Fast automatic registration of range images from 3D imaging systems using sphere targets

Marek Franaszek[*,1)], Geraldine S. Cheok[1)], and Christoph Witzgall[2)]

Construction Metrology and Automation Group[1)],

Mathematical and Computational Sciences Division[2)],

National Institute of Standards and Technology, Gaithersburg, MD 20899 USA

Abstract

The use of 3D imaging systems (e.g., laser scanners) in construction has grown significantly in the past decade. Range images acquired with such systems often require registration. This paper describes an automatic method to rapidly locate spheres and perform a registration based on three pairs of matching points (centers of fitted spheres) in two range images. The proposed method is directly applicable for regularly gridded datasets obtained with instruments that are typically used for construction applications and whose maximum ranges are greater than 50 m. A lab was scanned from two locations at three different scan densities. Four spheres were located in the lab, and the total number of points hitting the four spheres was a small fraction (< 0.01 %) of all the points in the dataset. At the highest scan density, the registration of two datasets with 6.4 x 10^6 and 3.4 x 10^6 points is obtained in less than 30 s. At the medium scan density, two range images with 1.6 x 10^6 and 0.8 x 10^6 points can be registered in less than 2 s.

Keywords: 3D imaging system, automated object recognition, sphere fitting, target-based registration

[*] Corresponding author. Tel: +1 301 975 6408; fax: +1 301 869 6275. *E-mail*: marek@nist.gov

1. Introduction

The applications for three-dimensional (3D) imaging systems are widespread and include generation of 3D models, surveying and mapping, reverse engineering, quality control, forensics, autonomous vehicle navigation, and historic preservation. 3D imaging systems are line-of-sight instruments [1], and thus, parts of a scanned scene which are occluded from one instrument position need to be obtained by positioning the instrument at other locations to fill in the missing data. Therefore, to build a 3D model of an scanned object or scene, registration of all point clouds to one coordinate system is required [2].

A registration process can consist of two parts: 1) an initial estimate of transformation parameters so the two coordinate systems are roughly aligned; 2) final tuning of the transformation parameters. The first task can be performed either fully or semi-automatically, with some user intervention (e.g., manual identification of features/points common to both datasets). This task can be accomplished either with or without the use of targets. Depending on the application, the initial estimate may be all that is required. The second task is usually performed automatically using algorithms such as ICP (Iterative Closest Point [3]). Although accuracy of the registration is the ultimate goal, the main concern for an initial, automatic registration is the execution time because large datasets need to be searched quickly. In the second part, the focus is shifted from algorithm speed to the accuracy of the final transformation.

Target-free registration is the desired goal for registration. The obvious reasons being that placement of targets in a scene may not be possible and placement of targets require planning and set-up time. The algorithms which perform automatic target-free, pair-wise registration generally follow one of two main strategies: 1) automatic identification of local features (e.g., planes, lines, surface patches, curvatures) common to both datasets [4-8]; or 2) maximization of the correlation of global characteristics (e.g., Extended Gaussian Images) [9]. The performance of the first strategy depends on the presence of selected features in images to be registered and therefore the strategy is not well generalized (e.g., datasets which do not contain features used earlier to train a classifier are prone to many mismatches). The second strategy requires relatively large overlap regions. Thus, the time for planning and placing targets for targeted registration may be offset by the additional scans necessary due to the need for large overlap regions needed for target-free registration. The algorithms that perform the target-free registration may be fast (run time of a few seconds for large datasets [8]) but both strategies require preprocessing of the data to segment the images and extract features or geometric primitives which requires extra time [10] (on the order of minutes). Until issues such as large overlap regions, the ability to quickly segment the data for identifiable features, and assurance that target-free registration is as accurate and robust as target-based registration are resolved, target-based procedures are more commonly used for initial registration.

Two types of targets are in use: planar or 3D targets. The use of targets for pair-wise registration requires a tradeoff between two conflicting conditions: 1) targets should be well distributed within the measurement volume; 2) two scans to be registered should have small overlap region (to reduce data acquisition time). Planar targets with high contrast or highly reflective regions are commonly used. Reflective targets enable quick and automatic target identification based on the returned intensity values. However, planar targets do not allow direct registration of two datasets with small overlap unless they are re-oriented to face the instrument (stick-on planar targets cannot be re-oriented and therefore requires the placement of more of these targets [11]). The registration of scans with several intermediate viewpoints can cause accumulation of registration error and the manual re-orientation limits the application of 3D imaging systems in a fully automated environment.

The most common type of 3D target (the second type of targets) is a sphere as it remains the same from different viewpoints. Properly placed sphere targets can be seen from more viewpoints without the need to re-orient the spheres and this gives a user more flexibility in locating an instrument and to reduce the number of required scans [12]. Correctly identified spheres in two or more datasets provide a well defined set of common points (sphere centers) which may be used for registration [13-16]. As long as at least three common spheres are identified in every acquired data set, all scans (regardless of the amount of overlap region) can be directly registered to the same coordinate system, without the need for intermediate registrations. Commercial software packages commonly include a registration algorithm based on sphere targets but these packages require a user to identify the spheres. Some of these packages require separate high density scans of each sphere in addition to the actual scan of the entire scene. Such an approach severely limits the application of 3D imaging systems in a fully automated environment. The goal of the current study is to introduce a fast algorithm which automatically extracts three pairs of matching points (sphere centers) from two datasets, without prior scanning of individual spheres. Once the three common points are found, a transformation relating the two coordinate systems can be determined.

Automatic registration based on 3D targets requires a quick and reliable procedure to find the targets in a large range image. Automated target recognition (ATR) from data acquired by 3D imaging systems has been studied by many researchers [17]. Different approaches have been developed, like fuzzy scene matching [18], adaptive texture representation [19], maximum-likelihood surface estimator [20], or methods based on segmentation [21, 22]. Many approaches rely on surface fitting procedures with a variety of error functions used [23, 24]. Automated retrieval of CAD model objects specifically in construction range images has also been studied [25]. These procedures usually deal with large targets which occupy a substantial part of the scanned scene or they locate small targets which move with respect to the surrounding background. However, targets used for registration do not move. Additionally, they occupy a small portion of a scanned scene, and they are, therefore, prone to noise and having only a few measurements on the target. Because of this, currently available procedures cannot identify such targets quickly.

Raw data acquired by 3D imaging instruments are in a format of 2D images with measured range value $r_{m,n} = r(\theta_m, \varphi_n)$ assigned to every (m,n) pixel, and (θ_m, φ_n) are elevation and azimuth angles at which a given range is recorded. From these 2D images, datasets containing Cartesian coordinates $\mathbf{u}_{m,n} = [x_{m,n}\ y_{m,n}\ z_{m,n}]$ are exported and available to a user for further processing. We developed the algorithm for datasets measured on a regular grid, i.e. all data points $\mathbf{u}_{m,n}$ can be indexed by a pair (m,n), where $m=1,\dots, M_{max}$ and $n=1,\dots, N_{max}$. If a point cloud is unstructured and sequentially recorded points $\mathbf{u}_i = [x_i\ y_i\ z_i]$ cannot be mapped on $\mathbf{u}_{m,n}$ (because of 'holes' in the gridded data for some (θ_m, φ_n) due to the lack of a return signal), then the dataset needs to be preprocessed (e.g., using a binning procedure) so the correspondence between $\mathbf{u}_i$ and $\mathbf{u}_{m,n}$ can be established. We will not discuss this preprocessing procedure further and assume the dataset is on a regular 2D grid. It should be noted that every target finding procedure, including the one proposed in this paper, has to deal with the problem of unstructured datasets.

Our algorithm is able to rapidly and automatically locate spheres and to register two large datasets (each containing several millions of points) in less than 30 s using the fitted sphere centers. The algorithm checks every data point in both datasets as a possible sphere center. Checking every data point is necessary as sphere targets inserted in a scanned scene are generally represented in a dataset by a small fraction of all data points (below 0.01 %). In addition, range images obtained with 3D imaging instruments contain many discontinuities. Thus, any other strategies which look for sphere centers by checking only some selected search paths are at risk of missing the true sphere location. On the other hand, if the sphere targets represent more than, say 5 % of all data points, then there may be more efficient algorithms for locating the spheres than ours.

The paper is organized as follows: in section 2 the algorithm is introduced, in section 3 the choice of algorithm parameters is discussed, in section 4 the experiment is described, the results are presented in section 5, and section 6 presents a discussion of the results and conclusions.

2. Algorithm

Our proposed algorithm consists of two major steps: 1) Automatic location of spheres: for every dataset, a list of possible candidates of sphere centers is created (list includes both true and false positives); 2) Registration: for every pair of datasets which are subject to registration, a pair of congruent triangles is determined. All vertices of a single triangle are chosen from one list of candidates and the selected pair of triangles minimizes a defined error function. Then, the isometry which transforms one triangle into another is the initial transformation that yields an initial estimate of the registration of one dataset to another.

2.1 Automatic Location of Spheres

The core of the automation is the location of the spheres. Step 1 of the location algorithm can be partitioned into six functional blocks: a) Three Points Filter: a quick search for an initial list of candidate sphere centers; b) Geometrical Constraints Filter: calculation of the error function and other features (available as output resulting from the error calculation) for every accepted candidate and keeping only the points which have feature values satisfying certain user defined conditions; c) Selection of unique sphere centers (two or more points retained in the previous step can belong to the same sphere and they are not considered as unique centers); d) Sphere Fitting: refinement of sphere locations by running an optimizer; e) Data-free Zone Filter: extension of the Three Points Filter; f) Sorting: sort resulting list based on error values. These steps are described in more detail in the following sections.

Step 1A: Three Points Filter

Spheres used for target-based registration are placed in such a way that they can be seen from different scanner positions. This implies there has to be some free space around each sphere to ensure the view is not obstructed by other objects. This property can be used as a filter - those points which don't have this feature can be eliminated as candidates of sphere centers. For every range $r_{m,n}$ measured by a 3D imaging instrument at elevation and azimuth angles (θ_m, φ_n), a proxy sphere of known radius R is placed in Cartesian coordinates at the location given by a vector $\mathbf{U}_{m,n} = [X_{m,n}\ Y_{m,n}\ Z_{m,n}\ ]$, defined by

$$\mathbf{U}_{m,n} = \mathbf{u}_{m,n} \frac{R + r_{m,n}}{r_{m,n}} \tag{1}$$

where vector $\mathbf{u}_{m,n} = [x_{m,n}\ y_{m,n}\ z_{m,n}\ ]$ is defined by the Cartesian coordinates of a point in the dataset and

$$r_{m,n} = \sqrt{x_{m,n}^2 + y_{m,n}^2 + z_{m,n}^2} \tag{2}$$

is the measured range. This defines a circular cone with an apex at the scanner center and aperture equal to $2\beta_{\mathrm{m,n}}$ (see Fig. 1), where

$$\sin(\beta_{m,n}) = \frac{R}{R + r_{m,n}} \tag{3}$$

If there is indeed a sphere centered at $\mathbf{U}_{m,n}$, then there should be no other points in the region around this sphere. This region is bounded by four user selected constants: outer and inner radius G_{max} and G_{min} (see Section 3 for the reason why G_{min} is greater than R) and the range limits D_{min} and D_{max}, see Fig.1. These constants are used for all points $\mathbf{u}_{m,n}$ in a dataset.

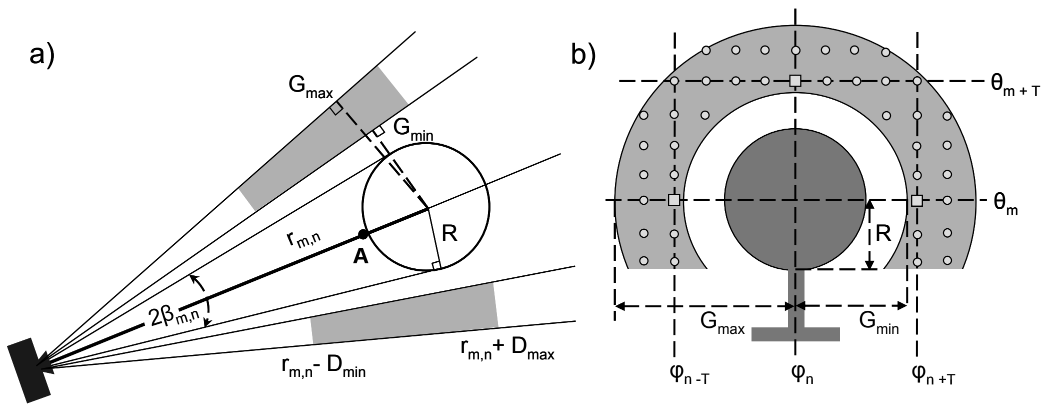

Fig.1 Geometrical relations used in the algorithm: a) top view of the scanner and sphere – the processed data point is shown as Point A at distance $r_{m,n} = r(\theta_m, \varphi_n)$ from the scanner. The grey region around a sphere target should not contain data points; b) view from instrument in scanning direction - the light grey represent the same region as shown in Fig. 1a, the three small squares correspond to three data points checked in step 1A. Small circles in grey region represent all other data points checked in step 1E. Sphere center located at $(\theta_m, \varphi_n, R+r_{m,n})$.

This step of the algorithm is repeated for every point in a dataset and in order to speed up the process, we check only three selected points in the marked region. In Fig.1b, these points are marked as small squares corresponding to three points from dataset located at $\mathbf{u}_{m,n-T}$, $\mathbf{u}_{m,n+T}$ and $\mathbf{u}_{m+T,n}$. Index T is calculated as

$$T = \gamma_{m,n} / \delta \tag{4}$$

where δ is the angular increment used in scanning and

$$\sin(\gamma_{m,n}) = G_{\min} / (r_{m,n} + R) \tag{5}$$

Spheres used as targets for registration are generally placed in such way that $G_{min} << r_{m,n}$ and small angle approximation can be used in (5). When different angular increments δ_θ and δ_φ are used for scanning along the elevation and azimuth directions, then two different indices T_θ and T_φ have to be used. If any of the three ranges $r_{m,n-T}$, $r_{m,n+T}$ or $r_{m+T,n}$ falls within the limits ($r_{m,n}$ - D_{min} , $r_{m,n} + D_{max}$), then $\mathbf{U}_{m,n}$ is rejected as a possible sphere center, otherwise a corresponding point from dataset $\mathbf{u}_{m,n}$ is accepted and kept for further processing. The size of the region marked in Fig. 1 depends on two factors: 1) how the sphere is mounted to the base; 2) where it is placed. Fig. 2a,b show examples of two different mountings and how they affect the value of inner radius G_{min}. Fig. 2c,d show how good or bad sphere placement may impact the choice of the three other parameters: in Fig. 2c G_{max}, D_{min} and D_{max} can be large while in Fig.2d they all have to be small. Generally, it is advantageous to place all spheres in such way that the grey region marked in Fig.1 is as large as possible. Detailed discussion on the choice of these and other parameters is given in section 3.

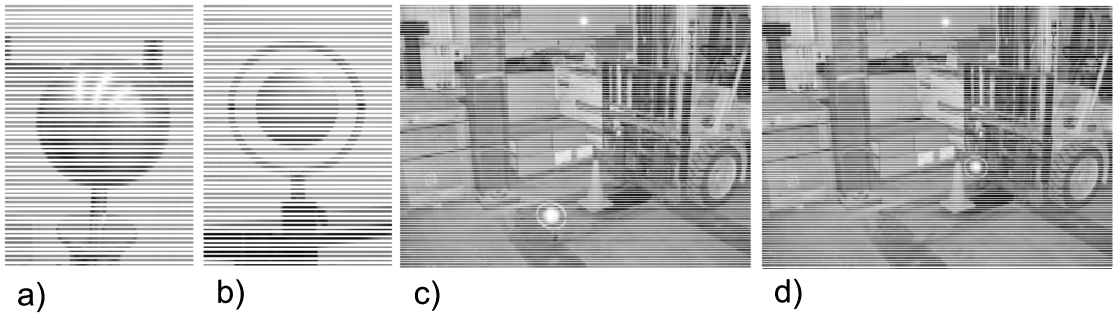

Fig.2 Different sphere mounting: a) sphere on a single stem, parameter G_{min} is related to sphere radius R; b) sphere inside a ring, G_{min} is related to outer radius of a ring. Image in c) shows an example of good sphere placement, with a lot of free space around the sphere, while d) shows an example of bad sphere placement in a cluttered part of the scanned scene.

Step 1B: Geometrical Constraints Filter

For every point $\mathbf{u}_{m,n}$ from a dataset which passed the test from step 1A, a cone depicted in Fig. 1a can be defined. Three different categories of points may be distinguished from points inside the cone: category A points are located behind the sphere and are shadowed by it (squares in Fig. 3); category B points are in front of the sphere (triangles in Fig. 3); and category C points are hits on the sphere surface (indicated by dots). Due to noise, points in category C will not lie perfectly on the surface of the sphere but are scattered within a small distance on both sides of its surface. Assignment of points to category C and B depends on the user defined parameter ψ which specifies the maximum allowed deviation of experimental points from the surface of the front hemisphere. A rule for choosing the right value of ψ is discussed in section 3. All points within the cone which have ranges larger than $r_{m,n} + 1.5R$ are assigned to category A. All points with ranges smaller than $r_{m,n}$ - ψ are assigned to category B.

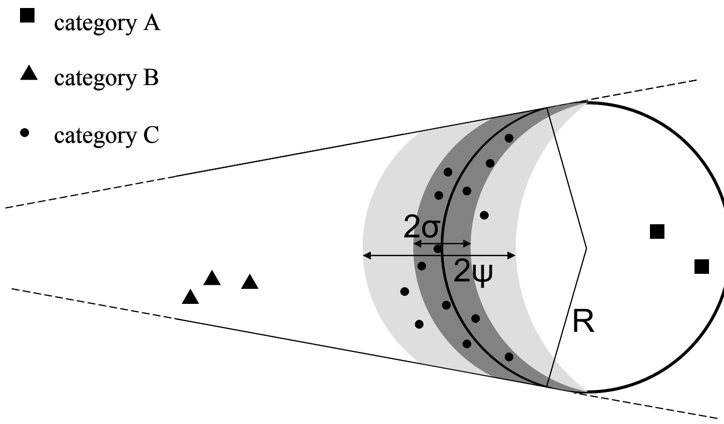

Fig.3 Selecting points inside a cone for sphere fitting: only points in the category C contribute to the error function. Dark grey region is determined by the level of instrument noise σ, light grey region extends beyond dark grey and is determined by parameter ψ, where $\psi = S\,\sigma$ with $3 \leq S \leq 5$. All points in the category C are located within both grey regions.

All points within the cone are in category C if their ranges satisfy $|r_{m,n} - t_{m,n}| \leq \psi$, where $(\theta_m, \varphi_n, t_{m,n})$ is a point on the front hemisphere. Those points are shown in Fig.3 inside both grey regions and only they contribute to the error function defined by

$$Err(\mathbf{U}_{m,n}) = \sqrt{P(\mathbf{U}_{m,n})} \Big/ N_c \qquad (6)$$

with

$$P(\mathbf{U}_{m,n}) = \sum_{j=1}^{N_c} [\sqrt{(x_j - X_{m,n})^2 + (y_j - Y_{m,n})^2 + (z_j - Z_{m,n})^2} - R]^2 \qquad (7)$$

where a location of sphere center $\mathbf{U}_{m,n} = [X_{m,n}\ Y_{m,n}\ Z_{m,n}\ ]$ is related to the currently processed data point $\mathbf{u}_{m,n}$ by (1) and $\mathbf{u}_j = [x_j\ y_j\ z_j\ ]$ is the location of j-th data point from category C and N_c is the total number of points in category C.

At any arbitrary location $\mathbf{U}_{m,n}$, all three categories are usually present. However, if the point is the center of a true sphere, there cannot be any points (category A points) behind it. This cannot be said about the points from category B as a true sphere may be partially occluded by other objects which are located between the sphere and the scanner. In addition to the error function, a fill ratio $Fill(\mathbf{U}_{m,n})$ is calculated

$$Fill(\mathbf{U}_{m,n}) = N_c / N_{cone} \tag{8}$$

where N_{cone} is the total number of points inside a cone. After step 1B is completed, only those points are kept for which the number of points in category A is zero, $N_c > N_{min}$ and $Fill(\mathbf{U}_{m,n}) \geq T_{fill}$, where N_{min} and T_{fill} are user-defined thresholds. A discussion on the choice of these two parameters is given in section 3.

Step 1C: Selection of Unique Sphere Candidates

All unique minima of the error function are determined from the points accepted in step 1B - subset 1B. First, from subset 1B, the point $\mathbf{U}_{m1,n1}$ with the smallest error function is set as the global minimum. Next, all points in subset 1B are checked, and if their distance to the minimum $\mathbf{U}_{m1,n1}$ is smaller than R, then these points are flagged. Next, the location of a new minimum $\mathbf{U}_{m2,n2}$ is determined from the remaining points in subset 1B (i.e., points that have not been flagged thus far) and again, all points closer than R are flagged. The process continues until all points in subset 1B are exhausted. The final outcome of this step of our algorithm is a list of K points $\{\mathbf{U}_{m1,n1}, \mathbf{U}_{m2,n2}, \ldots, \mathbf{U}_{mK,nK}\}$ which defines a set of possible candidates for approximate locations of the sphere centers.

Step 1D: Sphere Fitting

Every point from the output list from step 1C becomes a starting point for a sphere fitting procedure. We use the error function defined by (6) and (7) for which an analytical expression for the gradient can be easily derived. We use the standard quasi-Newton optimization procedure as described in [26]. During the optimization, a sphere is fitted to the same set of N_c points from category C determined in step 1B. It should be noted that starting points for the fitting procedure are located at the nominally measured coordinates $\mathbf{U}_{m,n} = [X_{m,n}\ Y_{m,n}\ Z_{m,n}\ ]$ while the final optimized sphere locations may have any location $\mathbf{U}_k = [X_k\ Y_k\ Z_k]$ $(k=1,\ldots,K)$ which need not coincide with the nominal values and may be somewhere in between them.

Step 1E: Data-free Zone Filter

We repeat the check similar to that performed in Three Points Filter, but this time the region marked in Fig.1 is centered only at the K points corresponding to the locations of minima $\mathbf{U}_k$ determined in step 1D. More precisely, for every $\mathbf{U}_k$, the closest $\mathbf{U}_{p,q}$ related to data point $\mathbf{u}_{p,q}$ by (1) is found, and the region defined by the four parameters G_{min}, G_{max}, D_{min} and D_{max} is established, as in step 1A. Now, every point marked by a small circle in the grey region in Fig.1b can be checked. If there is a true sphere located at $\mathbf{U}_k$, then not only three points marked by small squares in Fig.1b should have ranges outside the interval ($r_{p,q}$ - D_{min} , $r_{p,q}$ + D_{max}) but the same should apply to all points marked as circles. If this is not true for any candidate sphere center $\mathbf{U}_k$ then this candidate is rejected. This step may further cut the list of possible candidates to N points where $N \leq K$.

Step 1F: Sorting

Finally, the list of all $\mathbf{U}_n$ candidates, $n=1,\ldots, N$, is sorted in increasing order of final error values which were determined in the optimization step 1D. This completes the first major step of our algorithm.

Steps 1A-1F are repeated for every dataset. Once a list of possible sphere centers is determined for every dataset, the step 2 is performed. Initial registration of one dataset to another is done by finding three pairs of matching points from the corresponding two lists obtained from step 1.

2.1 Registration

The task of registration is to transform two datasets so that they have a common coordinate frame. After the locating algorithm has produced a list of candidate spheres in a scan, it needs to be determined which of these spheres represents an actual target, and if so, which particular target. Once corresponding sphere centers have been determined, the desired rigid-body transformation can be easily determined. This whole procedure, Step 2, is described below and can be further subdivided into four functional blocks.

Step 2A: Match Paired Points

Two lists of possible sphere centers $\mathbf{U}_{n1}^{(1)}$ and $\mathbf{U}_{n2}^{(2)}$, $n1 = 1,\ldots, N^{(1)}$ and $n2=1,\ldots, N^{(2)}$, are read, together with the corresponding residual error values determined in step 1D. Then, a list of matching pairs $d_{i,j}^{(1)}$ and $d_{k,l}^{(2)}$ is created, where $d_{i,j}^{(1)}$ is the distance between $\mathbf{U}_i^{(1)}$ and $\mathbf{U}_j^{(1)}$, and similarly $d_{k,l}^{(2)}$ is the distance between $\mathbf{U}_k^{(2)}$ and $\mathbf{U}_l^{(2)}$. A pair $(d_{i,j}^{(1)}, d_{k,l}^{(2)})$

is added to a list when both *i* and *k* are less than the total number of true spheres *M* and $|d^{(1)}_{i,j} - d^{(2)}_{k,l}| < \varepsilon$, where ε is a user defined threshold. The condition imposed on *i* and *k* restricts matching distances to only those pairs of points which have one point chosen from the top *M* possible sphere centers, no restriction is set on a second point, i.e. $1 \leq j \leq N^{(1)}$ and $1 \leq l \leq N^{(2)}$. Both list $\mathbf{U}^{(1)}_{n1}$ and $\mathbf{U}^{(2)}_{n2}$ are sorted by increasing error value, and so, it is reasonable to expect at least one true sphere center among the first *M* candidates on each of the lists. The parameter ε is a tolerance threshold for matching $d^{(1)}_{i,j}$ with $d^{(2)}_{k,l}$ and it depends on how good the sphere fitting is at $\mathbf{U}^{(1)}_{i}, \mathbf{U}^{(1)}_{j}, \mathbf{U}^{(2)}_{k}$ and $\mathbf{U}^{(2)}_{l}$: if ε is selected too low, it can lead to a rejection of legitimate pair $(d^{(1)}_{i,j}, d^{(2)}_{k,l})$ and if ε is too large value, it can cause spurious matches to be included. The choice of ε is discussed in section 3. The final list should only contain unique pairs, i.e. $d^{(1)}_{i,j}$ is considered the same as $d^{(1)}_{j,i}$.

Step 2B: Determine Congruent Triangles

Based on the list of pairs $(d^{(1)}_{i,j}, d^{(2)}_{k,l})$ output from step 2A, a separate list of pairs $(\Delta^{(1)}_{p}, \Delta^{(2)}_{q})$ of congruent triangles is created. Each triangle is defined by its three vertices, $\Delta^{(1)}_{p} = (\mathbf{U}^{(1)}_{i}, \mathbf{U}^{(1)}_{j}, \mathbf{U}^{(1)}_{p})$ and $\Delta^{(2)}_{q} = (\mathbf{U}^{(2)}_{k}, \mathbf{U}^{(2)}_{l}, \mathbf{U}^{(2)}_{q})$. For each triangle, the lengths of two sides are calculated, i.e. for $\Delta^{(1)}_{p}$, in addition to the already calculated $d^{(1)}_{i,j}$, $d^{(1)}_{i,p}$ and $d^{(1)}_{j,p}$ are determined. Similarly, for triangle $\Delta^{(2)}_{q}$ its side lengths $d^{(2)}_{k,q}$ and $d^{(2)}_{l,q}$ are calculated. A pair of triangles $(\Delta^{(1)}_{p}, \Delta^{(2)}_{q})$ is added to a list when one of the two conditions is fulfilled: 1)

$|d_{i,p}^{(1)} - d_{k,q}^{(2)}| < \varepsilon$ and $|d_{j,p}^{(1)} - d_{l,q}^{(2)}| < \varepsilon$, or 2) $|d_{i,p}^{(1)} - d_{l,q}^{(2)}| < \varepsilon$ and $|d_{j,p}^{(1)} - d_{k,q}^{(2)}| < \varepsilon$ (at this stage, the correspondence of the vertices of $\Delta_p^{(1)}$ and $\Delta_q^{(2)}$ has not been established yet).

Step 2C: Select Best Matched Pair of Triangles

For every pair ($\Delta_p^{(1)}, \Delta_q^{(2)}$) of triangles on the list, an error function is determined as

$$E(\Delta_p^{(1)}, \Delta_q^{(2)}) = Err(\mathbf{U}_i^{(1)}) + Err(\mathbf{U}_j^{(1)}) + Err(\mathbf{U}_p^{(1)}) + Err(\mathbf{U}_k^{(2)}) + Err(\mathbf{U}_l^{(2)}) + Err(\mathbf{U}_q^{(2)}) \quad (9)$$

where individual errors (*Err)* are output from the optimization performed at step 1D. Then, a pair with the smallest value $E(\Delta_p^{(1)}, \Delta_p^{(2)})$ is selected for final processing.

Step 2D: Match Corresponding Vertices

In order to determine the parameters of the transformation which relates $\Delta_p^{(1)}$ to $\Delta_q^{(2)}$, a unique correspondence between vertices of both triangles has to be known. In each of the two triangles, the longest side is found. This side, defined by two vertices, determines uniquely the third vertex. Thus, the first pair of matching vertices ($a^{(1)}, a^{(2)}$) is determined. In a similar fashion, the next longest side and a vertex laying opposite to that side is found for both triangles. After the second pair of matching vertices ($b^{(1)}, b^{(2)}$) is determined, there is no more ambiguity and the correspondence between ($a^{(1)}, b^{(1)}, c^{(1)}$) and ($a^{(2)}, b^{(2)}, c^{(2)}$) is established and parameters of the sought transformation can be calculated. It should be noted that this procedure is applicable only to scalene triangles.

3. Choice of parameters

Our algorithm requires the following nine parameters to be specified: R, G_{min}, G_{max}, D_{min}, D_{max}, ψ, N_{min}, T_{fill} and ε. The choice of these nine parameters is a sensitive issue because some of them are interdependent. The algorithm requires careful planning *prior* to data acquisition to ensure that the input parameters are within their domain of applicability.

We assume that two factors are known at the onset: 1) the manufacturer reported uncertainties of the 3D imaging instrument; 2) the selected scan density (angular increment δ). Based on these two fundamental factors, the nine parameters used in the algorithm can be estimated. The noise level, σ, for an instrument is usually manufacturer specified, and σ can be used to select the sphere radius, R. We suggest that the ratio $\sigma/R < 0.1$ where σ is the largest specified value.

Once the radius is known, parameter ψ can be estimated. This parameter should be large enough to avoid rejection of legitimate hits on a sphere from category C. On the other hand, it should be small enough to exclude points in category C which clearly belong to category A or B. Thus, ψ should be some multiple of the noise level σ. We recommend calibrating the instrument so the noise dependence on measured range $\sigma(r)$ is known and then use $\psi(r) = S\,\sigma(r)$ with $3 < S < 5$.

Knowing the sphere radius and sphere mounting, the inner radius G_{min} can be determined. In order to avoid any interference from mixed pixels (which yield a spurious range reading when a laser beam hits the edge of an object) we suggest setting $G_{min} = 1.5D_0$. For spheres similar to that shown in Fig.2a, $D_0 = R$ while for the type of mounting shown in Fig.2b, D_0 is the outer radius of the ring on which the sphere is mounted. Of the two types of sphere mountings, the one shown in Fig. 2a is preferable as it does not require re-orientation of the sphere from different instrument locations and there will be no points from the outer ring. For other types of mounting, D_0 needs to be defined appropriately and G_{min} can be treated as a constant. Three other parameters: G_{max} , D_{min} and D_{max} depend on where the sphere is placed or, more precisely, how close it is to other nearby objects. As mentioned previously, the parameters define the free space around the sphere which should be as large as possible. Fig.2d shows an example of bad sphere placement: the sphere is located next to a forklift which forces G_{max} and D_{max} to be small. Similarly, a cone in front of the sphere prohibits a choice of large D_{min}. Low values of these parameters could cause a large number of false positives as many data points would pass the test in step 1A. Conversely, the selection of large values for G_{max}, D_{min} and D_{max} would reject candidate sphere centers if a sphere were placed as in Fig.2d. The user often has flexibility when placing the sphere targets and we recommend the following lower bounds to be satisfied for every sphere: $G_{max} \geq 2.0\ D_0$, $D_{min} \geq 9R$ and $D_{max} \geq 3R$.

The next parameter needed in our algorithm is ε - the upper bound for the length tolerance which is used when searching for congruent triangles. This parameter should be related to the magnitude of the error in sphere center location but without explicit knowledge of that error, we recommend setting $\varepsilon = 0.5R$.

The parameters we discussed so far all depend on the instrument noise level σ. The remaining two parameters: N_{min} and T_{fill} depend on the scan density. An upper limit on the distance from the instrument to the sphere is r_{max} and is defined when $\beta_{m,n}$ (Fig. 1) is equal to the angular increment δ. From (3), r_{max} can be derived by setting $\beta_{m,n}$ equal to δ which leads to

$$r_{\max} = R/\delta \qquad (10)$$

provided that δ is small and small angle approximation can be used. If the distance between a sphere and a scanner is larger than r_{max}, then the total number of all data points inside the cone (including all points in category C) cannot exceed 5, see Fig.4. This increases the risk of detecting false positives with a very low value of error function as obtained from (6) and (7). Therefore, we set the lower bound for points in category C to $N_{min} = 7$. This means that the largest actual distance between the instrument and the sphere accepted by major step 1 is limited to r_{max}.

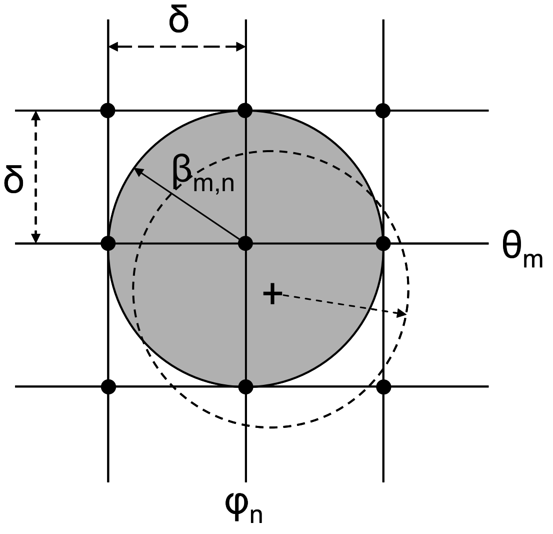

Fig.4 Geometric relations used to define parameter r_{max}. If a true sphere center is located at (θ_m, φ_n, $R+r_{m,n}$) then the largest possible number of hits on a sphere cannot exceed 5 for $r_{m,n} > r_{max}$. For $r_{m,n}$ equal to or even slightly smaller than r_{max} the actual number of hits may be as low as 3 if the sphere center (marked by a cross) does no coincide with nominal experimental bearings (θ_m, φ_n).

The last parameter T_{fill} could be set, in theory, to its maximum value of 1. However, there are two reasons why this should be avoided. First, the scanner may need to be placed in a location where a sphere is partially occluded and large T_{fill} could reject a legitimate sphere center. Second, a high value for T_{fill} requires that the value of ψ also be large so all points from category C are correctly classified. However, large values of ψ will lead to many false positives and such a situation should be avoided. The combination of small or medium values of ψ and high T_{fill} increases the risk of rejecting spheres whose distances to the scanner approach r_{max} . Such spheres will have a low number of points in category C (N_c) and even one or two points incorrectly excluded from this category will yield a substantial drop in the value of the *Fill* parameter. This, in turn, will cause rejection of such spheres when T_{fill} is large. On the other hand, too low a value of T_{fill} will also generate many false positives, and we, therefore, suggest keeping this parameter between moderate limits, like $0.5 \leq T_{fill} \leq 0.7$.

4. Experiment

Four anodized aluminum spheres, shown in Fig.2b, with radius R = (76.2 ± 0.5) mm and external radius of the mounting ring D_0 = (127± 0.5) mm, were placed throughout the laboratory. The lab [42 m (Length) x 10 m (Width) x 7 m (Height)] was scanned with a 3D imaging system from two different positions (see Fig. 5) with the sphere locations fixed. The 3D imaging system has a manufacturer specified range uncertainty of ± 10 mm. At each instrument position, three angular increments were used (δ=0.14°, 0.08° and 0.04°). Table I summarizes the experimental settings. In addition, every sphere was scanned individually at an angular increment 0.004° yielding high density datasets. In all scans, the same angular increment was used for both azimuth and elevation angles. The data from the individual sphere scans were manually segmented and used to fit spheres. The resulting sphere centers were used as ground truth for verifying the locations of the spheres calculated in the first step of our algorithm on the datasets of the entire scene.

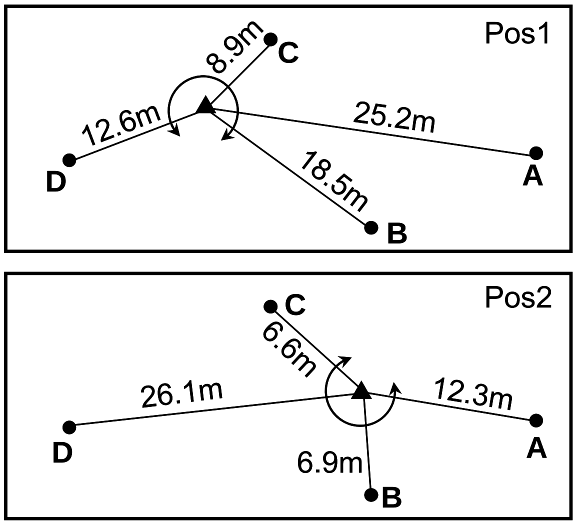

Fig.5 Schematic plan showing locations of four spheres (marked by dots) and a scanner (triangle) placed in two scanning positions Pos1 and Pos2. Semicircles around a scanner indicate the field of regard.

The residual values of the error function resulting from the sphere fitting of the individual sphere datasets scanned with $\delta = 0.004^{\circ}$ were used to develop a relationship between the instrument noise and range, shown in Fig.6.

Angular Increment [°]	Pos1				Pos2			
	#pts/col	#pts/row	*total #pts*	*Tcpu* [s]	#pts/col	#pts/row	*total #pts*	*Tcpu* [s]
0.04	1 176	5 526	6 498 576	21.5	701	4 926	3 453 126	3.6
0.08	590	2 764	1 630 760	1.6	352	2 464	867 328	0.3
0.14	339	1 581	535 959	0.2	203	1 409	286 027	0.05

Table I. Summary of experimental settings. For two scanner positions Pos1 and Pos2, #pts/col and #pts/row is the number of data points per column (elevation) and per row (azimuth). T_{cpu} is the longest time (among all runs with different parameters) needed to process step 1 for a given dataset.

To determine the sensitivity of the parameters, all six datasets (three angular increments in two scanner positions) were processed using the following ranges for the input parameters: $S = \{3.0, 3.5, 4.0, 4.5, 5.0\}$, where S is the scale factor between $\psi(r)$ and $\sigma(r)$, $T_{fill} = \{0.55, 0.60, 0.65, 0.7\}$, $D_{min} = \{9R, 12R\}$, $D_{max} = \{3R, 4R\}$, $G_{max} = \{2D_0, 2.5D_0\}$, $N_{min} = 7$ and $G_{min} = 1.5\ D_0$, which totals to 160 different combinations of

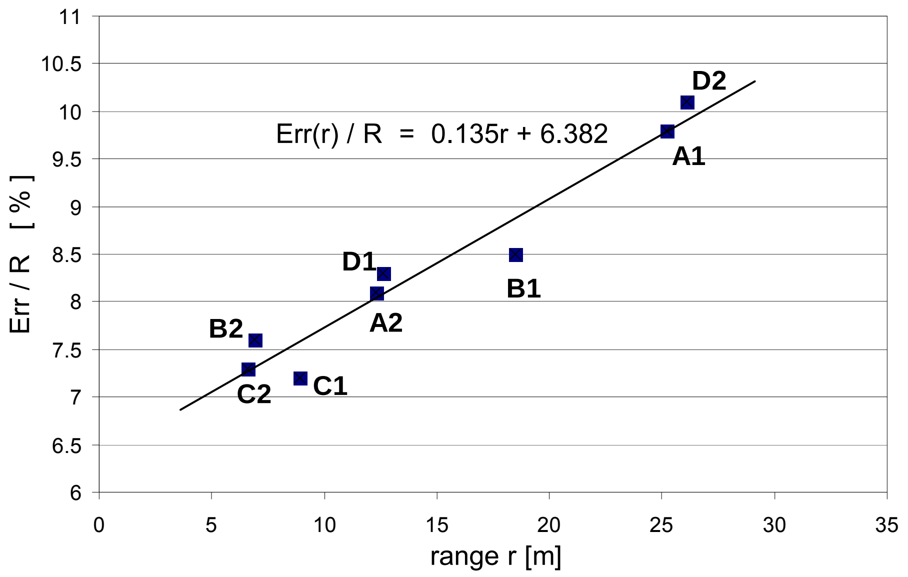

Fig.6 Residual values of error function of sphere fitting versus scanner-sphere distance. Labels A1,…, D2 correspond to spheres scanned in two instrument positions Pos1 and Pos2, as shown in Fig.5. The regression line was used to model the relationship between instrument noise σ and range *r*.

parameters. In addition, a more detailed study of the two most sensitive parameters, T_{fill} and S, was done. A grid of 50x50 points on [T_{fill} S] plane was built with $0.55 \le T_{fill} \le 0.7$ and $3.0 \le S \le 5.0$ with the remaining parameters fixed: $D_{min} = 12R$, $D_{max} = 4R$, $G_{max} = 2.5D_0$, $N_{min} = 7$ and $G_{min} = 1.5\ D_0$. For each of three angular increments δ, the second major step of the algorithm was run on a pair of two lists of possible sphere centers $\mathbf{U}_{n1}^{(1)}$ and $\mathbf{U}_{n2}^{(2)}$, corresponding to the two instrument locations. Step 2 was repeated for every list of pairs ($\mathbf{U}_{n1}^{(1)}$, $\mathbf{U}_{n2}^{(2)}$) created by step 1 for each combination of input parameters. The only input parameter required in step 2 was ε and it was kept constant at $0.5R$. The output of step 2 is a pair of congruent triangles ($\Delta_p^{(1)}, \Delta_q^{(2)}$). If all three vertices of the first triangle were close to three true sphere centers in a first coordinate system and the same was valid for the second triangle in a second system, then the algorithm was successful for a given combination of input parameters. Otherwise, the algorithm failed for that particular combination of input parameters.

5. Results

For both datasets, one from each instrument location, scanned with an angular increment δ = 0.04º, all 160 runs of the algorithm finished successfully, i.e., three matching sphere centers in both scanner positions were identified correctly. The same held for all 2500 runs while only varying parameters (T_{fill} , S) with the other parameters fixed. For δ = 0.08º, the algorithm finished successfully in 78/160 combinations of input parameters. A graph showing the performance of the algorithm on [T_{fill} S] plane is shown in Fig.7.

For the largest angular increment δ = 0.14º, none of 160 sets of input parameters yielded successful results. Similarly, all 2500 pairs of (T_{fill} , S) parameters failed to identify the three common sphere centers in both coordinate systems. Execution time T_{cpu} of step 1 was recorded (without input/output operations). For a given dataset, the time varied slightly for different sets of input parameters. The largest value of T_{cpu} recorded for every processed dataset is shown in Table I. The total time needed to estimate an initial transformation between two coordinate systems is equal to $T_{cpu}^{(1)} + T_{cpu}^{(2)}$, where $T_{cpu}^{(1)}$ and $T_{cpu}^{(2)}$ are times needed by step 1 of the algorithm to process a dataset acquired in the first and the second coordinate system, respectively. The processing time for step 2 is negligible (maximum time below 0.01s). All computations were performed on a PC with a 3.0 GHz processor.

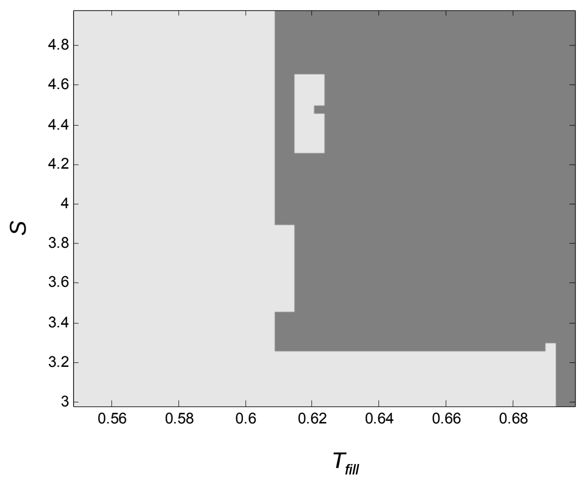

Fig.7 Performance of the algorithm as a function of T_{fill} and S on datasets acquired with $\delta = 0.08^o$, all other parameters were fixed. Light grey regions mark successes while dark grey regions were failures.

Fig. 8 shows the 2D intensity images of three regions of the lab section scanned with an angular increment of 0.04º from Pos1. Fig. 9 shows an overall perspective 3D view of the point cloud of the scanned scene. Fig. 10 presents four point clouds zoomed in on the individual spheres scanned from Pos1. For the cases when spheres were identified, the maximum calculated distance between the fitted sphere center and the ground truth was 0.05R or 3.81 mm.

6. Discussion and conclusions

The presented results demonstrate that the proposed algorithm can be useful in automating target-based registration when used with properly selected input parameters. The initial transformation needed to register two large 3D imaging datasets ($6.4x10^6$ and $3.4x10^6$ points for $\delta = 0.04^o$) can be performed in less than 30 s. For smaller datasets ($1.6 x10^6$ and $0.8 x10^6$ points for $\delta = 0.08^o$), the required processing time is less than 2 s. We have provided a set of rules to guide the choice of input parameters. Performance of the algorithm beyond the bounds investigated in this paper needs to be examined in future work.

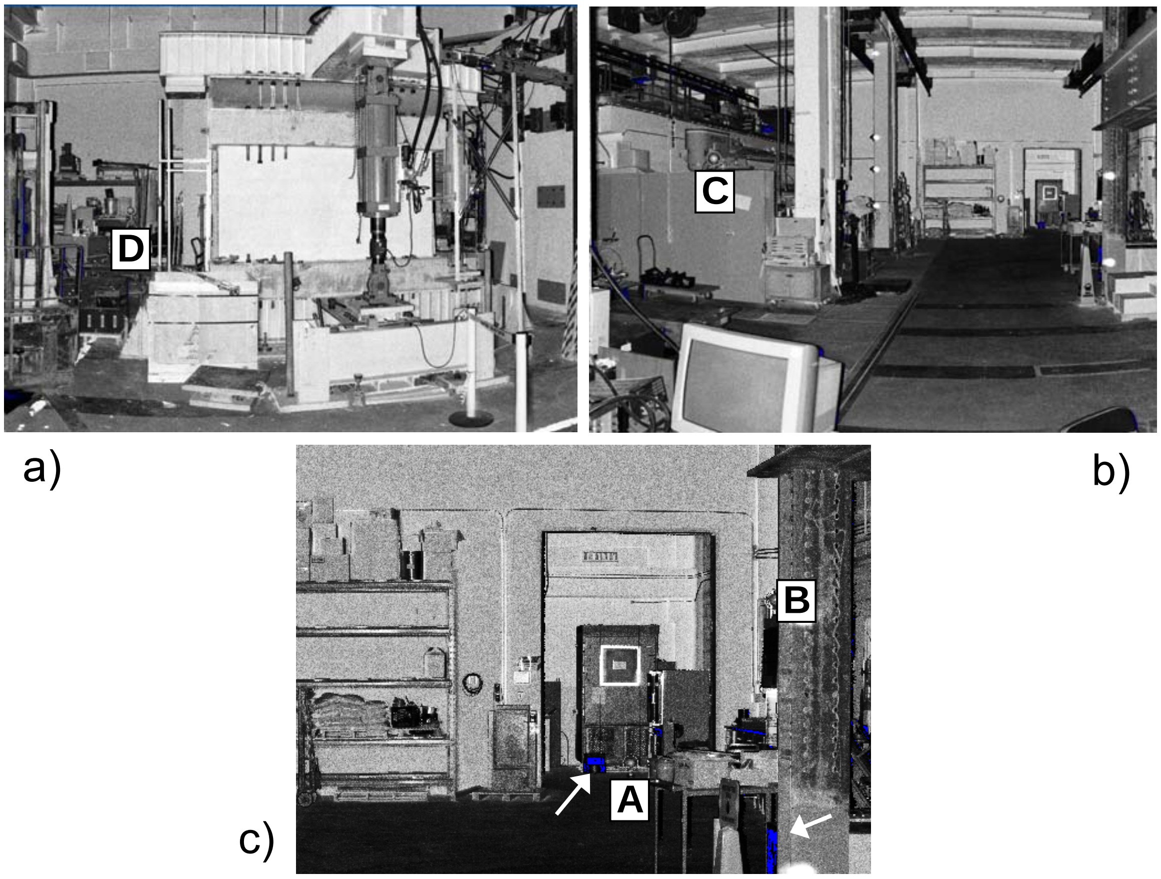

Fig.8 2D intensity images from the scans obtained with an angular increment 0.04º. Shown are three regions of the lab scanned from position Pos1: a) sphere D (mounted on tripod); b) sphere C on top of a cabinet; c) spheres A and B. Sphere B is partially occluded by a steel column. White arrows point toward color patches marking no-data regions, all spheres occupy less then 0.01 % of the scanned scene.

For two instrument locations, the pair of matching triangles is expected to be $\{A^{(1)}, B^{(1)}, C^{(1)}\}$ and $\{A^{(2)}, B^{(2)}, C^{(2)}\}$. This pair should have the smallest value of error function defined by (9) because $D^{(2)}$ is farther away from the scanner than $A^{(1)}$ and Fig.7 shows that the sphere fitting error increases with range. Based on this observation, it is useful to define the largest distance, R_{match}, between a vertex from the matching triangles and the corresponding instrument location. From Fig.5, R_{match} = 25.2m for sphere $A^{(1)}$.

Parameter R_{match} is related to the parameter r_{max} given by (10), and they both should be used in planning the locations of the spheres and instrument as well as the number of required spheres M prior to conducting the scans. We recommend that $R_{match} < r_{max}/2$ based on the observations presented in the following paragraph.

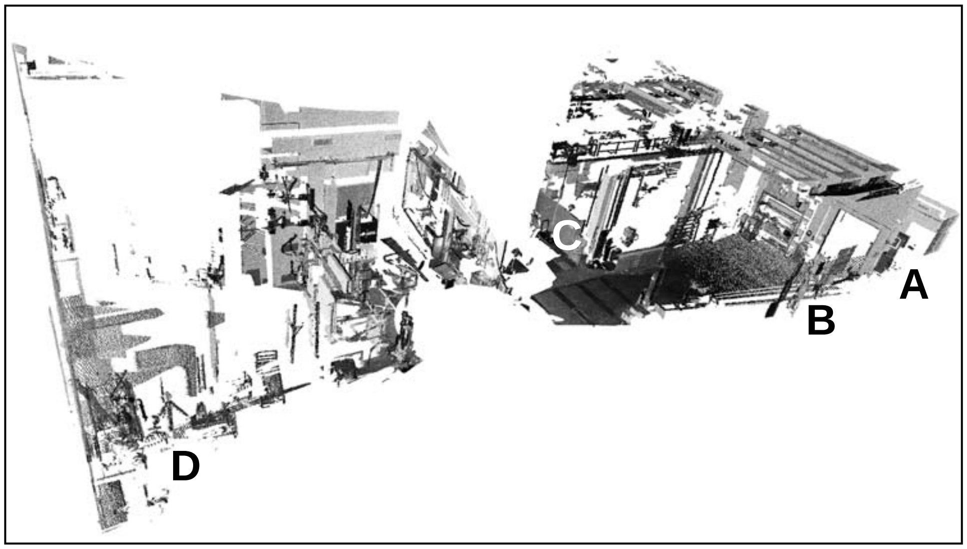

Fig.9 Perspective view of the 3D point cloud representing the dataset scanned from position Pos1, angular increment 0.04º.

The algorithm failed for all tested input parameters for datasets acquired with the largest angular increment $\delta = 0.14º$. It is not surprising because for this δ, the corresponding value of r_{max} is 31 m and the condition $R_{match} < r_{max}/2$ is violated (R_{match} is greater than $r_{max}/2$ by 60 %). Conversely, for the smallest $\delta = 0.04º$, the corresponding r_{max} is 109 m. In this case, the condition is satisfied and the algorithm succeeds for all 160 combinations of the input parameters. For the intermediate $\delta = 0.08º$, the corresponding $r_{max} = 54$ m and the condition $R_{match} < r_{max}/2$ is also violated but in this case, R_{match} is greater than $r_{max}/2$ by 7 % . In this case, for about 50 % of the cases, the algorithm was still able to correctly match the three sphere centers in both coordinate systems. It should be noted that the region where the algorithm was successful, contains a compact subset on [T_{fill} S] plane defined by $0.55 \leq T_{fill} \leq 0.6$ and $3 \leq S \leq 5$, (see Fig.7). However, since the size of this region depends on how much the condition $R_{match} < r_{max}/2$ is violated, we would not recommend using these values and we conservatively suggest that the condition for R_{match} be met.

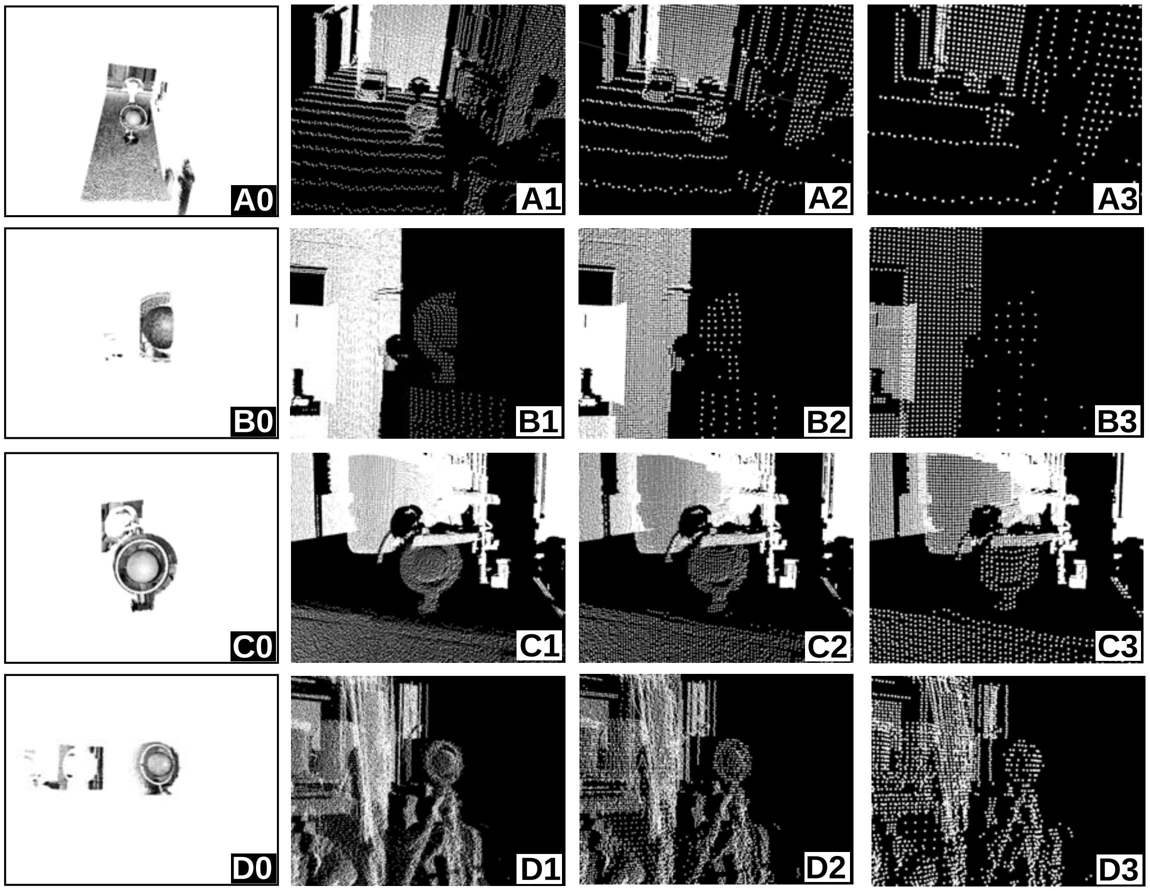

Fig.10 Zoomed-in 3D point clouds containing sphere targets scanned from position Pos1. Each row shows the same sphere, each column represents the same angular increment: 0.004º for images A0-D0, 0.04º for A1-D1, 0.08º for A2-D2 and 0.14º for A3-D3. Column 0 shows datasets used to define ground truth and each image in this column displays all points (except B0 where part of the points occluding a sphere was cropped out). Images in the remaining three columns (1-3) show only a small part (≈ 0.1%) of the corresponding full dataset. To enhance visualization, pixel sizes were increased for decreasing scanning density.

Although our algorithm provides an initial registration, it is very close to the final transformation. This is based on the assumption that if the sphere centers obtained from the algorithm are close to the ground truth values, then the transformation resulting from using these sphere centers will be close to the transformation obtained using the ground truth. The largest deviation between the center and ground truth was 0.05R or 3.81 mm while the smallest distance between two sphere centers (A and B) was 7.3 m.

Step 2 of the algorithm determines the correspondence between the spheres from two datasets without any prior knowledge of the distances between sphere centers. If the distances between the spheres are known, the observed performance of algorithm improved from 78/160 successes to 128/160 successes for angular increment $\delta = 0.08^{\circ}$. For angular increments of $\delta = 0.04^{\circ}$ and 0.14°, this additional knowledge did not change the success or failure ratio. An example of when the distances between the spheres are known would be a manufacturing facility where spheres are placed permanently throughout a facility and an example of when the distances are not known would be a construction site where placement of objects is temporary by necessity.

The order in which two first components of the step 1 are executed is important: we first perform a quick check in step 1A and then, for accepted points only, we calculate the error function given by (6) and (7) in step 1B. Step 1A filters out more than 99.9 % of all data points in an efficient manner: for every checked data point $\mathbf{u}_{m,n}$ and corresponding range $r_{m,n}$, there are only two floating point operations needed to calculate the index T in (4) and (5) (G_{min}/δ needs to be calculated only once), and a maximum of six *if* statements to check the ranges of the selected three points (in fact, since the *if* statements are executed sequentially, the corresponding loop is in most cases terminated after two or three steps). This small number of operations performed in step 1A reduces the processing even though every point in the dataset is checked. Again, it is emphasized that it is necessary to check every point because the targets occupy only a small fraction of the entire scene being scanned and the data contains a lot of discontinuities. Therefore, other procedures such as the Hough transform or those based on RANSAC approaches may take longer because they require more operations per data point. If the execution order of step 1A were swapped with step 1B, much time would be wasted in calculating the error function for data points which are not potential sphere centers. The number of operations per data point in our registration using sphere targets is comparable with those for registration using reflective planar targets. Thus, our algorithm may be an alternative in the situations when the use of planar targets would require additional scans or intermediate registrations of data sets with small overlap.

In summary, we have presented an algorithm which can be helpful in automating target-based registration using spheres. The transformation is determined by finding three matching points (sphere centers) in two datasets obtained from a 3D imaging system. Due to its speed, the algorithm may be very useful in automated construction processes which utilize a 3D imaging system. The algorithm can be very easily parallelized and used in a control loop together with modern fast scanners.

REFERENCES

[1] W. C. Stone, M. Juberts, N. Dagalakis, J. Stone, and J. Gorman, "Performance Analysis of Next-Generation LADAR for Manufacturing, Construction, and Mobility," *NISTIR 7117,* 2004.

[2] S. Hsu, S. Samarasekera, and R. Kumar, "Automatic registration and visualization of occluded targets using ladar data " in *SPIE Laser Radar Technology and Applications VIII*, 2003, pp. 209-220.

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

[4] C. Dold and C. Brenner, "Registration of Terrestrial Laser Scanning Data Using Planar Patches and Imaga Data," in *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'*, 2006, pp. 78-83.

[5] D. F. Huber and M. Hebert, "Fully automatic registration of multiple 3D data sets," *Image and Vision Computing,* vol. 21, pp. 637-650, 2003.

[6] T. Rabbani and F. v. d. Heuvel, "Automatic Point Cloud Registration Using Constrained Search for Corresponding Objects," in *7th Conference on Optical 3-D Measurement Techniques*, Vienna, Austria, 2005.

[7] I. Stamos and P. K. Allen, "Geometry and Texture Recovery of Scenes of Large Scale," *Computer Vision and Image Understanding,* vol. 88, pp. 94-118, 2002.

[8] I. Stamos and M. Leordeanu, "Automated Feature-Based Range Registration of Urban Scenes of Large Scale," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[9] A. Makadia, A. Patterson, and K. Daniilidis, "Fully Automatic Registration of 3D Point Clouds," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[10] S. W. Kwon, F. Bosche, C. Kim, C. T. Haas, and K. A. Liapi, "Fitting range data to primitives for rapid local 3D modeling using sparse range point clouds," *Automation in Construction,* vol. 13, pp. 67-81, 2004.

[11] D. Akca, "Full Automatic Registration of Laser Scanner Point Clouds," in *Optical 3D Measurement Techniques VI*, Zurich, Switzerland, 2003, pp. 330-337.

[12] Y. Reshetyuk, "Investigation and calibration of pulsed time-of-flight terrestrial laser scanners," in *Department of Transport and Economics, Division of Geodesy* Stockholm: Royal Institute of Technology (KTH), 2006.

[13] W. Gander, G. H. Golub, and R. Strebel, "Least-squares fitting of circles and ellipses," *BIT,* vol. 34, pp. 558-578, 1994.

[14] J. Garcia-Lopez, P. A. Ramos, and J. Snoeyink, "Fitting a set of points by a circle," *Discrete and Computational Geometry,* vol. 20, pp. 389-402, 1998.

[15] C. M. Shakarji, "Least-Squares Fitting Algorithms of the NIST Algorithm Testing System," *Journal of Research of NIST,* vol. 103, pp. 633-640, 1998.

[16] C. Witzgall, G. S. Cheok, and A. J. Kearsley, "Recovering Circles and Spheres from Point Data," in *Perspectives In Operations Research*, F. B. Alt, M. C. Fu, and B. L. Golden, Eds. New York: Springer, 2006, pp. 393-413.

[17] J. R. Beveridge, M. T. Stevens, and A. N. A. Schwickerath, "Toward Target Verification Through 3-D Model-Based Sensor Fusion," Colorado State University, Fort Collins 1996.

[18] J. Marjamaa, O. Sjahputera, J. M. Keller, and P. Matsakis, "Fuzzy Scene Matching in LADAR Imagery," in *IEEE International Fuzzy Systems Conference*, 2001, pp. 692-695.

[19] K. Messer, D. d. Ridder, and J. Kittler, "Adaptive texture representation methods for automatic target recognition," in *Proceedings on British Machine Vision Conference*, 1999, pp. 443 - 452.

[20] R. T. Whitaker and J. Gregor, "A maximum-likelihood surface estimator for dense range data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, pp. 1372-1387, 2002.

[21] X. Yu, T. D. Bui, and A. Krzyzak, "Robust Estimation for Range Image Segmentation and Reconstruction," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, pp. 530-538, 1994.

[22] D. Belton and D. Lichti, "Classification and Segmentation of Terrestrial Laser Scanner Point Clouds Using Local Variance Information," in *Image Engineering and Vision Metrology, Proceedings from ISPRS Commision V Symposium*, Potsdam, 2006, pp. 44 - 49.

[23] D. Marshall, G. Lukacs, and R. Martin, "Robust Segmentation of Primitives from Range Data in the Presence of Geometric Degeneracy," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, pp. 304 - 314, 2001.

[24] C. Grönwall, "Ground Object Recognition Using Laser Radar Data," in *Linköping Studies in Science and Technology* Linköping, Sweden: Linköpings Universitet, 2006.

[25] F. Bosche and C. T. Haas, "Automated retrieval of 3D CAD model objects in construction range images," *Automation in Construction,* vol. 17, pp. 499-512, 2008.

[26] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C, 2nd Edition*: Cambridge University Press, 1995.