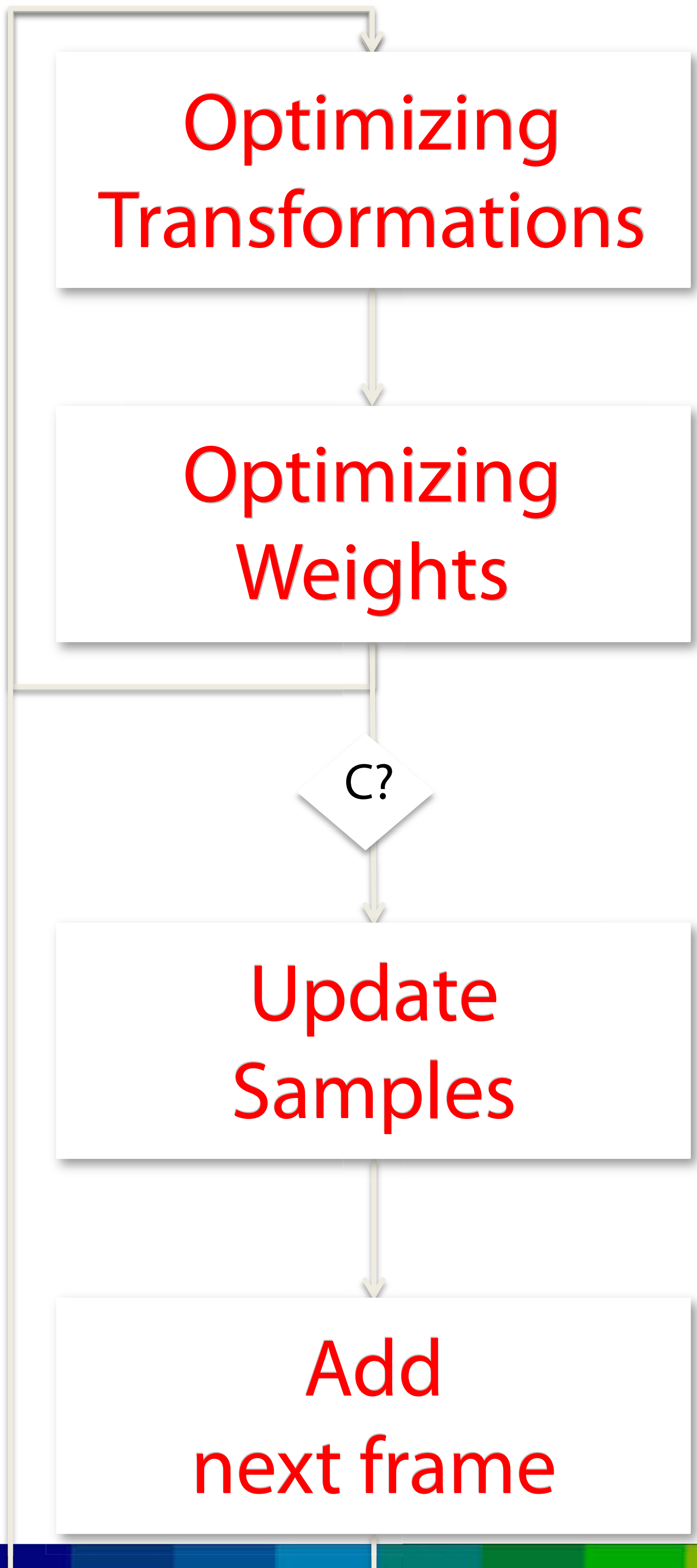
Label Optimization

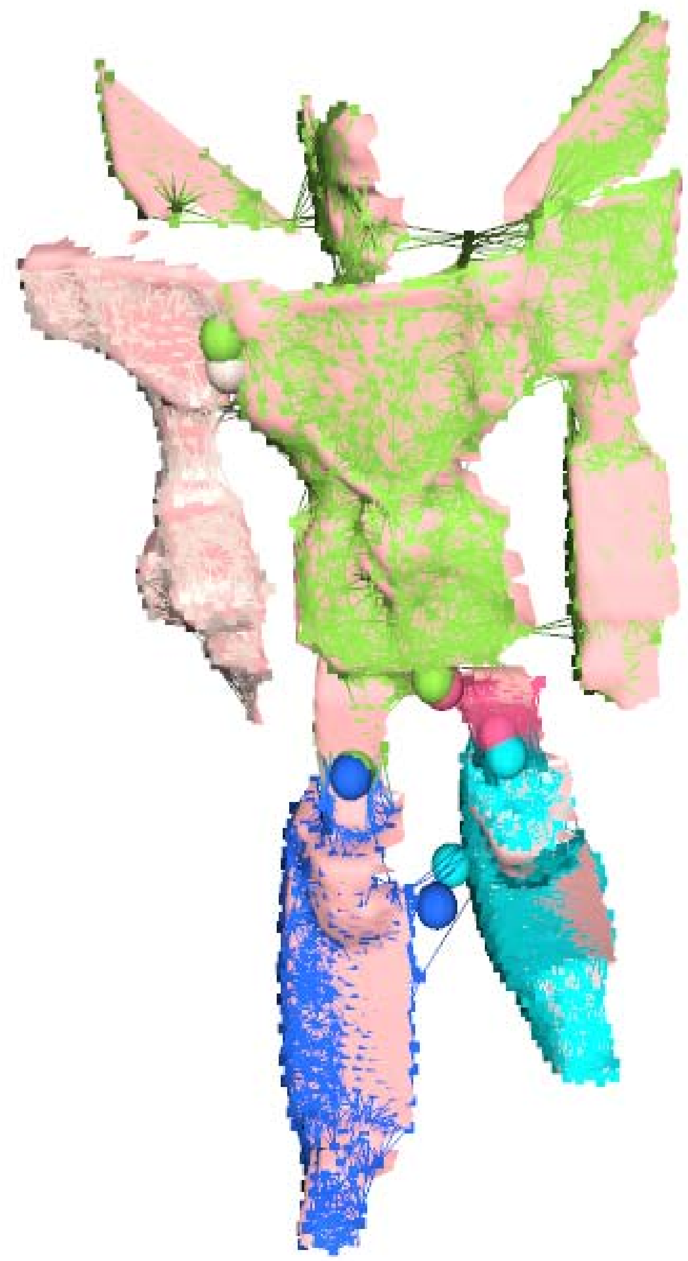
- **Graph Cuts [Boykov et al. 2002]**
  - Data constraint: minimize distance
  - Smoothness constraint: consolidate labels



Before After

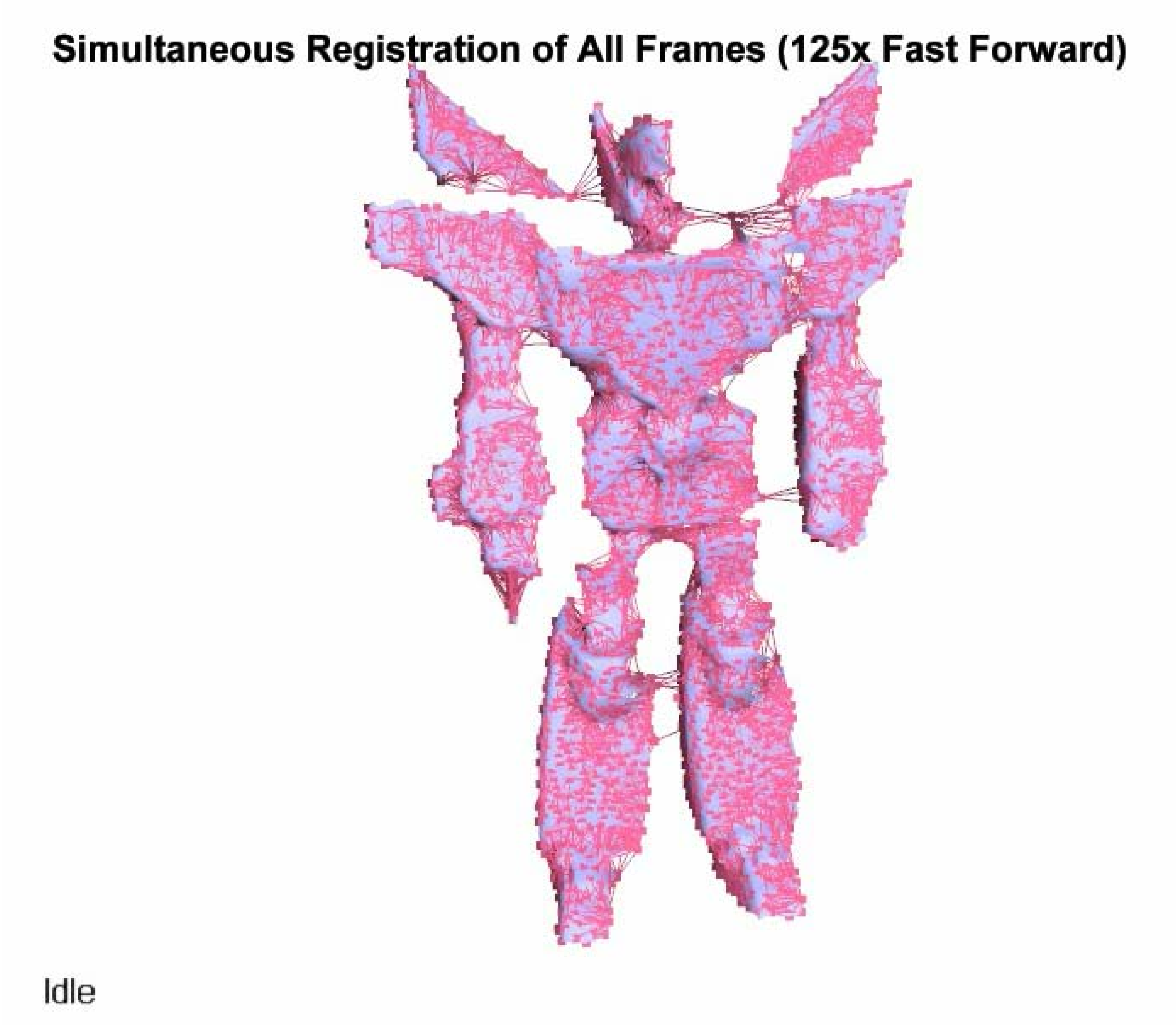
Global Refinement: Step Through





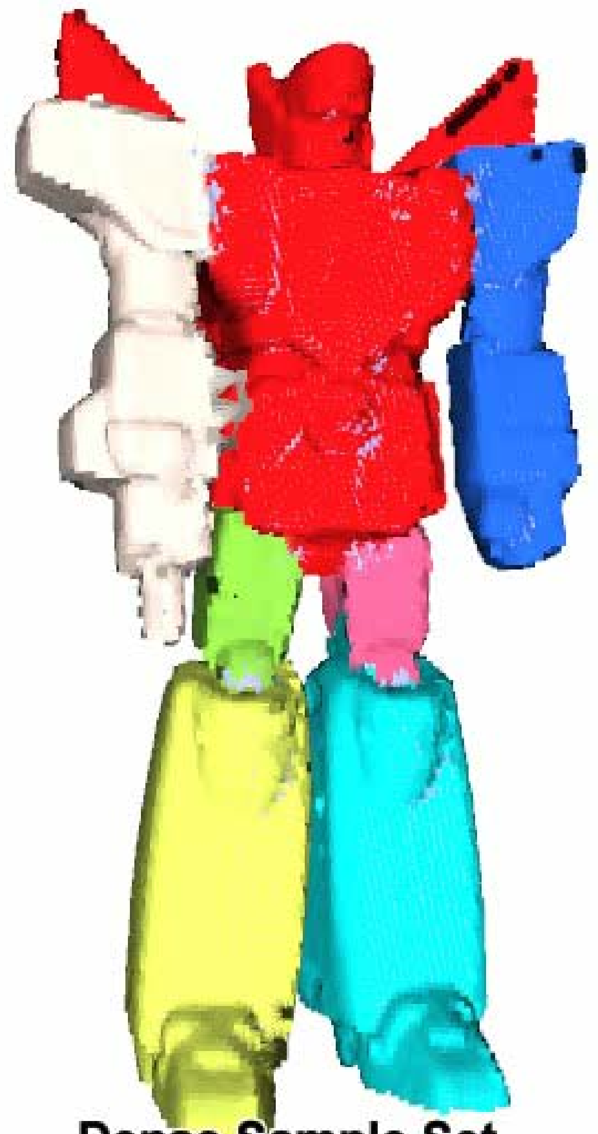
(Converged)

Global Refinement: Fast Forward



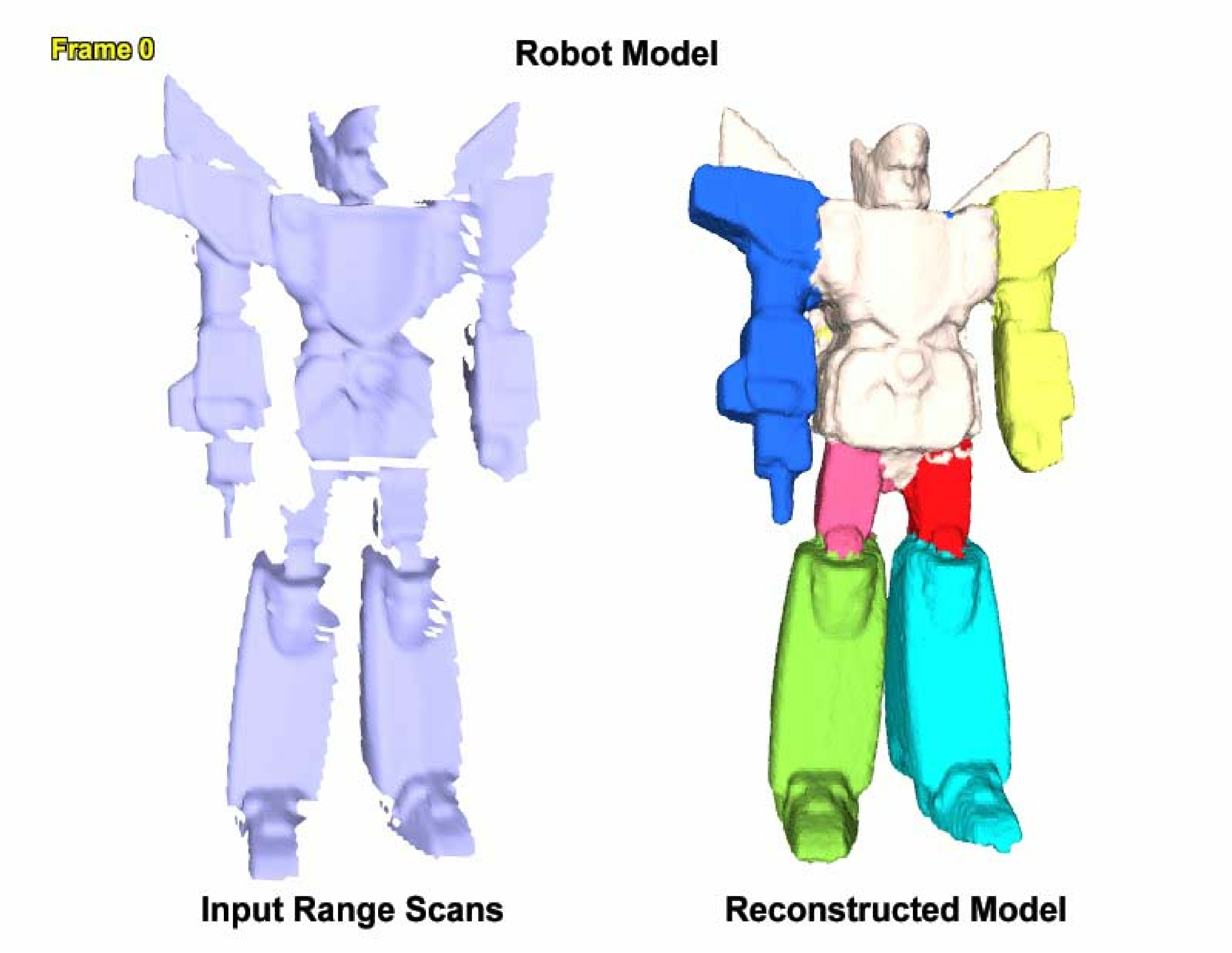
Post-processing

- Gather all frames into reference pose
- Resample surface, reconstruct mesh



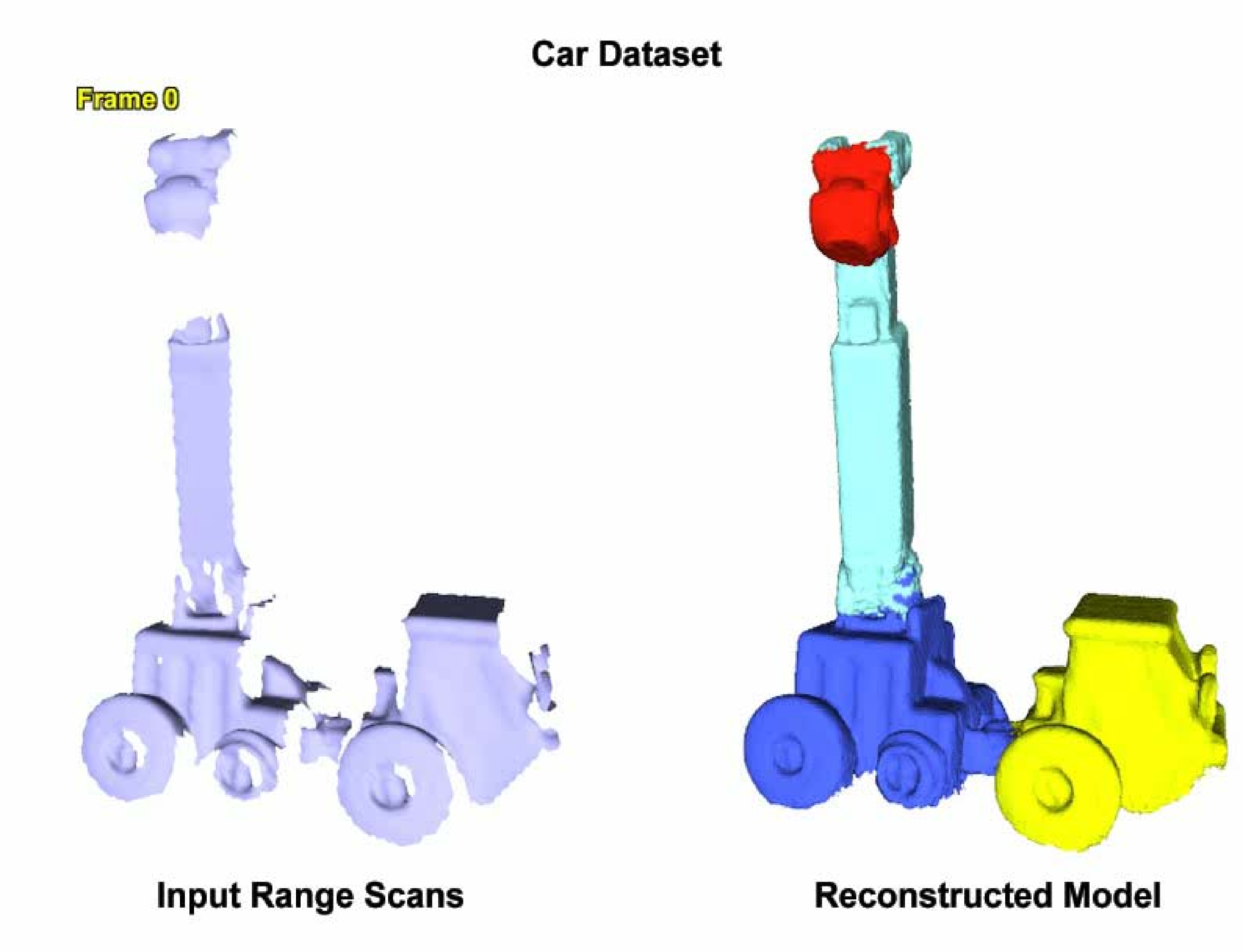
Results

Results: Registration



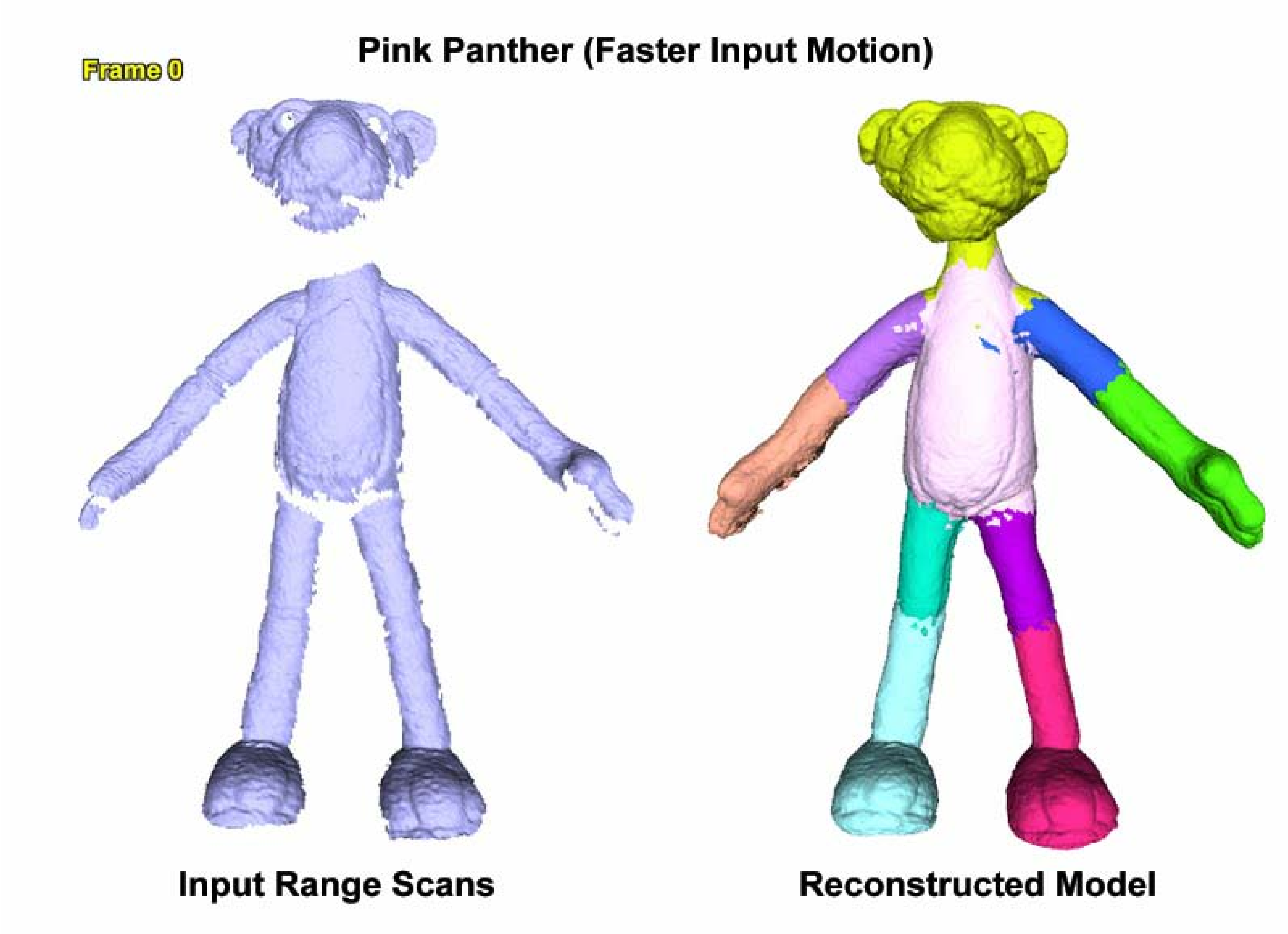
- Intel Xeon 2.5 GHz
- 90 Frames
- 7 Parts
- 0.84 million points
- 5000 DSG samples
- Total 165 mins
- 110 sec/frame

Results: Registration



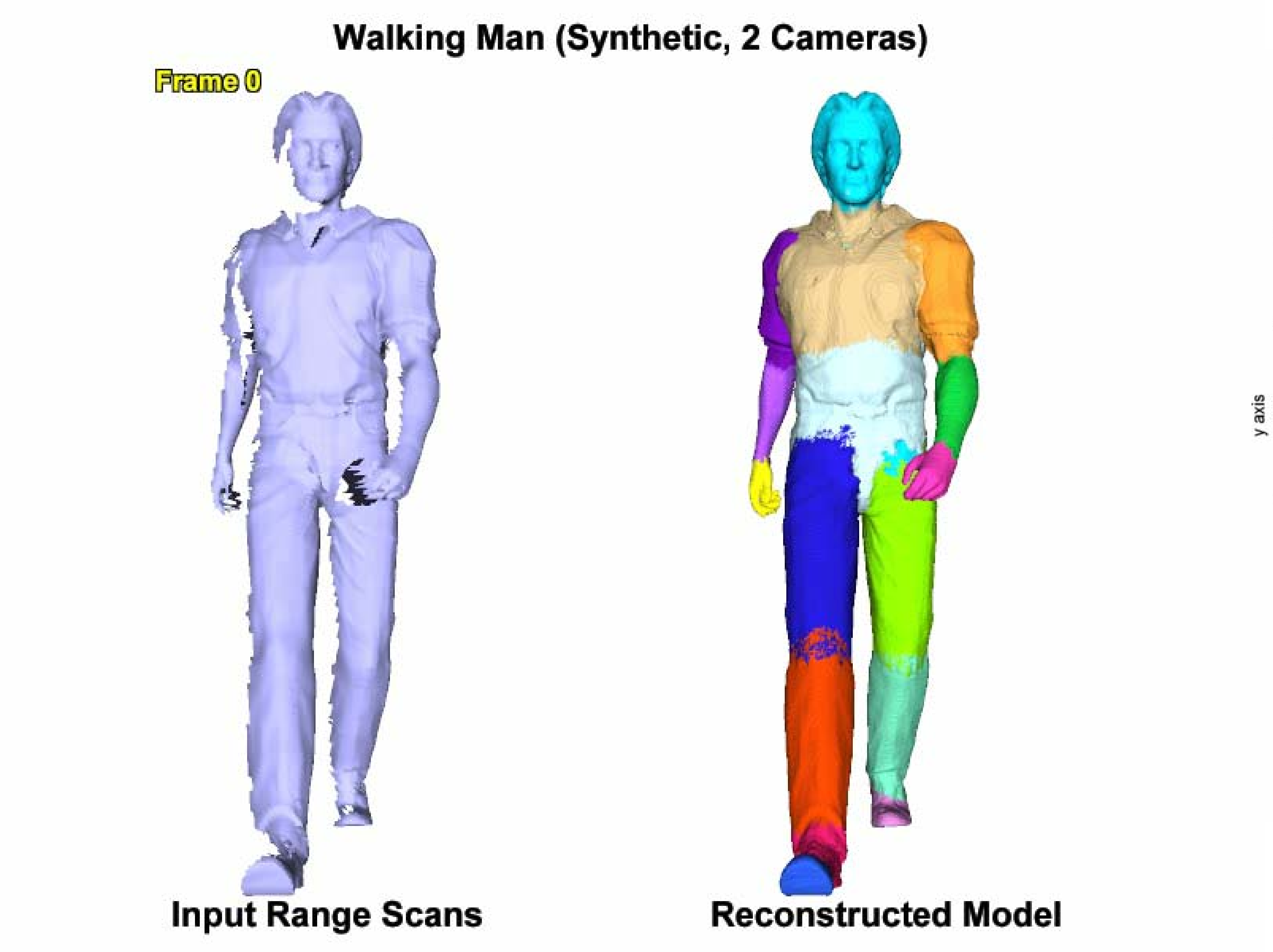
- Intel Xeon 2.5 GHz
- 90 Frames
- 4 Parts
- 0.48 million points
- 2700 DSG samples
- Total 66 mins
- 44 sec/frame

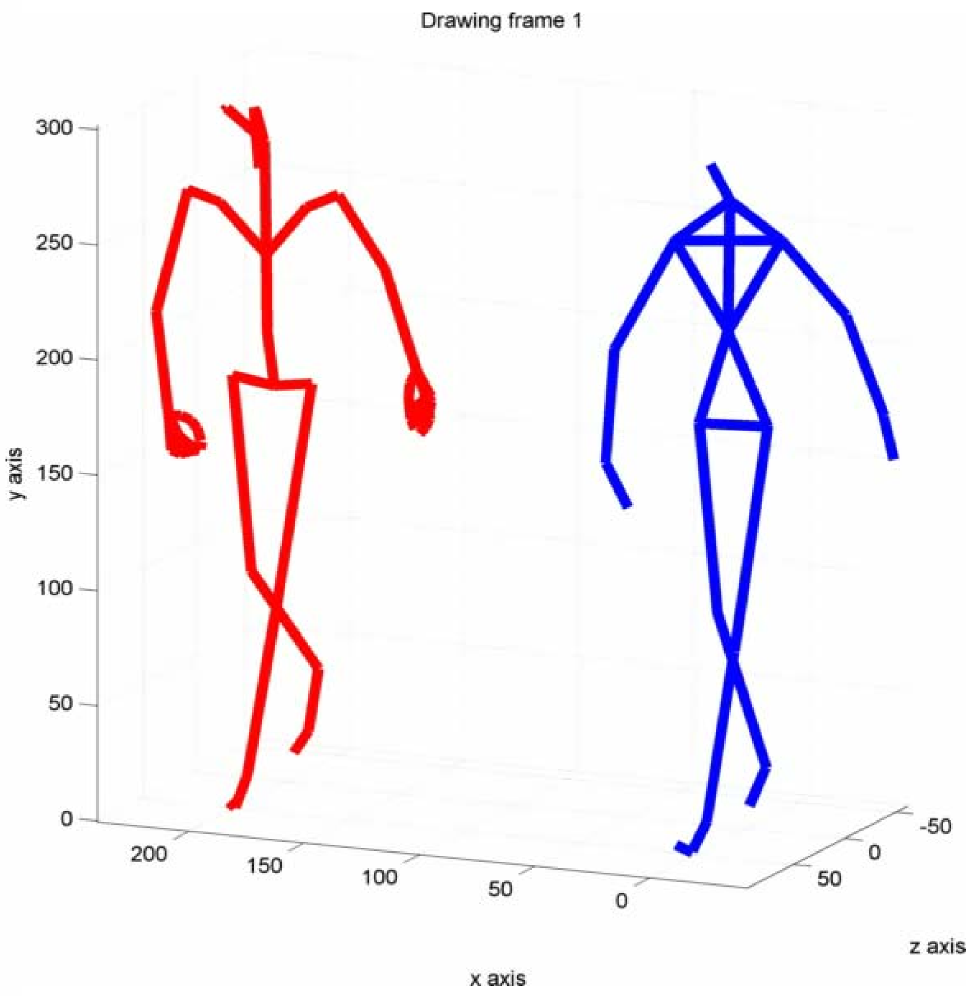
Results: Registration



- Intel Xeon 2.5 GHz
- 40 Frames
- 10 Parts
- 2.4 million points
- 4000 DSG samples
- Total 75 mins
- 113 sec/frame

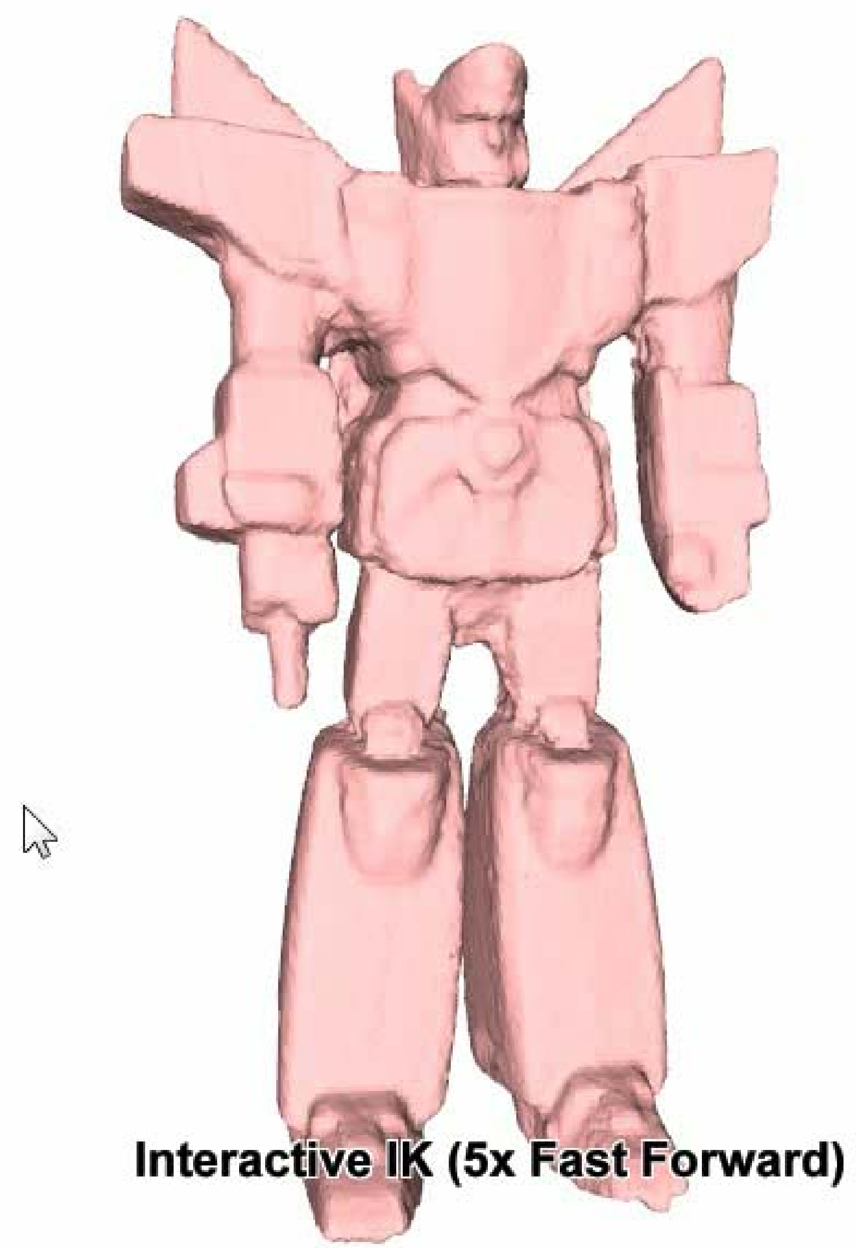
Ground truth comparison





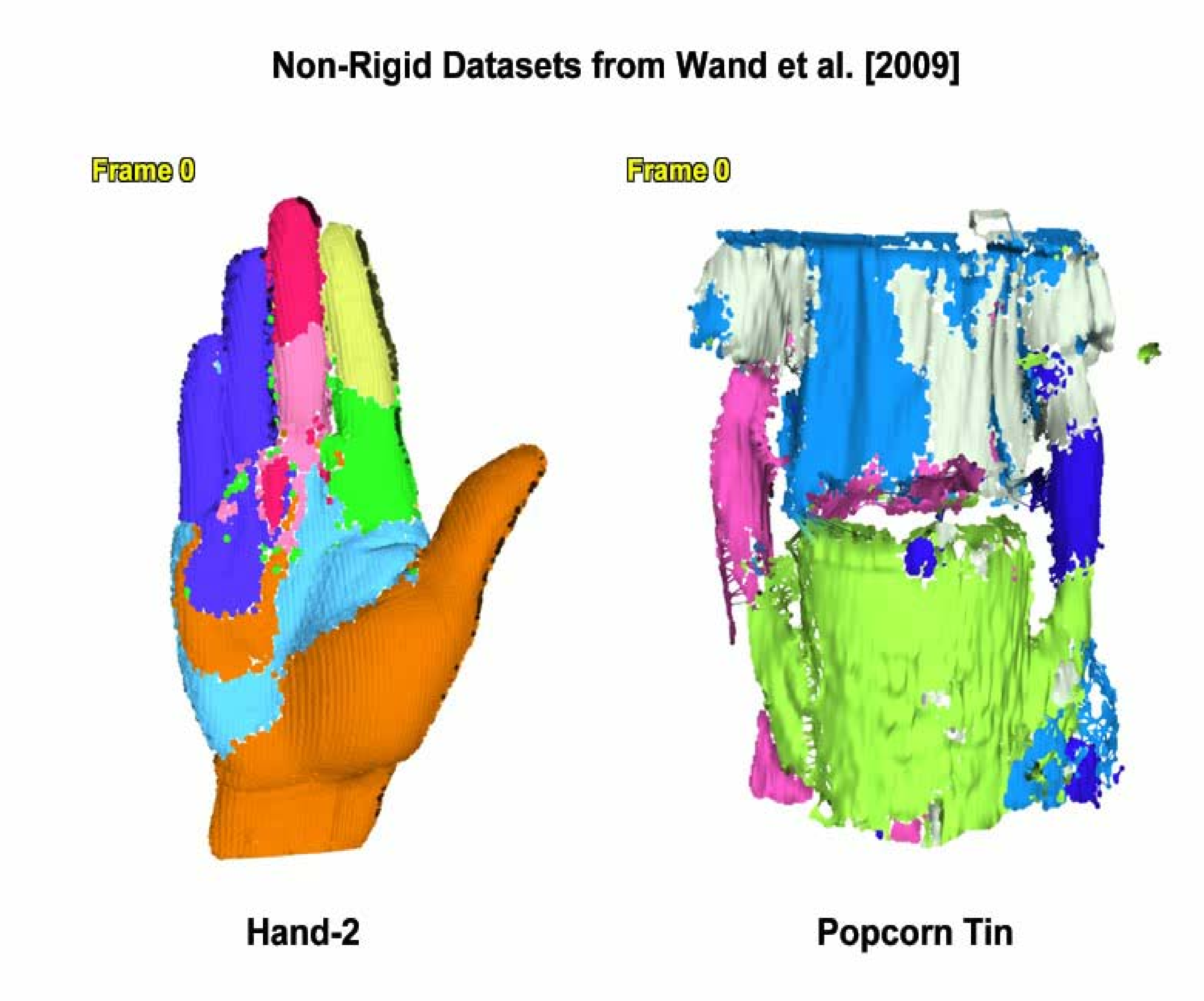
Red: Ground-truth
Blue: Reconstructed

Results: Inverse Kinematics



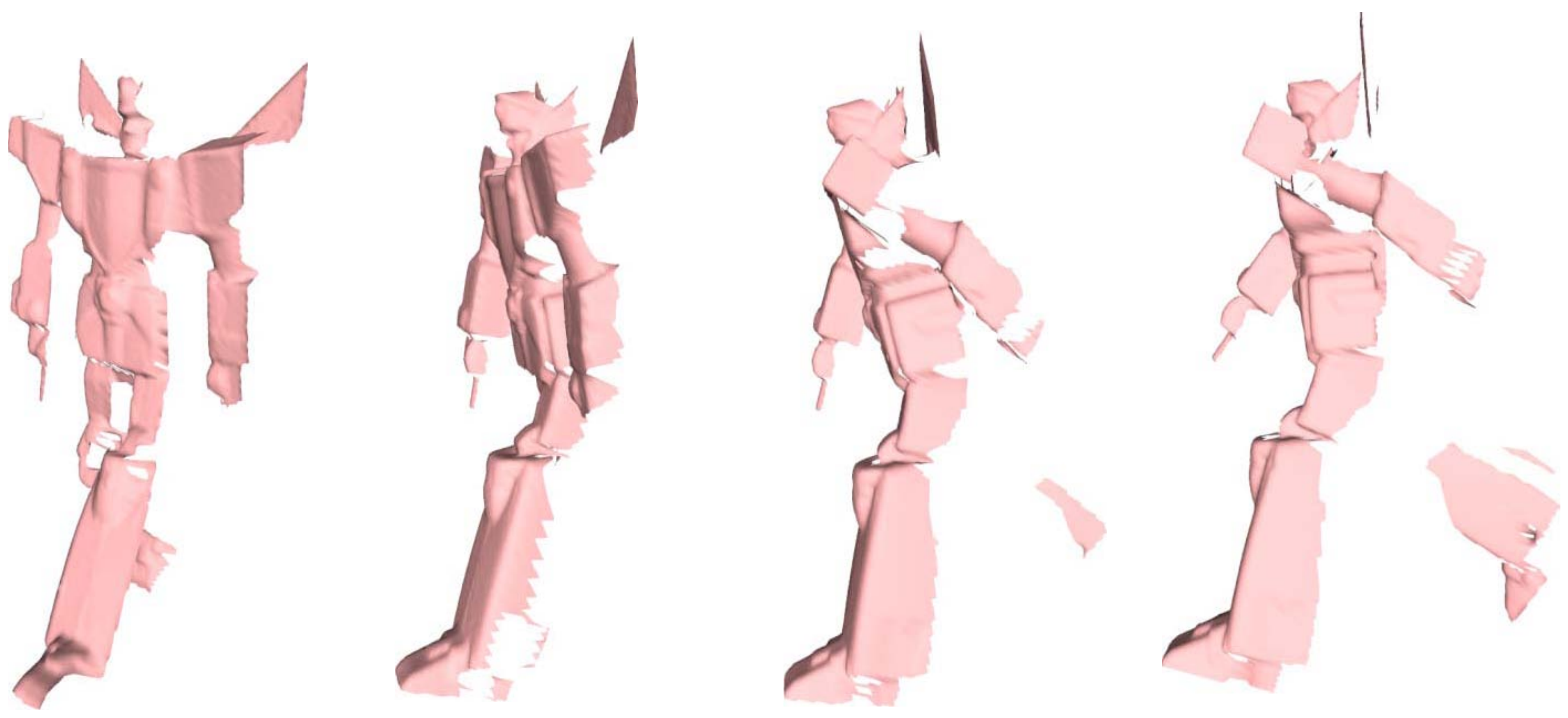
Limitations

Piecewise rigid approximation



Limitations

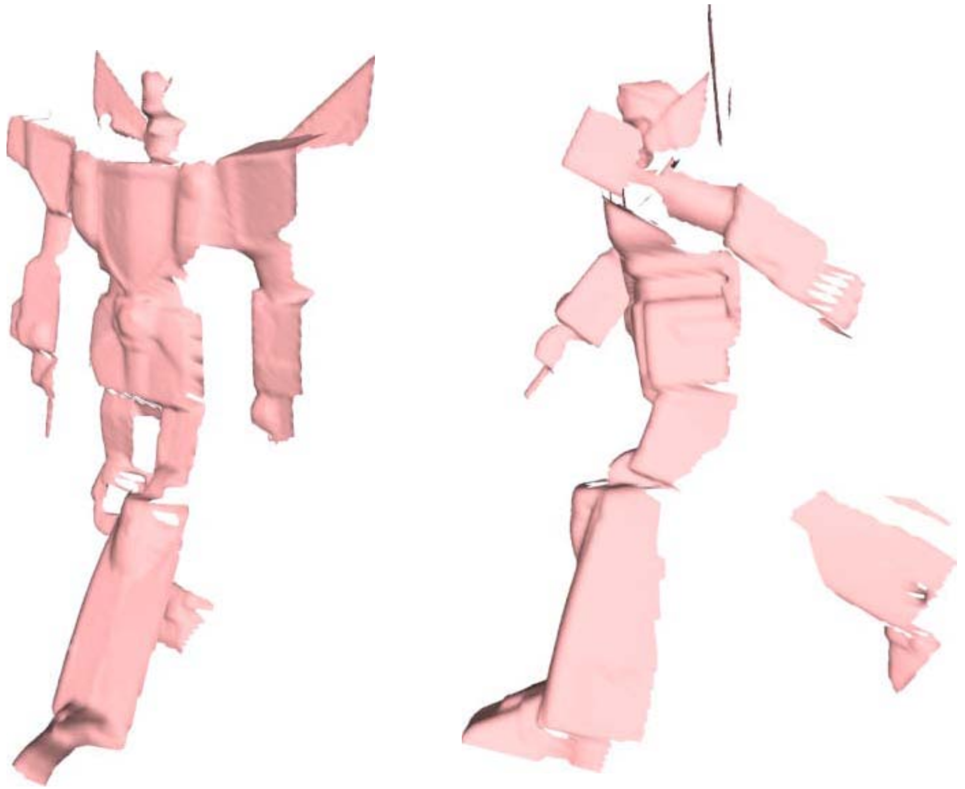
Needs sufficient overlap



Frame ***i*** Frame ***i+1*** Frame ***i+2*** Frame ***i+3***

Limitations

Needs sufficient overlap



Frame ***i***　　　　Frame ***i+1***

Articulated Global Registration

Implementation Details

Major Implementation Issues

Global registration T-step

Simple outline of the essential steps

Setting up the non-linear system for optimization

Global registration W-step

Setting up the graph-cut optimization

The algorithm in essence

At the end of the day: we have a huge "database" of closest-point correspondences.

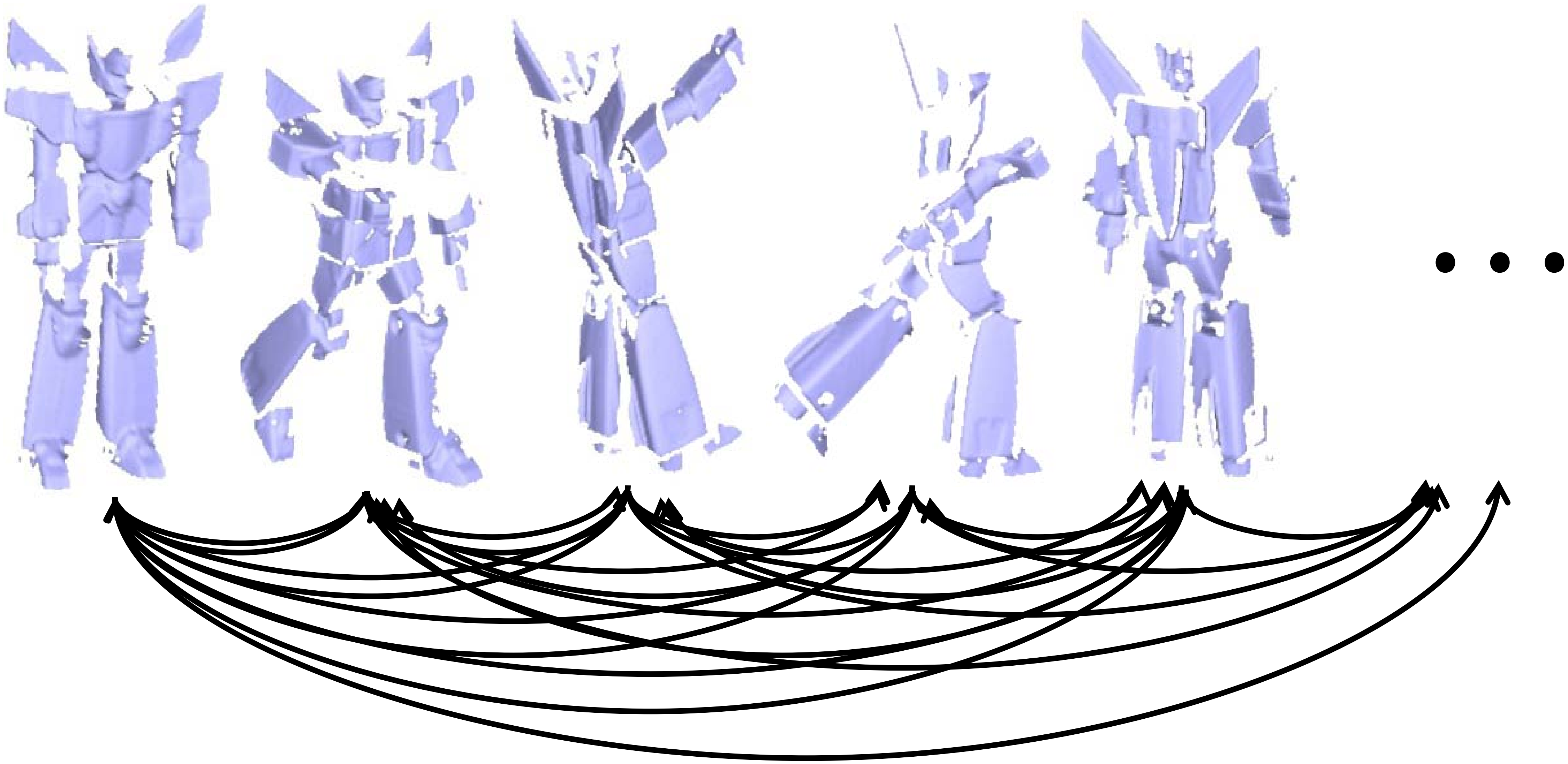
Each correspondence has the following info:

- Source point info, including
  - Frame of origin (f)
  - Original vertex position & normal in scan (x, n_x)
  - Weight (w)
- Target point info, including
  - Frame of origin (g)
  - Original vertex position & normal in scan (y, n_y)

Always a sample from the DSG!

Naïve method vs. DSG

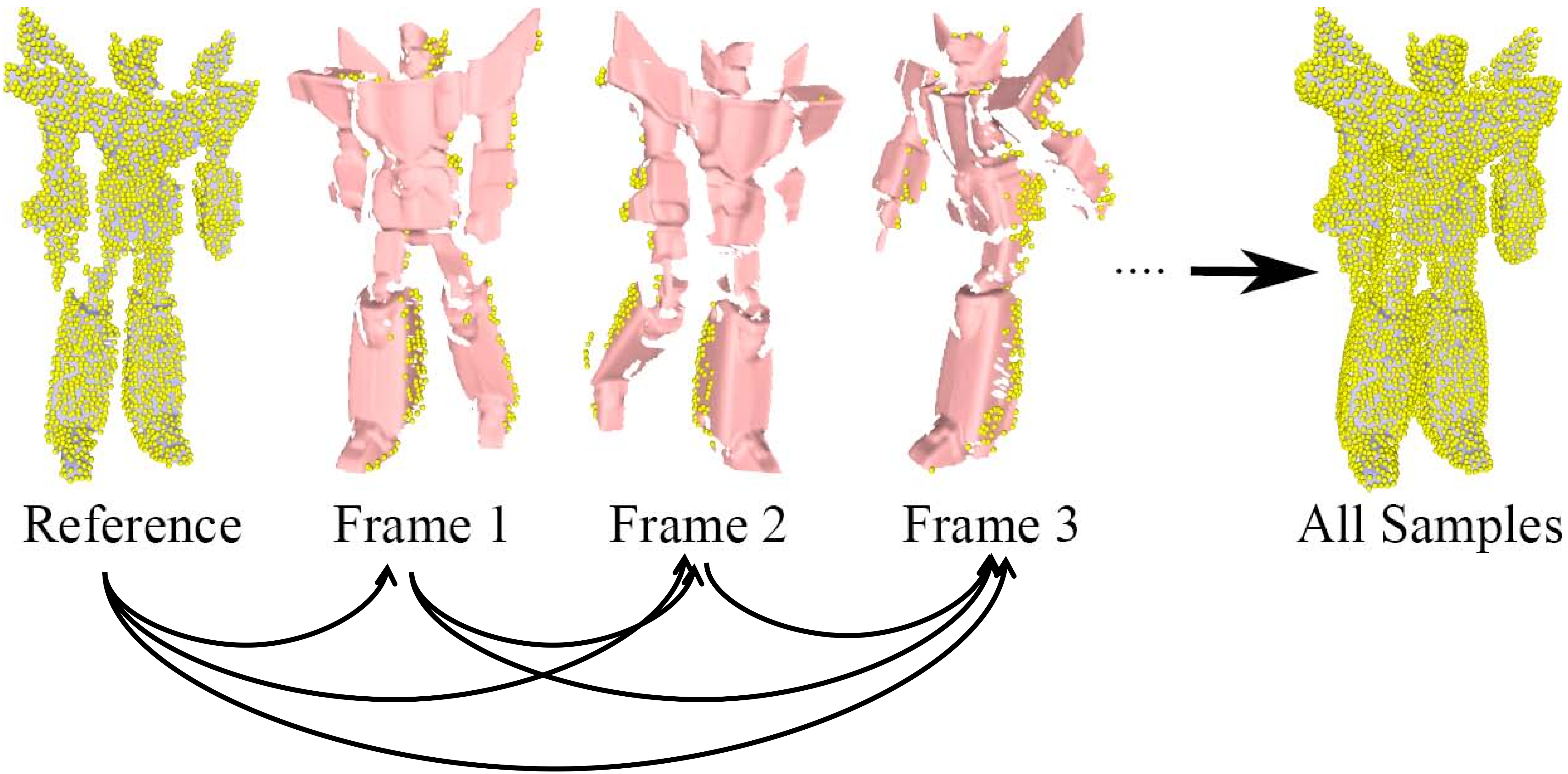
A simple way to setup the optimization:



O(n^2) combinations!!!

Naïve method vs. DSG

Using the DSG:



Life of a “sample point”

A sample point has multiple target points

- A target for each frame and for each transformation

Example

- A sample point from frame *f* has
  - a target point to frame $f+1$
  - a target point to frame $f+2$
  - a target point to frame $f+3$ (etc…)

How to find the target points?

- Transform from frame $f \rightarrow g$ (using current weight)
- The closest point is the target!

How to setup the optimization?

$$\operatorname*{argmin}_{\mathcal{T},\mathcal{W}} \quad \alpha\, \mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) + \beta\, \mathcal{E}_{\text{joint}}(\mathcal{T}) + \gamma\, \mathcal{E}_{\text{weight}}(\mathcal{W})$$

$$\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) = \sum_{\mathbf{x}\in S} \sum_{\text{Valid } \mathbf{y}_{j*}^{(g)}} d\left(T_{j*}^{(f\to\text{Ref})}(\mathbf{x}), T_{j*}^{(g\to\text{Ref})}\left(\mathbf{y}_{j*}^{(g)}\right)\right)$$

$$\mathcal{E}_{\text{joint}}(\mathcal{T}) = \sum_{\text{All } F_f} \sum_{\substack{\text{Valid Joints}\\ (\mathbf{u}_{ij},\vec{\mathbf{v}}_{ij})}} \sum_{t\in\mathbb{R}^3} \left\| {T_i^{(f\to\text{Ref})}}^{-1}(\mathbf{u}_{ij} + t\vec{\mathbf{v}}_{ij}) - {T_j^{(f\to\text{Ref})}}^{-1}(\mathbf{u}_{ij} + t\vec{\mathbf{v}}_{ij}) \right\|^2$$

$$\mathcal{E}_{\text{weight}}(\mathcal{W}) = \sum_{(\mathbf{x},\mathbf{y})\in E} I\left(\mathbf{w}_{\mathbf{x}} \neq \mathbf{w}_{\mathbf{y}}\right)$$

Non-linear optimization by linearization

We solve it by repeatedly linearizing the objective

How to linearize a rigid transformation $T = (R,t)$?

- $T(\mathbf{x}) = R\mathbf{x} + t$ (R = rotation matrix, t = translation)
- $T(\mathbf{x}) \sim (I + w^\wedge)\,\mathbf{x} + v$
  - $w^\wedge$ is a skew-symmetric matrix, v is a translation
  - This approximates the rotation about the identity
- To linearize about an arbitrary rigid transformation?
  - Apply the approximation as a "correction"
  - $T(\mathbf{x})' = T^{corr} * T(\mathbf{x}) = (I + w^\wedge)\,T(\mathbf{x}) + v$

How about an inverse $T^{-1} = (R^T, -R^T t)$?

- Note $R^T \sim (I + w^\wedge)^T = (I - w^\wedge)$
- Eventually $T^{-1}(\mathbf{x})' = T^{-1} * T^{corr\ -1}(\mathbf{x}) = T^{-1}\,[\,(\mathbf{x} - v) - w^\wedge(\mathbf{x} - v)\,]$

$$\sim T^{-1}\,[\,(\mathbf{x} - v) - w^\wedge\mathbf{x}\,]$$

E_{fit} boils down to:

$$\begin{bmatrix} -\widehat{(\mathbf{x}')} & I & \widehat{\left(\mathbf{y}_j^{(g)\prime}\right)} & -I \\ -\left(\vec{\mathbf{n}}_{\mathbf{y}}' \times \mathbf{x}'\right)^{\top} & \vec{\mathbf{n}}_{\mathbf{y}}'^{\top} & \left(\vec{\mathbf{n}}_{\mathbf{y}}' \times \mathbf{y}_j^{(g)\prime}\right)^{\top} & -\vec{\mathbf{n}}_{\mathbf{y}}'^{\top} \end{bmatrix} \begin{bmatrix} \omega_j^{(f)} \\ \upsilon_j^{(f)} \\ \omega_j^{(g)} \\ \upsilon_j^{(g)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}' - \mathbf{y}_j^{(g)\prime} \\ \vec{\mathbf{n}}_{\mathbf{y}}'^{\top} \left(\mathbf{x}' - \mathbf{y}_j^{(g)\prime}\right) \end{bmatrix}$$

First three rows: point-to-point constraint

Fourth row: point-to-plane constraint

- Hat operator ^ → "skew-symmetrizes" a vector
- ***x'*** (or ***y'***) = current transformation applied to ***x*** (or ***y***)
- Note: this constraint relates different frames *f* and *g*

E_{joint} boils down to:

$$\begin{bmatrix} R_i^{(f\to\mathrm{Ref})}\hat{\mathbf{u}}^{\top} & -R_i^{(f\to\mathrm{Ref})}{}^{\top} & -R_j^{(f\to\mathrm{Ref})}\hat{\mathbf{u}}^{\top} & R_j^{(f\to\mathrm{Ref})}{}^{\top} \end{bmatrix} \begin{bmatrix} \omega_i^{(f)} \\ v_i^{(f)} \\ \omega_j^{(f)} \\ v_j^{(f)} \end{bmatrix} = \begin{bmatrix} \mathbf{u}'_i - \mathbf{u}'_j \end{bmatrix}$$

Three rows for each joint constraint (in each frame)

- R_i and R_j are the current tfs (before correction)
- ***u'*** = current transformation applied to ***u***
- Note: this constraint doesn't relate different frames

Populating the linear system

1. **Simply plug in these formulas**
2. **Put numbers in the right location in the matrix**

Example: Suppose we have 3 frames and 2 bones.

- 2 corresp between f_0 & f_1 (one for b_0, one for b_1)
- 1 corresp between f_0 & f_2 (for b_0)
- 1 corresp between f_1 & f_2 (for b_1)
- 1 joint between b_0 and b_1 (applies to all frames)

Populating the linear system

Frame number

Bone number

$$\begin{bmatrix} w_0^0 \\ v_0^0 \\ w_1^0 \\ v_1^0 \\ \hline w_0^1 \\ v_0^1 \\ w_1^1 \\ v_1^1 \\ \hline w_0^2 \\ v_0^2 \\ w_1^2 \\ v_1^2 \end{bmatrix} =$$

1 joint between b_0 and b_1

(applies to all frames)

Solving the system

After constructing the matrix:

(1) Solve for the values of *w*, *v*

(2) Convert them to a rigid tf (exponential map)

(3) Apply correction

(4) Repeat until convergence (δ error < threshold)

A number of sparse linear solvers exist

-- We used TAUCS

Setting up the graph-cut optimization

We need to *solve* for the weights

- Evaluate distance to closest point for **all transformations** and **all frames**
  - This is different in the previous step, where we found the closest point only for the ***current*** transformation

Example *(B = number of bones)*

- Each sample point from frame *f* has
  - *B* targets to frame *f+1* *(one per transformation)*
  - *B* targets to frame *f+2*
  - *B* targets to frame *f+3* (etc…)

Setting up the graph-cut optimization

Use the same error term as before

- Data term for assigning bone "b" to a sample point x
  - Sum up the error for all frames, and average using the number of valid correspondences used in the sum
  - Special case: (a) rules for "invalidating" closest points exist. (b) If the closest point using the ***current*** weight is invalid, exclude all target points for that sample (in that frame). (c) Error in units of distance, not distance^2
- Smoothness term for assigning similar labels nearby
  - Use the "graph" part of the DSG, with constant error
- Can easily use existing graph-cut minimization code

Conclusions

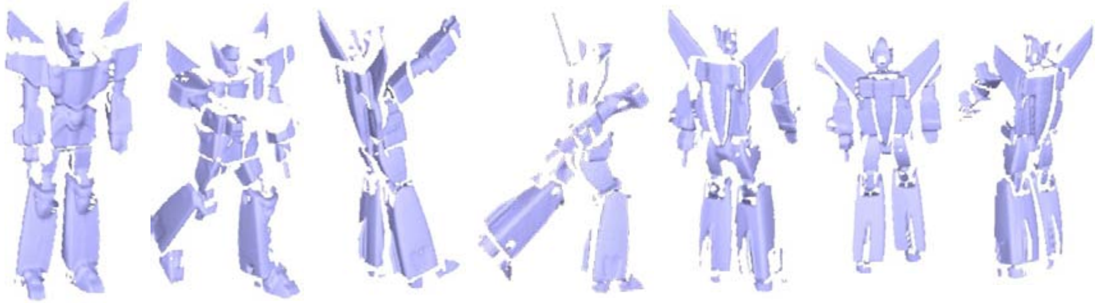
Articulated Global Registration

Contributions

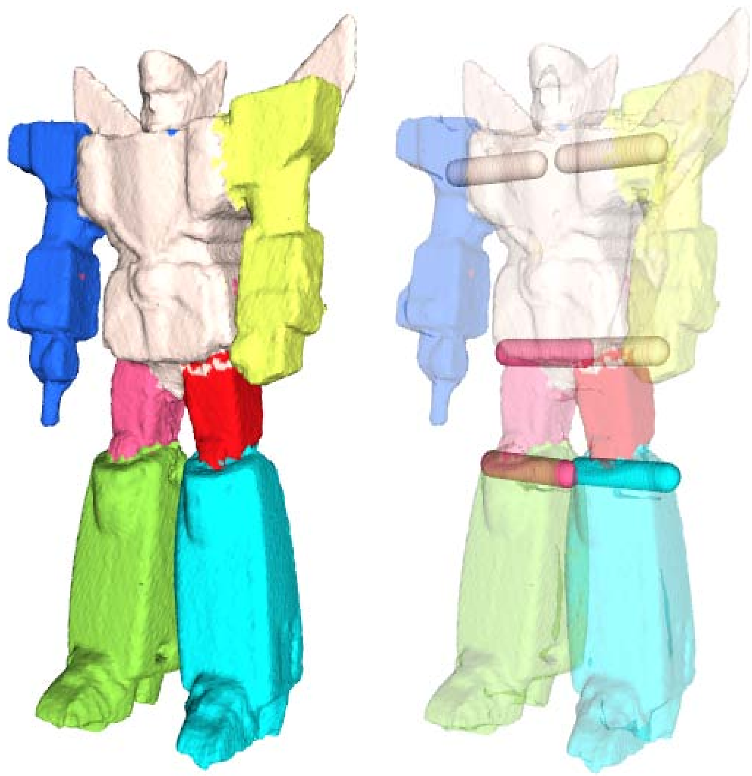
- Automatic registration algorithm for dynamic subjects
- No template, markers, skeleton, or segmentation needed
- Final result used directly to produce new animations

In the future

- Add non-rigid motion
- Reduce parameters
- Real-time



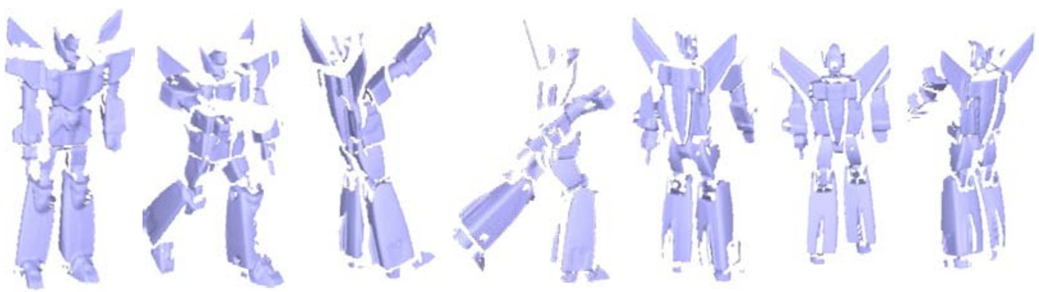
Input Range Scans



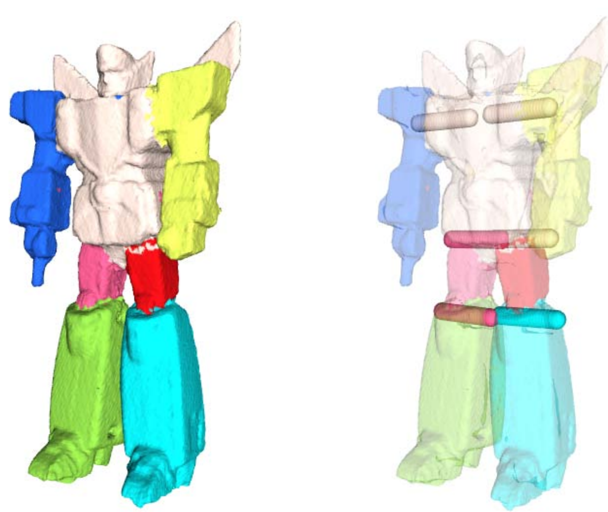
Reconstructed Poseable 3D Model

Thank you for your attention!!

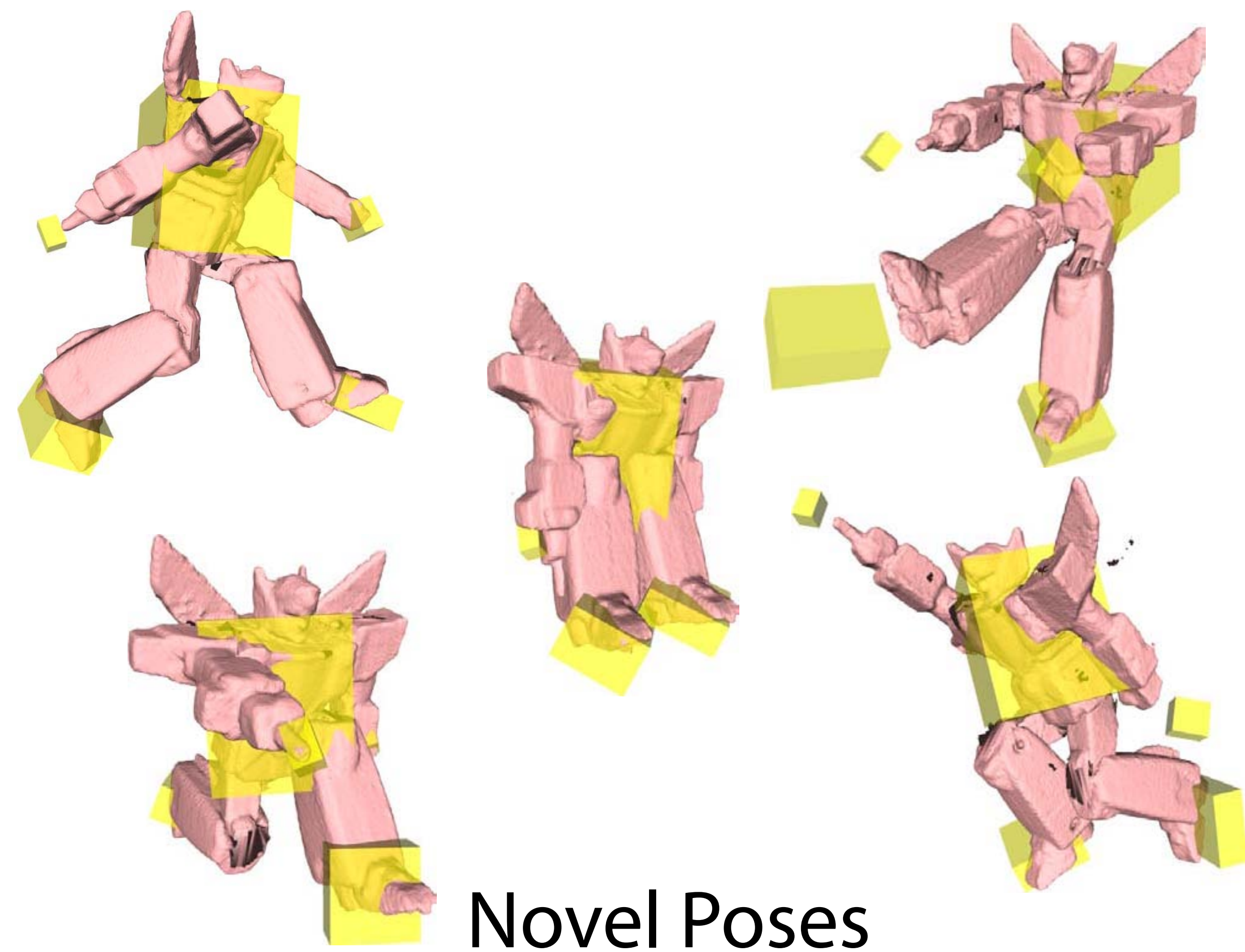
Questions?



Input Range Scans



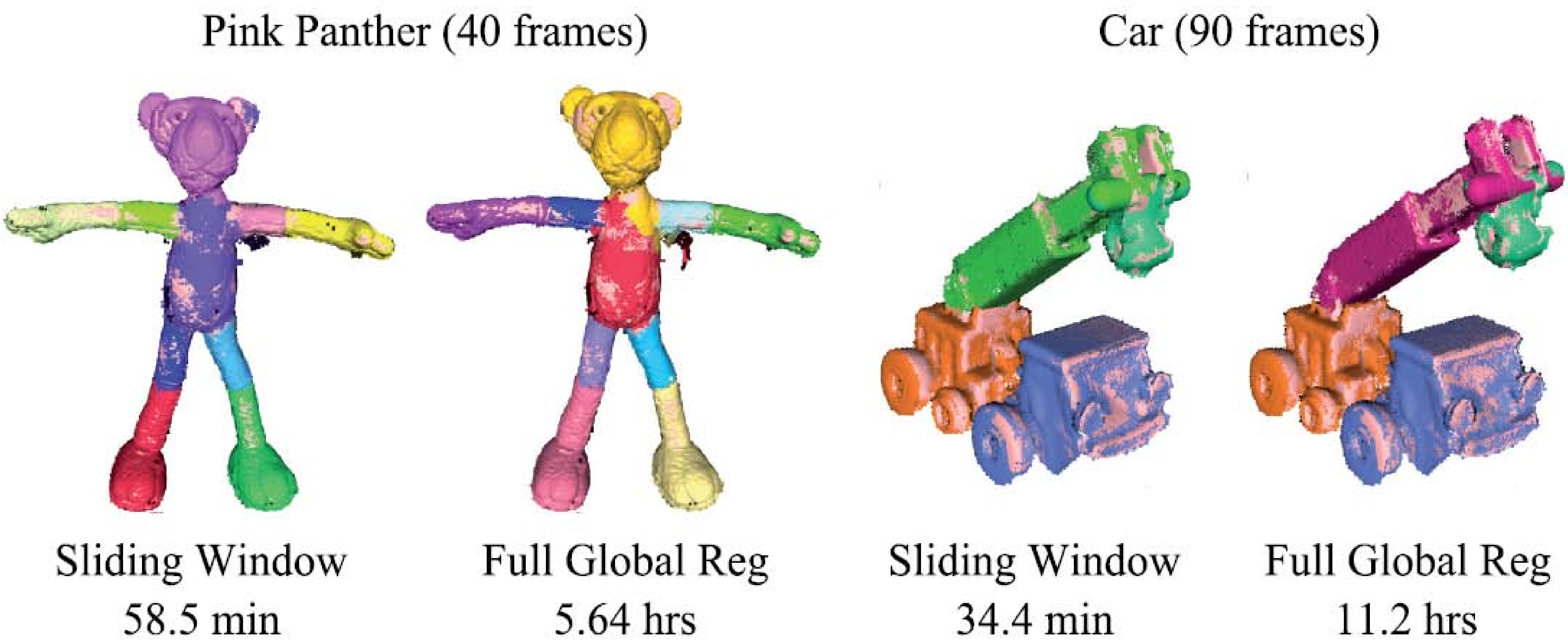
Reconstructed Poseable 3D Model



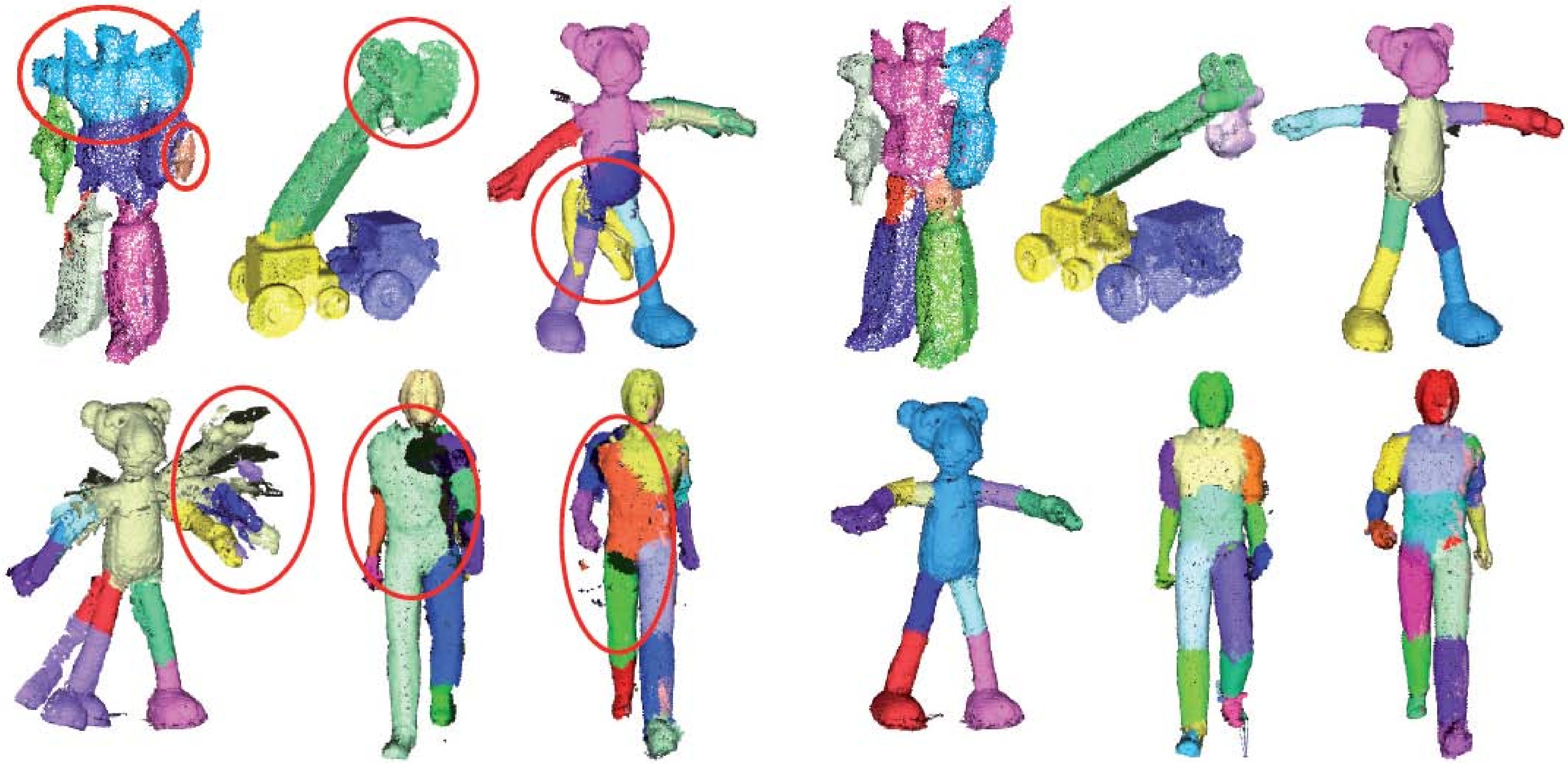
Novel Poses

Additional Comparisons

Sliding window comparison



Local vs. global comparison



(a) Using sequential registration (b) Using simultaneous registration