Simultaneous Registration of Multiple Range Views Satisfying Global Consistency Constraints For Use In Reverse Engineering*

David W. Eggert   Andrew W. Fitzgibbon   Robert B. Fisher

Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland EH1 2QL

eggertd, andrewfg or rbf@aifh.ed.ac.uk

Abstract

When reverse engineering a CAD model, it is necessary to integrate information from several views of an object into a common reference frame. Given a rough initial alignment of local 3-D shape data, further refinement is achieved using an improved version of the recently popular Iterative Closest Point algorithm. Data correspondence is determined by considering the merging data sets as a whole. A potentially incorrect distance threshold for removing outlier correspondences is not needed as in previous efforts. Incremental pose adjustments are computed simultaneously for all data sets, resulting in a more globally optimal set of transformations. Individual motion updates are computed using force-based optimization, with the data sets considered as connected by groups of springs. Experiments on both 2-D and 3-D data sets show that convergence is possible even for very rough initial positionings, and that the final registration accuracy typically approaches less than one quarter of the interpoint sampling resolution of the views.

Keywords: 3-D surface registration, Iterative Closest Point algorithm, motion estimation

Abbreviated title: Simultaneous Registration of Multiple Range Views

Address for correspondence: Robert Fisher
Department of Artificial Intelligence
University of Edinburgh
5 Forrest Hill
Edinburgh, Scotland EH1 2QL
Phone: (44) (0131) 650 3098
Fax: (44) (0131) 650 6899

*This work was funded by UK EPSRC Grant GR/H/86905.

Introduction

With the increased abilities of Computer Aided Manufacturing (CAM) systems over the past decade, it is becoming more and more desirable for manufacturing companies to develop computer databases of their inventory. Computer models of components can now be directly used by numerically-controlled milling machines in the creation process. While most new parts are now created using a Computer Aided Design (CAD) system, the need for models of old parts still exists. Generating these by hand is a time-consuming and tedious measurement task, even with the aid of computer-controlled coordinate measurement machines.

The solution is an automated process to reverse engineer the part. This procedure is generally composed of three basic steps; data acquisition, data registration and model estimation. Currently, local shape data of an object is usually acquired in the form of 3-D range images, either from a stereo camera setup, a variety of laser-based range finder systems, or from tactile probes. In order to build a complete model, multiple views of the object must be taken to achieve total coverage of the surface. The data in these views must then be aligned in a common coordinate system. The final step generates an approximation of the object's surface from the aligned 3-D points, typically in the form of a triangulated mesh, extracted quadric surfaces, or a potentially smoother spline representation.

The quality of the final model is directly related to the inaccuracies of each step: the sampling resolution and inherent noise in each sampled data point, the error in the computed alignment transformations, and the residual error of a surface fit to the combined data. It can be argued that the second of these three steps is perhaps most crucial to the final quality. The sampling accuracy of the range sensor establishes an error baseline, which can be expected to be lowered over time with advances in technology. The final surface approximation is generally controlled by either the desired number of triangular patches in the case of a mesh, or an increase in the order of the surface being fit to the data. Regardless of these error levels, if the surface detail from the various views is not properly aligned, one merely gets a finer approximation to an incorrect set of data.

The view registration process itself consists of two basic steps; generating initial estimates of the alignment transformations, and then refining these estimates. The first step is often accomplished by aligning a small set of features computed from the data (e.g., edges, surface patches, and high curvature points). This was often the only alignment done in early systems. But the quality then depends on finding a good and accurate set of features. Also, by its very nature, this ignores vast amounts of helpful information in the data set. Thus, additional techniques that use the underlying point data have been developed recently to further improve upon the initial estimates.

In this paper a point-based data registration refinement process is examined. In the next section previous techniques are reviewed and their limitations discussed. Then an improved algorithm, designed to overcome many of these limitations, is presented. This technique simultaneously solves for the interview transformations using more global correspondence constraints. Results of processing both 2-D and 3-D data sets using this algorithm are then given.

Registration Using the ICP Algorithm

A recently popular method of refining a given registration is the iterative closest point (ICP) algorithm, first introduced by Besl and McKay [1]. The algorithm is relatively straightforward. First, given a motion transformation that initially aligns two data sets to some degree, a set of correspondences is developed between features (usually points) in one set and the next. This is done using the simple metric: for each point in the first data set, pick the point in the second which is closest to it under the current transformation. From this set of correspondences an incremental motion can be computed which further aligns these points to one another. This find correspondence/compute motion process is iterated until some convergence criterion indicating proper alignment is satisfied.

Given the algorithm's simplicity, it performs quite well. But there are two major drawbacks. First, it is not obvious how the two set approach can be extended to handle multiple data sets. Second, proper convergence only occurs if one of the data sets is a subset of the other. The presence of points in each set that are not in the other leads to incorrect correspondences, which subsequently generates non-optimal transformations. Attempts to solve these two problems have led to several variants of the original algorithm.

Improving correspondence

One improvement to the basic algorithm changes the simple point-to-point correspondences used in many of the methods [1–6], to ones between a point and a location on the "surface" represented by the other data set. This potentially increases the accuracy beyond that of the sampling resolution.

The first such effort was due to Chen and Medioni [7]. A starting location was found as the data point in the second set that is closest to a line through the first point in the direction of its estimated surface normal. Then, the tangent plane at this "intersection" point is used as the surface approximation. The initial point is projected onto this plane to give the corresponding location. This technique has subsequently been used in other approaches [8–10]. A further minor improvement by Dorai *et al.* [11] involves incorporating estimates of sensor inaccuracies into the tangent plane calculations. Lastly, more accurate but time-consuming estimates of the surface have also been used; such as octrees [12], triangular meshes [13], and parametric surfaces [14].

Another improvement explored by fewer researchers uses more than the simple Euclidean distance [1,3,4,6,7,9–14] in determining the closest point. Higher dimensional feature vectors include the estimated surface normal [8], as well as the principal curvatures of the surface [2,5]. By properly weighting the different components of the feature vector, as by Feldmar and Ayache [5], fewer incorrect correspondences can be obtained during early iterations when points are farther apart.

Thresholding outliers

The previous improvements still do not deal with the issue of data sets that are not subsets of one another, as was the case in most of the early algorithms [1,2,6,7,9,11,14]. The solutions proposed to date invariably have involved imposing a heuristic threshold on either the distance allowed between points in a valid pairing [3–5,8,12,13] or the deviation of the surface normals of corresponding points [8,10]. Any point pairs with distances greater than the threshold are assumed to be incorrect. These thresholds are usually constants [3,5,8,12,13] related to the estimated accuracy of the initial transformations, and can be difficult to choose robustly. Only Zhang [4] has computed a dynamically adjustable threshold based on the distribution of the distance errors at each iteration.

Computational requirements

In all of the techniques, computing potential correspondences is generally the most time consuming step. In a brute-force approach [1,8,14], an $O(N^2)$ number of comparisons is performed to find N pairings. One way to reduce the actual time is to subsample the original data sets.

Criteria for subsampling include taking a simple fraction of the original number [2,3], using multiple scales of increasing resolution [13], or having subsets based on potential visibility under the current transform [10], points in areas away from surface discontinuities [7,11], points in areas of fine detail [9], and small random sets for robust transform estimation [6]. An alternative is to use the full original data sets, but organize the search using more efficient data structures such as the octree [12] or k-d tree [4,5]. The k-d tree [15] is even efficient, $O(N\ logN)$, when higher order features of the points are incorporated in the distance metric.

Computing Intermediate Motions

Once a set of correspondences has been determined, a motion transform must be computed that best aligns the points. The most common approach is to use one of several least squares techniques [16,17] to minimize the distances between corresponding points [1,4,6–11,13]. Alternatively, a Kalman filter has been used to compute the intermediate motion at each iteration [5]. In certain cases [1,4,5,13], point contributions are weighted based on the suspected noise of different portions of the data sets. More robust estimation using the least median squares technique (clustering many transforms computed from smaller sets of points) has been tried by Masuda and Yokoya [6].

Other techniques compute the motion transform via some form of search over the space of possible transforms, trying to minimize a cost function such as the sum of distance errors across all corresponding points. Steps in transform parameter space are computed based on the changing nature of the function. Such standard search strategies as Levenberg-Marquardt [12,14] and simulated annealing [3] have been used, in addition to others more heuristic in nature [2,9]. Correspondences must be periodically updated during the search to keep the error function current. Updating too frequently can drastically increase the amount of computation, while too few updates can lead to an incorrect minimization.

Initialization/Convergence of Search

As mentioned earlier, an ICP-based refinement occurs after some initial set of transformations has been determined. Some researchers assume that this estimate is determined by a previous process [4,6–8,14], possibly calculated using feature sets. Other prior estimates can be given by a rotary table [2,10,12], a robot arm [3], or even the user [13]. Most such estimates are assumed to be quite accurate so that using various distance thresholds during matching will prune outliers correctly.

Other researchers do their own feature-based alignment using such characteristics as principal moments [1] or axes [11], normals of distinctive points [2], positions of points with distinguishing principal curvatures [5] or similar triangles on a mesh representation of the data [9]. If these distinguishing features are absent, a uniform distribution of starting points can be processed [1].

All of the iterative search algorithms must use some set of criteria to detect convergence of the final transformation. For those techniques that compute intermediate motions using least squares methods, convergence is achieved when the transform implies a sufficiently small amount of motion [4,9,10], or the distance between corresponding points becomes suitably close [1,6–8,11,13]. The Kalman filter approach stops when the uncertainty in the computed transform reaches a desired level [5]. And the iterative searches [2,3,12,14] typically converge based on small changes in the parameters or error value, or if the shape of the cost function at the current value indicates a function minimum. All methods can be terminated if convergence is not detected after some maximal number of iterations.

View pairs vs. multiple views

The majority of the discussed techniques [1,4–6,8,9,11,12,14] were designed with only two data sets in mind. If one desires to merge multiple images, the naive approach of simply examining the sequence in pairs could be performed [2]. However, any errors in these computations will accumulate, leaving the first and last in the sequence aligned rather poorly. Therefore, a few methods have searched for a more globally optimal set of transforms.

The first of these, by Turk and Levoy [13], assumed that an additional continuous cylindrical data scan is available. Individual linear scans were registered to this image, which should have had commonalities with each of them. Unfortunately, not all scanners can produce such a base image. Chen and Medioni [7] incrementally registered data from successive views into a growing combined set. However, early calculation errors were still not corrected.

Two other techniques attempted to compute the motion transforms simultaneously. First, Blais and Levine[3] defined a consecutive set of transforms between pairs of images in the sequence. In addition, the transform between the first and the last images was taken as the composition of the intermediate ones. They then minimized the total cost function across all image pairs by searching in the combined transformation space. This high dimensional minimization is often difficult. In the second technique, Gagnon *et al.* [10] considered each view as being transformed into a common frame. Then a view's data could be matched to each other set through composed transforms. The combined set of correspondences was used to compute each data set's motion at each iteration. While these two methods did compute the transforms simultaneously, they still suffer from the thresholding difficulties of pairwise correspondences.

In summary, global optimization techniques have been developed, but they still rely on pairwise correspondence computations. These suffer from having to choose thresholds that reject incorrect pairings. In the following section a new global registration technique is presented which addresses the major problems of simultaneous solutions and distance thresholds, in addition to combining some of the best features of previous techniques with new ideas.

The Registration Algorithm

Given a group of N data sets, the goal of the registration is to compute a set of rigid transformations $\{\mathcal{T} \mid \boldsymbol{T_i} = [\boldsymbol{R_i}, \boldsymbol{t_i}],\ i = 1 \ldots N\}$, where $\boldsymbol{R_i}$ is a standard orthonormal rotation matrix and $\boldsymbol{t_i}$ is a translation vector. These transformations should align the data sets with minimal disparity. A high-level overview of the algorithm is shown in Figure 1. All view transforms are incrementally updated in a simultaneous manner so that the best global solution can be found. At each iteration correspondence is determined using point position and normal information. Point comparison is performed over a combined data set, where every point in one data set should have a corresponding point in another data set, eliminating the need for a distance threshold (noisy points should get removed in a preprocessing step). K-d trees are used to speed up point matching, and point projections onto tangent planes of corresponding points increase the final accuracy.

Incremental motion computations are made using a force-based approach. Imaginary springs connect corresponding locations to generate interpoint forces. A time step simulation is run to update the motion of each data set based on the net forces and torques applied by the springs. Correspondences are periodically updated over time. Finally, hierarchical sized sets of data are processed to decrease overall computation time without sacrificing eventual accuracy. Convergence is detected when the amount of motion is sufficiently small. In the following sections each of the stages of this algorithm is discussed in detail.

- Initialization
  - Median filter data in each view (7 x 7 window).
  - Determine sampling resolution and depth discontinuity boundaries for each view using threshold (d_d)
  - Gaussian smooth data (7 x 7 window, σ) and compute surface normals ($\mathbf{n}_p$) in each view
  - Determine number of subsampling passes based on data size.
  - Initialize view transforms (T), dynamic spring system (μ_v ,μ_ω, λ ,$\mathbf{v}$,ω, $\mathbf{a}$, α) and simulation time step (Δt)
- Subsampling Passes
  - For increasing sample sizes
    - Select random subset of interior and boundary points in views
    - Generate k-d trees of points in each view
    - Initialize point feature (location,normal) weight ratio (s_n ,w_n)
    - Data set convergence
      - While standard deviation of point distance error distribution still decreasing
        - Data correspondence
          - Compute correspondence of non-boundary points using normal k-d trees
          - Compute correspondence of boundary points using specialized k-d trees with threshold (β)
          - Determine number of changed correspondences from previous iteration
          - Update feature weight ratio (w_n) based on number of changed correspondences
        - Motion estimation
          - While incremental motion still significant compared to threshold (δ_m)
            - Update correspondence locations with projections onto tangent planes ($\mathbf{p}$, $\mathbf{q}$)
            - Compute spring forces ($\mathbf{F}$), torques ($\mathbf{M}$)and moments of inertia ($\mathbf{I}$) from current data positions ($\mathbf{p}$, $\mathbf{q}$)
            - For each view's data set
              - Update angular position (θ), velocity (ω) and acceleration (α) of points using torques ($\mathbf{M}$), moments of inertia ($\mathbf{I}$) and current time step (Δt)
              - Update linear position ($\mathbf{l}$), velocity ($\mathbf{v}$) and acceleration ($\mathbf{a}$) of points using forces ($\mathbf{F}$) and current time step (Δt)
              - Adjust velocity damping factors (μ_v , μ_ω) based on new accelerations ($\mathbf{a}$,α)
              - Compute incremental transformation ($R_{\Delta\theta}$, t_Δ) based on updated position ($\mathbf{l}$, θ), compose with current transformation (R_v , t_v)
            - Adjust time step (Δt) based on new distance error (E)

Figure 1: High level chart of registration algorithm

Initialization

Several operations are performed before the actual iterative process begins. First a median-type filter is used to remove noisy data in each range image based on a neighborhood (7×7) of depth values in the scan grid. Following this, points on the occluding side of step discontinuities are detected, where a discontinuity is said to occur for points separated by more than a multiple of the average sampling resolution (here, the depth discontinuity threshold, $d_d = 10 *$ sampling resolution). These occluding points are important in constraining proper alignment of surface boundaries in a manner not used in previous algorithms.

Next, the remaining data is smoothed with a Gaussian kernel ($\sigma = 2.5$) before surface properties are computed. All three position components are independently calculated based on positions of neighbors in a 7×7 window in the range scan. At each point a tangent plane is fitted to the window of smoothed data around the point to estimate the surface normal. (The smoothed data is only used in the normal calculation. Original data is used in all other computations.) Principal curvatures were not used in this implementation. The inherently noisy nature of these second-order features, plus the additional computation burden, outweighed the potential discriminatory power. On the other hand, the usefulness of the surface normal justified their computations.

Additionally, we must initialize the state of the dynamic spring system. It is assumed that the starting view transformations, $\mathcal{T}$, that align the views in a common coordinate frame, are obtained from an external process, such as the rotary table or robot arm mentioned previously. Later experiments will show that these transforms can be quite rough in their accuracy without preventing proper convergence. The remaining state consists of inertial properties (velocity damping factors $\boldsymbol{\mu_v}$ and $\boldsymbol{\mu_\omega}$) of the data sets, as well as the spring strength (λ) and the simulation step size (Δt). In the beginning it is assumed that each group of points is at rest, with both zero velocity ($\boldsymbol{v}$ and $\boldsymbol{\omega}$) and acceleration ($\boldsymbol{a}$ and $\boldsymbol{\alpha}$), there is no velocity damping, the spring strength is constant ($\lambda = 1000$), and the simulation time step is $\Delta t = 0.005$. The number of subsampling passes for the simulation is a function of the total number of points in each view. The actual schedule is discussed in a later section.

Determining point correspondences

In order to determine the point that is "closest" to another, a distance metric is needed to define *closest*. In a manner similar to Feldmar and Ayache [5], the metric used combines both point location and normal information. The actual metric is:

$$E = \| \boldsymbol{p} - \boldsymbol{q} \|^2 + w_n * s_n^2 * \| \boldsymbol{n_p} - \boldsymbol{n_q} \|^2 \quad (1)$$

Here, $\boldsymbol{p}$ and $\boldsymbol{q}$ are the coordinates of two points and $(\boldsymbol{n_p}, \boldsymbol{n_q})$, their associated unit normal vectors. Since the difference in magnitude of errors in position and normal can be large, it is important to scale the dimensions of these components of the feature vector to similar ranges. The value s_n is chosen as the ratio of the size of a bounding box in 3-D about the combined data sets to the maximal normal difference (bounded by a value of 2 for unit normals). The value of w_n is used to control the steadily decreasing contribution of the normal information over time (the exact schedule is discussed later).

Equation (1) is not the only distance metric used. It will be shown in later experiments that only using simple point correspondences as in equation (1) can lead to premature convergence. Therefore, an additional metric is used based on those points labelled as occluders as:

$$E = b_{vp} * \| \boldsymbol{p} - \boldsymbol{q} \|^2 \quad (2)$$

Here, b_{vp} has a value of one if the surface normal of point $\boldsymbol{q}$ is back-facing with respect to the viewpoint associated with point $\boldsymbol{p}$. Otherwise b_{vp} has a value of infinity. This models the relation that' the proper correspondent of a point on an occluding boundary is merely the nearest point on the hidden portions of the other surfaces with respect to the current data set's viewpoint.

The correspondence search

A basic assumption of the global approach presented here is that each portion of the object has been observed in at least two data sets. (Limitations of this assumption are discussed later). Given this assumption, a proper corresponding location on a surface does exist; it just needs to be found. Therefore, we avoid the need for any distance thresholds, as in the pairwise view processing of previous methods. This is a significant advantage of our approach.

The use of k-d trees can improve the efficiency of searching for correspondences. Ideally one would like to represent all of the data points across all views in a single k-d tree. But, since the interpoint distance relationships are constantly changing between data sets, this would require continual reconstruction of the tree. On the other hand, since the points of a single data set are rigid, a tree representing each view can be constructed in the beginning and never modified.

A k-d tree partitions data points into regions of space bounded by dividing planes that are perpendicular to a particular axis of the k dimensions. At each level in the tree the dimension which provides maximal distinction (here k = 6 for three position and three normal components) is used to divide the points at a node into two sets. Because the dividing planes are perpendicular to the axes, the scaling of a dimension (such as the normal components by w_n in equation (1)), does not change the relative ordering of the points and the partition. Thus the single k-d tree for a view can still be used in many different searches as the value of ω_n changes.

Determining the overall closest point is simply a matter of searching the k-d tree for each data set other than the current one to find its corresponding closest point. The globally closest point is found with a simple linear comparison of these closest points. The search for the closest point in a given tree is done using a depth-first traversal with pruning. By going down the tree one moves into smaller and smaller regions which contain the search point. At a leaf which contains only one point, an upper bound on the distance to the closest point is updated using the distance to the data point in the leaf's cell (actually, the distance norms of equation (1) are left squared to reduce computation). In the remainder of the search, a portion of the tree can be ignored if the distance to the boundary of the corresponding region of space is greater than the current upper bound.

A good initial estimate of the maximal distance to the closest point can lead to drastic reductions in the percentage of the tree traversed. An effective value can be found by using the correspondences from a previous iteration. Here, the distance to the old corresponding point under the current transform is a good upper bound. Near convergence the number of changing correspondences is small, and the search of the k-d tree requires only a single traversal to the bottom to verify that the old point is still the best.

A second tree with $k = 3$ (for the three position coordinates) is used to search each data set according to equation (2). A single modification to the search involves the method of update to the maximal distance estimate. Rather than always updating the value when a leaf is reached, the distance is changed only if the back-facing normal test labels the point as "invisible". In practice a tolerance angle of $\beta = 10°$ is used to model the inability of real range sensors to detect steeply angled surfaces with respect to the viewer.

Interpoint forces

In order to obtain a registration with potentially greater accuracy than the sampling resolution, it is necessary to find the closest location on the surface near the corresponding point of a given point. A variant of the tangent plane projection method of Chen and Medioni[7] is used, which is seen as a compromise between the improved accuracy of a higher-order surface fit and the associated increased computation time.

One drawback of Chen and Medioni's method is that it is doubly sensitive to any noise in the surface normals. The calculation of a corresponding point by "intersecting" a normal with a surface can be significantly in error during early iterations prior to rotational alignment, and even later due to noise. The search for correspondence detailed here avoids much of this error. However, once the closest point has been chosen, the surface location is still found by projecting the search point onto the associated tangent plane as in previous methods.

To see the potential effects of this process, consider the simple 2-D example in Figure 2. Assume that the black points are a subset of the white as seen from a different angle to the left, and that the rightmost point was labelled as an occluding point. The point-to-point correspondences are indicated by the solid lines. Note the rightmost occluding point links to a point

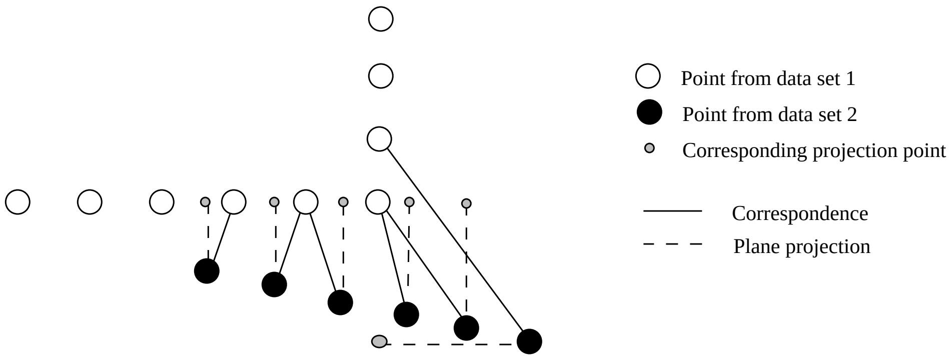


Figure 2: Mapping of corresponding points. The points from data set 2 correspond to the indicated points in set 1 via the solid lines. The points project onto the associated tangents of corresponding points via dashed lines. The rightmost point of set 2 is labelled occluder and therefore maps to the nearest hidden point around the corner.

around the corner (it does not map to the corner point because the normal for this point is assumed to be consistent with the other points on the horizontal line, making the corner point "visible"). The projections onto the tangents are indicated by dashed lines. The non-occluder pairings will tend to pull the two lines into being parallel, while the occluding point moves the ends of the lines together. If the rightmost point was instead handled as a non-occluder, it would map to the corner point, causing the two lines to never overlap properly without a force that acted along the line of points. Using direct point-to-point mappings everywhere as in the original ICP process can also lead to problems in this example. For instance, as the region of overlap increases during motion in the figure, the desire of points to stay still when incorrectly mapped to very close neighbors will eventually outweigh the desire of the others to move, causing premature convergence. Further examples of this will be seen in the experiments.

In order to align the corresponded data sets, there must be attractive forces to pull them together. Connected between each point and its corresponding projection location is an imaginary spring of natural length zero. This spring generates a force with magnitude, $F = \lambda\ d$, where d is the distance between the points, and λ is the spring tensile strength ($\lambda = 1000$ for regular points and $1000 * \text{pass}^2$ for occluders. Increasing an occluder's spring constant helps quicker convergence). The spring force is directed along the line connecting the points and is reflexively applied at both ends. The cumulative effect of the forces from each point will govern the movement of the data sets toward one another.

Computing motion transforms

Rather than use some of the more popular least squares solutions [1,4,6–11,13] to compute intermediate motions, a simulation of the dynamic spring system is used. The reason is that the effects of any significantly incorrect correspondences are compounded when the best alignment is computed in a least squares manner, in addition to the premature convergence problems just mentioned. With a dynamic system it is possible to move in the direction of an intermediate solution without being totally committed to it. Also, inertia from previous motions can help constrain any excessive, and possibly incorrect, motion tendencies. In the example of Figure 2, this makes it seem as if the corresponding end of each spring is not attached to a specific point, but rather to a line. This degree of freedom allows the data to move in the direction of the tangent lines under the influence of the single force from the occluding point. This point to line mapping technique has been used previously to help solve other alignment problems [18].

One method of simulating this dynamic system would be an exact *finite element analysis* [19], in which each data set truly moves simultaneously. Spring lengths and end positions would constantly be changing, resulting in continuously varying forces on each data set. However, this process is rather involved due to the computational complexities. In a less exact scenario, forces can be assumed constant over a properly small amount of time, thus allowing the movement of each set to be computed independently. This movement can be decomposed into a translational movement of the center of mass of a view's points, along with a rotation of the points about that center. We now give the details of this approximate solution.

A single time step

Translational movement is a consequence of the cumulative effect of the forces generated both by the springs attached from each data point to other view's data surfaces, and from other view's points to locations on the current surface. Relating each force to motion is done according to the law, $\boldsymbol{F_i} = m_v\, \boldsymbol{a_i}$, where $\boldsymbol{F_i}$ is the force directed by a particular spring, $\boldsymbol{a_i}$ is the corresponding acceleration that it generates, and m_v is the mass of the view's points. The overall equation for a view's acceleration is then given by:

$$\boldsymbol{a_v} = \frac{\sum_{i=1}^{n_v} \lambda\, \boldsymbol{d_i} + \sum_{j=1}^{n_o} \lambda\, \boldsymbol{d_j}}{m_v} = \frac{\lambda}{n_v}\, \left(\sum_{i=1}^{n_v} \boldsymbol{d_i} + \sum_{j=1}^{n_o} \boldsymbol{d_j}\right) \qquad (3)$$

Here, n_v and n_o are the number of points in the view's data set and the other data sets, respectively, that are contributing forces. The distances between points and their projections onto corresponding tangent planes are given by d_i and d_j. If each point is assumed to have unit mass, then $m_v = n_v$.

By assuming that the forces, and therefore acceleration, remain constant for a time, Δt, the change in location and velocity can be calculated using the equations:

$$\begin{aligned} \boldsymbol{l_t} &= \boldsymbol{l_{t-1}} + \boldsymbol{v_{t-1}}\, \Delta t + 0.5\, \boldsymbol{a_{t-1}}\, \Delta t^2 &\quad (4) \\ \boldsymbol{v_t} &= \boldsymbol{v_{t-1}}\, \boldsymbol{\mu_v} + \boldsymbol{a_{t-1}}\, \Delta t &\quad (5) \end{aligned}$$

Position is updated through the velocity and acceleration over the given time step. The new velocity is a function of the old velocity and acceleration. An additional term, $\boldsymbol{\mu_v}$, is included here as a matrix with zeroes off the diagonal and damping values along the diagonal for each velocity component. As time passes, the momentum of a set of points can get quite large, causing oscillations as correspondences change. While the reversal of acceleration from these changes can slow these oscillations, additional damping of the previous velocity in this case is also helpful. The damping can be relaxed as velocity and acceleration again become synchronized.

Rotational movement is governed by a similar set of equations. Each force, when applied at the associated point, generates a torque (or moment) with respect to the center of mass of the points. The cumulative effect of these moments causes rotation about the center. Recall in two dimensions that rotation in the plane caused by a single force is governed by the simple relation, $M_i = I_i\ \alpha_i$, where α_i is the angular acceleration, $I_i =\ m_v\ \parallel \boldsymbol{r_i} \parallel^2$ is the moment of inertia, and $M_i = \parallel \boldsymbol{r_i} \times \boldsymbol{F_i} \parallel$ is the moment for the radius, $\boldsymbol{r_i} = \boldsymbol{p_i} - \boldsymbol{c_m}$. Here the force, $\boldsymbol{F_i}$, is applied at point, $\boldsymbol{p_i}$, with respect to the center of mass, $\boldsymbol{c_m}$.

In 3-D, the relationship of moments to angular motion is more complex and governed by the set of scalar equations (referred to as the Euler equations of motion [20]):

$$\begin{aligned} M_x &= I_{xx}\ \alpha_x + \omega_y\ \omega_z\ (I_{zz} - I_{yy}) \\ M_y &= I_{yy}\ \alpha_y + \omega_x\ \omega_z\ (I_{xx} - I_{zz}) \\ M_z &= I_{zz}\ \alpha_z + \omega_x\ \omega_y\ (I_{yy} - I_{xx}) \qquad (6) \end{aligned}$$

Here, ω_j, α_j, M_j, and I_{jj} are the j^{th} components of the angular velocity, angular acceleration, moment and moment of inertia, respectively. If the angular velocity is known, equations (6) can be rewritten to calculate the angular acceleration due to a set of forces as:

$$
\begin{aligned}
\alpha_x &= \frac{\sum_{i=1}^{n_v} [(\boldsymbol{r_i} \times \boldsymbol{F_i})_x - \omega_y\, \omega_z\, n_v\, (r_{iz}^2 - r_{iy}^2)] + \sum_{j=1}^{n_o} [(\boldsymbol{r_j} \times \boldsymbol{F_j})_x - \omega_y\, \omega_z\, n_v\, (r_{jz}^2 - r_{jy}^2)]}{n_v\, (\sum_{i=1}^{n_v} r_{ix}^2 + \sum_{j=1}^{n_o} r_{jx}^2)} \\
\alpha_y &= \frac{\sum_{i=1}^{n_v} [(\boldsymbol{r_i} \times \boldsymbol{F_i})_y - \omega_x\, \omega_z\, n_v\, (r_{ix}^2 - r_{iz}^2)] + \sum_{j=1}^{n_o} [(\boldsymbol{r_j} \times \boldsymbol{F_j})_y - \omega_x\, \omega_z\, n_v\, (r_{jx}^2 - r_{jz}^2)]}{n_v\, (\sum_{i=1}^{n_v} r_{iy}^2 + \sum_{j=1}^{n_o} r_{jy}^2)} \\
\alpha_z &= \frac{\sum_{i=1}^{n_v} [(\boldsymbol{r_i} \times \boldsymbol{F_i})_z - \omega_x\, \omega_y\, n_v\, (r_{iy}^2 - r_{ix}^2)] + \sum_{j=1}^{n_o} [(\boldsymbol{r_j} \times \boldsymbol{F_j})_z - \omega_x\, \omega_y\, n_v\, (r_{jy}^2 - r_{jx}^2)]}{n_v\, (\sum_{i=1}^{n_v} r_{iz}^2 + \sum_{j=1}^{n_o} r_{jz}^2)} \qquad (7)
\end{aligned}
$$

where r_{ab} is the b^{th} component of the radius vector for the a^{th} point.

Then, making the approximation that angular acceleration remains constant for small Δt, the change in angular position and velocity can be found using equations similar to (4) and (5):

$$\boldsymbol{\theta_t} = \boldsymbol{\theta_{t-1}} + \boldsymbol{\omega_{t-1}}\, \Delta t + 0.5\, \boldsymbol{\alpha_{t-1}}\, \Delta t^2 \qquad (8)$$

$$\boldsymbol{\omega_t} = \boldsymbol{\omega_{t-1}}\, \boldsymbol{\mu_\omega} + \boldsymbol{\alpha_{t-1}}\, \Delta t \qquad (9)$$

Here, each component of the angular velocity is damped using the appropriate element of the diagonal matrix, $\boldsymbol{\mu_\omega}$. If the above approximation is valid, these calculations are much simpler and more reasonable, than attempting to directly solve the set of differential equations in (7).

Using the values of $\Delta \boldsymbol{l} = \boldsymbol{l_t} - \boldsymbol{l_{t-1}}$ and $\Delta \boldsymbol{\theta} = \boldsymbol{\theta_t} - \boldsymbol{\theta_{t-1}}$ from equations (4) and (8), one can compute the elements of an incremental transform relative to the center of mass, $\boldsymbol{R}_{\Delta \boldsymbol{\theta}}$ and $\boldsymbol{t}_{\Delta \boldsymbol{l}}$. These can then be converted to incremental transforms in the common coordinate frame as:

$$\boldsymbol{R}_\Delta = \boldsymbol{R}_{\Delta\boldsymbol{\theta}} \qquad \boldsymbol{t}_\Delta = (\boldsymbol{I} - \boldsymbol{R}_{\Delta\boldsymbol{\theta}})[\boldsymbol{R_v}\, \boldsymbol{c_m} + \boldsymbol{t_v}] + \boldsymbol{t}_{\Delta \boldsymbol{l}} \qquad (10)$$

where, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{R_v}$ and $\boldsymbol{t_v}$ are the current transform for the view, and $\boldsymbol{c_m}$ is the original center of mass. This incremental transform can then be composed with the current one to yield the new total transform for the view into the common frame. The common frame is initially established by setting all view transforms to the identity.

Iteration control

The overall iterative process is a set of nested cycles as seen in Figure 1. The outermost cycle controls the hierarchical subsampling scheme for the data. For a given sampling, point correspondences are found and motion computed until convergence occurs. Since determining correspondence between data points is an expensive operation, it is done as few times as possible. Several motion steps can be made before new pairings are needed. Thus the motion computation itself is also iterative, continuing until movement due to a set of forces is minimal (minimal motion, δ_m, is 1% of the total motion so far in a cycle).

Subsampling the data

Larger data sets mean more computation. Therefore, if approximate alignment can be obtained using reduced data sets, efficiency is enhanced. In the algorithm, a starting sample size of 100 points is used. For subsequent passes we increase this data set size by a factor of ten, until on the final pass all of the original data is used. Points for a subsample are selected randomly, with the constraint that occluding points, which are important for boundary alignment, are specially selected so that a quantity proportional to the number of regular points is obtained. The k-d trees are then constructed for each new subsample. Because the mass of a view is based on the number of points, reasonable motion continuity between samplings is maintained as the increased number of forces is balanced by the increased mass.

Adjusting controlling parameters

Several of the controlling parameters in the previous equations are adjusted either during a subsampling step, or during each motion step. The parameter, w_n, decides the weight of normal to position information during correspondence, see equation (1). This value changes based on sample size and motion steps. Processing of the first subsample involves large changes in rotational alignment, where normal information is most useful. Therefore, w_n is set to one for the entire pass. During the second pass focus shifts towards translational alignment, where normals are less important. After each new set of pairings is computed, the number of changed correspondences is recorded. During the second pass, w_n is set to the ratio of this quantity and the total sample size. Thus it should gradually go from one towards zero as the pass progresses. For all subsequent passes w_n is set to zero, since the data should be very nearly aligned, requiring only small translation updates.

The next two parameters adjusted are the damping matrices, $\boldsymbol{\mu_v}$ and $\boldsymbol{\mu_\omega}$. Beginning with diagonal values of one, a diagonal component is reduced by 10% each time the associated acceleration and velocity directions are opposed, and increased by 100% when agreement again occurs, until the value of one is reached. In this way, oscillations are damped rather smoothly, and then the system is released to begin again quickly.

The final parameter is the time step, Δt. This should be maintained at a value that is not so large as to violate the constant force assumption, yet large enough to continue reasonable amounts of movement. After a motion step, the velocity damping factors are examined. If a majority of the views are not being damped, things are in order, and the time step is increased by 0.1%. If not, it means oscillations are occurring as the result of assumptions being violated, and therefore the time step is reduced by the same amount. The initial length of the time step is set at 0.005.

Convergence of sampled data

Convergence of sampled data is detected based on the distances between points and their corresponding tangent planes. The distribution of the signed distance of points to planes is computed after each motion. If the standard deviation of this distribution decreases then the surfaces are still integrating. If not, the next larger sample is processed until finally convergence is achieved with the original data.

Experimental Verification

In this section the properties of the registration process are examined. These properties include the radius of convergence with regard to starting configuration, the accuracy and repeatability of the force-based optimization, as well as the rate of convergence. The algorithm was implemented in C, and uses values for the various controlling parameters as summarized in Table 1. Both 2-D and 3-D data sets are used to emphasize key features of the algorithm.

Utility of data normals and occlusion points

Before presenting the complete experimental results, it is instructive to examine how the various characteristics of the current algorithm provide more potential than previous methods. For this we use a simple 2-D data set consisting of four corner views of a rectangle as seen in Figures 3.a and 3.g. The data is simulated, but contains depth quantization noise. Five different versions of the algorithm are compared, all of which use the force-based motion computations, but correspondences are determined in different ways. The first is the traditional direct mapping

Table 1: Algorithm parameter initial values and update methods

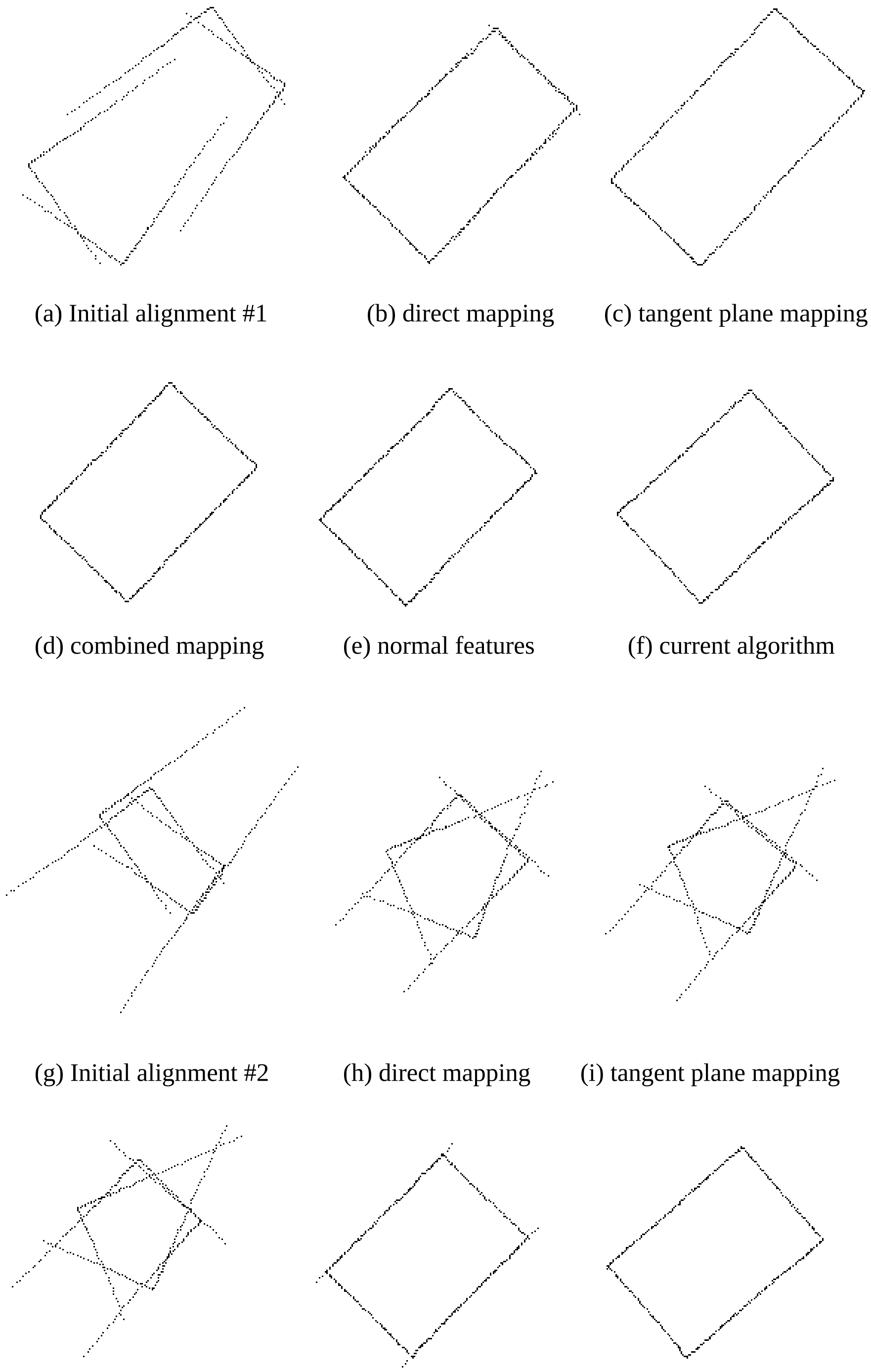
Parameter	Initial value/Update method
Image Processing	median filter and smoothing window size - 7×7 Gaussian smoothing kernel ($\sigma = 2.5$)
d_d	$10 *$ sampling resolution
s_n	data diameter / 2
β	$10°$
w_n	Pass 1 - $w_n = 1$, Pass ≥ 3 - $w_n = 0$ Pass 2 - $w_n = \#$ changing correspondences / # points, only if decreasing
λ	1000 for regular points, $1000 *$ pass^2 for occluders
δ_m	1% of total movement for current point pairings
$\boldsymbol{\mu_v}$, $\boldsymbol{\mu_\omega}$	Initially all diagonal components of matrix are 1.0, decrease an element by 10% to dampen, increase by 100% to undampen
Δt	Initially 0.005, decrease 0.1% each step when system is damped, increase by 0.1% if system undamped

of closest points based strictly on position. The second computes projections of points onto the corresponding tangents. The third additionally maps occluder points to the nearest hidden point. The fourth again uses direct mapping between points, but the normal at the point is used as in equation (1). The fifth algorithm is that presented here which combines all of these elements.

The first example, Figures 3.a - 3.f, shows the results on data sets which have been pushed out from their desired position. The first two algorithms do not fully contract to the correct shape, due to the missing occlusion constraints which the third version provides. The use of normal features also helps to contract the shape towards its proper size. For this case, the latter three algorithms all produce comparable results. The second example, Figures 3.g - 3.l, contracted the data sets in from their desired position. Here, the normal features are a necessity in establishing correspondences that do not bind the tips together incorrectly. However, when using only the normal features, the ends do not come into full alignment. Only the algorithm presented here, which combines normal features and occluding point mapping, converges upon the correct shape in both instances.

Optimization convergence properties

In this section 2-D data is again used to examine the convergence properties of the algorithm. These data sets consist of eight views of a cross-section of a real 3-D object as seen by a triangulation-based range sensor and shown in Figure 4.a. The goal of this experiment is to



(a) Initial alignment #1 (b) direct mapping (c) tangent plane mapping

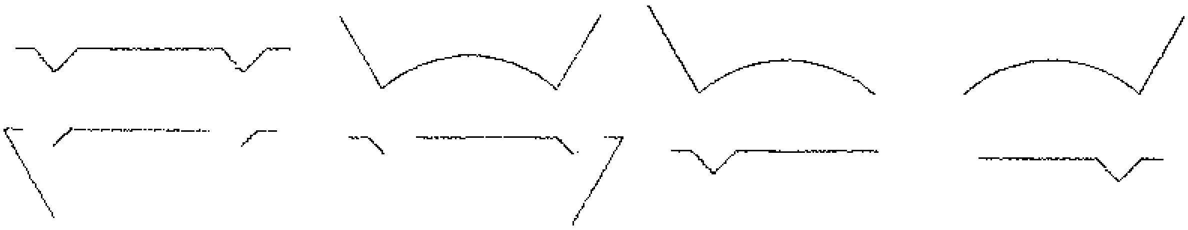
(d) combined mapping (e) normal features (f) current algorithm

(g) Initial alignment #2 (h) direct mapping (i) tangent plane mapping

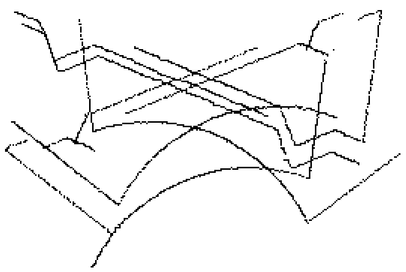
(j) combined mapping (k) normal features (l) current algorithm

Figure 3: Alignment of 4 corner views of a rectangle. Each view contains 128 simulated points generated with quantization error. Given the starting positions in (a) and (g), algorithms with 5 different characteristics converged to the registrations shown.

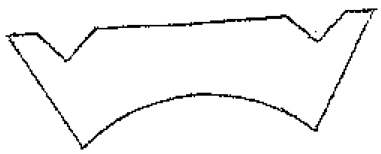
determine the necessary quality of the initial transforms to insure proper convergence. Three sets of tests were run, the results of which are summarized in Table 2. First, the algorithm was run on the data from starting positions provided by the acquisition process. The resulting answer was taken to be the "true" registered position. The data was then disturbed from this configuration by varying amounts and the algorithm run on it.



(a) Eight views of example 2d outline



(b) Initial position of views,
rotation = $\pm$25 degrees,
translation = $\pm$25% of size



(c) Registered set of views

Figure 4: Alignment of 8 views of cross-section of a real object of size $50 \times 125mm$. Each view in (a) contains between 336 and 501 real points measured by a triangulation-based range sensor. An initial position with rotation error of $\pm$ 25° and translation error of $\pm$ 25% of the size of the polygon is shown in (b), with final registration in (c).

The perturbance of the data took three forms; rotation about the center of mass in the plane of the data only, translation of the center of mass in the plane only, and combined rotation and translation. The change in rotation for each view was $\pm\theta$, where θ was increased in five degree increments as shown in Table 2.a. Thus two views were basically either in alignment, or 2θ out of alignment. Translation amounts were handled similarly, with the amount of movement equal to a fraction of the diameter of the data sets. For each level of disturbance, 25 (of the possible 256) data configurations were generated, with the direction of rotation or translation being determined randomly. The three sections of Table 2 show the number of times the "true" configuration was achieved, how many iterations the process took, and how close the converged positions were to one another.

Looking at the results in these tables one can draw a few conclusions.

- The average rate of convergence is rather independent of the level of perturbation from the starting point. The average number of iterations does not change by more than 20%, but the absolute rate of convergence is expected to be different for other data sets.
- The system repeatably converges to the same set of transforms. The difference between

Table 2: Results of convergence tests for object in Figure 4. At each error setting 25 tests were performed. Average angle and translation errors from "true" converged values were measured, along with total distance error sums and standard deviation of error distances. A registration was deemed correct if the total distance error was less than 400 mm. All distance entries are in mm.

$\Delta\theta$	# converged correctly	avg iterations	avg std dev	avg total distance	avg angle (correct)	avg trans (correct)	avg angle (incorrect)	avg trans (incorrect)
$\pm 0°$	25	16.8	0.1615	343.0	0.060°	1.04	-	-
$\pm 5°$	25	18.3	0.1609	343.4	0.060°	1.02	-	-
$\pm 10°$	25	20.8	0.1611	343.6	0.063°	1.08	-	-
$\pm 15°$	25	19.2	0.1606	342.4	0.064°	1.10	-	-
$\pm 20°$	24	19.5	0.1856	382.1	0.075°	1.29	4.35°	70
$\pm 25°$	18	17.7	0.5094	881.0	0.075°	1.28	12.52°	226
$\pm 30°$	14	19.3	0.7953	1359.6	0.080°	1.37	14.57°	297

(a) Convergence results for rotation only changes

$\Delta x, y$	# converged correctly	avg iterations	avg std dev	avg total distance	avg angle (correct)	avg trans (correct)	avg angle (incorrect)	avg trans (incorrect)
$\pm 0\%$	25	17.2	0.1612	343.8	0.062°	1.05	-	-
$\pm 5\%$	25	17.0	0.1611	343.2	0.064°	1.10	-	-
$\pm 15\%$	25	20.5	0.1610	343.8	0.078°	1.34	-	-
$\pm 25\%$	25	19.7	0.1611	344.3	0.060°	1.02	-	-
$\pm 35\%$	24	18.3	0.1818	385.4	0.061°	1.05	1.48°	32
$\pm 40\%$	20	20.3	0.4036	772.7	0.062°	1.06	6.19°	118
$\pm 45\%$	14	18.8	0.5150	910.3	0.086°	1.46	4.33°	68

(b) Convergence results for translation only changes

$\Delta\theta$ $\Delta x, y$	# converged correctly	avg iterations	avg std dev	avg total distance	avg angle (correct)	avg trans (correct)	avg angle (incorrect)	avg trans (incorrect)
$\pm 0°, 0\%$	25	14.2	0.1601	343.3	0.077°	1.32	-	-
$\pm 5°, 5\%$	25	19.1	0.1611	344.1	0.070°	1.19	-	-
$\pm 10°, 10\%$	25	18.3	0.1613	344.3	0.063°	1.08	-	-
$\pm 15°, 15\%$	25	19.2	0.1610	342.8	0.064°	1.11	-	-
$\pm 20°, 20\%$	24	21.4	0.2397	489.1	0.067°	1.15	11.51°	193
$\pm 25°, 25\%$	21	20.0	0.3868	711.9	0.080°	1.39	10.01°	183
$\pm 30°, 30\%$	7	19.4	1.7935	3149.9	0.060°	1.03	17.13°	366

(c) Convergence results for rotation and translation changes

the answers of correct convergences was less than 0.1° in rotation and 1.5% of the diameter of the object in translation. Note that small changes in rotation can almost be compensated for with different translations to achieve very similar transforms. The average total distance error, as well as the standard deviation of the point to tangent distance, indicate that a consistent minimum was being converged upon. Similar error variance was observed in the zero perturbation cases (top lines of each table), because the first pass of registration based on the small random subsamples pulls the search from its starting point to different locations each time.

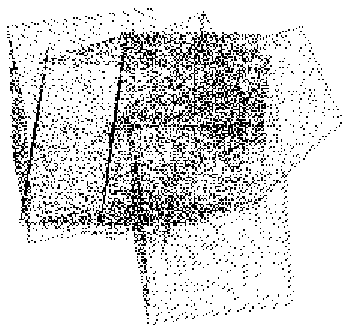
- When incorrect registration occurs, it is rather easy to detect. For the example object in Figure 4, the algorithm was more susceptible to rotational errors than translational errors. The amount of allowable angular error is related to the rotational symmetry of the object. For instance, rotating one of the views of the rectangle in Figure 3 by more than 45° would lead to incorrect correspondence. For the object in Figure 4, reliable results can be obtained for rotational errors of ±20° and translational differences up to one third of the object size. An example starting position in this extreme range for which proper registration was achieved is seen in Figure 4.b. Our algorithm appears to have a much greater radius of convergence than that demonstrated by previous methods based on this example.

3-D data analysis

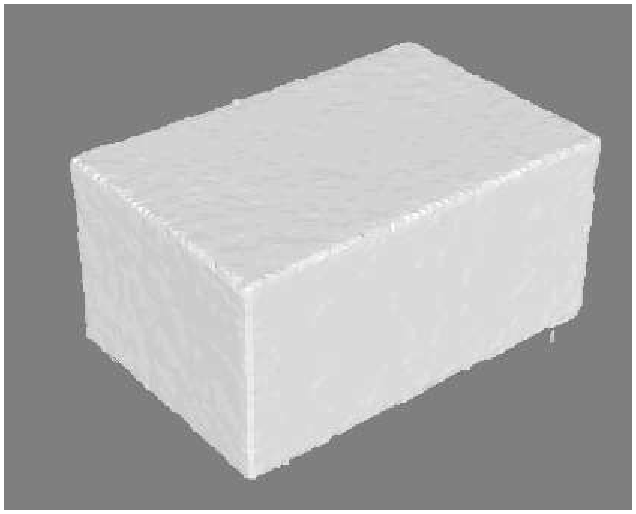
In order to demonstrate that the algorithm also works on 3-D data sets, multiple views of several objects were merged by the system. The first, a rectanglular block, is shown at the top of Figure 5. The image on the left shows the point sets of each view in their starting position (randomly perturbed from the initial registration of acquisition by approximately 10° in all rotation angles and 25% of the object size in translation). Eight views of this object were synthetically generated at a resolution of 128×128, but they did contain quantization error in the depth values. The image on the right shows the result of a triangulation process due to Hoppe *et al.*[21], as applied to the automatically registerd data. This triangulation has two distinguishing features. First, the edges of the block are rounded. This is a characteristic of many triangulation schemes, especially that of Hoppe *et al.* Second, there are certain small patches sticking out from the edges. These are due to the low sampling resolution. Since the viewing rays do not hit the object exactly along each true edge of the block, straight lines may not result. Some of the extremely sampled edge points give rise to the outlying small triangles in the reconstruction.

The views of the second object in the middle of Figure 5 were also synthetically generated, but at a resolution of 256×256. Here it can be seen that the resulting triangulation has edges that are less rounded, and the small extra patches are no longer present. The presence of small features and a curved surface on this object did not stop the algorithm from determining the proper registration.

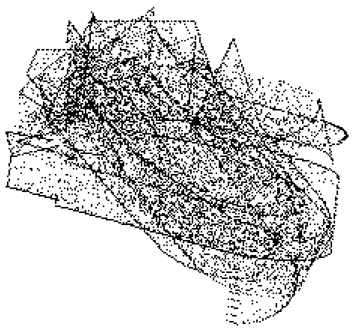
The first example of processing data from a real sensor is shown in the bottom of Figure 5. This object again possesses a combination of planar and curved surfaces. Here it can be seen that the edges of the triangulation are worse than in the synthetic case. This is not the



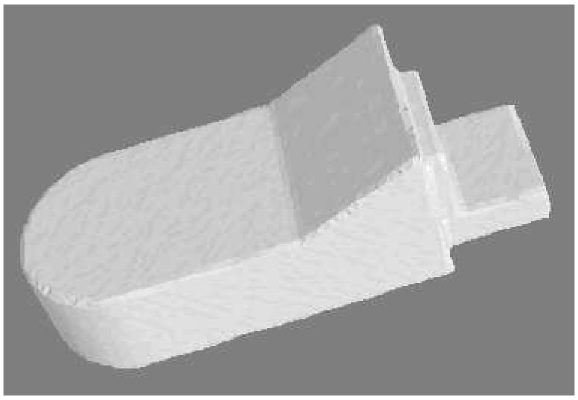
(a) initial registration of rectangular box



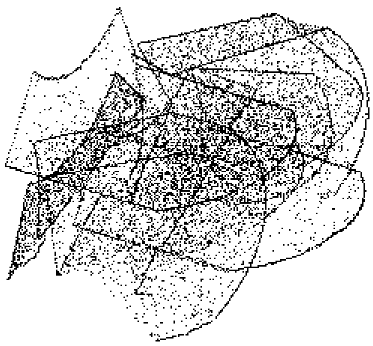
(b) triangulation of registered box



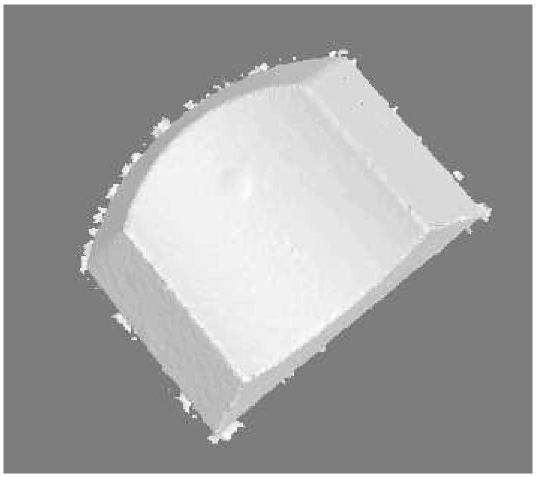
(c) initial registration of widget 1



(d) triangulation of registered widget 1



(e) initial registration of widget 2



(f) triangulation of registered widget 2

Figure 5: Registration results for three objects, two of them synthetic (top and middle) data sets, the other (bottom) data from a real sensor. Initial positions of the point sets are shown in the left column, while triangulations of the registered data are depicted on the right.

fault of the registration algorithm, but rather it is due to the data obtained by the sensor. Contributing factors for the erroneous data are: this particular object is metallic, a troublesome material for most laser scanning devices; laser scanners are known to produce worse data at depth discontinuities due to an averaging of the response over two surfaces; and steeply angled surfaces with respect to the sensor are foreshortened because edges are not sampled as closely as on surfaces perpendicular to the viewing rays. Here, the algorithm has settled on a registration which aligns the common observed areas. The non-common portions then protrude from the edges, resulting in the extra small patches. This happened mainly along the border of the bottom of the object, which appeared larger in one view than the others.

The final example, shown in Figure 6, depicts the results for the well-known Renault part. Here the positions of the point sets after registration at the intermediate subsampling resolutions are also given. As can be seen, the majority of the rotational alignment is achieved using the smallest subsamples. Using the next size up completes a majority of the translational movement, while the final two passes perform minor adjustments. The average motion on the first pass was $15°$ in each rotation dimension and 125 units in the translation dimensions. The motion of the second pass was approximately 10% of this amount, while the amount of motion in the final two passes was only about 1% of that in the second pass.

The quality of the final registration can be seen in comparing the image of the triangulation (Figure 6.f) to an actual photo of the object (Figure 6.e). Here the object was painted before acquiring the range data to reduce specular reflections where possible, obviously improving the quality over that in the previous example. But, there are still a few outlying clusters of erroneous points which the median filter was not able to remove. However, notice how the mold lines of the part have been registered nicely, while there are no other surface discontinuities that would indicate a misregistration.

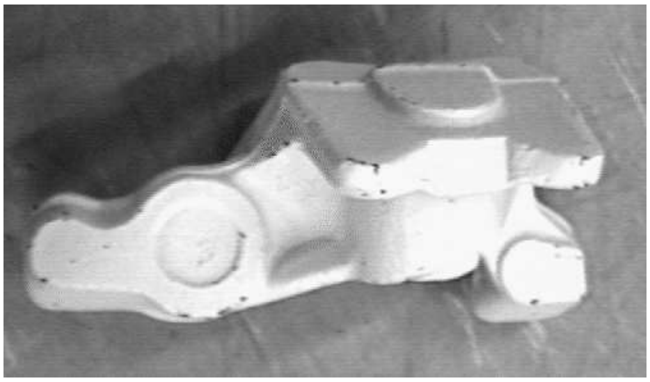
A full summary of the registration results for all four objects is given in Table 3. Of first interest are the columns stating the sampling resolution and the average distance of a point from the tangent plane of its corresponding point. Here, the point to plane distances are 10 - 15 % of the sampling resolution for the synthetic data sets, and 20 - 25% for the real data sets. This is an encouraging result, since in general processes that attempt to locate features to accuracies less than the sampling rate (for instance, subpixel interpolation), rarely achieve below 10% of the sampling rate.

(a) initial alignment of 10 views

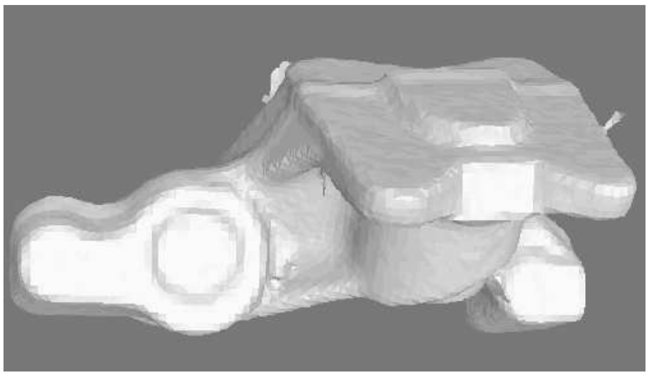
(b) registration of 100 points per view

(c) registration of 1000 points per view

(d) registration of 10,000 points per view



(e) actual photo of imaged Renault part



(f) rendered image of triangulation of data points

Figure 6: Registration results for the Renault part. The position of the point sets after each stage are shown, along with the final triangulation. For comparison, an actual image of the part is also shown beside it.

A slightly more discouraging statistic is execution time. The table clearly shows the non-linear dependence on the data set size. The simple box with only 30,000 points needed only 8 minutes to process on a Sparc 5 workstation, but the Renault part, containing half a million points, required a full day. However, when the time for the Renault part is broken down, about ten minutes was spent in the initialization phase, and a similar amount of time in the first two passes in which 99% of the data movement occurred. Thus, a fairly large decrease in the execution time could be achieved at the expense of only a little accuracy.

Table 3: Results of registration algorithm on 3-D objects.

Object	type	# views	view size	total points	sampling resolution	avg point distance	# of iterations	time
box	synthetic	8	128 x 128	31,464	0.0497	0.00738	18	8 min
widget # 1	synthetic	8	256 x 256	151,380	1.397	0.139	31	1 hr
widget # 2	real	8	$\sim$ 250 x 250	273,730	0.610	0.121	33	3 hr
Renault part	real	10	$\sim$ 225 x 400	471,760	0.663	0.174	24	1 day

It is difficult to compare the results given here to those of previous efforts, either because completely different data sets were used, or few actual numbers have been reported. Only two of the efforts performing multiple view registration have reported quantitative accuracy and timing results. Blais and Levine stated that the processing of six views (of size 256×256) of an owl figurine yielded an average distance between corresponding points of 1.55 mm for images with a point measurement error (related to the sampling resolution) of 0.625 mm. This took 83 hr to compute on an SGI workstation. The initial registrations were off by approximate 4° in rotation and 8 mm in translation. This suggests superior results are being obtained by our algorithm.

Gagnon *et al.* processed 8 views of a teapot, each containing approximately 10,000 points (the sampling resolution was not stated). From an initial registration having an average point distance error of a few tenths of a millimeter, they achieved a final registration with a point error of less than 10 micrometers, an order of magnitude improvement. This took approximately 30 minutes to compute on a Sparc 10 workstation. Again this indicates either comparable or superior performance by the algorithm given here.

Conclusions and Future Work

In this paper an algorithm was presented for performing a refinement on an initial set of transformations that register multiple range views of an object. This algorithm has several advantages over previous registration methods.

- The radius of convergence is larger than that of previous efforts. Errors in rotation of 20° and translations up to 25% of the object size can still be compensated for properly on most objects. However, characteristics of the object shape, such as rotational symmetry, are the deciding factors in determining the particular radius of convergence.

- The set of transformations is solved for simultaneously, rather than pairwise incrementally, leading to a better global solution.

- Correspondence is not determined on a pairwise view basis. The use of a global data set eliminates the need for distance thresholds (assuming each part of the object has been seen at least twice). This also implies that the views need not be part of an actual sequence in which the changes between views are small.

- Using k-d trees, in combination with lists of previous correspondences, increases the performance of the correspondence process considerably.

- Extended point features such as surface normals are used in early passes to help convergence, but only point positions are used at the end due to the inherently noisier values of the normals.

- The concept of linking a point to a surface tangent via a spring leads to less restrictive motion calculations that should eliminate premature convergence. However, additional special correspondences between occluding points and hidden surface regions are necessary to ensure motion parallel to these tangent planes.

- The use of spring forces between corresponding locations allows for the use of a dynamic simulation as a search method for the proper transformations, which has proven to be fairly robust with respect to initial transformation variations.

- The accuracy of the final registration is on a par with most other sublocalization algorithms, approaching 10 - 25% of the sampling resolution.

- The use of hierarchically-sized data sets leads to quicker convergence. Faster times and slightly less accurate results can be achieved if the full data set is not processed.

Given this, there are still avenues open for future work. These could include:

- The use of uncertainty in sensor readings to determine point weightings as by others[1,4,5,13] could lead to potentially more accurate results.

- A further analysis of the use of curvatures as a point correspondence feature, especially on curved objects, is needed to see if potential benefits can be made to outweigh drawbacks.

- Better convergence can always be achieved by improving the correspondence computation. An investigation is planned into whether using a signed distance function to estimate the surface, as in the triangulation method of Hoppe *et al.*, can lead to better correspondences.

- Try to relax the assumption that each portion of the object surface be viewed at least twice so that possibly fewer views may be used. Currently if a surface portion is seen only once, the algorithm may still perform well (i.e., when the surface area in question is quite uniform and there is other constraining data visible in the view). Otherwise, a less accurate model in the singly viewed areas may result due to the lack of data redundancy or incorrect correspondences.

REFERENCES

1 Besl, P J and McKay, N D 'A method for registration of 3-D shapes' *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol 14 No 2 (1992) pp 239-256

2 Potmesil, M 'Generating models of solid objects by matching 3D surface segments' *Proceedings of the 8th International Joint Conference on Artificial Intelligence* Karlsruhe, West Germany (Aug 1983) pp 1089-1093

3 Blais, G and Levine, M D 'Registering multiview range data to create 3D computer objects' *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol 17 No 8 (1995) pp 820-824

4 Zhang, Z 'Iterative point matching for registration of free-form curves and surfaces' *International Journal of Computer Vision* Vol 13 No 2 (1994) pp 119-152

5 Feldmar, J and Ayache, N 'Rigid and affine registration of smooth surfaces using differential properties' *Proceedings of the 3rd European Conference on Computer Vision* Stockholm, Sweden (May 1994) pp 397-406

6 Masuda, T and Yokoya, N 'A robust method for registration and segmentation of multiple range images' *Computer Vision and Image Understanding* Vol 61 No 3 (1995) pp 295-307

7 Chen, Y and Medioni G 'Object modelling by registration of multiple range images' *Image and Vision Computing* Vol 10 No 3 (1992) pp 145-155

8 Lu, F and Milios, E E 'Robot post estimation in unknown environments by matching 2D range scans' *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Seattle, WA (June 1994) pp 935-938

9 Bergevin, R, Laurendeau, D and Poussart, D 'Registering range views of multipart objects' *Computer Vision and Image Understanding* Vol 61 No 1 (1995) pp 1-16

10 Gagnon, H, Soucy, M, Bergevin, R and Laurendeau, D 'Registration of multiple range views for automatic 3-D model building' *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Seattle, WA (June 1994) pp 581-586

11 Dorai, C, Weng, J and Jain, A K 'Optimal registration of multiple range views' *Proceedings of the 12th IAPR International Conference on Pattern Recognition* Jerusalem, Israel (Oct 1994) pp 569-571

12 Champleboux, G, Lavallee, S, Szeliski, R and Brunie, L 'From accurate range imaging sensor calibration to accurate model-based 3-D object localization' *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Champaign, IL (June 1992) pp 83-89

13 Turk, G and Levoy, M 'Zippered polygon meshes from range images' *Proceedings of SIGGRAPH 94* Orlando, FL (Jul 1994) pp 311-318

14 Menq, C H, Yau, H T and Lai, G Y 'Automated precision measurement of surface profile in CAD-directed inspection' *IEEE Transactions on Robotics and Automation* Vol 8 No 2 (1992) pp 268-278

15 Preparata, F and Shamos, M *Computational Geometry: An Introduction* Springer Verlag (1985)

16 Sabata, B and Aggarwal, J K 'Estimation of motion from a pair of range images: A review' *Computer Vision, Graphics and Image Processing: Image Understanding* Vol 54 No 3 (1991) pp 309-324

17 Lorusso, A, Eggert, D W and Fisher, R B 'A comparison of four algorithms for estimating 3-D rigid transformations' *Proceedings of the 6th British Machine Vision Conference* Birmingham, UK (1995) pp 237-246

18 Hill, A, Cootes, T F and Taylor, C J 'Active shape models and the shape approximation problem' *Proceedings of the 6th British Machine Vision Conference* Birmingham, UK (1995) pp 157-166

19 Kardestuncer, H *Finite Element Handbook* McGraw-Hill, New York (1987)

20 Hughes, J H and Martin, K F *Basic Engineering Mechanics* Macmillan (1977)

21 Hoppe, H, DeRose, T, Duchamp, T, McDonald, J and Stuetzle, W 'Surface Reconstruction from Unorganized Points' *Computer Graphics* Vol 26 No 2 (1992) pp 71-78