Multi-frame Registration Using Reduced Deformable Models

Will Chang
Department of Computer Science and Engineering
University of California, San Diego
wychang@cs.ucsd.edu

October 27, 2008

1 Introduction

Registering multiple scans is a bit more involved than the pairwise case, especially when registering non-rigid 3D models. A straightforward extension of pairwise registration to the multiple scan case is to register scans sequentially, merging the geometry as we complete each registration. We call this the "accumulation" strategy, because we are gradually accumulating geometry to complete a 3D model. This approach has been used previously to register multiple frames of a rigid body. The advantages of this strategy are:

- **Efficiency:** a new frame only needs to be registered once to the growing 3D model
- **Large Overlap:** the model incorporates all of the data seen so far, therefore providing a large probable overlap to new subsequent frames. A large overlap helps to establish correspondences and the ICP to converge.

However, the critical disadvantage of this approach is:

- **Accumulation Error:** registration error that slowly grows as more and more frames are added to the model.

An example of this is shown below [5]:

This example shows an intermediate result of a real-time shape acquisition system. A user is slowly rotating a vase object in front of a projector and camera, and the system automatically registers and accumulates geometry as new data is acquired. We can see that one end of the vase doesn't align with the other side, once we go all the way around. This is a classic example of accumulation error.

The key question we will tackle in this report is this: **How can we design an efficient registration algorithm that avoids accumulation error and results in an accurate global registration?**

2 Simultaneous Registration

It's difficult for the accumulation strategy to obtain a good global registration, because once a frame is incorporated to the model, there is no provision to correct possible registration errors in the future. Therefore it's desirable to have a global correction step that is able to correct possible misregistrations in all previously registered frames. This can be accomplished by simply optimizing for the registration of all frames at once. We call this strategy "simultaneous registration."

In our setup, we have n input frames that we would like to align to a common reference frame and pose. We have three quantities to model when we are working with an articulated object:

- **Correspondences** indicating equivalent points in multiple frames,
- **Transformations** describing the motion of each frame,
- **Weights or Segmentation** that partition the object into independently moving rigid parts.

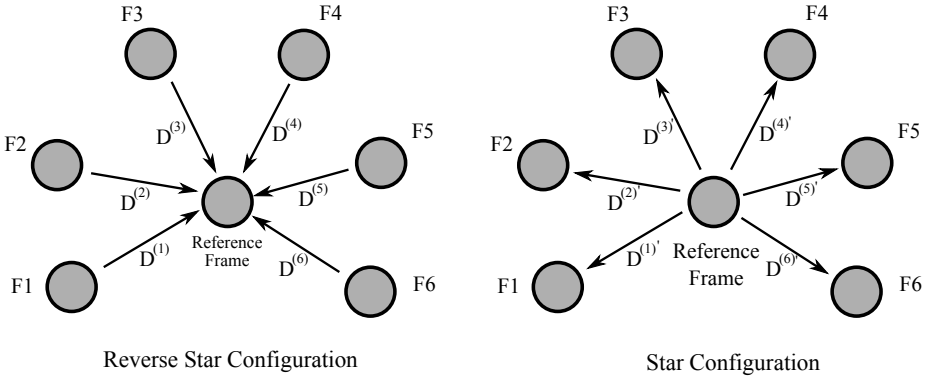
Correspondences and transformations are coupled quantities in the sense that a set of correspondences determine a transformation, and a transformation aligning two frames determines correspondences (e.g. via closest point). The strategy we would like to take is inspired by our previous work, where we optimize the correspondences and transformations in one step (called the T-step) and the

weights in another step (the W-step), and alternate between these steps until the optimization converges. The difference with our previous work is that in each step, instead of working with a pair of frames, we would like to simultaneously align multiple frames at once.

Reference weight function. In order to have a consistent weight function for all frames, we define a single weight function on the reference frame. This will have important consequences for the optimization.

2.1 Optimizing the transformations (T-step)

To solve for the transformations, we would like to alternate between estimating correspondences and transformations in ICP-like fashion. In general, we have a set of correspondence pairs between arbitrary frames $(x^{(i)}, x^{(j)})$ where $x^{(i)}, x^{(j)}$ are points in frame i and j, respectively. These correspondences are tentative, and we expect them to be updated using closest-point, normal-shooting, or projection strategies [6]. Now, the idea is to optimize the transformations in all frames. The optimization problem depends on the configuration of transformations between frames, and we would like to use the "star configuration," illustrated below.



Reverse Star Configuration

Star Configuration

We will use mainly the reverse star configuration, but also use the star configuration as it is convenient.

Distance Term. In the **reverse star** configuration, a given correspondence $(x^{(i)}, x^{(j)})$ constrains the transformation ${D^{(j)}}^{-1} \circ D^{(i)}$. This is the transformation from frame i to frame j, expressed as a composition by first applying the transformation from frame i to the reference frame using $D^{(i)}$, and then the transformation from the reference frame to frame j using ${D^{(j)}}^{-1}$. With the point-to-plane metric, we can write this constraint as

$$\left({D^{(j)}}^{-1} \circ D^{(i)}(x^{(i)}) - x^{(j)}\right) \cdot n^{(j)} = 0,$$

where $n^{(j)}$ is the normal corresponding to $x^{(j)}$ in frame j. Equivalently we can measure the error by transforming everything to the reference frame:

$$\left(D^{(i)}(x^{(i)}) - D^{(j)}(x^{(j)})\right) \cdot D^{(j)}(n^{(j)}) = 0.$$

Since we want to minimize all correspondences, we would like to solve for the motion of all frames minimizing the objective function

$$\underset{D^{(i)} \forall\, i}{\operatorname{argmin}} \sum_{\forall\, (x^{(i)}, x^{(j)})} \left(\left(D^{(i)}(x^{(i)}) - D^{(j)}(x^{(j)}) \right) \cdot D^{(j)}(n^{(j)}) \right)^2 .$$

Weight Function. The difficulty is that the weight function is defined on the reference frame, and not on each individual frame as we'd like. First, recall the LBS model:

$$D^{(i)}(x) = \sum_k w_k(x) \left(R_k^{(i)} x + t_k^{(i)} \right),$$

Here, the problem is that we cannot look up $w_k(x)$ directly using x. Instead we need to move x to the reference frame in order to look up the weight value. However, to move x to the reference frame, we need to calculate $D^{(i)}(x)$ and thus we need to know the weight value $w_k(x)$! As a result, we have kind of circular dependency as we see in the following equation:

$$D^{(i)}(x) = \sum_k w_k \left(D^{(i)}(x) \right) \left(R_k^{(i)} x + t_k^{(i)} \right).$$

How do we resolve this circular dependency? Matthias suggested an iterative solution. To look up weight values in frame i, using the current transformation, we transform the reference frame to frame i and look up the weight value in the *transformed* weight grid. Once we have these weights, we solve for the transformation, and we iterate this process until convergence. This approach would require a good initialization of the transformation, so that we do not have to extrapolate far away from the reference weight grid. We will talk about initialization later.

Numerical Solution. Writing out the entire equation, we arrive at the expression

$$\underset{R_k^{(i)}, t_k^{(i)}\, \forall i,k}{\operatorname{argmin}} \sum_{\forall\, (x^{(i)}, x^{(j)})} \left((A - B) \cdot C \right)^2 .$$

where

$$\begin{aligned}
A &= D^{(i)}(x^{(i)}) = \sum_k w_k \left(D^{(i)}(x^{(i)}) \right) \left(R_k^{(i)} x^{(i)} + t_k^{(i)} \right), \\
B &= D^{(j)}(x^{(j)}) = \sum_k w_k \left(D^{(j)}(x^{(j)}) \right) \left(R_k^{(j)} x^{(j)} + t_k^{(j)} \right), \\
C &= D^{(j)}(n^{(j)}) = \sum_k w_k \left(D^{(j)}(x^{(j)}) \right) \left(R_k^{(j)} n^{(j)} \right).
\end{aligned}$$

In order to solve this equation, we follow the solution by [3] of linearizing the rotation and solving the resulting linear system of equations, iterating according to the Levenberg-Marquardt algorithm. We could also have an additional step

projecting the linearized transformation to a rigid transformation, but we need to evaluate the weight function which is only available on the reference frame. A simpler approach would be to skip this step and directly update the 6 DOF consisting of the axis-angle of the rotation and the translation vector.

Initialization. In order to initialize the solution, we introduce each frame into the simultaneous registration in a sequential manner. Suppose that we have simultaneously registered frames from 1 through frame i. To incorporate a new frame $i+1$, we first keep the weights and all transformations fixed. Then, we solve for directly for the transformation from the reference frame to frame $i+1$, ${D^{(i+1)}}'$ (as in the star configuration), using ${D^{(i)}}^{-1}$ as the initial guess. This is solving for the simplified problem

$$\underset{{D^{(i+1)}}'}{\operatorname{argmin}} \sum_{\forall\,(x^{(j)},x^{(i+1)})} \left(\left({D^{(i+1)}}'\left(D^{(j)}(x^{(j)})\right) - x^{(i+1)}\right)\cdot n^{(i+1)}\right)^2,$$

where $0 \leq j \leq i$, and $D^{(j)}(x^{(j)})$ is a precomputed quantity using all the previously solved transformations, mapping all the correspondence source positions to the reference frame. We write this in this way as opposed to

$$\underset{D^{(i+1)}}{\operatorname{argmin}} \sum_{\forall\,(x^{(j)},x^{(i+1)})} \left(\left(D^{(j)}(x^{(j)}) - D^{(i+1)}(x^{(i+1)})\right)\cdot D^{(i+1)}(n^{(i+1)})\right)^2$$

because this would require us to additionally look up the weight function value in frame $i+1$, using the (possibly) bad initialization for $D^{(i+1)}$.

Joint constraint. Since we keep the weights fixed in this step, we can easily compute the joint constraint term and include it in the optimization. Fortunately this term doesn't involve any correspondences, so we don't have to worry about this business of looking up the weight function for certain points in certain frames.

2.2 Optimizing the weights (W-step)

In this step we keep the transformations fixed and solve for the weights. We use a discrete labeling approach as we have in our previous work: find an optimal assignment of labels to the grid cells, so that the correspondence error is minimized, and the assignment results in contiguous bones. We solve this optimization problem using graph cuts, and the key here is to specify the data term and the smoothness term. The smoothness term can be the same as before: a constant penalty when the labels at neighboring cells differ. The data term must incorporate the distance error of all frames simultaneously.

Data Term. In this term, we need to quantify how a decision to assign label l to cell c affects the distance error over all frames. Recall the distance error is:

$$\underset{D^{(i)}\forall\, i}{\operatorname{argmin}} \sum_{\forall\,(x^{(i)},x^{(j)})} \left(\left(D^{(i)}(x^{(i)}) - D^{(j)}(x^{(j)})\right)\cdot D^{(j)}(n^{(j)})\right)^2.$$

The labels represent *bones*, which in term specify a rigid transformation for each frame. Thus, by assigning a label to a cell, we are assigning a set of rigid transformations $\{D_l^{(1)}, D_l^{(2)}, \ldots, D_l^{(n)}, \}$ to the cell, one for each frame. The main question is, exactly which correspondences does cell c affect? (**Note:** I'm not 100% sure this is the right formulation, so please take this section with a grain of salt.)

We analyze this problem by considering a single correspondence $(x^{(i)}, x^{(j)})$. The cell c, along with the label l affects this correspondence if either

- the point $D_i^{(l)}\left(x^{(i)}\right)$ lies in cell c, or
- the point $D_j^{(l)}\left(x^{(j)}\right)$ lies in cell c.

Another way to think about this is to transform the grid cell c according to label l in all frames, and finding all the points contained within the transformed cell c in all frames. Finally, once we find all these affected correspondences, we can directly calculate the total distance error of assigning label l to cell c. (**Note:** this is a bit strange to think about, because the correspondences affected by the cell c will change depending on which label l we use. I'm not sure if this is entirely correct.)

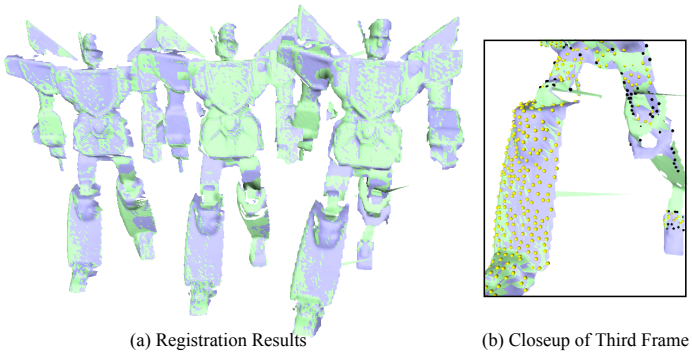
3 Robustness Issues

3.1 Reliable correspondences

A crucial step is to obtain reliable initial correspondences throughout our optimization. For this purpose, we can use the spin image feature matching + RANSAC framework that we used for our previous work. We may further improve the speed of this process, we can compare spin images on the GPU, and we can use the improved PROSAC framework [1] to obtain faster convergence.

Another strategy is to take the spectral clustering approach of Huang et al. [2] to find initial correspondences. This approach requires geodesic distances, which are not readily available in our situation. It would be interesting to investigate possible extensions.

One of the advantages of the accumulation strategy is the large overlap between the accumulated geometry and a new frame. We would also like to take advantage of the geometry of all previous frames whenever we introduce a new frame to the optimization. For this purpose, I would like to use correspondence tracks: tracked feature points that are present in multiple frames. By tracking correspondence beyond just neighboring frames, I hope to obtain correspondences even when there is little overlap between the last frame and the newly introduced frame. Figure 1 shows an example of this problem, where there's a bad registration of the right leg of the robot in the third frame, using correspondences only between neighboring frames. Geometry in the previous frames could have provided more correspondences to align the leg correctly.



(a) Registration Results (b) Closeup of Third Frame

Figure 1: Using correspondences only between neighboring frames, there is little overlap in the right leg (third frame), where most of the selected correspondences are unactivated (indicated by black points). However, more geometry of the right leg in previous frames could have been used.

3.2 Occluded bones

Sometimes we will not be able to observe particular parts of the object, and some bones may be occluded completely. In these situations, we would like to prevent the optimization from optimizing occluded bones. In addition to techniques suggested by previous work [4], we can throw out points that cause errors when they are mapped to the reference pose. Here's a list of strategies:

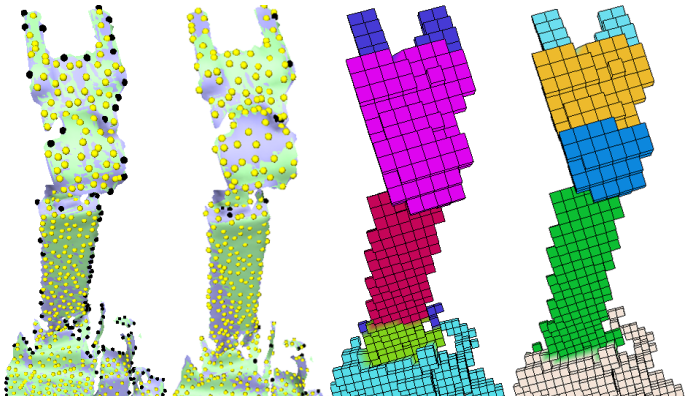
- Threshold on number of correspondences
- Threshold on distance error
- Throw out inverse mapping errors

When we decide that a bone is occluded, we exclude it from the optimization of the transformations, and rely on the joint constraint term to position the bone correctly in that frame.

3.3 Preventing over-consolidation of bones

Another issue with our current approach is that if the correspondences are undersampled, in the graph cut step the smoothness term tends to take over and consolidate two rigid parts into a single label. Although we could increase the number of correspondences or decrease the smoothness weight, these are parameters to tweak and we'd like to have a more principled approach. Here are two possible strategies:

- Including boundary correspondences with small distance (see Figure 2). Instead of rejecting all correspondences on the boundary, we include boundary correspondences when the distance is very small. This helped with parts of the toy truck example.



(a) No Boundary (b) With Boundary (c) No Boundary (d) With Boundary

Figure 2: Results with and without boundary correspondences. Black points indicate unactivated correspondences, whereas yellow points indicate activated correspondences. As a result, the graph cut segmentation is able to separate the two rigid parts more reliably, as shown on the right.

- Spatially varying smoothness weight. In our previous work we had a step to reuse labels that were not assigned at all. For regions that are consolidated, we observed that the reuse step divided these regions for refinement, but the subsequent graph cut step re-consolidated the region, resulting in this region to be split and consolidated repeatedly. Instead we could vary the weight of the smoothness term over the grid (a common practice in the vision literature) and decrease the smoothness weight for the regions that are split frequently.

4 Implementation Plan

To implement this algorithm, I plan to re-write major parts of my existing code for the multi-frame case.

- Numerical solution for multi-frame optimization: I will extend the current code to optimize for multiple frames simultaneously.
- Use per-cell labels instead of per-point weights: by doing this I hope to simplify many parts of the optimization code.
- Correspondence Tracks: implement data structure that maintains correspondences through multiple frames.

I plan on allocating at least 1 week of full-time coding (starting October 27, 2008) to implement the multi-frame registration algorithm, and I will try my best to finish the implementation within this time. In the future, I plan to implement various extensions and improvements including

- Multi-resolution adaptive grid
- PROSAC for initial correspondences
- Parallel spectral clustering for initial correspondences

A Appendix: Why work in the reverse star configuration?

Actually the two configurations are equivalent except that the direction of the transformations are reversed. If we had used the star configuration instead, we would have obtained the following distance term:

$$\underset{D^{(i)}\forall\, i}{\operatorname{argmin}} \sum_{\forall\,(x^{(i)},x^{(j)})} \left(\left({D^{(i)}}^{-1}(x^{(i)}) - {D^{(j)}}^{-1}(x^{(j)})\right) \cdot {D^{(j)}}^{-1}(n^{(j)})\right)^2.$$

Since we want to linearize the transformations using cross products for rotations, expressing the inverse of a transformation is less convenient than specifying the transformation directly, since there is no unique inverse for a cross product.

References

[1] Ondrej Chum and Jiri Matas. Matching with prosac: Progressive sample consensus. In *CVPR*, pages 220–226, 2005.

[2] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. *Computer Graphics Forum (Proceedings of SGP)*, 27(5):1449–1457, 2008.

[3] P. J. Neugebauer. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *International Journal of Shape Modeling*, 3(1/2):71–90, 1997.

[4] Yuri Pekelny and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *EUROGRAPHICS*, 27(2), 2008.

[5] Szymon Rusinkiewicz, Olaf A. Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *SIGGRAPH*, pages 438–446, 2002.

[6] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Proceedings of 3DIM*, pages 145–152, 2001.