Geometrically Stable Sampling for the ICP Algorithm

Natasha Gelfand
Stanford University
ngelfand@cs.stanford.edu

Leslie Ikemoto
Stanford University
leslie@cs.stanford.edu

Szymon Rusinkiewicz
Princeton University
smr@cs.princeton.edu

Marc Levoy
Stanford University
levoy@cs.stanford.edu

Abstract

The Iterative Closest Point (ICP) algorithm is a widely used method for aligning three-dimensional point sets. The quality of alignment obtained by this algorithm depends heavily on choosing good pairs of corresponding points in the two datasets. If too many points are chosen from featureless regions of the data, the algorithm converges slowly, finds the wrong pose, or even diverges, especially in the presence of noise or miscalibration in the input data. In this paper, we describe a method for detecting uncertainty in pose, and we propose a point selection strategy for ICP that minimizes this uncertainty by choosing samples that constrain potentially unstable transformations.

1. Introduction

When building three dimensional models using a range scanner, multiple views are usually required due to the limited field of view of the scanner and the presence of occlusions. Registration of these views is typically performed pairwise, using a variant of the Iterative Closest Point algorithm (ICP) [3, 2]. This algorithm starts with two meshes and an initial estimate of the aligning rigid-body transform. It then iteratively refines the transform by alternately choosing corresponding points in the meshes and finding the best translation and rotation that minimizes an error metric based on the distance between them.

Since ICP is a non-linear local search algorithm, it suffers from many problems commonly associated with local searches, such as slow convergence (due to shallow error landscapes) and the tendency to fall into local minima. The point selection strategy and the choice of error metric to be minimized play a large role in both the rate of convergence and the accuracy of the resulting pose. A discussion of these issues can be found in [11].

Poor alignment between a pair of meshes can come from several sources. Noise in the input data can cause ICP to converge to a local minimum. The frequency of local minima in the error landscape depends on input geometry and on the minimized distance metric. The point-to-plane error metric of Chen and Medioni [3] makes the ICP algorithm less susceptible to local minima than the point-to-point metric of Besl [2]. Pottman and Hofer [9] show that if the two meshes are close to each other, the point-to-plane distance is the best approximation for the true distance between the two surfaces. This metric also has an advantage that it allows the two surfaces to “slide” against each other in the flat and spherical regions, which do not contain enough information to fully constrain the transform. However, if too many point-pairs come from such featureless regions, the algorithm can fail to converge because of lack of constraints. In this case, the cause of poor convergence and poor final pose is the shallow error landscape that results from too much sliding. We will call geometry that does not have enough constraints for good convergence “unstable.”

In most 3D scanning systems, pairwise registration is usually followed by a global relaxation algorithm [10], which spreads the accumulated alignment error over a set of views. Since a single mesh usually has several partners in this set, poor pose for one mesh can easily be propagated to its partners. Even if two views are aligned correctly, a shallow error landscape around the minimum can cause them to be pulled apart during global relaxation. Finally, if the output surface model is to be reconstructed from the input views by some sort of averaging [4], misaligned features can become blurred. For all these reasons, we would like the final pose to be both correct and well-constrained.

Several methods have been proposed for evaluating and improving the stability of the final pose between two meshes. Once a set of point-pairs has been selected, the presence of sliding can be detected by analyzing the covariance matrix used for error minimization [13, 12, 5]. The chosen point set can then be altered to provide the best constraints for the final pose. Guehring [5] addresses the problem of maximizing stability of the transform by assigning weights to existing point-pairs based on their contribution to the covariance matrix. However, since stability analysis is performed after the point-pairs have already been chosen, this reweighting may not constrain sliding since not enough constraining points may have been chosen in the first place. Simon [12] has developed several algorithms for iteratively adding and removing point-pairs to provide the best-conditioned covariance matrix. We will discuss his approach in Section 3.

We propose a technique for identifying whether a pair of meshes will be unstable in the ICP algorithm by estimating the covariance matrix from a sparse uniform sampling of the input. We then develop a sampling strategy that tries to minimize this instability by drawing a new set of sample points primarily from stable, i.e. "lock and key", areas of the input meshes. This technique extends the normal space sampling proposed by Rusinkiewicz and Levoy [11]. Unlike [11], our approach deals with both translational and rotational uncertainties in registration.

2. Geometric Stability of ICP

In this section we describe a method based on 6x6 covariance matrixes for determining if a pair of meshes will be unstable if aligned using a point-to-plane error metric. This discussion is similar to the analysis of Menq [8] and Simon [12].

2.1. Error Minimization

Each iteration of the ICP algorithm proceeds as follows. Let P and Q be two meshes or two point sets with associated normals. These normals can either be computed by averaging the normals of adjacent faces (for a mesh), or can be provided externally (if no connectivity information is available). A set of points is chosen on P, and for each point the corresponding closest point is found on Q. This forms a set of k point-pairs $(\mathbf{p}_i, \mathbf{q}_i)$, where each $\mathbf{q}_i$ has normal $\mathbf{n}_i$. (In many implementations, source and target meshes are then exchanged and the point selection is repeated. Here we will use only one mesh as the source to simplify the presentation.) We then try to find a rigid-body transformation, composed of a rotation R and a translation $\mathbf{t}$, that minimizes the sum of squared distances of each $\mathbf{p}_i$ to the plane tangent to Q at $\mathbf{q}_i$. The alignment error is given by:

$$E = \sum_{i=1}^{k} ((R\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i) \cdot \mathbf{n}_i)^2 \qquad (1)$$

If the rotation that minimizes E is small, Equation 1 can be solved by linearizing the rotation matrix R. This is equivalent to treating the transformation of each point $\mathbf{p}_i$ as a displacement by a vector $[\mathbf{r} \times \mathbf{p}_i + \mathbf{t}]$, where $\mathbf{r} = (r_x, r_y, r_z)$ is a (3 × 1) vector of rotations around the x, y, and z axes, and $\mathbf{t} = (t_x, t_y, t_z)$ is the translation vector. Substituting and expanding, we therefore wish to find a 6-vector $[\mathbf{r}^T \ \mathbf{t}^T]$ that minimizes:

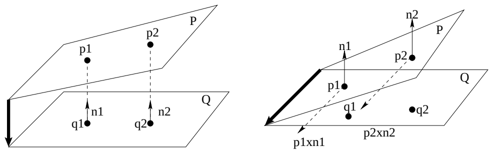

Figure 1: (a) Force vectors (dashed) exerted by points in Q on points in P. Resulting translation vector is in bold. (b) Torque vectors (dashed) exerted by points in Q on points in P. Q's normals are attached to points on P for clarity. Resulting rotation vector is in bold.

$$E = \sum_{i=1}^{k} ((\mathbf{p}_i - \mathbf{q}_i) \cdot \mathbf{n}_i + \mathbf{r} \cdot (\mathbf{p}_i \times \mathbf{n}_i) + \mathbf{t} \cdot \mathbf{n}_i)^2 \qquad (2)$$

To understand the terms of Equation 2, we can imagine that each pair $(\mathbf{p}_i, \mathbf{q}_i)$ applies to P a translational "force" in the direction of $\mathbf{n}_i$ and a rotational "torque" around the axis $\mathbf{p}_i \times \mathbf{n}_i$ (Figure 1).

The last two terms of Equation 2 show that the amount by which the point-plane distance will change if a given point-pair i is moved by a transformation vector $[\Delta\mathbf{r}^T \ \Delta\mathbf{t}^T]$ is given by:

$$\Delta d_i = [\Delta\mathbf{r}^T \ \Delta\mathbf{t}^T] \begin{bmatrix} \mathbf{p}_i \times \mathbf{n}_i \\ \mathbf{n}_i \end{bmatrix} \qquad (3)$$

From Equation 3, we see that points whose corresponding normal vectors $\mathbf{n}_i$ are perpendicular to $\mathbf{t}$, or whose torque vectors $\mathbf{p}_i \times \mathbf{n}_i$ are perpendicular to $\mathbf{r}$ do not change the error E (Figure 2 (a),(c)). In a more general setting, Δd_i is zero for point-pairs whose 6-vector of torques and forces is orthogonal to the transformation vector.

We solve for the aligning transform by taking partial derivatives of Equation 2 with respect to the transform parameters. This results in a linear system $C\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is the 6 × 1 vector of transformation parameters, $\mathbf{b}$ is the residual vector, and C is a 6 × 6 covariance matrix of the "torque" and "force" components contributed by each point pair:

$$C = FF^T = \qquad (4)$$

$$\begin{bmatrix} \mathbf{p}_1 \times \mathbf{n}_1 & \dots & \mathbf{p}_k \times \mathbf{n}_k \\ \mathbf{n}_1 & \dots & \mathbf{n}_k \end{bmatrix} \begin{bmatrix} (\mathbf{p}_1 \times \mathbf{n}_1)^T & \mathbf{n}_1^T \\ \dots & \dots \\ (\mathbf{p}_k \times \mathbf{n}_k)^T & \mathbf{n}_k^T \end{bmatrix}$$

The matrix C encodes how much the alignment error will change when the mesh P is moved from its optimum alignment to Q (where the error is 0) by the transformation $[\Delta\mathbf{r}^T \ \Delta\mathbf{t}^T]$:

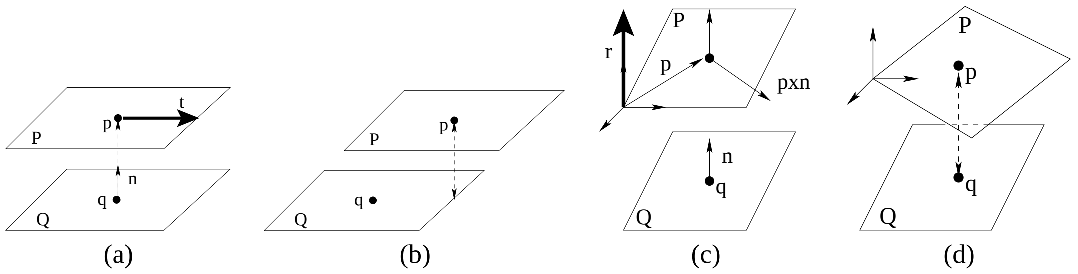

Figure 2: Two unstable surfaces sliding against each other. (a)-(b) Point-pairs whose corresponding normals $\mathbf{n}$ are orthogonal to translation direction $\mathbf{t}$ maintain the same point-plane distance. Thus, this point pair exerts no constraint on translation $\mathbf{t}$. (c)-(d) Point-pairs whose torques (along $\mathbf{p} \times \mathbf{n}$) are orthogonal to the rotation vector $\mathbf{r}$ also maintain the same point-plane distance. Thus, this point pair exerts no constraint on rotation by vector $\mathbf{r}$.

$$\Delta E = \left[\Delta \mathbf{r}^T \; \Delta \mathbf{t}^T\right] C \left[\begin{array}{c} \Delta \mathbf{r} \\ \Delta \mathbf{t} \end{array} \right] \tag{5}$$

The transformations for which this increase is comparatively small correspond to directions where the input meshes can slide relative to each other.

2.2. Stability of the Solution

Certain types of geometry lead to a covariance matrix that is not full rank, which means that the minimizing transform is not unique. The simplest example is two planes. Without loss of generality, suppose they are parallel to the xy-plane. Once the planes are aligned with each other, there are still three degrees of freedom: translation in the xy-plane of one plane relative to the other and rotation around the z axis. Neither of these transformations changes the point-plane alignment error (Figure 2 (b), (d)). This corresponds to our intuitive notion that there is not enough information in the input data to fully constrain all the motions. Other combinations of unconstrained rotations and translations are possible. Figure 3 shows a familiar shape for each possible combination of unconstrained rotations and translations. Each of these shapes will exhibit sliding when aligned to a copy of itself.

We can identify the unconstrained transformations by expressing C in terms of its eigenvalues and eigenvectors. If any of the eigenvalues are small, the corresponding eigenvector defines a transformation that can move two meshes from their optimum alignment with only a small increase in error.

Sliding between a pair of meshes can occur even if the input has enough features to constrain most motions. An example is two planar regions with indentations or incisions. Examples of such input in shown in Figure 4. If the size of these “lock and key” features is small and only a subset of the mesh points are used in the alignment algorithm, most of the points used in the registration will come from areas that are planar. If the data has no noise, the small number of points from the “lock and key” areas should be sufficient to resolve the ambiguity in the transform and bring the meshes into alignment. In reality, noise in point positions and normals in the flat areas will overwhelm the contribution of the points sampled from the features, and the algorithm will fail to converge.

There are several ways to approach the problem of sliding. We can try to reduce the noise by smoothing the meshes. This can have an undesirable side effect of smoothing away the features that provide the valid constraints. We can try to use other constraints, such as color [1, 15]. We can also add more points to be used for minimization of Equation 2. Just adding more points will not improve convergence, since they are as likely to come from the flat areas as from the parts of the meshes that provide the constraints. We would like, therefore, to be able to detect whether the input data has any rotational or translational instability, identify if there are any features that can better constrain the unstable transformations, and sample those features more densely.

3. Improving ICP’s Stability Through Sample Selection

In this section we describe a greedy algorithm for selecting samples from the input meshes in a way that will constrain transformations which have small associated error change under the uniform sampling model.

The two techniques that are the most similar to our approach are those of Simon [12] and Rusinkiewicz [11]. Simon developed several hill climbing algorithms for selecting a set of points on one of the input meshes that has the best potential for constraining all transformations when another mesh is aligned with it. These algorithms are especially well-adapted for dealing with noisy data, but do not address the problem when matching areas are only a subset of the input meshes. They are also designed for cases when only a very small number of points is required for alignment. As a result, they are too expensive to be used when large number of points are to be selected for minimization.

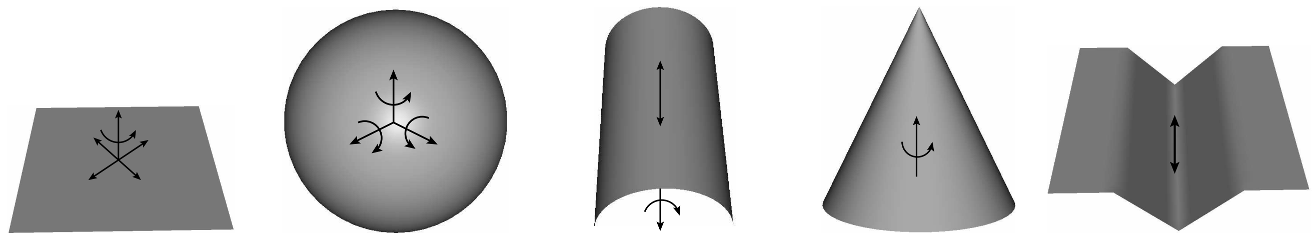

Figure 3: Some examples of simples shapes that are unstable. For each shape, the corresponding covariance matrix will have some number of small eigenvalues, and for those, the corresponding eigenvector specifies the direction of instability. Below each figure, the number and types of the instabilities are noted. A helix, which has one unstable screw motion, is missing, but helical shapes are not likely to arise in scanned data.

Rusinkiewicz [11] proposed a technique called normal-space sampling that is aimed at constraining translational sliding of the input meshes. When drawing samples from a mesh, the algorithm tries to ensure that the normals of the selected points uniformly populate the sphere of directions. The algorithm can be viewed as trying to equalize the eigenvalues of eigenvectors of C that correspond to translations. We will use a similar approach to create a basis all six eigenvectors of C.

3.1. A Measure of Stability

In previous sections we showed that the point-to-plane error metric is susceptible to sliding in the presence of too few constraints. When the two input meshes are far away from each other, this sliding can help us by preventing the algorithm from getting stuck in a wrong local minimum. However, we do not want the meshes to slide once they get close to their correct alignment. The goal of our sampling strategy, therefore, is to select samples that will constrain the transformations when the alignment gets close to the correct pose.

As discussed in Section 2.2, we can detect if the chosen set of point-pairs has any unconstrained transformations by examining the eigenvalues of the covariance matrix C. Let $\mathbf{x}_1 \ldots \mathbf{x}_6$ be the eigenvectors of C with the corresponding eigenvalues $\lambda_1 \geq \ldots \geq \lambda_6$. Each eigenvector corresponds to a general screw motion that can be described as a rotation around an axis and a translation along that axis. If any of the λ_k are small compared to λ_1, the corresponding eigenvector corresponds to a sliding direction. Our measure of stability, therefore, will be the condition number of the matrix C: $c = \frac{\lambda_1}{\lambda_6}$. The goal of the sampling strategy is to keep c as close to 1 as possible.

However, the part of the transformation vector $[\mathbf{r}^T \ \mathbf{t}^T]$ that corresponds to rotation depends on the term $\mathbf{p}_i \times \mathbf{n}_i$. This means rotations are dependent on distance of the point $\mathbf{p}_i$ from the origin (which is the center of rotation when $\mathbf{r}$ is applied to each $\mathbf{p}_i$). As is common with PCA methods, we will shift the center of mass of the points $\mathbf{p}$ to the origin. However, the magnitude of rotations can still be incompatible with the magnitude of translations, since a point $\mathbf{p}_i$ can be arbitrarily far from the center of mass. Therefore, after shifting the center of mass, we will scale the point set so that the average distance of points $\mathbf{p}_i$ from the origin is 1. This has the effect of equalizing the maximum amount of displacement that can be contributed by a point due to "torque" (i.e. rotation) to the amount of displacement due to "force" (i.e. translation).

Finally, we only want to add those constraints that will pull the meshes to the alignment that is the global optimum. That is, we want to make the error landscape around the global minimum steep, while keeping the landscape shallow around the local minima to allow the algorithm to escape. The global minimum is achieved when the points in P align exactly with their correct mates in Q. In this case the normals in Equation 2 are the same for points $\mathbf{p}_i$ and $\mathbf{q}_i$. Therefore, to constrain the correct transformations, we should analyze and constrain the covariance matrix that is computed using both points and normals from the mesh P. We will call this matrix C_P.

3.2. Optimizing the Measure

As discussed above, our measure of stability is the condition number c of the matrix C_P. In order to optimize our stability measure, we first need an estimate of what the eigenvectors of the linear system would be if uniform sampling were used. Given a single mesh, we can directly compute its covariance matrix using Equation 4, where the $\mathbf{n}_i$ associated with the points $\mathbf{p}_i$ come from the mesh P. However, in a registration problem only those points that lie in the overlap between two meshes should contribute to the matrix computation. Thus, we may obtain the estimate of the covariance matrix as follows:

A1 Let S_P be a set of points randomly selected from P. The size of S_P should be chosen so that once the points outside of the overlap area are discarded, there are still enough points to reliably determine the covariance matrix for the overlap region. The number of points depends on the size of the overlap region between the two meshes, the resolution of the mesh, and the magnitude of noise in the input data. In our experience with the Forma Urbis Romae dataset [7], for meshes that overlap by 25%, the number of points necessary for the eigenvectors to stabilize is on the order of several hundred.

A2 For each $\mathbf{p} \in S_P$, we need to determine whether it belongs to the overlap area. We find its closest point $\mathbf{q}$ in Q and check if it lies on the boundary of the mesh Q. If $\mathbf{q}_i$ belongs to the boundary, then $\mathbf{p}$ is outside the overlap area [14] and we discard it. Otherwise, it is added to the set of overlap points O_P.

A3 We form the covariance matrix C_P of the points in O_P and compute its eigenvectors $\mathbf{x}_1 \ldots \mathbf{x}_6$. We compute C_P according to Equation 4, but use both points and normals from P.

We now use these computed eigenvectors to obtain a better sampling of the mesh P in the overlap region.

B1 Let S be the initial set of candidate points. Ideally, S will contain all points on P that belong to the overlap area. We will discuss how to obtain the set S later. Form a 6-vector $\mathbf{v}_i = [\mathbf{p}_i \times \mathbf{n}_i^P, \mathbf{n}_i^P]$ for each point in S. Notice that here $\mathbf{n}_i^P$ is the normal of the point $\mathbf{p}_i$ as opposed to the normal of its closest point mate.

B2 Form six sorted lists $L_1 \ldots L_6$. Each list L_k contains the vectors $\mathbf{v}_i$ sorted in decreasing order based on the magnitude of the dot product $\mathbf{v}_i \cdot \mathbf{x}_k$. The magnitude of this dot product determines how much a given point constrains each eigenvector $\mathbf{x}_k$. Hence, points in each list are sorted in order of decreasing contribution to geometric stability.

B3 We now try to equally constrain all eigenvectors of C_P. We will maintain an estimate of how each eigenvector is constrained by the already chosen points. Let $t_1 \ldots t_6$ be the sums of $(\mathbf{v}_i \cdot \mathbf{x}_k)^2$ over the already chosen points. $(\mathbf{v}_i \cdot \mathbf{x}_k)^2$ is the amount of error incurred if the point $\mathbf{p}_i$ is moved from its optimum position by the transformation $\mathbf{x}_k$. Therefore, we can think of these totals as our current estimate of the eigenvalues. We choose the next point from the sorted list that has the smallest total. This corresponds to the most unconstrained eigenvector.

B4 Let $\mathbf{p}$ be the chosen point. We compute $(\mathbf{v}_i \cdot \mathbf{x}_k)^2$ for each eigenvector $\mathbf{x}_k$ and update the running totals.

Notice that this sampling strategy does not take into account the mesh Q. We can think of this strategy as constraining the all transformations when P is aligned to a copy of itself in the overlap region. Assuming that we are aligning two ideal, overlapping scans of the same object, this exactly corresponds to constraining the covariance matrix when we reach the global minimum. Once the points are sampled from P, we can compute their closest points in Q. We then proceed with the rest of the ICP algorithm as usual, now using the normals from Q for the minimization of Equation 2.

4. Accelerations and Enhancements

The sampling algorithm as it is described above contains two sources of inefficiency.

First, in Step B2, we have to perform 6 sorts of the set of vectors formed in Step B1. To reduce the cost of these sorts, we instead sort the points into a specified number of bins. The points are left unsorted within each bin. Although not optimal, this still produces a good sampling, and the approximation error can be bounded by the size of the bins. Thus, the second step can be done in time proportional to $|S|$.

Second, in Step B1, we need to form the set S of points in the overlap between P and Q. A brute-force approach would be to test each point in P for overlap with Q as described in Step A2. Even using an efficient nearest-neighbor data structure such as a k-d tree, this can be expensive for large meshes. It can also be wasteful if we only intend to use a small set of points for computing the aligning transform.

A simple improvement that we implemented in our system is to process all the points in P regardless of whether they are in the overlap area. This allows us to delay the overlap test until Step B3. At that time, we can perform this test the same way as in Step A2 of the matrix estimation algorithm. If $\mathbf{p}_i$ does not belong to the overlap area, we do not update the totals and choose the next point. This method is more efficient than the brute force approach (if we use fast sorting), since we perform only as many nearest-neighbor tests as dictated by the sampling rate. We can use the closest points computed as a result of the overlap test for minimization of Equation 2. In practice, this makes the amount of wasted work inversely proportional to the size of overlap region. With this implementation, ICP using our sampling strategy takes about 5 times longer per iteration than ICP using uniform sampling when input meshes overlap by half their area.

A faster solution is to use the set of points computed in by the initial eigenvector estimation (Steps A1-A3) to generate more points in the overlap area. We can generate more such points by crawling the mesh P starting as these seed points. If we are given a point cloud with no connectivity, we can crawl the k-d tree used for the closest point computation instead. This allows us to quickly generate a large set of points in the overlap area for the set S and avoids wasting work performing nearest-neighbor tests for points that are clearly outside the overlap area. The small number of points outside the overlap area generated by this method can be discarded when their mates in Q are determined for error minimization. This optimization makes our algorithm

3 times slower per iteration than conventional ICP. We expect that with a more careful implementation we can make our sampling strategy perform comparably to ICP with uniform sampling.

5. Results

We have applied our sampling algorithm to several types of synthetic and real data.

The first test case is two planar patches with two grooves forming an X (Figure 4). Each patch has independently added Gaussian noise. This test case is similar to the one used by Rusinkiewicz [11] for normal-space sampling. Figure 5 shows the convergence rates for aligning these patches using uniform sampling, normal-space sampling, and our covariance-based sampling. Both normal-space and covariance sampling are able to find the correct alignment, while uniform sampling does not align the grooves correctly. Normal-space sampling takes more iterations to converge since distributing the points equally throughout the sphere of normals puts an equal number of points in the flat areas of the patches as it does in the grooves. Covariance sampling instead picks only those points that form a good basis for the normals.

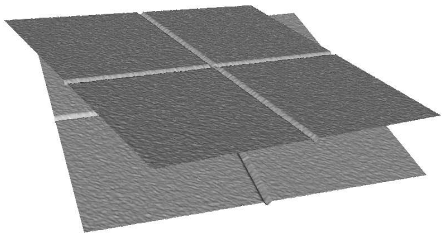

Figure 4: Two planar patches with 1 mm deep grooves. Each patch has independently added zero-mean Gaussian noise with variance 0.05 mm. Initial condition number is 66.1. Condition number after selecting 30% of the points with our algorithm is 3.7.

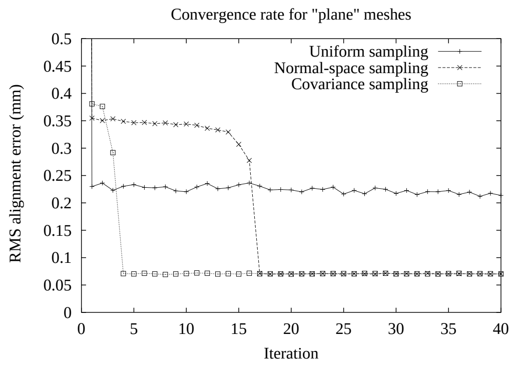

Figure 5: Convergence rates for "incised plane" meshes for uniform, normal-space and covariance sampling.

Figure 8 shows the points picked by the sampling algorithm to constrain the eigenvectors of the covariance matrix. To simplify the visualization, we use a smaller version of the incised plane model and assume that the entire mesh is within the area of overlap. The initial covariance analysis reveals three unstable eigenvectors with approximately equal eigenvalues: two translations in the xy plane and rotation around z. Notice that most of the points are picked from the areas in the grooves, since they are the ones that constrain the unstable eigenvectors. A few points from the corners are picked to additionally stabilize the rotations around the diagonals.

Figure 6 shows two spherical patches with grooves and noise. Here, covariance sampling in the only method that finds the pose that correctly aligns the grooves (Figure 7).

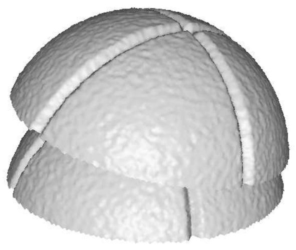

Figure 6: Two spherical patches with 1 mm deep grooves. Each patch has independently added zero-mean Gaussian noise with 0.05 mm variance. This dataset has three unstable rotations. Initial condition number is 26.9. Condition number after selecting 30% of the points with our algorithm is 4.1.

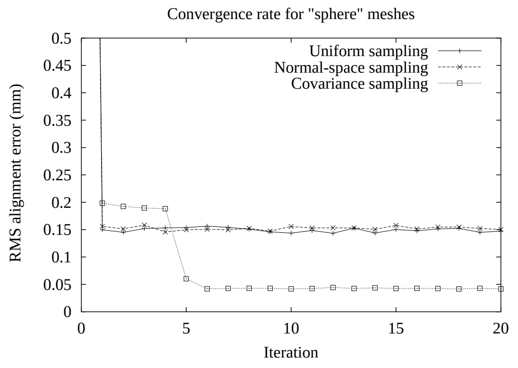

Figure 7: Convergence rates for "incised sphere" meshes for uniform, normal space and covariance sampling

We have also applied our algorithm to real scan data. Figure 9(a) shows the sampling of two scans from the Forma Urbis Romae dataset [7]. Similar to the "incised plane" example, these meshes exhibit translational sliding in the plane and rotational sliding around the vector perpendicular to the plane of the meshes. Most of the samples are placed into the incisions on the scans to constrain the scans from sliding and rotating in their common plane. It took

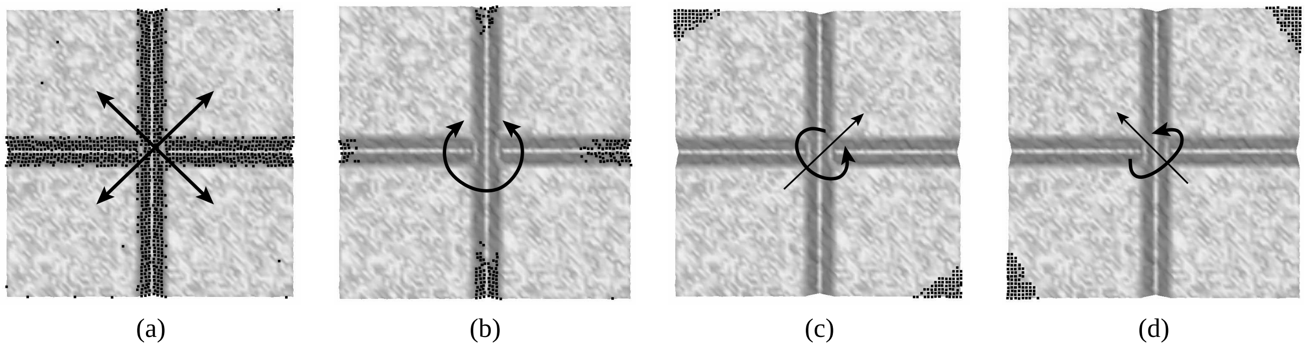
Figure 8: Points picked by our sampling algorithm for a patch with two grooves. (a) Points constraining two unstable translational eigenvectors. (b) Points constraining the unstable rotation. (c)-(d) Two remaining rotations are stable so they only require a few points. The eigenvector corresponding to translation in z is well constrained by the already picked points and does not contribute to the sampling.

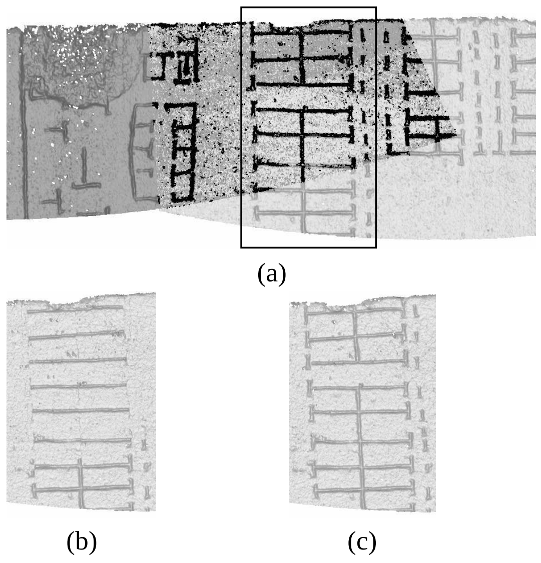
Figure 9: Aligning two scans of Forma Urbis Romae fragment 033abc. (a) Points selected by our sampling strategy are in black. Notice that in the outlined region there are relatively fewer constraints to prevent horizontal sliding than vertical sliding. (b) Therefore uniform sampling cannot align the vertical grooves in the outlined region as evidenced in this Z-buffer rendering of the two meshes by the fact that the vertical grooves are obscured. (c) Covariance sampling produces the correct alignment making all the grooves visible.

ICP 25 iterations to converge to the correct alignment (Figure 9(c)) from a rough manual positioning of the scans using our sampling strategy. Each input mesh contains about 300,000 points, and the algorithm was subsampling 10% of the points from each mesh to be used in alignment. With these settings, each iteration of ICP using our stable sampling took 5 seconds on a 400MHz Pentium II. One iteration of ICP using uniform sampling took 1.5 seconds, however when started from the same position, uniform sampling is unable to correctly align the vertical grooves (Figure 9(b)).

We have performed some initial experiments with using the output of geometrically stable ICP in the global relaxation algorithm of Pulli [10] using the Forma Urbis Romae dataset. The results seem to suggest that scans that are aligned pairwise using our sampling strategy "hold together" better than those aligned using uniform sampling. Figure 10 shows the residual error between the pair of scans from the Forma Urbis dataset examined above after the entire set of views has been processed by the global relaxation algorithm. Scans aligned with uniform sampling (Figure 10 (a)) have been pulled apart by as much as a millimeter, while those aligned by our algorithm (Figure 10 (b)) stayed together. A system for global registration of meshes that uses our sampling is presented in a companion paper [6].

We also investigated the influence of noise on the performance of our sampling strategy. Since the algorithm prioritizes the points based on their influence of the covariance matrix, it is possible that it can favor areas with significant noise, since the points there can look like good features for the algorithm to sample. Smoothing the input data can eliminate some of the false features and improve the sampling. However, if the meshes are smoothed too much, the sampling algorithm can still fail since the true features will be smoothed away. Figure 11 shows success and failure cases of smoothing the noisy data to improve the sampling.

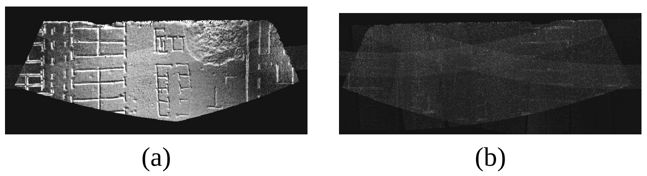
Figure 10: A visualization of residual error in the overlap portion of the pair of scans in Figure 9 after they and their partners have been processed by Pulli's global registration [10]. Meshes in (a) were aligned using uniform sampling. Meshes in (b) using our geometrically stable algorithm. Error is in mm, black corresponds to 0, white to 1. The maximum error in (a) is over 1 mm, while the maximum error in (b) is 0.3 mm.

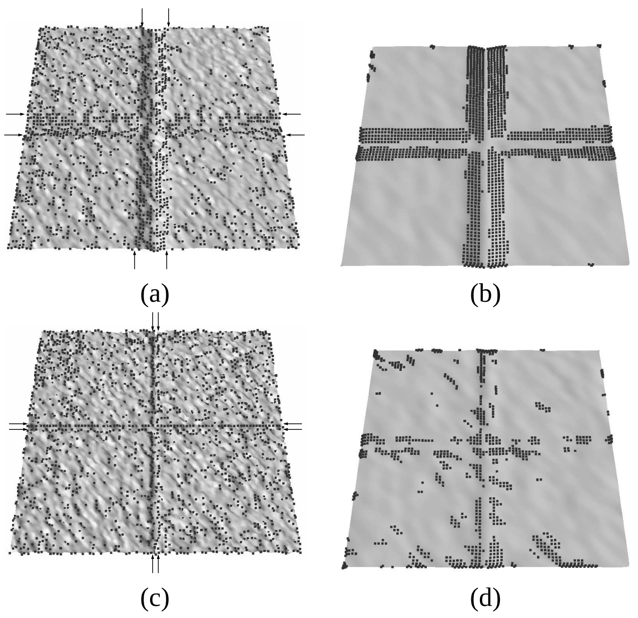

Figure 11: Effect of noise on covariance sampling. (a) A noisy patch with a cross in the center. The width of the grooves, indicated by black arrows, is 10 mm, the depth is 1 mm, the mean height of the noise is 0.2. Since the groove is shallow, the normals of points in the groove are comparable to normals of the noisy flat areas and the algorithm cannot distinguish between features and noise. (b) Performing 6 iterations of simple smoothing by averaging neighbors removes most of the noise but keeps the feature. (c) A similar patch, but the width of the groove is only 1 mm. (d) Since the size of the feature is comparable to the size of the noise, smoothing removes the noise and most of the feature, which means all areas of the patch now look identical, and covariance-based sampling fails.

6. Conclusions

We have presented a point selection strategy that improves geometric stability of the ICP algorithm. This technique is aimed at sampling those features of the input meshes that provide the best convergence of the algorithm to the correct pose. The sampling strategy is based on estimating the transformations that can cause unstable sliding in the ICP algorithm and picking points that best constrain this sliding.

Several directions are possible for future work. The current technique treats all eigenvectors of the covariance matrix the same and tries to constrain them equally. However, the geometry of the input meshes outside the overlap area can have an effect on how we want to constrain the transformations within the overlap area. In particular, if a mesh extends far beyound the overlap area, small misalignments in rotation can become amplified. We would like to investigate sampling methods that take this leverage into account, and in general are able to assign different weights to different eigenvectors.

While we only address the stability of pairwise alignment of meshes in this paper, a similar stability analysis can be applies to a larger collection of meshes, e.g. to the global relaxation step of the mesh alignment pipeline. Point selection for maximizing stability of a large set of scans is substantially more difficult than the pairwise step, since we have to consider how sliding of a single scan pair will affect the entire system. We also discuss this issue in more detail in a companion paper [6].

Acknowledgements. This work was supported by the National Science Foundation Research Grant IIS-0113427. The datasets used in this work were provided by the Digital Michelangelo and Forma Urbis Romae projects. Thanks to Daniel Russel and Leo Guibas for useful discussions.

References

[1] F. Bernadini, I. Martin, and H. Rushmeier. High-quality texture reconstruction from multiple scans. *IEE Trans. on Visualization and Computer Graphics*, 1991.

[2] P. Besl and N. McKay. A method for registration of 3-D shapes. *Trans. PAMI*, 14(2), 1992.

[3] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Proc. IEEE Conf. on Robotics and Automation*, 1991.

[4] B. Curless and M. Levoy. A volumnetric method for building complex models from range images. *Proc. SIGGRAPH'96*, 1996.

[5] J. Guehring. Reliable 3D surface acquisition, registration and validation using statistical error models. *Proc. 3DIM*, 2001.

[6] L. Ikemoto, N. Gelfand, and M. Levoy. A hierarchical method for aligning warped meshes. *Proc. 3DIM 2003*.

[7] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3-D scanning of large statues. *Proc. SIGGRAPH*, 2000.

[8] C.-H. Menq, H.-T. Yau, and G.-Y. Lai. Automated precision measurement of surface profile in CAD-directed inspection. *IEEE Trans. on Robotics and Automation*, 1992.

[9] H. Pottman and M. Hofer. Geometry of the squared distance function to curves and surfaces. *Vienna Institute of Technology Technical Report No.90*, January 2002.

[10] K. Pulli. Multiview registration for large datasets. *Proc. 3DIM*, 1999.

[11] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. *Proc. 3DIM*, 2001.

[12] D. Simon. *Fast and Accurate Shape-Based Registration*. Ph.D Dissertation. Carnegie Mellon University, 1996.

[13] A.J. Stoddart, S. Lemke, A. Hilton, and T. Renn. Estimating pose uncertainty for surface registration. *Proc. British Machine Vision Conference*, 1996.

[14] G. Turk and M. Levoy. Zippered polygon meshes from range images. *Proc. SIGGRAPH*, 1994.

[15] S. Weik. Registration of 3-D partial surface models using luminance and depth information. *Proc. 3DIM*, 1997.