Simultaneous Registration of Multiple Range Views For Use In Reverse Engineering of CAD Models[1]

David W. Eggert
Department of Computer Science
University of New Haven
West Haven, CT 06516

eggert@vision.newhaven.edu

Andrew W. Fitzgibbon
Department of Engineering Science
University of Oxford
Oxford, United Kingdom OX1 3PJ

awf@robots.ox.ac.uk

Robert B. Fisher
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland EH1 2QL

rbf@dai.ed.ac.uk

Abbreviated title: Simultaneous Registration of Multiple Range Views

Address for correspondence: David Eggert
Department of Computer Science
University of New Haven
300 Orange Ave.
West Haven, CT 06516
Phone: (203) 932-7097
Fax: (203) 932-7261

[1]This work was funded by UK EPSRC Grant GR/H/86905. A shorter version of this manuscript [1] recently appeared at the 1996 International Conference on Pattern Recognition.

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 in several images, further refinement is achieved using an improved version of the recently popular Iterative Closest Point algorithm. Improved 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, by considering the data sets as implicitly 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 images.

List of Symbols

$\mathcal{T}$ - view transformation set
$\boldsymbol{T}$ - 3-D transformation
$\boldsymbol{R}$ - transformation rotation matrix
$\boldsymbol{t}$ - transformation translation vector
$\boldsymbol{I}$ - identity matrix
$\boldsymbol{l}$ - 3-D location
$\boldsymbol{v}$ - linear velocity
$\boldsymbol{\mu_v}$ - linear velocity damping matrix
$\boldsymbol{a}$ - linear acceleration
$\boldsymbol{\theta}$ - rotation angle
$\boldsymbol{\omega}$ - angular velocity
$\boldsymbol{\mu_\omega}$ - angular velocity damping matrix
$\boldsymbol{\alpha}$ - angular acceleration
Δt - simulation time step
N_p - number of sampling passes
$\boldsymbol{S_S}$ - point sampling schedule
δ_m - minimal motion threshold
E - distance error metric
s_n - normalization scale factor
w_n - point normal weighting factor
b_{vp} - point visibility weighting factor
β - surface normal angle tolerance
k - dimension size of feature vector
N - number of points in a data set
n_v - number of corresponding points in view
n_o - number of corresponding points in all other views
$\boldsymbol{p}$,$\boldsymbol{q}$ - corresponding data points
$\boldsymbol{n_p}$,$\boldsymbol{n_q}$ - corresponding surface normals
$\boldsymbol{F}$ - spring force
λ - spring strength
d - distance between two points
$\boldsymbol{M}$ - rotational moment
m_v - mass of points in data set
I - moment of inertia
$\boldsymbol{r}$ - radius for moment
$\boldsymbol{c_m}$ - center of mass of data set
d_d - depth discontinuity threshold
σ - Gaussian kernel size

1 Introduction

With the increased abilities of Computer Aided Manufacturing (CAM) systems over the past decade, it is becoming more and more desirable for companies to develop computer databases of their inventory. Computer models of components can now be directly used in the creation process by numerically-controlled milling machines. While most new parts are now created using a Computer Aided Design (CAD) system, the need for models of old parts is a problem. 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 a 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 images 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 using the aligned 3-D points. Typical surface forms include a triangulated mesh, extracted quadric surfaces, and various spline representations.

The level of quality of the final model is directly related to the inaccuracies of each step: the sampling resolution between data points and the inherent noise of the sensor, the error in the computed alignment transformations, or the residual error of a surface fit to the registered 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 adjusting the desired number of triangular patches in the case of a mesh, or increasing the order of the surface being fit to the data. Regardless of how one reduces these errors, if the shape information from the various views is not properly aligned, the result is merely 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 performed in early systems. But this quality depends on finding a very accurate set of features. Also, by its very nature, vast amounts of helpful information in the data set are ignored. Thus, additional techniques which use the underlying point data have recently been developed to further improve upon the initial estimates.

In this paper a point-based data registration refinement process is examined. In the next section many of the 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, as incorporated in a force-based optimization. Results of processing both 2-D and 3-D data sets with this algorithm are given.

2 A History of 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 [2]. 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 each set. 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 still performs quite well and has been shown to monotonically converge under the stated assumptions [2, 3]. But there are two major drawbacks.

First, 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. Secondly, it is not obvious how the two set approach can be extended to handle multiple data sets. Attempts at solving these two problems have led to several variants of the original algorithm.

2.1 Improving correspondence

The first improvement to the basic algorithm changes the simple point-to-point correspondence used in many of the methods [2, 4, 5, 6, 7, 8, 9, 10, 11], to that between a point and a location on the "surface" represented by the other data set. This potentially increases the integration accuracy beyond that of the sampling resolution.

The first such effort was due to Chen and Medioni [12]. They begin by finding the data point in the second set that is closest to a line through the point in the first set in the direction of its estimated surface normal. Then, the tangent plane at this "intersection" point is used as the surface approximation. The first set's data point is projected onto this plane to give the corresponding location. This technique has subsequently been used in other approaches [13, 14, 15, 16]. A further minor improvement by Dorai *et al.* [17] 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 [18], triangular meshes [3, 19, 20], and parametric surfaces [3, 21].

Most researchers have used the simple Euclidean distance in determining the closest point [2, 3, 5, 6, 8, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21]. Fewer have used higher dimensional feature vectors, such as including the estimated surface normal [13], principal curvatures of the surface [4, 7], and other surface properties [9]. By properly weighting the different components of the feature vector, as done by Feldmar *et al.* [7, 9], fewer incorrect correspondences can be obtained during early iterations when points are farthest apart.

2.1.1 Thresholding outliers

Most of the early algorithms were limited by the original assumption that one data set was a subset of the other [2, 3, 4, 8, 11, 12, 14, 17, 20, 21]. Proposals to bypass this limitation have involved imposing a heuristic threshold on either the distance allowed between points in a valid pairing [5, 6, 7, 9, 10, 13, 18, 19], the deviation of the surface normals of corresponding points [13, 15], and rigidity constraints between pairs of correspondences [16]. Any point pairs which exceed these thresholds or constraints are assumed to be incorrect. These thresholds are usually predefined constants related to the estimated accuracy of the initial transformations, and can be difficult to choose robustly [5, 7, 13, 16, 18, 19]. Dynamically adjustable thresholds have been based on both the distribution of the distance errors at each iteration [6, 9], and a fuzzy set classification of inlier and outlier correspondences [10].

2.1.2 Computational requirements

In all of the techniques, computing potential correspondences is generally the most time consuming step. In a brute-force approach [2, 3, 10, 13, 21], an $O(N^2)$ number of comparisons is performed to find N pairings. One way to reduce the actual time, with a potential loss of accuracy [3], is to subsample the original data sets. Criteria for subsampling include taking a simple fraction of the original number [4, 5], using multiple scales of increasing resolution [19, 20], producing subsets based on potential visibility under the current transform [15], or taking points in areas away from surface discontinuities [12, 16, 17], in areas of fine detail [14], and in small random sets for robust transform estimation [8]. A more accurate and slower alternative is to use the full original data sets, but organize the closest point search using efficient data structures such as the octree [18] and k-d tree [6, 7, 9, 20]. The k-d tree [22] is even efficient, $O(N\ logN)$, when higher order features of the points are incorporated in the distance metric [7, 9].

2.2 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 [23, 24] to minimize the distances between corresponding points [2, 3, 6, 8, 9, 10, 12, 13, 14, 15, 16, 17, 19]. In certain cases [2, 6, 7, 11, 19], individual 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 *et al.* [8, 20]. Alternatively, a Kalman filter has been used to track the intermediate motion at each iteration as new correspondences are computed [6, 7].

More involved techniques compute the motion transform via some form of search in the space of possible transforms, trying to minimize a cost function such as the sum of distance errors across all corresponding points. Movements in transform parameter space are computed based on the changing nature of the function. Such standard search strategies as Levenberg-Marquardt [18, 21] and simulated annealing [5] have been used, in addition to others more heuristic in nature [4, 11, 14]. 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.

2.2.1 Initialization and convergence of searches

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 [3, 6, 8, 9, 10, 11, 12, 13, 20, 21], possibly calculated using feature sets. Other prior estimates can be given by a rotary table [4, 15, 18], a robot arm [5], or even the user [16, 19]. Most such estimates are assumed to be quite accurate so that using one of the various distance thresholds during matching will prune outliers correctly.

Other researchers do their own feature-based alignment prior to refinement using such characteristics as principal moments [2] or axes [16, 17], normals of distinctive points [4], positions of points with distinguishing principal curvatures [7] or similar triangles on a mesh representation of the data [14]. If these distinguishing features are absent, a uniform distribution of starting points can always be processed [2].

All of the ICP 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 [6, 14, 15], or the distance between corresponding points becomes suitably close [2, 3, 8, 9, 10, 12, 13, 16, 17, 19, 20]. The Kalman filter approach stops when the uncertainty in the computed transform reaches a desired level [7]. The iterative searches of parameter space [4, 5, 11, 18, 21] 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. Any method can be terminated if convergence is not detected after some maximal number of iterations.

2.3 View pairs vs. multiple views

The majority of the discussed techniques [2, 3, 6, 7, 8, 9, 10, 13, 14, 16, 17, 18, 21] 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 consecutive pairs could be performed [4, 16]. However, any errors in these computations will accumulate, leaving the first and last frames in the sequence rather poorly aligned. Certain methods, which search for a more globally optimal set of transforms, have since been developed.

The first of these, by Turk and Levoy [19], assumes that an additional continuous cylindrical data scan is available. Individual linear scans are registered to this image, which should have commonalities with each of them. Unfortunately, not all scanners can produce such a base image. Chen and Medioni [12] and Masuda *et al.* [20] incrementally register data from

successive views into a growing combined set. While matters are improved, early calculation errors are still not corrected.

Three other techniques have attempted to compute the motion transforms simultaneously. In the first, Blais and Levine [5] define a consecutive set of transforms between pairs of images in the sequence, including a transform between the first and the last images, taken as the composition of the intermediate transforms. They then minimize a total cost function involving all the image pairings by searching in the large, combined transformation space. This high dimensional minimization is often difficult. In the second technique, Bergevin *et al.* [15] consider each view as being transformed into a common coordinate frame. Then each view's data can be matched to each other set through composed transforms. The combined set of correspondences from all possible pairings is used to compute each data set's motion using a least squares computation at each iteration. While these two methods do compute the set of transforms simultaneously, they still suffer from the thresholding difficulties in determining pairwise correspondences.

The final method, developed independently by Stoddart and Hilton [11], is most similar to that presented in the next section. They also perform a force-based optimization by simulating the interconnection of the data sets with springs between corresponding points. The exact method of solving this physical system's equations is different from that here. Their key assumption is that all proper correspondences are known ahead of time, a rather strong assumption.

In summary, global optimization techniques have been developed, but they still rely on pairwise correspondence computations (or assume that they are already known). These computations suffer from the need to choose appropriate thresholds that reject incorrect pairings. In the following section a new global registration technique is presented which addresses the major problems of simultaneous optimization and distance thresholds, through a combining of some of the best features of previous techniques with new ideas.

3 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 in a common coordinate frame. 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 both point position and surface normal information. Closest point searches are performed using a combined data set in which every point should have a corresponding point (a basic assumption), eliminating the need for a distance threshold (extremely noisy data points should get removed in a preprocessing step). K-d trees are used to speed up point searching, and data point projections onto tangent planes of corresponding points will 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 compute 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. Final 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.

3.1 Initialization

Several operations are performed before the actual iterative process begins. First a median-type filter is used to remove noisy pixels in each range image based on the characteristics of a neighborhood (7×7) of depth values in the scanning grid. Following this, points on the occluding side of step discontinuities are detected, where a discontinuity is said to occur for points separated by a distance which is more than a multiple of the average image sampling

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 ($\mu_{\mathbf{v}}$, 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.

resolution (here, the depth discontinuity threshold is $d_d = 10 * \text{sampling resolution}$). These occluding points will be important in constraining proper alignment of surface boundaries in a manner not used in previous algorithms.

Next, the remaining valid data is smoothed with a Gaussian kernel ($\sigma = 2.5$) before surface properties are computed. All three position components (x, y, z) are independently calculated based on positions of neighboring points in a 7×7 window of 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 this normal calculation. The original data is used in all other computations.) Principal curvatures were not used in this implementation. Initial tests indicated that the inherently noisy nature of these second-order features, plus the additional computation burden, outweighed their potential discriminatory power. In contrast, the potential usefulness of the surface normals justified their computations.

Lastly, we must initialize the state of the dynamic spring system, which is represented by the vector $\{ \mathcal{T}, \boldsymbol{v}, \boldsymbol{\omega}, \boldsymbol{a}, \boldsymbol{\alpha}, \boldsymbol{\mu_v}, \boldsymbol{\mu_\omega}, \lambda, \Delta t, N_p, \boldsymbol{S_S} \}$. It is assumed that the starting view transformations, $\mathcal{T}$, 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 approximate without preventing proper convergence. In the beginning it is also assumed that each data set is at rest. And so the linear and angular velocities are zero ($\boldsymbol{v} = 0$ and $\boldsymbol{\omega} = 0$), as are the corresponding accelerations ($\boldsymbol{a} = 0$ and $\boldsymbol{\alpha} = 0$). The remaining state consists of inertial properties of the data sets (the velocity damping factors, $\boldsymbol{\mu_v}$ and $\boldsymbol{\mu_\omega}$), the spring strength (λ), simulation step size (Δt), number of subsampling passes (N_p), and the actual sampling schedule ($\boldsymbol{S_S}$). Initially, there is no velocity damping, the spring strength is constant at $\lambda = 1000$, and the simulation time step starts with $\Delta t = 0.005$. The number of subsampling passes for the simulation is computed as a function of the total number of points in each view. And from this the actual sampling schedule is determined as discussed in a later section.

3.2 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 [7], 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 box in 3-D surrounding the combined data sets to the maximal normal difference (a value of 2 for unit normals). The value of w_n is used to control the contribution of normal information over time (this is chosen to be steadily decreasing, the exact schedule is discussed later).

Eq. (1) is not the only distance metric used. It will be shown in later experiments that using only simple point correspondences as with Eq. (1) can lead to premature convergence. An additional metric is used for those points, $\boldsymbol{p}$, labeled as occluders in the range image as:

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

Here, b_{vp} has a value of one if the surface normal of a 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.

3.2.1 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 in the whole of the data sets; 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 the current approach.

This should not be interpreted to mean that the closest points identified by the metrics in Eqs. (1) and (2) are necessarily the true corresponding points. Depending on the amount of error in the motion transforms, many of the determined correspondences are incorrect. However, if a majority of these points are (1) correctly corresponded or (2) correspond to a point in the direction of the correct point, then the computed interpoint forces described in the next subsection should pull the point sets into closer convergence. As long as the registration is approximately correct, then any errors caused by mis-correspondences should cancel out to some degree. Only when the transforms are so much in error that the majority of determined correspondences associate points to others in the wrong direction is convergence unlikely to occur.

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 for best performance. 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 within 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 that are 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 in Eq. (1)) 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 value (such as scaling the normal components by w_n in Eq. (1)), does not change the relative positioning

of the points in the partition. Thus a single k-d tree for each view can still be used in many different searches, even as the value of ω_n changes.

Determining the overall closest point is simply a matter of searching the k-d trees of each data set other than the current one to find their corresponding closest points. 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 descending in the tree one moves into smaller and smaller regions of space which contain the search point. At a leaf, which is a region containing only one point, an upper bound on the distance to the closest point can be updated using the distance to the data point in the leaf's cell (actually, the distance norms of Eq. (1) are left squared to reduce computation). In the remainder of the tree traversal, a sub-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 pruning of large sub-trees during the traversal. An effective value comes from the correspondence of the previous iteration. The distance to the old corresponding point under the current transform is a good upper bound. Near final convergence the number of changing correspondences is small, and each search of a 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 for occlusion points according to Eq. (2). A single modification to the above search involves the method of updating 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.

3.2.2 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 an estimate of the surface near the

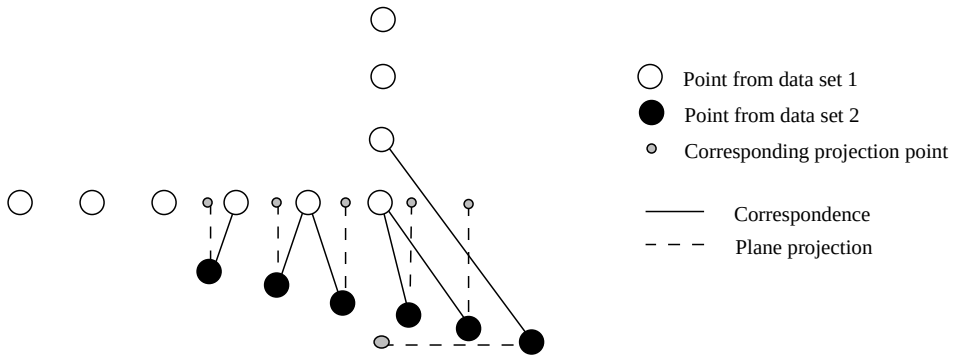


Figure 2: Mapping of corresponding points. The points from data set 2 correspond to the indicated points in set 1 via the solid lines. The points project onto the associated tangents of corresponding points via dashed lines. The rightmost point of set 2 is labeled occluder and therefore maps to the nearest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

corresponding point of a given pairing. A variant of the tangent plane projection method of Chen and Medioni [12] is used. This 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 tangent-plane method is that it is doubly sensitive to any variation 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 to a lesser degree later due to noise. The correspondence search detailed here avoids much of this error. Once the closest point has been found as above, the associated surface location is found by projecting the search point onto the accompanying 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 off to the left, and that the rightmost black point was labeled as an occluding point. The point-to-point correspondences are indicated by the solid lines. Note the rightmost occluding point links to a point around the corner (it does not map to the corner point because the back-facing normal test labels it as "visible"). The projection of points 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 helps align the ends of the lines. If the rightmost point was instead handled

as a non-occluder, it would only map to the corner point, causing the two lines to never overlap properly. Even using direct point-to-point mappings everywhere as in the original ICP process can lead to problems in this example. For instance, as the region of overlap increases over several iterations, 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 these problems will be seen in the experiments.

One way to align the corresponded data sets is to provide attractive forces that pull them together. Connected between each point and its corresponding projection location will be 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 $1000 * (\text{sampling pass})^2$ for occluders. Increasing an occluder's spring constant helps quicken convergence). Each spring force is directed along the line connecting the endpoints and is reflexively applied at both ends. The cumulative effect of the forces from all points will govern the movement of the data sets toward one another.

3.3 Computing motion transforms

Rather than use some of the more popular least squares techniques to compute intermediate motions, a simulation of the dynamic spring system is used. Force-based optimization in different forms [11, 25] has recently been used effectively in registration processes. 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 contributing to the premature convergence problems just mentioned. With a dynamic spring 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 along the way. 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 also been used previously to help solve other alignment problems [26].

One method of simulating this dynamic system would be an exact *finite element analysis* [27], in which each data set truly moves simultaneously. Spring lengths and end positions are constantly changing, resulting in continuously varying forces on each data set. However, the computational complexities of this process are rather involved due to the changing correspondences. In a less exact scenario, forces could be assumed constant over a properly small amount of time, thus allowing the movement of each set to be computed independently for each time step. This movement can be decomposed into a translational movement of the center of mass of a view's points due to a resultant force, along with a rotation of the points about that center due to a resultant torque. We now give the details of an approximate solution to this force-based optimization.

3.3.1 Computations during a single time step

Translational movement is a consequence of the cumulative effect of the forces generated by the springs attached from each data point of a set to the other views' data surfaces, and from the other views' points to locations on the current set's 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 data set's acceleration is then given by:

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

Here, n_v and n_o are the number of points contributing forces in the view's data set and the other data sets combined, respectively. The associated distance between a point and its projection onto a corresponding tangent plane is given by d_i or d_j. Then, if each point is assumed to have unit mass, $m_v = n_v$.

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

$$\boldsymbol{l_t} = \boldsymbol{l_{t-1}} + \boldsymbol{v_{t-1}}\ \Delta t + 0.5\ \boldsymbol{a_{t-1}}\ \Delta t^2 \tag{4}$$

$$\boldsymbol{v_t} = \boldsymbol{v_{t-1}}\ \boldsymbol{\mu_v} + \boldsymbol{a_{t-1}}\ \Delta t \tag{5}$$

Position is updated according to the velocity and acceleration over the given time step. The new velocity is a function of the old velocity and acceleration. An additional factor, $\boldsymbol{\mu_v}$, is included here as a matrix with elements that are zero off the diagonal and damping values for each velocity component along the diagonal. As time passes, the undamped momentum of a data set 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 (simulating a viscous fluid through which the points are moving). The damping can be relaxed carefully 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. In two dimensions the rotation of a body in the plane caused by a single force, $\boldsymbol{F_i}$, applied at point, $\boldsymbol{p_i}$, with respect to the center of mass, $\boldsymbol{c_m}$, is governed by the simple relation, $M_i = I_i\ \alpha_i$, where α_i is the angular acceleration, $I_i = m_v\ \| \boldsymbol{r_i} \|^2$ is the moment of inertia, and $M_i = \| \boldsymbol{r_i} \times \boldsymbol{F_i} \|$ is the moment for the radius, $\boldsymbol{r_i} = \boldsymbol{p_i} - \boldsymbol{c_m}$.

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 [28]):

$$M_x = I_{xx}\ \alpha_x + \omega_y\ \omega_z\ (I_{zz} - I_{yy})$$

$$M_y = I_{yy}\ \alpha_y + \omega_x\ \omega_z\ (I_{xx} - I_{zz})$$

$$M_z = I_{zz}\ \alpha_z + \omega_x\ \omega_y\ (I_{yy} - I_{xx}) \tag{6}$$

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 and assumed constant, Eq. (6) can be rewritten to calculate the angular acceleration due to a set of forces as:

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

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

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

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

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

Here, each component of the angular velocity is damped using the appropriate element of a diagonal matrix, $\boldsymbol{\mu_\omega}$. If the above approximations are valid, these calculations are much simpler and more reasonable, than attempting to directly solve what is actually a 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 Eqs. (4) and (8), one can compute the elements of an incremental transform relative to the center of mass, $\boldsymbol{R_{\Delta\theta}}$ and $\boldsymbol{t_{\Delta l}}$. These can then be converted to incremental transforms in the common coordinate frame as:

$$\boldsymbol{R_\Delta} = \boldsymbol{R_{\Delta\theta}} \qquad\qquad \boldsymbol{t_\Delta} = (\boldsymbol{I} - \boldsymbol{R_{\Delta\theta}})[\boldsymbol{R_v}\,\boldsymbol{c_m} + \boldsymbol{t_v}] + \boldsymbol{t_{\Delta l}} \tag{10}$$

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 latest transform for the view into the common frame. The common frame is initially established by setting all view transforms to the identity.

3.4 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 in the above simulation can be made before new pairings are needed. Thus the motion computation for a given set of correspondences is also iterative, continuing until movement due to the set of forces is minimal (minimal motion, δ_m, is defined as 1% of the total motion so far for the pairings).

3.4.1 Subsampling the data

Larger data sets mean more computation. Therefore, if approximate alignment can be obtained using reduced-size data sets, efficiency is enhanced. However, it has been shown that large data sizes are needed to achieve high accuracy [3], so a hierarchical sampling scheme, S_S, is used. In this algorithm, a starting sample size of 100 points is chosen. The total number of sampling passes, N_p, is determined by increasing this data set size by a factor of ten on each subsequent pass, until on the final pass all of the original data is used. Points for a subsample are selected randomly, with the constraint that a number of occluding points, which are important for boundary alignment, are specially selected so that the quantity is proportional to the number of regular points picked. The respective regular and occlusion based k-d trees are then constructed for each new subsample. Because the mass of a view is based on its number of points, reasonable motion continuity between samplings is maintained as the increased number of forces is balanced by the increased mass.

3.4.2 Adjusting the controlling parameters

Several of the controlling parameters in the previous equations are adjusted either at the start of a subsampling step, or during each motion computation. The parameter, w_n, decides the proportional weighting of surface normal to point position information during correspondence (see Eq. (1)). This value is changed based on the sample size and number of motion steps taken. 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 during this pass, the number of changed correspondences is recorded. Then w_n is set as the ratio of this quantity to 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, and the inherently noisier normal values only tend to interfere with this process.

The next two parameters adjusted are the damping matrices, $\boldsymbol{\mu_v}$ and $\boldsymbol{\mu_\omega}$. Beginning with diagonal values of one (representing no damping), a diagonal component is reduced by 10% each time its 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, while the system is released quickly after things are corrected.

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 per iteration. After each motion step, the velocity damping factors are examined. If a majority of the data sets 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 value of the time step is set at 0.005.

3.4.3 Measuring convergence of sampled data

Convergence of a sampling pass is detected based on the distances between points and their corresponding tangent planes. The distribution of the signed distances of points to planes is computed after each complete motion computation for a set of pairings. If the standard deviation of this distribution continues to decrease then the surfaces are still integrating. If not, the next larger sample is processed until finally convergence is achieved with the original data.

4 Experimental Verification

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

4.1 The necessity of using 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 registration 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 two different initial alignments in Figures 3.a and 3.g. This data is simulated, but contains depth quantization noise. Five different versions of the algorithm are compared in Figure 3, all of which use the force-based motion computations, but point correspondences are determined in different ways. The first is the traditional direct mapping of closest points

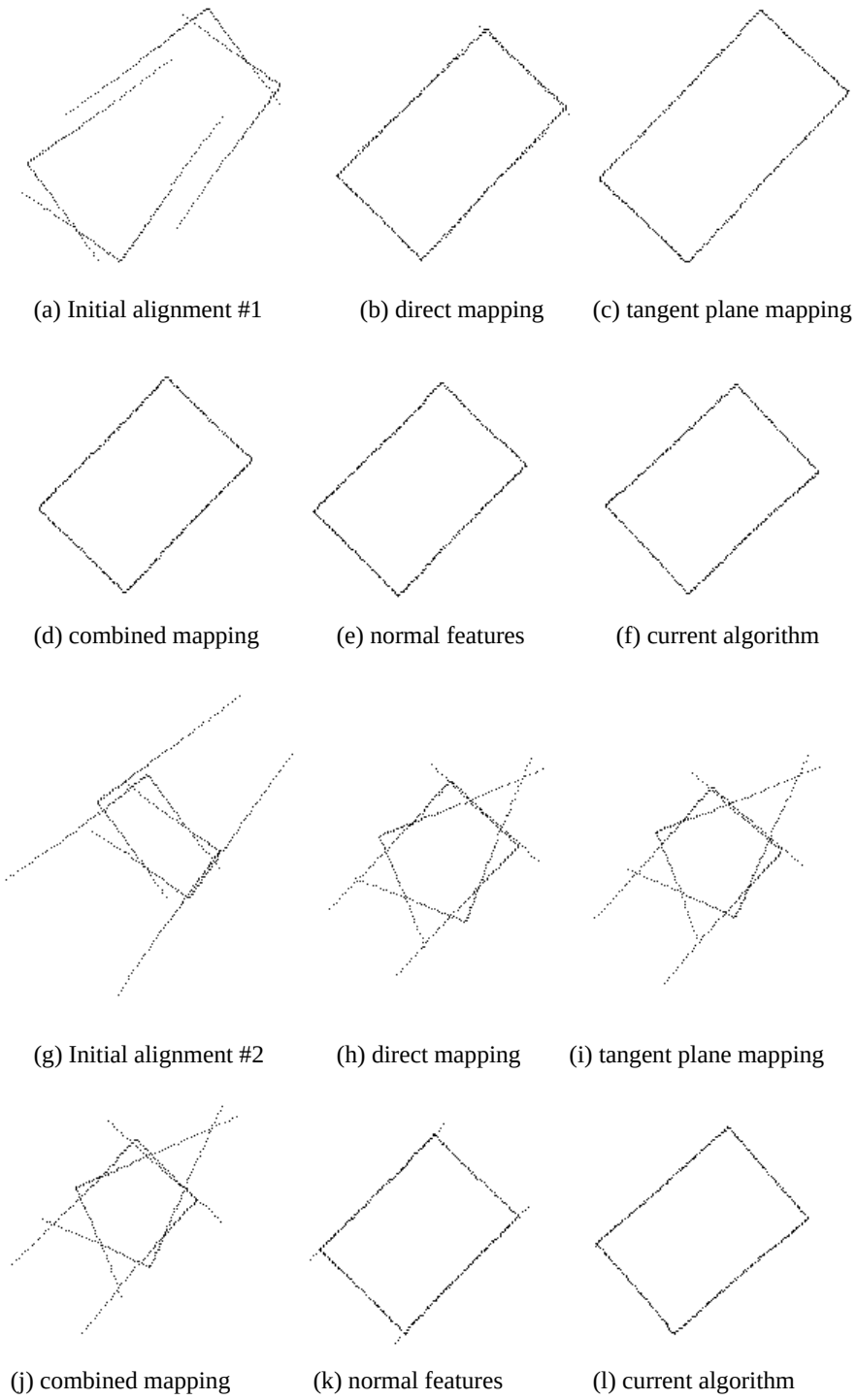
[2]The implementation is available to interested researchers via anonymous ftp using the URL ftp://ftp.dai.ed.ac.uk/pub/vision/3dicp.tar.gz. The implementation includes source code, documentation and a demonstration. Use of the code is freely allowed provided this paper is cited in any resulting publications.

Table 1: Initial values and update methods for algorithm parameters.

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 2 - $w_n =$ # changing correspondences / # points, only if decreasing Pass ≥ 3 - $w_n = 0$
λ	1000 for regular points, $1000 *$ (sample 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

based strictly on position. The second computes projections of points onto their corresponding tangents. The third additionally maps occluding points to the nearest hidden point. The fourth again uses only direct mapping between points, but the surface normal at the point is included in the feature vector as in Eq. (1). The fifth algorithm is what has been presented here and combines all of these elements.

The first example, Figures 3.a - 3.f, shows the results of running the algorithm on data sets which have been pushed out from their desired position. The first two algorithms do not fully contract the data 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, all of the latter three algorithms produce comparably correct results. The second example, Figures 3.g - 3.l, began by contracting the data sets in from their desired position. Here, the use of normal features is a necessity in establishing correspondences that do not bind the edge tips together incorrectly. However, when using only the normal features, these ends still do not come into full alignment. Only the algorithm presented here, which combines normal features and occlusion point mapping, converges upon



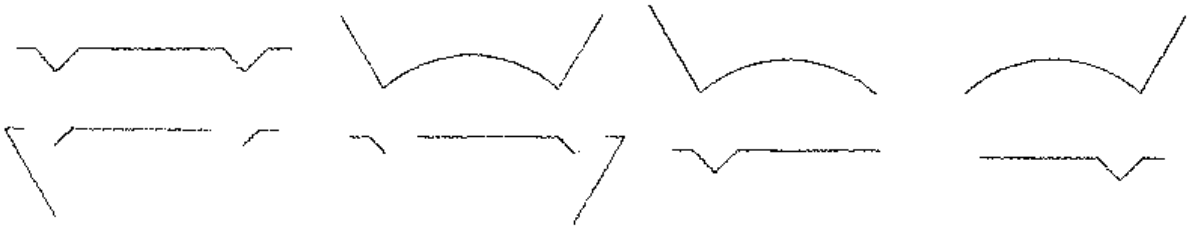
(a) Initial alignment #1 (b) direct mapping (c) tangent plane mapping

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

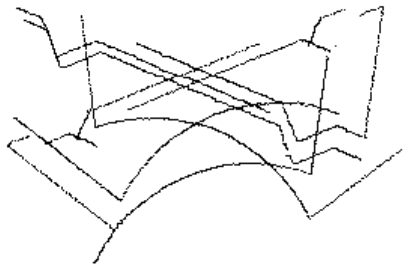
(g) Initial alignment #2 (h) direct mapping (i) tangent plane mapping

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

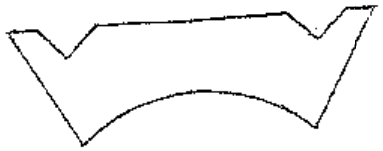
Figure 3: Example of various registration processes using 4 corner views of a rectangle. Each view contains 128 simulated points generated with quantization error. Given the starting positions in (a) and (g), algorithms with 5 different characteristics converged to the registrations shown.



(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: Example of alignment of 8 views of a cross-section of a real object of size $50 \times 125mm$. Each view in (a) contains between 336 and 501 real data 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 the associated final registration in (c).

the correct shape in this instance.

4.2 Convergence properties of the force-based optimization

In this section another 2-D data set is used to examine the convergence properties of the algorithm. This data set, shown in Figure 4.a, consists of eight views of a cross-section of a real 3-D object as seen by a triangulation-based range sensor.

4.2.1 Convergence using the entire data set

The goal of this experiment is to determine the size of the radius of convergence. Three sets of tests were run, the results of which are summarized in Table 2. Before starting these, the algorithm was first run on the data set from a fairly accurate starting position provided by the acquisition process. The resulting answer was said to be the "true" registered position.

Table 2: Results of convergence tests for data sets in Figure 4. For each error setting 25 tests were performed. Average angle and translation errors from "true" converged values were measured, along with the total distance error and standard deviation of the error distribution. 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 combined rotation and translation changes

The data was then disturbed from this configuration in varying amounts and the algorithm run on each positioning.

The perturbation of the data took three forms; only rotation in the plane about the center of mass of the data, only translation of the center of mass in the plane, 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 either rotationally in alignment, or 2θ out of phase. Translation amounts were handled similarly, with the amount of movement being equal to a fraction of the diameter of the data sets. For each level of disturbance 25 data configurations were generated (out of the possible $2^8 = 256$). Each direction of rotation and translation was determined randomly. The three sections of Table 2 show the number of times the "true" registration was achieved, how many iterations the process took, and how close the "correct" converged positions were to one another.

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

1. The average rate of convergence (approximately 20 iterations) is basically independent of the level of perturbation from the starting point. The average number of iterations did not change by more than 20%. However, the absolute rate of convergence is expected to be different for other data sets.

2. The system repeatably converges to the same set of "correct" transforms.The difference between answers for correct convergences was less than 0.1° in rotation and 1.5% of the object diameter in translation. Even this level of difference is bigger than it seems, since small changes in rotation can almost be compensated for with different translations to achieve very similar motion. The average total distance error, as well as the standard deviation of the error distribution, indicate that a consistent minimum was being converged upon. Rather surprisingly, similar error variance was observed in the zero perturbation cases (top lines of each table). This is because the first registration pass, which uses a small random subsample, pulls the search from its starting point, requiring further refinement.

3. 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 expectedly lead to incorrect correspondence. For the object in Figure 4, reliable results were obtained for rotational errors up to ±20° and translational differences up to one third of the object size. An example starting position near the edge of the convergence radius is seen in Figure 4.b, with its corresponding registration in 4.c. The radius of convergence demonstrated by this example appears to be much greater than that of previous methods.

4.2.2 Convergence using subsets of the data

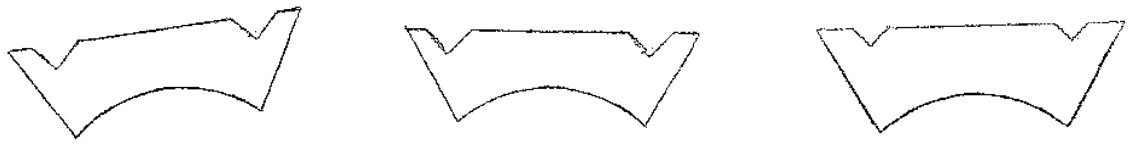
A last set of experiments on 2-D data sets was performed to analyze how important it is to view all portions of the object's surface at least twice. This was done by looking at convergence results generated by using only subsets of the first six views of the 2-D outline in Figure 4. Table 3 shows the average point distance error, over all of the subsets ranging from six views to only two views. These tests were run with initial starting configurations in error by ±15° in rotation and ±15% of the object size in translation. As expected, the error grew as the number of views used decreased. Visually reasonable convergence was achieved on the original set of six views, all combinations of five views, and several of the sets containing only four views. The remaining smaller subsets did not have sufficient surface coverage for the algorithm to respond properly. Figure 5 shows sample convergence images for various subsets. From this data, certain conclusions can be made:

1. For an average object with minimal amounts of small detail, it does not take a large number of views to achieve full surface coverage with significant overlap. The six view convergence in Figure 5.a has at least double surface coverage, and mostly three deep overlap, while the group of four in 5.b is almost two deep everywhere.

2. The algorithm is capable of producing reasonable convergence results in the presence of small amounts of singly-viewed surface portions and even small gaps. Figures 5.b - 5.d all possess single surface coverage of the two indentations in the shape, while the

Table 3: Convergence results using subsets of the data views found in Figure 4.a. Initial positions of views were in error by $\pm 15°$ in rotation and $\pm 15\%$ of the object size. Average error is calculated for all combinations of views for the given subset sizes.

Number of views in data set	Average point distance	Maximum point distance	Minimum point distance
6	0.139	-	-
5	0.255	0.453	0.161
4	1.261	3.024	0.352
3	3.401	6.732	1.741
2	5.578	13.712	1.636



(a) Proper convergence using all 6 views (b) Near convergence using views 1 - 3, 6 (c) Near convergence using views 2 - 6

(d) Near convergence using views 3 - 6 (e) Near convergence using views 1 - 5 (f) Partial convergence using views 2 - 4

(g) Partial convergence using views 5 - 6 (h) Wrong convergence using views 3, 4, 6 (i) Wrong convergence using views 1, 3, 5, 6

Figure 5: Examples of final convergence alignments for subsets of the data views found in Figure 4.a with initial positions in error by $\pm 15°$ in rotation and $\pm 15\%$ of the object size in translation.

rest of the outline contains double or triple coverage depending on the subset of views. All of these data sets still achieved satisfactory convergence. Performance does degrade slightly for the subset in Figure 5.e, in which there is a small portion of the outline that was never observed. But of course there are limits to the amount of singly viewed

surface the algorithm can handle. Figures 5.h and 5.i show typical incorrect convergence alignments that result when an entire side of the outline is seen only once. In these cases the large errors introduced by these singly-viewed points are too overwhelming.

3. The algorithm can be used to some degree on partial surface shapes, as long as the majority of the partial surface has been viewed twice. Figure 5.f shows an example convergence for approximately one half of the shape. Not all of this partial shape had double coverage, and so a noticeable misalignment resulted. Figure 5.g shows a more extreme case in which about half of the observed partial shape was seen only once, causing an understandably poor alignment.

4.2.3 Generalization of convergence results

The absolute quantification of the convergence properties of the algorithm as presented for the given object is, of course, not applicable to all objects. However, similar experiments run on other 2-D shapes have given results that are qualitatively similar to those shown here. The algorithm is most susceptible to initial rotational error, the tolerable amount generally being a function of both the rotational symmetry and uniqueness of the object surface. The less symmetric and more unique the surface, the greater the chance of convergence in general. Initial convergence is usually rapid, whether toward a correct or incorrect transformation set, followed by a longer fine-tuning stage.

For an object of average complexity, a set of 6-8 views can typically produce enough duplicate coverage of the object surface to allow convergence. If the set of singly-viewed surface areas is relatively evenly distributed around the object, this also enables the algorithm to tolerate a greater amount of missing data, although with an expected loss in overall accuracy.

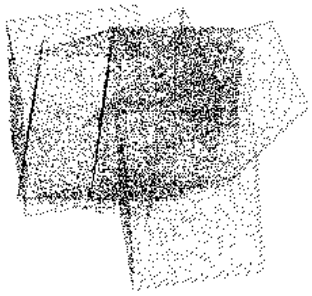
4.3 Analysis of performance on 3-D data

In order to demonstrate that the algorithm also works on 3-D data sets, multiple views of several objects were processed by the system. The first, a rectangular block, is shown at the

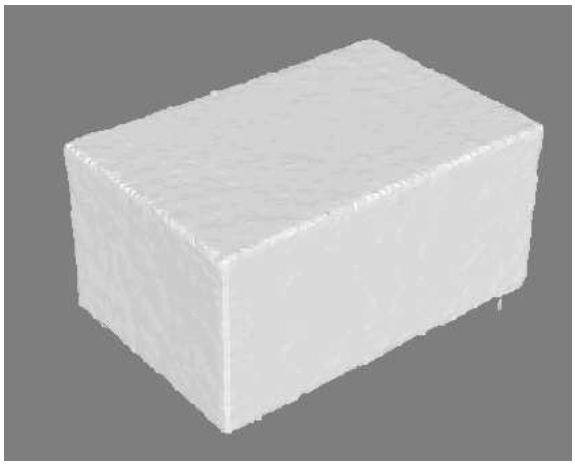
top of Figure 6. Eight images of this object were synthetically generated at a resolution of 128×128, with quantization error introduced in the depth values. The image on the left in (a) shows the point sets of each view in their initial configuration (actually perturbed from the positions of the simulated acquisition by approximately 10° in all rotation angles and 25% of the object size in each translation component). The image on the right in (b) shows the result of a triangulated mesh process due to Hoppe *et al.* [29], as applied to the automatically registered data. This triangulation has two distinguishing features. First, the edges of the block are rounded. This is an artifact of many triangulation schemes, especially that of Hoppe *et al.* Second, there are certain small patches sticking out from the edges. These are a function of the low sampling resolution. Since the simulated viewing rays do not always hit the object exactly along each true edge of the block, straight sample lines may not result. Some of the extremely sampled edge points give rise to the small protruding triangles in the reconstruction.

A second synthetically generated set of images, but this time at a resolution of 256×256, was created for the second object in the middle of Figure 6. Here it can be seen that the triangulation in (d), produced from the registered data in (c), has edges that are less rounded, and the small extra patches are no longer present because of the higher image resolution. Notice also that the presence of small features and a curved surface on this object did not stop the algorithm from determining the proper registration.

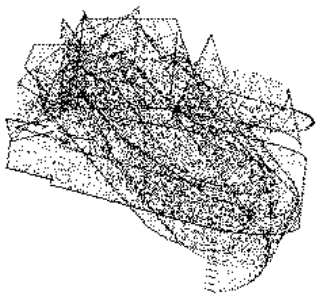
The first example of processing data sets from a real sensor is shown in the bottom of Figure 6 (the ability of the algorithm was again tested by additionally perturbing the data in (e) from the acquisition positions). This object also possesses a combination of planar and curved surfaces. Here it can be seen that the edges of the resulting triangulation in (f) are worse than those in the synthetic cases of (b) and (d). This is not the fault of the registration algorithm, but rather a result of the quality of the data obtained by the sensor. Several contributing factors for this erroneous data were: this particular object is metallic, a troublesome material for most laser scanning devices due to reflections; laser scanners are also known to produce inferior data along depth discontinuities due to an averaging of the



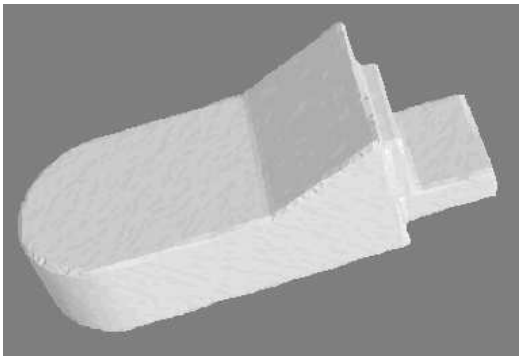
(a) initial registration of rectangular box



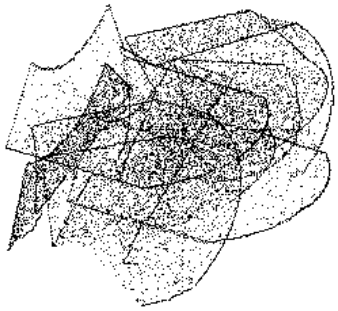
(b) triangulation of registered box



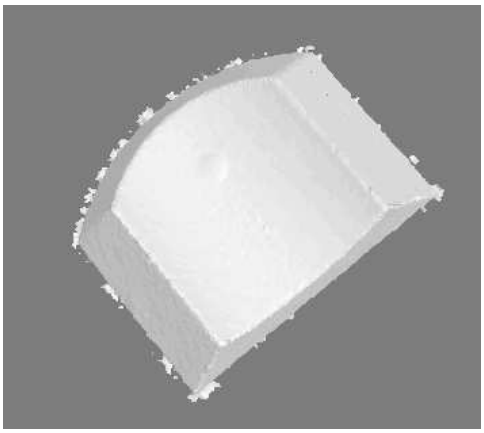
(c) initial registration of widget 1



(d) triangulation of registered widget 1



(e) initial registration of widget 2



(f) triangulation of registered widget 2

Figure 6: Registration results for three objects. Two of the data sets were synthetic (top and middle), while the other data set (bottom) came from a real sensor. Initial positions of the point sets (additionally perturbed from the acquisition locations) are shown in the left column, while triangulated meshes generated from the registered data are depicted on the right.

response over two disparate surfaces; and lastly, steeply angled surfaces with respect to the sensor become foreshortened when the true edges are not sampled as closely as on surfaces more perpendicular to the viewing direction. All these factors lead the algorithm to settle on a registration which aligns the commonly observed areas. The non-common portions subsequently protrude along the edges, resulting in the observed extra small patches. This was most pronounced along the border of the bottom of the object, which appeared larger in one view than in the others.

The final example, shown in Figure 7, depicts the results of processing the well-known Renault part (again initially perturbed in (a)). Here the configurations of the point sets after registration of each intermediate subsample are given ((b) - (d)). As can be seen, the majority of the rotational alignment is achieved using the smallest subsampling. Use of the next resolution completes a majority of the translational movement. The final two passes perform only minor adjustments. The average amount of motion on the first pass was 15° in each rotation dimension and 125 units in the translation dimensions (the part is roughly 1000 units long). The magnitude of motion during 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.

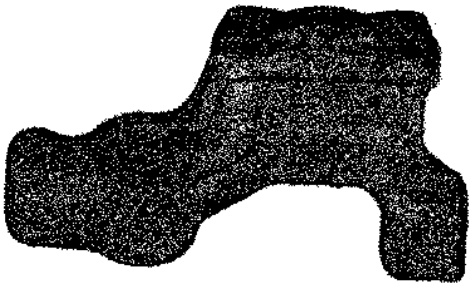
The quality of the final registration is demonstrated by comparing the rendered image of the triangulation (Figure 7.f) to an actual photo of the object (Figure 7.e). For this experiment, the metallic object was painted before acquiring the range data to reduce the effects of specular reflections where possible. This obviously improved the quality over that in the previous example. But, there still exist a few outlying clusters of erroneous points which the median filter did not remove. Even given the presence of these points, notice how the mold lines of the part have been registered nicely, without the introduction of other surface discontinuities that typically indicate a misregistration.

A full summary of the registration results for all four objects is given in Table 4. Of first interest are the columns indicating the sampling resolution of the images and the final

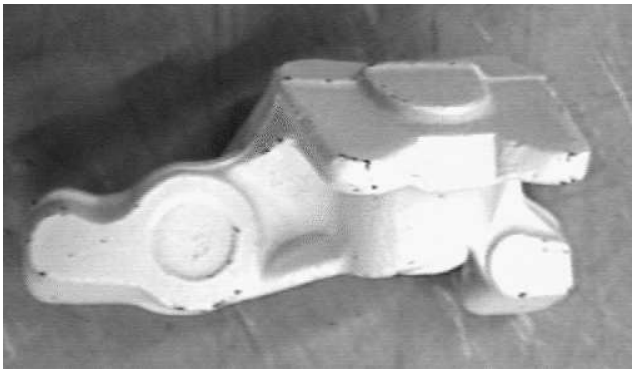
(a) initial alignment of 10 views

(b) registration of 100 points per view

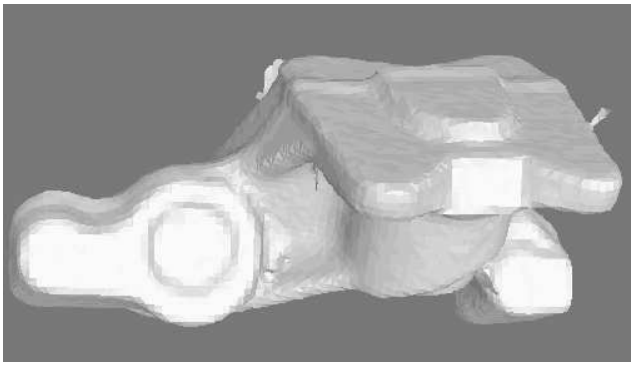
(c) registration of 1000 points per view



(d) registration of 10,000 points per view



(e) actual photo of imaged Renault part



(f) rendered image of triangulation of data points

Figure 7: 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.

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 of the synthetic data sets, and 20 - 25% of the resolution of the real data sets. This is an encouraging result when compared

Table 4: Results of registration algorithm on 3-D objects shown in Figures 6 and 7. Data set statistics are provided. The average point error measurement is given for each object, along with the number of processing iterations and the total execution time.

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

to analogous processes that attempt to locate features to accuracies less than the sampling rate. (For instance, the accuracy of subpixel interpolation of edges rarely falls below 10% of the sampling rate.)

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

4.3.1 A comparison to previous efforts

It is difficult to directly compare the results given here to those of the three previous efforts at simultaneous registration, because the few actual numbers reported are for different error measures on unique data sets. The work of Stoddart and Hilton [11] is the most difficult to compare. Using simulated data sets (with additions of Gaussian noise to the point coordinates) and predetermined correspondences, final error residuals approaching the noise level were achieved in a few iterations given starting transformation errors of 10° in rotation and 20%

of the object size in translation. Only small data sets of less than 100 points were processed, requiring less than a second to complete.

The other two research efforts have reported some quantitative accuracy and timing results on real data. Blais and Levine [5] 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 hours to compute on an SGI workstation. The initial registrations were off by approximately 4° in rotation and 8 mm in translation.

Bergevin *et al.* [15] report results for the processing of sequences of 8 views of a teapot and a toy soldier. Each view contained approximately 10,000 data points with an unstated sampling resolution (see Table 5 for actual values). The error measure given in these cases was the average signed point to tangent plane distance, which is different from the average absolute point to plane distance used here. It is harder to judge the actual convergence accuracy using their measure since the individual errors can cancel out. From initial registrations having an average distance error of a few tenths of a millimeter, they achieved final registrations with errors of less than 10 micrometers, an order of magnitude improvement, indicating that the error distribution had indeed become unimodal. Results for the teapot were slightly superior to those of the toy soldier. Each process took approximately 30 minutes to compute on a Sparc 10 workstation.

In order to try and provide direct comparisons, an attempt was made to acquire the data sets in [5] and [15]. As it turns out, these data sets were created at the same place. Unfortunately the owl dataset found in [5] no longer exists. However, the teapot and toy soldier data, along with two other objects depicted in Figure 8, were available [3]. Unfortunately [4] the form of these data sets violates certain of the fundamental assumptions of the algorithm presented here. These range images were acquired by using a rotary table and imaging at

[3]The authors would like to thank the Visual Information Technology Group of the National Research Council of Canada at Ottawa, Ontario for providing the data sets featured in Figure 8.

[4]This fact was not understood until the data sets were obtained.

Table 5: Results of registration algorithm on 3-D objects shown in Figure 8. Error measurements include the average point error at the nominal alignment provided with the data, as well as final alignment errors when using forces from boundary points (b) and when not using them (nb). Initial alignment errors were 5° in rotation and 10% of the object size in translation.

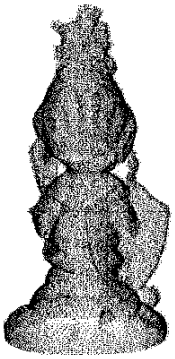
Object	# views	view size	total points	sampling resolution	aligned avg point distance	final avg point dist.(b)	final avg point dist.(nb)
toy soldier	8	256x256	84,813	0.939 mm	0.616 mm	0.449 mm	0.271 mm
pencil sharpener	8	256x256	121,429	0.923 mm	0.427 mm	0.426 mm	0.247 mm
teapot	12	256x256	106,604	1.823 mm	0.999 mm	1.395 mm	0.424 mm
stacked blocks	8	256x256	81,614	0.843 mm	0.709 mm	1.453 mm	0.697 mm

different angular intervals. While these scans did allow the majority of the observed surface to be seen twice, not all of the object was recorded. Most notably the bottom of each object was never seen. And any surfaces parallel to the bottom, or occluded with respect to the stationary camera as the object rotated were missing. In their current state these data sets would not be sufficient to produce an entire object model.

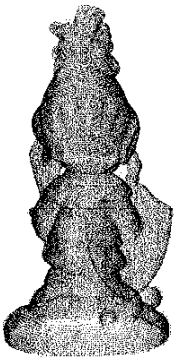
It was seen in an earlier section what effects incomplete data can have on the algorithm. But, as a further demonstration of the algorithm's limits, these data sets were processed anyway. Beginning with modest amounts of error in the initial transforms (5° rotation error, 10% object size translation error, as shown in the leftmost column of Figure 8), the final converged point sets shown in the middle column of Figure 8 were obtained. The level of convergence is depicted here using point distributions rather than triangulations because the algorithm used earlier [29] does not produce very desirable results in the areas of missing data. The number of processing iterations and the total execution time were similar to those used on the widgets, namely, about 30 iterations requiring a few hours. The final error values are listed in the next-to-last column of Table 5.

Looking at the final point distributions it can be seen that the regions of missing data cause poor alignment for most of the objects. The actual final error is basically a direct reflection of the amount of surface missed during the scans. The toy soldier and pencil sharpener have

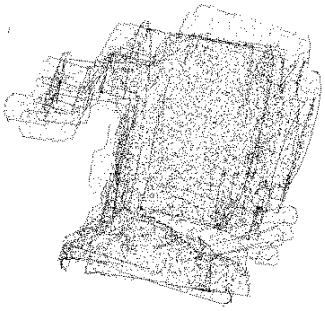
(a) Initial registration of toy soldier



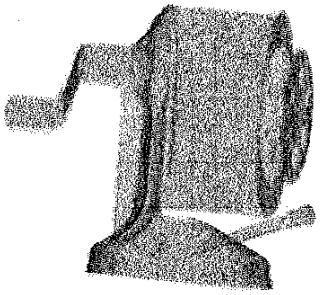
(e) Registered toy soldier (using boundary points)



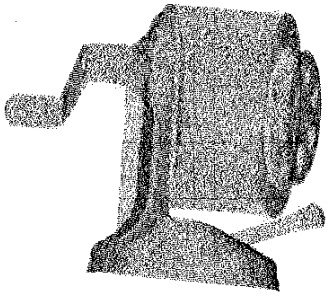
(i) Registered toy soldier (no boundary points)



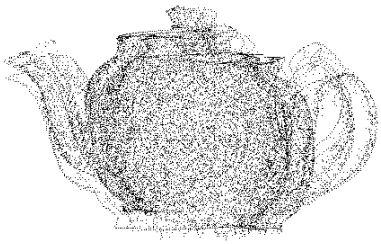
(b) Initial registration of pencil sharpener



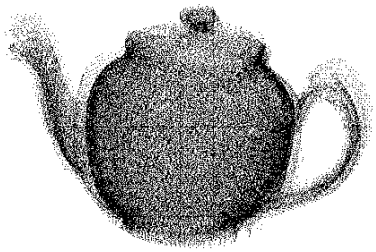
(f) Registered pencil sharpener (using boundary points)



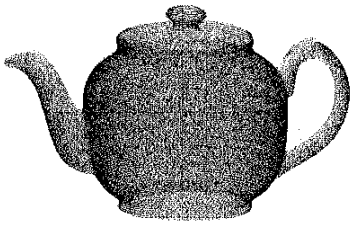
(j) Registered pencil sharpener (no boundary points)



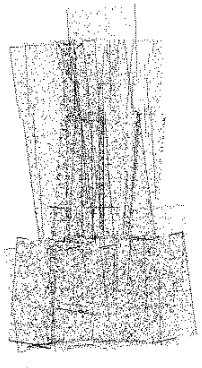
(c) Initial registration of teapot



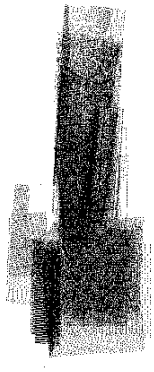
(g) Registered teapot (using boundary points)



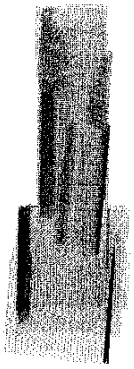
(k) Registered teapot (no boundary points)



(d) Initial registration of stacked blocks



(h) Registered stacked blocks (using boundary points)



(l) Registered stacked blocks (no boundary points)

Figure 8: Registration results of four additional objects. Point distributions in the leftmost column indicate initial data alignments. Those in the middle column indicate convergence results when alignment forces from boundary points were used. The distributions in the rightmost column show the results when these forces were ignored.

sufficiently varying surfaces such that any missing patches are small, aside from the bottom. However, the teapot and stacked blocks have large surfaces parallel to the object bottom which were never imaged. The calculated forces due to the boundary points along these missing areas were large because the proper corresponding points did not exist. This can cause incorrect convergence, as indicated for the stacked blocks.

Given that the boundary points seemed to be the major source of trouble on these images, another processing pass was made in which the forces from these boundary points were ignored. Starting from the same initial configurations, the final alignments shown in the rightmost column of Figure 8 were achieved, the actual errors of which are listed in the final column in Table 5. The degree of improvement varies from object to object. For the toy soldier and pencil sharpener, which converged well previously, only a factor-of-two improvement occurred. This can be seen in the tighter alignment of the surface boundaries in the figure. For the teapot, a great increase in the amount of convergence occurred. All three of these objects have surfaces which vary enough to counteract the missing boundary points. However, the stacked blocks still did not align well. This is because almost all of the surfaces are parallel and aligned along a common axis, which is one of the situations in which the boundary points are most needed.

As a point of reference to determine the quality of these two sets of results, one can look at either the sampling resolutions of the images or the average point distance error of the nominal alignment provided with the data (which was fairly accurate). The version of the algorithm which used forces generated by boundary points improved upon the nominal alignment only for the toy soldier and pencil sharpener objects. The other two, having started at initial positions farther away than the nominal alignment, did not converge as well. The version of the algorithm which ignored boundary point forces improved upon the nominal alignment in all cases (similar error values were obtained whether the process started at the nominal alignment or at the error alignment used here). Unfortunately the error value for the stacked blocks is deceptive. When planar surfaces are involved it is possible to have small point to plane distances, yet still have misalignment as indicated in Figure 8.

When compared to the level of accuracy obtained on the first data sets in Table 4, the final error levels are again about 25% of the sampling resolution. Thus, what appears to be a proper convergence from looking at Figure 8, is verified numerically. The exception is the stacked-blocks object. The combination of axially aligned planar surfaces and large areas of missing surface were too much for the algorithm to handle.

And lastly, even after running the algorithms on common data sets, it is still hard to compare the results given here with those of Bergevin *et al.* [15]. The range of convergence of the algorithm demonstrated here is greater (5° rotation, 10% translation error versus less than 1° rotation and less than 1% translation error). Their stated error values are smaller than that of this algorithm. However, their measure is the average signed distance, whereas ours is the average absolute distance. One would always expect the signed distance value to be less than its absolute counterpart, and it should be considerably less due to error cancellation once a unimodal error distribution has been achieved. Therefore, any lack in performance of the current algorithm is at most minimal. Lastly, the execution times reported here are on a slower machine (roughly half the speed), and rather approximate. Considering that most of the time is spent in the final stages of convergence, our timings could be adjusted to similar lengths with minimal loss of accuracy (much less than 1%). Finally, the display of converged point distributions or triangulations, unless done with the exact same software, can be very deceptive. Therefore, the results presented here are encouraging and indicate that either comparable or superior performance is being achieved by the current algorithm.

5 Conclusions and Future Work

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

1. The radius of convergence is larger, as shown in the experiments. On most objects, the algorithm can still compensate properly for initial errors in rotation of 20° and

translations up to 25% of the object size. However, characteristics of the object shape, such as rotational symmetry, are the true deciding factors in determining a particular radius of convergence.

2. The set of transformations is solved for simultaneously (as some others do), rather than pairwise incrementally, leading to a provenly better global solution.

3. Correspondence is not determined on a pairwise view basis. The use of a global data set for searching 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 typically the changes between camera positions are small. Thus additional clarifying images may be added later without difficulty.

4. Using k-d trees, in combination with lists of previous correspondences, greatly increases the performance of the correspondence search process, especially near final convergence.

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

6. The concept of linking a data point to its corresponding surface tangent via a spring leads to more accurate motion calculations that should eliminate premature convergence. In addition, special correspondences between occluding points and hidden surface regions are necessary to ensure sufficient motion parallel to these tangent planes occurs.

7. The use of implied spring forces between corresponding locations allows for the use of a dynamic physical simulation as a search method for the properly optimized transformations. This has proven to be fairly robust with respect to initial transformation variations.

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

9. The use of hierarchically-sized data sets leads to quicker convergence. Faster times, at the expense of 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:

1. The actual use of uncertainty in sensor readings to determine point weightings as done by others [2, 6, 7, 19] could lead to potentially more accurate results.

2. A further analysis of the use of robust curvature estimates [30] as a point correspondence feature [7] on early sampling passes, especially on curved objects, is needed to see if potential benefits can be made to outweigh drawbacks.

3. Better convergence can always be achieved by improving the correspondence computation. An investigation is planned into whether using recently developed signed distance functions [29, 31] to estimate the combined data set surface can lead to better correspondences without large computational overhead.

4. For many reasons it is desirable to relax the assumption that each portion of the object surface be viewed at least twice. Currently the algorithm may still perform well (i.e., when the singly-viewed surface area is rather uniform and there is other constraining data), but there are limits. Lack of data redundancy may result in misalignments or a less accurate model in the singly-viewed areas. It is hoped that the use of the signed distance surface estimates mentioned above will allow the non-overlapped areas to map to themselves and not obstruct the convergence process. Otherwise, it may be necessary to finally use some adjustable threshold mechanism.

5. A further analysis of the tradeoff between execution time and final accuracy is needed. The rate of convergence curve needs to be examined over all sampling passes.

6. The current algorithm will be extended to apply to other registration problems. Areas of promise being investigated by others currently include constrained forms of non-rigid motion [7], and other data modalities in the medical imaging field [9].

References

[1] D. W. Eggert, A. W. Fitzgibbon and R. B. Fisher, Simultaneous registration of multiple range views for use in reverse engineering, in *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1996, vol. A, pp. 243-247.

[2] P. J. Besl and N. D. McKay, A method for registration of 3-D shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**, 1992, 239-256.

[3] D. Brujic and M. Ristic, Analysis of free form surface registration, in *Proceedings of the IEEE International Conference on Image Processing, Lausanne*, 1996.

[4] M. Potmesil, Generating models of solid objects by matching 3D surface segments, in *Proceedings of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe*, 1983, pp. 1089-1093.

[5] G. Blais and M. D. Levine, Registering multiview range data to create 3D computer objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**, 1995, 820-824.

[6] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, *International Journal of Computer Vision* **13**, 1994, 119-152.

[7] J. Feldmar and N. Ayache, Rigid and affine registration of smooth surfaces using differential properties, in *Proceedings of the 3rd European Conference on Computer Vision, Stockholm*, 1994, pp. 397-406.

[8] T. Masuda and N. Yokoya, A robust method for registration and segmentation of multiple range images, *Computer Vision and Image Understanding* **61**, 1995, 295-307.

[9] J. Feldmar, G. Malandain, J. Declerck and N. Ayache, Extension of the ICP algorithm to non-rigid intensity-based registration of 3D volumes, in *Proceedings of the Workshop on Mathematical Methods in Biomedical Image Analysis, San Fransisco*, 1996, pp. 84-93.

[10] B. Krebs, P. Sieverding and B. Korn, A fuzzy ICP algorithm for 3d free form object recognition, in *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1996, vol. A, pp. 539-543.

[11] A. J. Stoddart and A. Hilton, Registration of multiple point sets, in *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1996, vol. A, pp. 40-44.

[12] Y. Chen and G. Medioni, Object modelling by registration of multiple range images, *Image and Vision Computing* **10**, 1992, 145-155.

[13] F. Lu and E. E. Milios, Robot pose estimation in unknown environments by matching 2D range scans, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle*, 1994, pp. 935-938.

[14] R. Bergevin, D. Laurendeau and D. Poussart, Registering range views of multipart objects, *Computer Vision and Image Understanding* **61**, 1995, 1-16.

[15] R. Bergevin, M. Soucy, H. Gagnon, D. Laurendeau, Towards a general multi-view registration technique, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18**, 1996, 540-547.

[16] C. Dorai, G. Wang, A. K. Jain and C. Mercer, From images to models: Automatic 3D object model construction from multiple views, in *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1996, vol. A, pp. 770-774.

[17] C. Dorai, J. Weng and A. K. Jain, Optimal registration of multiple range views, in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Jerusalem*, 1994, pp. 569-571.

[18] G. Champleboux, S. Lavallee, R. Szeliski and L. Brunie, From accurate range imaging sensor calibration to accurate model-based 3-D object localization, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign*, 1992, pp. 83-89.

[19] G. Turk and M. Levoy, Zippered polygon meshes from range images, in *Proceedings of SIGGRAPH 94, Orlando*, 1994, pp. 311-318.

[20] T. Masuda, K. Sakaue and N. Yokoya, Registration and integration of multiple range images for 3-D model construction, in *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1996, vol. A, pp. 879-883.

[21] C. H. Menq, H. T. Yau and G. Y. Lai, Automated precision measurement of surface profile in CAD-directed inspection, *IEEE Transactions on Robotics and Automation* **8**, 1992, 268-278.

[22] F. Preparata and M. Shamos, *Computational Geometry: An Introduction*, Springer Verlag, 1985.

[23] B. Sabata and J. K. Aggarwal, Estimation of motion from a pair of range images: A review, *Computer Vision, Graphics and Image Processing: Image Understanding* **54**, 1991, 309-324.

[24] D. W. Eggert, A. Lorusso and R. B. Fisher, Estimating 3-D rigid body transformations: a comparison of four major algorithms, *Machine Vision and Applications* **9**, 1997, pp. 272-290.

[25] Y. Kita, Force-based registration method using attribute values, in *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1996, vol. B, pp. 34-39.

[26] A. Hill, T. F. Cootes and C. J. Taylor, Active shape models and the shape approximation problem, in *Proceedings of the 6th British Machine Vision Conference, Birmingham*, 1995, pp. 157-166.

[27] H. Kardestuncer, *Finite Element Handbook*, McGraw-Hill, New York, 1987.

[28] J. H. Hughes and K. F. Martin, *Basic Engineering Mechanics*, Macmillan, 1977.

[29] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, Surface reconstruction from unorganized points, *Computer Graphics* **26**, 1992, 71-78.

[30] F. P. Ferrie, S. Mathur and G. Soucy, Feature extraction for 3-D model building and object recognition, in *Three-Dimensional Object Recognition Systems* (A. K. Jain and P. J. Flynn, Eds.), pp. 57-88, Elsevier Science Publishers, 1993.

[31] A. Hilton, A. J. Stoddart, J. Illingworth and T. Windeatt, Reliable surface reconstruction from multiple range images, in *Proceedings of the 4th European Conference on Computer Vision, Cambridge*, 1996, pp. 117-126.