Partial 3D Shape Matching Using Large Fat Tetrahedrons

J.S.M. Vergeest,
A. Kooijman, Y. Song

Delft University of Technology
Landbergstraat 15, NL-2628 CE
Delft, The Netherlands

j.s.m.vergeest@tudelft.nl

ABSTRACT

We present a method of automatic alignment of two partially matching 3D shapes. The algorithm selects large fat tetrahedrons (LFT) composed of 4 vertices in one shape and exhaustively searches in the other shape for sets of 4 vertices being compatible with the tetrahedron. By selecting such salient tetrahedrons that are relatively wide and fat, although also being not too unlikely to be contained in the overlap region, the cost of search can be reduced. The method is relatively insensitive to noise and not depending on the existence of local shape features nor on feature correspondences. When implemented on a GPU in Cuda, two point sets of 40,000 each can be aligned within seconds. The method is intended to support interactive 3D scan registration applications.

Keywords

Scan view registration, partial shape matching, fat tetrahedron, GPU, Cuda

1. INTRODUCTION

Partial shape matching is an essential step in the reconstruction of geometric objects. One important application is 3D scanning of physical objects. To construct a geometric model from a physical object, multiple scan views are taken, each consisting of range data, *i.e.* 3D points representing the object's surface. Since the orientation of the object relative to the scanning device is different for different scan views, the collection of points from all scan views do not as such represent the object's surface. First the points need to be aligned to each other, that is be transformed to a common coordinate system. The process of aligning the scan views is called scan view registration. From the aligned point sets the surface of the object can be reconstructed, either fully or partially, depending on the coverage of the scan views. If the surface can be fully reconstructed, it can be assumed to represent the boundary of a volume, or solid model. Then a solid model can be derived, which can serve as input to a CAD (Computer-Aided Design) system for further modeling and processing. To base a design on an existing object or on reuse of precedent models is an important paradigm in some industries, such as industrial design engineering. Such a method cannot be successful unless occasional users of scanning devices can easily operate the system. However, the registration task is, even nowadays, still an impeding factor. In practice the user could be a stylist who has manually created a clay model of a future household device. Whereas taking the scan views of the clay model is a commonsense task to him/her, the registration of view pairs is not. The scanning system's manufacturer normally offers an interactive software package, allowing the user to designate correspondences he/she observes among the scan views, as to provide a starting position for a shape matching algorithm, typically based on the ICP (Iterative Closest Point) method. Each scan view has to be aligned, by the user, with scan view(s) aligned previously. Generally this way of operating the scanner is perceived as slow and tedious, both by incidental users and by trained operators.

To improve the operation speed and ease of 3D scanning, the equipment can be enriched with mechanical or magnetic location/orientation trackers attached to the scanner or to the object being sensed or to both of them. Another, commercially available,

solution is based on the detection of optical markers attached to the object being scanned.

If no additional devices or markers are applied and if one does not want to rely on human intervention, the registration problem depend on automatic partial shape matching. Several approaches have been reported to the problem of partial shape matching. From here on we assume that the input data consists of unordered point sets only. That is, we will not rely on preprocesses that generate surface meshes, nor on additional information such as color, texture or material properties of the scanned object. We focus on the kernel problem of matching two point sets. Most methods make use of geometric descriptors and/or feature points. A geometric descriptor is rule to assign a characterization vector to each point in a data set, where the vector is typically computed from points in the neighborhood around the point. The geometric descriptor can take many forms, including moments, FFT coefficients, spin images etc [Johnson 1999]. Since geometric descriptors are invariant under rigid body transformations, they can be used to detect correspondences between two shapes. However, the number of points in a scan view can be large, possibly leading to too many descriptors, causing too long processing time. Then one can attempt to define feature points of shape. Feature points are invariant under rigid body transformations as well, and they are typically small in number. If correct feature points can be generated within the overlap region of two scan views, then the approximate transformation can be easily derived and be applied to the point sets to bring them into a position from which ICP can be successfully used. However, feature points are commonly derived from point differences (for example local curvature) and hence sensitive to noise, which may cause wrong correspondence assignments. Defining a feature using integrated quantities rather than using derivatives reduces the influence of noise [Gelfland 2005]. Another approach to diminish sensitivity to noise and data outliers is taken by [Aiger 2008]. He collects sets of 4 planar points in both point clouds. If a particular set from one point cloud is approximately congruent with one from the other point cloud, a candidate corresponding pair of 4-points sets is found. If the 4-points set is relatively wide, then the method is less sensitive to noise. In this paper we will not further review existing work on partial shape matching. We refer to [Gelfland 2005] and references therein for a more extended description of registration methods.

Our approach is inspired by the 4-points congruent sets as in [Aiger 2008]. We look for 4-points sets which define a large fat tetrahedron (LFT). The assumption is that the geometry of a large tetrahedron is relatively rare and therefore can serve to detect correspondences in the two point clouds. However, since true correspondence exist in the overlap region only, a bound must be set to the maximum size of the tetrahedrons. Secondly, since the number of fat tetrahedrons in point sets can be very large, a straightforward comparison of two sets of tetrahedrons (each derived from one point cloud) would not be efficient. Our algorithm derives a limited number of fat tetrahedrons from one point cloud. Then each tetrahedron is tested for being approximately congruent to any point neighborhood of the other point set. Although the latter step of the algorithm is expensive, it can be easily parallelized and the correct approximate transformation can be found in about 10 seconds in a Cuda implementation.

In the next section the LFT algorithm will be described. In section 3 we present the achievements of the method. Conclusions and recommendations are given in section 4.

2. THE LFT ALGORITHM

Let two point sets A and B be given, originating from sampling of a portion of the surface of a three-dimensional object. There may exist subsets $A' \subseteq A$ and $B' \subseteq B$ such that A' and B' are samples of the same subsurface of the object. A' and B' are then said to represent an overlap region of the samples.

Let a set of sets B_i be a partitioning of B be defined as follows. A three-dimensional grid is constructed, aligned with a bounding box of B. The grid has the size of the bounding box of B. The block-shaped grid elements all have the same size and have index i, $i = 1, ..., n_G$, where n_G is the number of grid elements of B. Each grid element encloses zero or more points of B. Each point of B is enclosed by exactly one grid element. B_i is the set of points enclosed by grid element indexed i.

Let A' and B' be the largest overlap of A and B, informally defined as follows. Assuming that A and B are range images of a physical object, let S_A and S_B informally be defined as the portions of the object's surfaces represented by A and B, respectively. Then, both A' and B' represent a superset of the surface $S_A \cap S_B$. Depending on the extent of $S_A \cap S_B$, A' and B' each may contain zero up to as many points as the cardinality of A and B, respectively.

Let $B'_i = B_i \cap B'$. Our search strategy is based on the assumption that the overlap region is connected and has the extent of at least the size of a grid element. In such cases there might exist sets B_i containing multiple points of B_i'. A property of any point of B' is that its Euclidian distance to A is relatively small, provided that A and B are defined in the same coordinate system. However, since A and B originate

from independent sampling processes, they will in general be defined in different coordinate systems. The difference between the two coordinate systems can be described by a rigid body transformation M, such that MB and A are defined in the same coordinate system, where MB is the set of points of B to which transformation M has been applied. Let b_{ij} be the jth point in B_i, with $j = 1, ..., n_i$, where n_i is the number of points in B_i.

We could naively define a search algorithm as follows:

Exhaustive search algorithm

```
for r = 1, ..., n_r //  loop over possible transformations
  for i = 1, ..., n_G  // loop over grid elements
    n_ri = 0           // number of nearby points so far
    for j = 1, ..., n_i // for points in one grid element
      for s = 1, ..., n_A // loop over points in A
        if | M_r b_ij – a_s| <  δ then n_ri++ // evaluate dist.
      endfor
    endfor
  endfor
endfor,
```

where r defines the parameters of a rigid body transformation M_r. One must then assume that the problem can be solved by considering a finite number n_r of transformations. Typically, r is a 6-dimensional vector defining translation and rotation in 3D space. n_r is the total number of configurations achieved by stepping over finite intervals of translation and rotation. The points contained in A are denoted by a_s with $s = 1, ..., n_a$. The value n_{ri} is the number of those points in grid element i that are close to A after applying transformation M_r to B. When $n_{ri} > 3$ (or any other threshold) we could define r as a candidate parameter set for an approximate alignment transformation. The value of δ (in the algorithm) is typically set to a small multiple of the sampling spacing of A and B.

It is well-known that exhaustive sampling is generally too slow for obtaining scan view registration within seconds, as would be needed for the purpose of interactive 3D scanning. The critical factors here are the step sizes in the six dimensions of configuration space, that define the number n_r of point cloud distance computations. However, using the GPU and adaptive step sizes might prove exhaustive search be feasible in the future [Kooijman 2009].

To reduce the number of distance calculations, we need to extend the algorithm in order to select "promising" transformations. If, for a particular transformation Mr, an (even small) number of points in a grid element is, after transformation, very close to A then these points could be taken as candidates for being points in the overlap region. However, in regions where the object's surface has low curvature, points in the overlap region poorly define the proper transformation M_r. Therefore we need also to consider the degree of planarity of the points close to A. When a small set of points (4 or more) of B_i is close to A *and* if these points are sufficiently non-planar, then the transformation to match this set with A is a relatively good candidate of the M_r we are looking for. Relying on this principle we base the algorithm on matching 4 points to A, where the 4 points are contained in the same grid element. The 4 points, denoted l_1, l_2, l_3, l_4, are selected from B_i such that they form L, a large least-planar set, or large fat tetrahedron (LFT) as follows: l_1 and l_2 are the points in B_i which are furthest apart. l_3 is the point in B_i furthest from the line through l_1 and l_2, that is it maximizes $|(l_3 - l_1) \times (l_2 - l_1)|$. l_4 is the point in B_i furthest from the plane defined by l_1, l_2 and l_3, that is it maximizes $| ((l_3 - l_1) \times (l_2 - l_1)) \cdot (l_4 - l_1) |$.

If L is contained in B' then the directed Hausdorff distance of M_rL to A will be small for the proper transformation M_r. To test whether this is the case and to determine M_r we proceed as follows. At first, we apply a translation to L such that point l_1 coincides with a given point $a_s \in A$ (the following procedure will be repeated for all points in A). We will leave point l_1 unchanged but otherwise rotate L around this pivot point to find out whether the remaining three points can be positioned close to A, which will be the case when there exist points m_2, m_3, $m_4 \in A$, such that $| M_r l_j - m_j |$ is small for $j = 2, 3, 4$. Since (l_1, l_2) is the longer edge of tetrahedron L, all candidate points m_2, m_3 and m_4 must be on or be contained in the sphere of radius $|l_2 - l_1|$ centered at a_s. Let us define $A_s \subseteq A$ as the set of points of A either on or inside this sphere. Candidate points m_2 will be on the surface of sphere up to some accuracy depending on sampling spacing and the precision of the sampling process. Let $m_2' \in A_s$ be such a candidate point. Then L is rotated about point l_1 such that point l_2 comes closest to m_2', which implies that a_s, l_2 and m_2' are collinear. Now there is one degree of freedom left for L, namely its rotation about the line through l_1 and l_2. If point l_3 would (approximately) hit any point m_3 in A_s then L has attained a configuration which should be further evaluated. If, then, point l_4 is close to any point m_4 in A_s, L is called being matching to A. The implied transformation M_r can be determined, as described later. All points in B_i are then subjected to transformation M_r and their distance to A is determined. If a sufficient fraction of the points are close to A then M_r is saved as a candidate alignment transformation.

Algorithm based on LFT (Large Fat Tetrahedron)

for $i = 1, \ldots, n_G$ // loop over grid elements
  select l in B_i // l is large fat tetrahedron
    for $s = 1, \ldots, n_A$ // loop over points in A
      move L such that $l_1 = a_s$ // anchoring 1st point of *LFT* to A
      determine A_s // points of A in sphere at l_1, of radius $|l_1 - l_2|$
      for moves of L such that l_2 reaches A_s // move 2nd point of *LFT* to A
        for moves of L such that l_3 reaches A_s // move 3d point of *LFT* to A
          if l_4 near A_s then evaluate goodness of matching, using points in M_rB_i
          endif
        endfor
      endfor
    endfor
  endfor

The number of candidate transformations M_r depends on the various distance criteria that should be set in the algorithm. As a final step, the candidate transformations are used to compute the set M_rB, involving all points of B, and the degree of matching of the set to A is determined. The transformation producing the best alignment is the outcome of the method. The goodness of an alignment is defined as the number of points in M_rB closer to A than a predefined threshold distance.

The transformation M_r can be written in explicit form as follows. We define transformation M_1 to bring point l_1 to a_s as translation $T(a_s - l_1)$. Let us apply M_1 to L. Subsequently L is transformed by M_2 such that the vector $l_2 - a_s$ is rotated about direction vector $d_1 = (l_2 - a_s) \times (m_2' - a_z)$ to reach direction $(m'_2 - a_s)$, which requires the rotation angle

$$\alpha_1 = \text{acos}((l_2 - a_s) \cdot (m_2' - a_z)) / (|l_2 - a_s| \, |m_2' - a_z|).$$

The axis of rotation goes through point a_s, that is the rotation is indeed around point l_1 of L. Therefore $M_2 = T(a_s)\ R(d_1,\ \alpha_1)\ T(-a_s)$, where $R(d_1,\ \alpha_1)$ is the rotation of angle α_1 about the line through the origin, with direction d_1. An implementation note: the program gets a bit simpler and slightly more efficient if we initially transform all points a_i with $T(-a_s)$, that is point cloud A is moved such that a_s reaches the origin and from then on $T(a_s)$ would become identity. The next transformation, M_3, is a rotation about the current direction vector $l_2 - a_s$ by angle α_2. This angle is determined by point m_3 and computed as the angle between the vectors n_1 and n_2, the vectors normal to the triangles $(a_s,\ l_2,\ l_3)$ and $(a_s,\ l_2,\ m_3)$, respectively, that is $n_1 = (l_2 - a_s) \times (l_3 - a_s)$, $n_2 = (l_2 - a_s) \times (m_3 - a_s)$ and $\alpha_2 = \text{acos}(n_1 \cdot n_2) / (|n_1|\ |n_2|)$. The rotation form n_1 toward n_2 is in positive sense about vector $d_2 = n_1 \times n_2$. Thus $M_3 = T(a_s)\ R(d_2\ ,\ \alpha_2)\ T(-a_s)$. The total transformation gets $M_r = M_3\ M_2\ M_1$, and accounting for some canceling translations we obtain:

$$M_r = T(a_s)\ R(d_2,\ \alpha_2)\ R(d_1,\ \alpha_1)\ T(-l_1). \qquad (1)$$

M_r can be interpreted as a transformation of corner l_1 of the LFT to the origin, then two rotations about lines through origin and finally translating l_1 to point a_s. The explicit matrix form of transformation $R(v, \alpha)$, $v = (v_0, v_1, v_2)$ is

$$R(v,\alpha) = U \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} U^T, \qquad (2)$$

with

$$U = \begin{pmatrix} u_{00} & u_{01} & u_{02} \\ u_{10} & u_{11} & u_{12} \\ u_{20} & u_{21} & u_{22} \end{pmatrix}, \text{ and}$$

$$\begin{pmatrix} u_{02} \\ u_{12} \\ u_{22} \end{pmatrix} = \frac{1}{\sqrt{v_0^2 + v_1^2 + v_2^2}} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix},$$

$$\begin{pmatrix} u_{01} \\ u_{11} \\ u_{21} \end{pmatrix} = \frac{1}{\sqrt{v_0^2 + v_1^2}} \begin{pmatrix} -v_1 \\ v_0 \\ 0 \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} u_{00} \\ u_{10} \\ u_{20} \end{pmatrix} = \begin{pmatrix} u_{01} \\ u_{11} \\ u_{21} \end{pmatrix} \times \begin{pmatrix} u_{02} \\ u_{12} \\ u_{22} \end{pmatrix}.$$

The transformation of equation (1) can be implemented efficiently using this form to transform larger number of points in B_i as to evaluate the closeness of M_rB_i and/or M_rB to A.

3. IMPLEMENTATION AND RESULTS

First, to illustrate the steps of the LFT algorithm we apply it to two scan views (A and B) from the Stanford Bunny data set [Stanford 2009]. The scan views consist of about 40,000 points each. The relative position and orientation of A and B are completely arbitrary, as a result of the particular scanning process. The scan views were decimated to 3185 and 2406 points, respectively (Figure 1) to increase the processing speed. In Figure 2 scan view B is shown as well as one of the tetrahedrons. This tetrahedron happened to be contained in the overlap region. The tetrahedron was selected by the algorithm as follows in this particular example run. The subdivision grid for points B was chosen to consist of 5×5×5 elements. Out of these 125 elements, 8 contained more than 96 points, which was a lower threshold (discussed later), as set for this particular run. In each of these 8 grid elements the LFT was constructed as described in Section 2. Each of the 8 tetrahedrons was sufficiently fat (criterion discussed later) and was subjected to the exhaustive test of congruence with any subset of A. Depending on the criterion for points of a tetrahedron to coincide with a point of A, the number of sets in A congruent to a tetrahedron was typically between 20,000 and 200,000, which is the gross number of candidate transformations per each grid element. For each transformation, the fraction of points M_rB_i (that is the points inside the current grid element) closer than some preset distance to A was computed. If this fraction exceeded a predefined threshold, the number of points in M_rB close to A was determined. The latter computation, involving the full point set B, occurred relatively seldom, between 0 and 300 times only, which is less than 1% of the number of candidate transformations.

In Figure 3 the number of points in M_rB close to A is shown for the candidate transformations M_r, where the x-axis represents the amount of rotation implied by matrix M_r. A couple of almost similar transformations, having rotation angle near 90 degrees, produce the highest score. From the ground truth information that we have about the scan views, we can conclude that the correct alignment was obtained. The points M_rB, which are matching to shape A are shown, together with points A in Figure 4.

The computation of this partial shape matching process took 450s on an ordinary PC, which could be reduced to about 10s when implemented on a GPU, not including a couple of seconds time to import and decimate the point clouds.

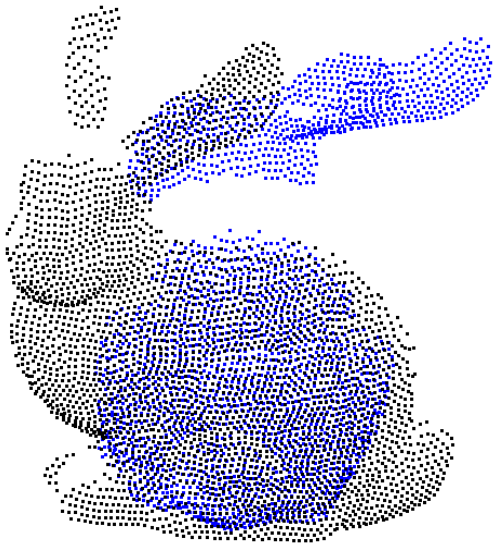

Figure 1. Two scan views (*A* and *B*) of Bunny before alignment.

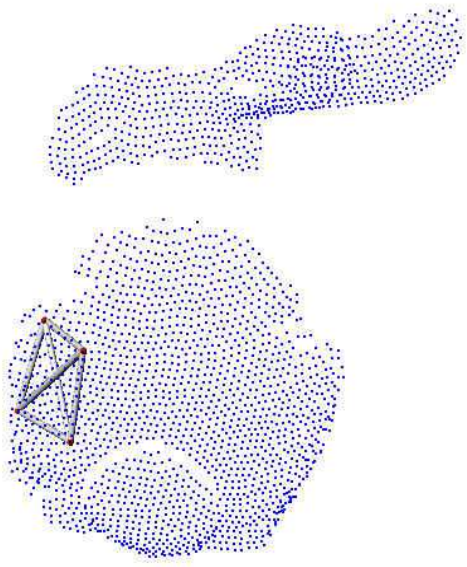

Figure 2. Points in scan view *B* and the best performing LFT depicted by its 4 edges.

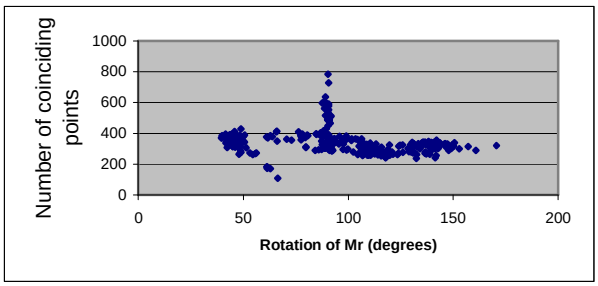

Figure 3. Number of points of M_rB close to A versus rotation angle of candidate transformations M_r.

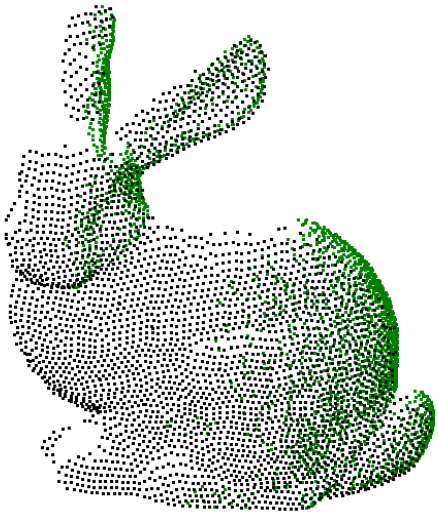

Figure 4. Result of the alignment; point sets *A* and *Mr B*.

As a next example two scan views of a human face, consisting of about 18,000 points each, were taken using our Minolta Vivid700 laser scanner, see Figure 5. Since the overlap region of the two views was relatively large, a good fit could be obtained with sets down-sampled to 9% of the original cardinality. The rotation angle plot is shown in Fig. 6. The computation time for this particular on CPU (not GPU) was about 40s, not including file read/write and cloud decimation, which took together 10s. The best transformation matrix obtained by the algorithm from the down-sampled data differed only little compared to the one from the original data. The amounts of rotations, for example, of the two transformations differed by 0.6 degrees. It is an indication that the result of the algorithm is suited for further processing by common ICP-based registration software.

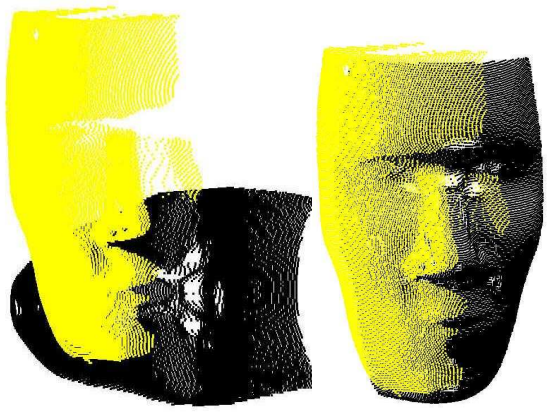

Figure 5. Two scan views before (left) and after partial shape matching (right).

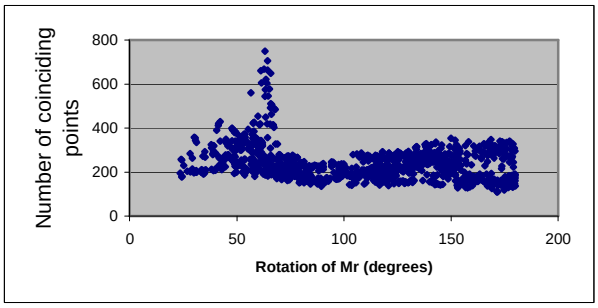

Figure 6. Number of points of M_rB close to A versus rotation angle of candidate transformations M_r for the scan views shown in Figure 5.

In Figure 7 two scan views of an automobile shape are shown. The physical, scaled, model was about 25cm long and very much simplified, containing no detailed shape features. The top view of the car model also contains data from the left side of the car, as can be seen in Figure 7, however that part of the surface is recorded at a small incident angle at relatively poor accuracy. Yet, the tetrahedron selected by the algorithm in scan view *B* (see Figure 8) could be matched to the correct location in scan view *A*, thus providing a good initial match, shown in Figure 9. The tetrahedron is quite large because the size of each grid element is relatively large. We did not remove "outliers" such as data due to the supporting platform and therefore the bounding box got large and so did the grid elements. It can be noted that the background data of scan view *A* has not affected the outcome of the algorithm but has slowed down the search process, since each point in *A* is tested by the LFT algorithm as a potential corner point of the tetrahedron. Using the full data, without decimation of the point clouds (containing about 10,000 points each), the computation time is 15s on GPU. When the point clouds are reduced to about 2000 each, the computation is accelerated

dramatically to less than 1s on GPU or 19s on a CPU. The resulting transformation was still approximately correct, *i.e.* sufficiently accurate to reach fine registration using a commercial tool based on ICP or a general purpose minimizer based on steepest descent.

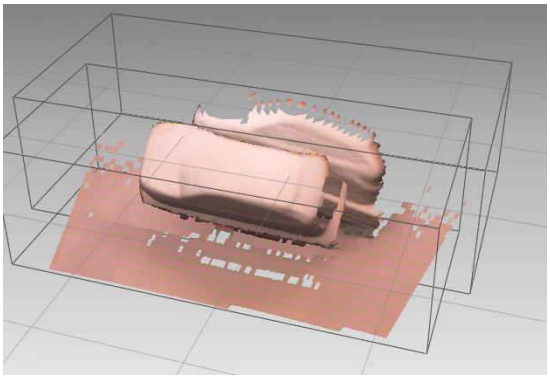

Figure 7. Top view and left side view of a simple car model.

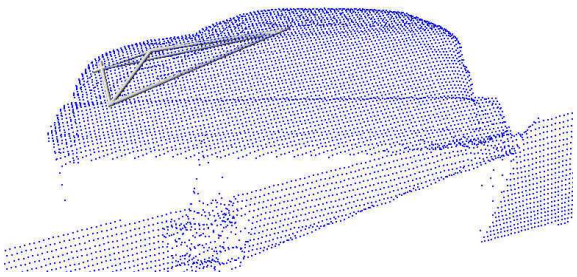

Figure 8. Point cloud *B* representing the left side view of the car and an LFT leading to the correct transformation M_r. The LFT is depicted by its four edges.

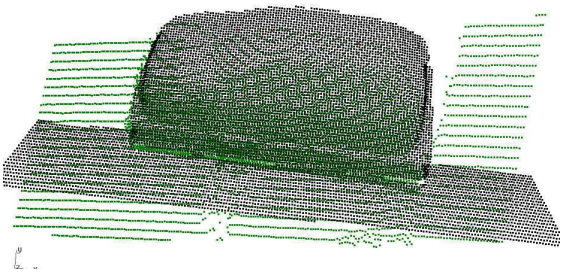

Figure 9. Correct matching of the two scan views from the car model.

4. DISCUSSION AND CONCLUSIONS

The partial shape matching algorithm based on large fat tetrahedrons (LFT) has been evaluated using a, as yet still small, number of data sets.

Both the goodness of the transformation M_r and the amount of computation time spent by the algorithm depend on the settings of various tolerance criteria. For example, if the thickness of the spherical surface of A_s is too small, the correct position of l_2 could be missed; however if the thickness is taken too large then the number of candidate tetrahedrons can become excessive. Similar reasoning can be applied to the number of grid elements subdividing the bounding box, and other parameters mentioned. In our implementation, the chief parameter is the "assumed scanning distance" *d*, which is the typical closest distance between the measured points. All other geometric thresholds in the algorithm are currently defined proportional to *d*.

The algorithm has provided the correct transformation for the partial match in all cases we investigated so far. More examples are being studied.

The distance parameter *d* is critical with respect to computation time. For example, whereas the matching example of Figure 3 took 10s with *d* set to 1.5 units, the computation took 70s with $d = 2.0$. The outcome of the algorithm was the same, but there were relatively many false candidate tetrahedron placements (4 million compared to 0.5 million at $d = 1.5$). The increase of the number of false hits can be seen by comparing Figure 10 to Figure 3.

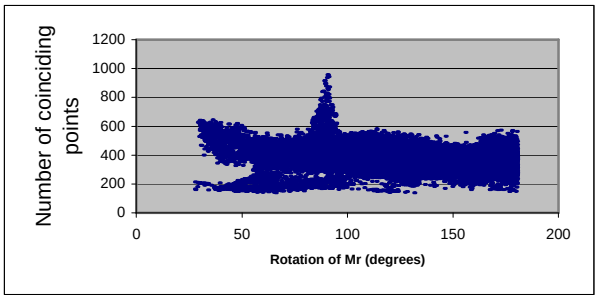

Figure 10. Number of points of M_rB close to *A* versus rotation angle of candidate transformations M_r for the same scan data samples as in Figure 3, with larger parameter *d*.

If the grid element size gets smaller than the overlap region of the two scan views then the algorithm would no longer be able to find a correct match. There should also be a lower limit on the number of points in a grid element to decide whether an LFT will be considered. We have set the default threshold to $n_B / n_G^{2/3}$, which is appropriate for point sets *B* representing a smooth surface.

The numerical tests indicate that the method is not very sensitive to down-sampling of the point clouds and hence could be scalable. The down sampling algorithm that we applied simulates lower precision scanning, and was intentionally not designed to

preserve sharp curvatures. Indeed, the LFT method is not dependent on high curvature features, but on "medium-size" aspects, which are preserved under down-sampling.

In addition the method discards flat regions, including the planar "outlier" points as in the car model, since flat tetrahedrons are by definition non-fat. However, if a grid element contains a mixture of data from the model and from a supporting table, wrong tetrahedrons could emerge.

As mentioned, the computation time depends strongly on the degree of decimation and the values to which the threshold parameters are set. From the numerical experiments we learned how to tweak these parameters to settings which make the computation very fast; however, for a given pair of point clouds it is not *a priory* known how to chose the optimal parameters. The algorithm could be extended with pre-scans of the data in order to estimate an optimal setting.

Since the computation time to match two scan views remains below 10s using the GPU or even the CPU, the LFT method seems very suitable to practically support interactive and mobile scanning processes.

As mentioned, the data samples we used to evaluate the algorithm originate from repositories, such as in [Stanford 2009], or from measurements we conducted ourselves. To our knowledge there is not yet a suitable database containing data samples that could be used for benchmarking purposes for partial shape matching. Indeed, there are databases containing data samples to evaluate shape retrieval algorithms. However, these data samples represent shapes, not partial shapes. The authors look forward to suggestions to achieve at a repository of partial shapes to benchmark scan view registration methods.

REFERENCES

[Aiger 2008] Aiger, D., Niloy, M., Cohen–Or, D. 2008. 4–Points Congruent Sets for Robust Pairwise Surface Registration. ACM Trans. Graph. 27, 3.

[Johnson 1999] Johnson, A.E. and M. Hebert, "Using spin-images for efficient multiple model recognition in cluttered 3-D scenes," IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 5, pp. 433–449, 1999.

[Gelfland 2005] Gelfland, N., Mitra, N. J., Guibas, L. J., Pottmann, Robust global registration. In Proc. Symp. Geometry Processing, Eurographics, pp 197–206.

[Kooijman 2009] Kooijman, A., Vergeest, J.S.M., A GPU-Supported Approach to Registration of 3D Scan Data. Proceedings of the Gravisma 2009, V. Scala (Ed), in press.

[Stanford 2009] Stanford Scan data Repository, http://graphics.stanford.edu/data/3Dscanrep