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
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

Initialized Frames

Global Refinement

Optimized labels, motion, joints, and geometry
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

Frame $i$  

Frame $i+1$
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
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
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

- Optimizing Transformations
- Optimizing Weights
- Update Samples
- Add next frame

(Converged)
Global Refinement: Fast Forward

Simultaneous Registration of All Frames (125x Fast Forward)

Idle
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

Walking Man (Synthetic, 2 Cameras)

Input Range Scans
Reconstructed Model

Red: Ground-truth
Blue: Reconstructed
Results: Inverse Kinematics

Interactive IK (5x Fast Forward)
Limitations

Piecewise rigid approximation

Non-Rigid Datasets from Wand et al. [2009]

Hand-2  Popcorn Tin
Limitations

Needs sufficient overlap
Limitations

Needs sufficient overlap
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 \((y, n_y)\)
Naïve method vs. DSG

A simple way to setup the optimization:

\[ O(n^2) \text{ combinations!!!} \]
Naïve method vs. DSG

Using the DSG:

Reference  Frame 1  Frame 2  Frame 3  All Samples
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?

$$\arg \min_{T,W} \alpha \mathcal{E}_{\text{fit}}(T,W) + \beta \mathcal{E}_{\text{joint}}(T) + \gamma \mathcal{E}_{\text{weight}}(W)$$

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

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

$$\mathcal{E}_{\text{weight}}(W) = \sum_{(x,y) \in E} I \left( w_x \neq w_y \right)$$
Non-linear optimization by linearization

We solve it by repeatedly linearizing the objective

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

- $T(x) = Rx + t$ ($R =$ rotation matrix, $t =$ translation)
- $T(x) \approx (I + w^\wedge) x + v$
  - $w^\wedge$ 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”
- $T(x)' = T^{corr} * T(x) = (I + w^\wedge) T(x) + v$

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

- Note $R^T \sim (I + w^\wedge)^T = (I - w^\wedge)$
- Eventually $T^{-1}(x)' = T^{-1} * T^{corr}^{-1} (x) = T^{-1} [ (x - v) - w^\wedge (x - v) ]$
  \[ \sim T^{-1} [ (x - v) - w^\wedge x ] \]
$E_{\text{fit}}$ boils down to:

$$
\begin{bmatrix}
-\hat{x}' \\
-\hat{n}_y \times x' \\
-\hat{n}_y^T (\hat{n}_y \times y_j^{(g)})^T \\
-\hat{n}_y^T \\
\end{bmatrix}
\begin{bmatrix}
I \\
\hat{n}_y^T \\
\hat{n}_y^T (\hat{n}_y \times y_j^{(g)})^T \\
-I \\
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}
\omega_j^{(f)} \\
v_j^{(f)} \\
\omega_j^{(g)} \\
v_j^{(g)} \\
\end{bmatrix}
\end{bmatrix}
= \begin{bmatrix}
x' - y_j^{(g)'} \\
\hat{n}_y^T (x' - y_j^{(g)'}) \\
\end{bmatrix}
$$

First three rows: point-to-point constraint

Fourth row: point-to-plane constraint

- Hat operator $\hat{\cdot}$ $\rightarrow$ “skew-symmetrizes” a vector
- $x'$ (or $y'$) = current transformation applied to $x$ (or $y$)
- Note: this constraint relates different frames $f$ and $g$
\( E_{\text{joint}} \) boils down to:

\[
\begin{bmatrix}
R_i^{(f\rightarrow\text{Ref})}^T & -R_i^{(f\rightarrow\text{Ref})}^T & -R_j^{(f\rightarrow\text{Ref})}^T & R_j^{(f\rightarrow\text{Ref})}^T
\end{bmatrix}
\begin{bmatrix}
\omega_i^{(f)} \\
v_i^{(f)} \\
\omega_j^{(f)} \\
v_j^{(f)}
\end{bmatrix}
= \begin{bmatrix}
\dot{u}_i' - \dot{u}_j'
\end{bmatrix}
\]

Three rows for each joint constraint (in each frame)

- \( R_i \) and \( R_j \) are the current tfs (before correction)
- \( \dot{u}' = \) 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

\[
\begin{pmatrix}
\mathbf{w}_0^0 \\
\mathbf{v}_0^0 \\
\mathbf{w}_1^0 \\
\mathbf{v}_1^0 \\
\mathbf{w}_0^1 \\
\mathbf{v}_0^1 \\
\mathbf{w}_1^1 \\
\mathbf{v}_1^1 \\
\mathbf{w}_0^2 \\
\mathbf{v}_0^2 \\
\mathbf{w}_1^2 \\
\mathbf{v}_1^2
\end{pmatrix}
\begin{pmatrix}
w_0 \\
w_1 \\
w_0 \\
w_1 \\
w_0 \\
w_1 \\
w_0 \\
w_1 \\
w_0 \\
w_1 \\
w_0 \\
w_1
\end{pmatrix}
\begin{pmatrix}
n \\
1 \\
n \\
n \\
n \\
n \\
n \\
n \\
n \\
n \\
n \\
n
\end{pmatrix} = \begin{pmatrix}
\end{pmatrix}
\]

2 joints between frame 0 and frame 1

Frame number

Bone number

(assumes b0, b1, b2, and b3)
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 ($\delta$ 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*  \((B = \text{number of bones})\)

- 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 distance²

• 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
Thank you for your attention!!

Questions?

Input Range Scans

Reconstructed Poseable 3D Model

Novel Poses
Additional Comparisons
Sliding window comparison

Pink Panther (40 frames)

- Sliding Window: 58.5 min
- Full Global Reg: 5.64 hrs

Car (90 frames)

- Sliding Window: 34.4 min
- Full Global Reg: 11.2 hrs
Local vs. global comparison

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