Published in
Pattern Recognition (In: Lecture Notes in Computer Science) 2449, 91-99, 2002
which should be used for any reference to this work

Fast ICP algorithms for shape registration

Timothée Jost and Heinz Hügli

Pattern Recognition Group, Institute of Microtechnology, University of Neuchâtel,
Breguet 2, CH-2000 Neuchâtel, Switzerland.
timothee.jost@unine.ch, heinz.hugli@unine.ch

Abstract. Shape registration plays an important role in applications such as 3D object modeling or object recognition. The iterative closest point (ICP) algorithm is widely used for the registration of geometric data. One of its main drawback is its time complexity $O(N^2)$, quadratic with the shape size N, which implies long processing time, especially when using high resolution data. Several methods were proposed to accelerate the process. One of the most effective one uses a tree search (k-D tree) to establish closest point relationships and reduces the complexity to O(N logN). This paper reviews several of the existing methods and proposes and analyses a new, even less complex ICP algorithm, that uses a heuristic approach to find the closest points. Based on a local search it permits to reduce the complexity to O(N) and to greatly accelerate the process. A comprehensive analysis and a comparison of the considered algorithm with a tree search method are presented.

1 Introduction

Shape registration plays an important role in today's computer vision. It consists in finding the correct alignment of two or more sets of data. For example, 3D object modeling generally requires to assemble several complementary views into a single representation. Quality inspection and 3D object recognition are other examples of applications of the registration.

A common approach for shape registration uses techniques based on the intrinsic properties of measured surfaces, or geometric matching. One of the best known and widely used low level geometric matching algorithms is the iterative closest point (ICP) algorithm [3]. At each iteration, it first creates closest point correspondences between two sets of points P and X (or more generally geometric data) and then minimizes the average distance between both sets by finding an optimal rigid transformation. There exist several implementations and variations of it which mainly vary in the way they handle the point pairings. A recent overview and comparison of the different methods can be found in [7].

The main practical difficulty of the ICP algorithm is that it requires heavy computations. Its complexity is $O(N_p N_x)$, where N_p and N_x basically represent the number of points of the data sets. Matching detailed high resolution shapes (>20000 points) takes so much time on current computers that there is a real need for ways to reduce ICP computation time. Several solutions to speed up the algorithm have been proposed and can be divided into 3 categories: reduction of the number of iterations, reduction of the number of data points and acceleration of the closest point search.

Besl [3] and later Simon [9] proposed variations named *"accelerated ICP"* which use a linear or quadratic extrapolation of the registration parameters to reduce the

number of iterations. Typical results from these authors showed reductions of the computation time by a factor of 3 and 4 respectively.

Some authors proposed to use a *coarse to fine strategy* [11][10]. They execute the first iterations using a lower resolution, like 1/4 or 1/5 of the points, and finish with fine matching using full resolution. Zhang [11] found a reduction factor of about 2 to 4 using this strategy. Chen and Medioni [5] and Brett [4] proposed to use subsets of the surface points sitting respectively in smooth areas (for robust line plane intersections) and in high curvature areas (to keep significant features).

Benjemaa [1] proposed to project points into Z-buffers and then to perform local searches in them. This method reduces the complexity of closest point search to $O(N_p)$ but it needs a very good starting approximation of the matching and the Z-buffers need to be updated at each iteration. Finally, Besl [3] also suggested that using a k dimensional binary search tree, *k-D tree* [2], would greatly decrease closest point search time. Theoretically, searching the closest point in a k-D tree having N entries is of complexity O(logN). Thus, using a balanced k-D tree in the case of ICP, the global complexity of the algorithm becomes $O(N_p \log N_x)$. Practical results [9] showed gains of about 15 for meshes containing ≈2500 pts. and one can note here that the gain increases with the number of points, due to the reduction of the global complexity. Both theoretical and practical results confirm the importance of closest point search speedup to create fast ICP algorithms.

The "neighbor search" closest point algorithm is presented in this paper. It is designed to be adapted to the ICP algorithm and to optimally exploit, for speedup, the closest point neighborhoods of the two surfaces to be aligned. Specifically, it assumes that two neighbors on a surface possess closest points on the other surface that are neighbors and uses this property to obtain a first approximation of the closest point. It then refines the result with a local search. This method permits to avoid a global search for most points and leads to a closest points search (and ICP) algorithm that provides a complexity of $O(N_p)$. The neighborhood either exists a priori or can be built in range images, triangulated meshes and clouds of points, which are the common ways of obtaining and storing the 3D data measurements to be used in most applications that require registration.

Section 2 describes the "neighbor search" closest point algorithm. Some experiments and results using this new heuristic method are presented in section 3 as well as a comparison with a k-D tree search. Finally, conclusions and a future work discussion can be found in 4.

2 Neighbor Search Closest Point Algorithm

2.1 Neighborhood Relationship Hypothesis

The proposed algorithm assumes the existence of a neighborhood relationship between the two sets of points P and X. The relationship hypothesis is that two neighbors in a data set possess closest points that are neighbors in the other data set. Formally, the principle of this neighborhood relationship is exposed in figure 1: given a point $\mathbf{p_k}$ in data set P and its corresponding closest point $\mathbf{x_k}$ in data set X, the closest point $\mathbf{x_i}$ of $\mathbf{p_i}$, if $\mathbf{p_k}$ belongs to neighborhood V of $\mathbf{p_i}$, $V(\mathbf{p_i})$, is found in the neighborhood V' of $\mathbf{x_k}$, $V'(\mathbf{x_k})$.

The proposed idea towards a faster search is to use good approximations of the closest points instead of exact closest points. The neighborhood relationship is used to get a first approximation of the closest point and, then, a local search can be performed to refine the result instead of an exhaustive one: if $\mathbf{p_i}$ possesses a neighbor $\mathbf{p_k}$ in data set P, with a known closest point $\mathbf{x_k}$ in data set X, finding the closest point of $\mathbf{p_i}$ can be reduced to searching the closest point in the neighborhood V' of $\mathbf{x_k}$, V'($\mathbf{x_k}$).

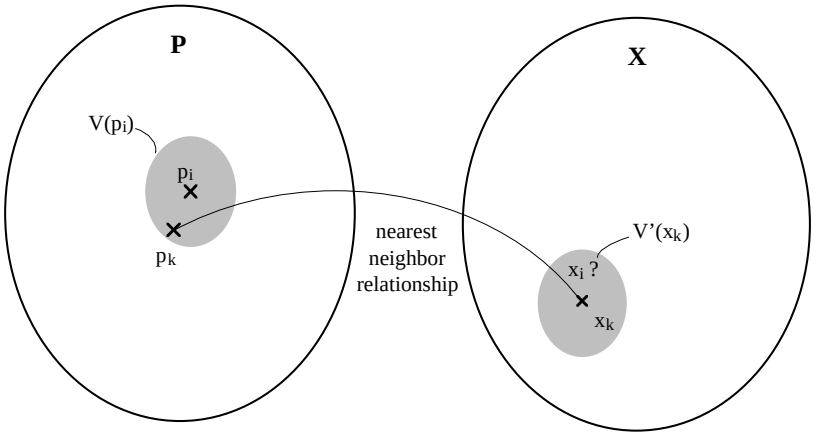

Fig. 1. The neighborhood relationship assumption

2.2 Basic Algorithm

The following procedure formulates the closest point search algorithm:

```
-input:      data sets P and X, with associated neighborhood,
             V(P) and V'(X)
-output:     for each p_i of P, an approximation x_i of its
             closest point in X
-procedure   neighbor_closest_point_search (P, X)
             for (each p_i of P) do:
             if ( ∃ x_k closest point of p_k ∈ V(p_i),) then:
                    x_i = search_closest_point (p_i, V'(x_k) )
             else:
                    x_i = search_closest_point (p_i, X)
```

It appears that for each point $\mathbf{p_i}$, the closest point search is performed either in the full set X or only in the neighborhood V'($\mathbf{x_k}$), depending if at least one neighbor of $\mathbf{p_i}$ has already a known closest point. Formally, this closest point search algorithm has therefore a theoretical best case complexity of $O(N_p)$. This is better than the best case using k-D tree, $O(N_p \log N_x)$ and lets us expect an overall gain in speed over using a tree search.

The worst case complexity corresponds to the conventional full search discussed so far. A good suggestion here is to use a tree search, instead of an exhaustive search, when searching the full set X - other potential ideas would be to use a temporal cache or a heuristic method like 3 steps algorithm, in range images -. Using this method, it is interesting to note that the neighbor search algorithm acceleration worst case is basically equal to the tree search $O(N_p \log N_x)$, as long as the cost of the local search is smaller than a tree search of course.

2.3 Data Structure Considerations

Structured data sets are needed to have a defined neighborhood. In many cases, it already exists a priori or at least can be built. Generally, applications using 3D registration rely on either range images, clouds of points or triangle meshes as their input data. A neighborhood exists in a range image, where direct neighbors in the image are also neighbors on the surface. It also exists in the case of triangle meshes, where neighborhood is defined by belonging to the same triangle edge. In the case of a cloud of points, an existing triangulation algorithm (like the ball-pivoting algorithm for example) could be used to create a triangle mesh. Such a neighborhood is needed in both the P mesh, to obtain closest point approximation, and the X mesh, to make the local search.

Of course, the order in which points $\mathbf{p_i}$ of P are scanned is also important. Using a random method is a bad idea, as it would create a high number of global searches and pushes complexity toward the worst case. Consequently, the basic idea is to scan points using a diffusion method, such as the next point to be scanned is chosen in the neighborhood of the points that already have a known neighbor.

2.4 Algorithm Applied to Range Images

In a range image, each point generally possesses 8 direct neighbors (except points on borders). A very basic algorithm is considered here. Neighborhood V is 3x3 in P and neighborhood V' is nxn in X. We choose to scan the points of range image P *row by row, starting from upper left*. That way, the possible direct neighbors of $\mathbf{p_i}$ with a known closest point $\mathbf{p_k}$ can be found on the previous point in the same row and in the previous row (see image P on figure 2). Those 4 possible candidates are just checked sequentially and the first one that possesses a known closest point is chosen as $\mathbf{p_k}$. Normally, any of the candidate neighbors possesses a known closest point, except for the first scanned point and in case of missing data points.

Once $\mathbf{p_k}$. and its corresponding closest point $\mathbf{x_k}$. are known, the local closest point search of $\mathbf{p_i}$ is done in a square neighborhood zone of size nxn, centered around the approximation $\mathbf{x_k}$ (see image X in figure 2). If no $\mathbf{p_k}$ can be found, a global search is performed in X, as suggested previously. Of course, the bigger the zone, the better the approximation, but the longer the search, so a compromise has to be found. Bigger search zone also implies a bigger neighborhood, which reduces the number of unconnected subsets in the range image.

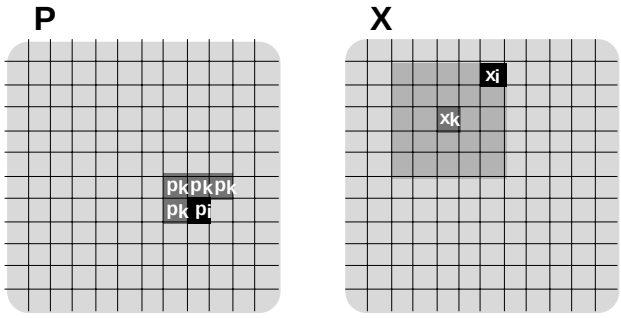

Fig. 2. Neighbor search using range images

3 Experiments and Results

In order to validate the proposed approach, the new neighbor search algorithm has been tested on different data sets and compared to an optimized ICP using a k-D tree search, similar to the one proposed by Zhang [11]. The comparison must focus on two things: computation speed and matching quality.

3.1 Matching Quality Comparison

Two measures can be considered to examine the quality of the matching procedure: the matching error and the domain of convergence. To compare the matching error, the algorithm must converge and the resulting alignment has to be the same or at least in the same error range as when matching using exact closest points.

To examine the domain of convergence, we use a method presented in [6] that permits to define and compare domains of successful initial configurations (SIC). A successful initial configuration is basically defined by a relative initial positioning of the two data sets that leads to a successful matching. The initial configuration space possesses 6 dimensions, so examining all of it isn't conceivable due to both heavy computations and difficulty to handle the results. The idea consists in examining a subset of "interesting" initial configurations that are liable to converge successfully. The different initial configurations are defined in a three-dimensional space by placing one surface on several points of the circumsphere of the other one. The results are plotted in 2D in a SIC-map. Each successful initial configuration is represented by a black sector (fig. 4). The higher the number of black sectors, the bigger the SIC domain is.

3.2 Matching Experiment

The presented experimental data consist in two different, partially overlapping data sets, which typically represents the problem of views registration for virtual modeling. The data are range images of the surface of an irregular substrate measured by an AFM and their overlap is approximately 30% of their total surface. Figure 3 shows a render of the two coarsely aligned sets. The considered meshes contain approximately 930 points.

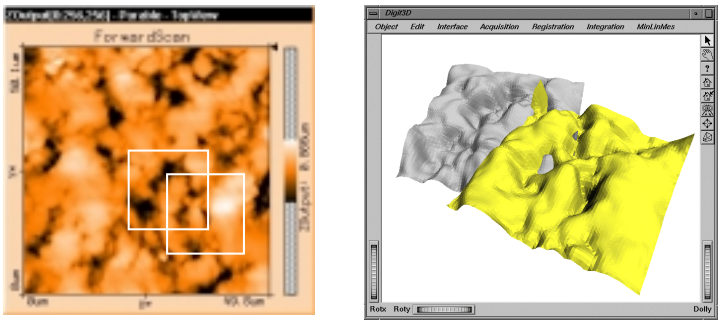

Fig. 3. Range image and surface rendering of two partially overlapping AFM data sets

Matching Quality. Trying to register both meshes using the sole point to point distance implies that the ICP algorithm practically never converges to the right positioning. Therefore, normals are added to the distance measurement, as shown by schütz [8], to improve the matching quality.
A comparison of the average errors show that they do not differ significantly in case of successful convergence. The registration error using k-D tree is about 1% in translation and 2° in rotation. There is a very small increase of the error in translation and respectively 0.2° to 0.8° in rotation when using 9x9 and 5x5 search zones.

The SIC-maps of figure 4 show that the range of successful initial configuration is practically reduced to nothing when using a 5x5 neighbor search zone but that it is even bigger when using a 9x9 zone than when using a tree search!

One can finally note that a small increase of the number of iterations of ICP can be observed when using neighbor search but the cost is negligible.

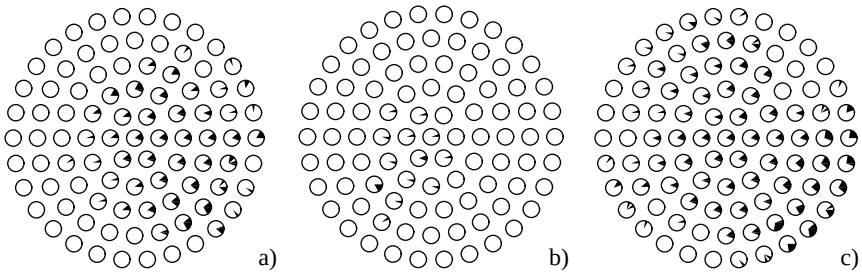

Fig. 4. SIC-maps of the AFM data, using point and normal distance (k=6):a) tree search b) neighbor search with a 5x5 zone c) neighbor search with a 9x9 zone

Search Time. Figure 5 graphically presents the average closest point computation time per point of P, for the partially overlapping sets. The neighbor search method is compared with tree search at different resolutions. Two values of k-D tree are given at each resolution, a minimal time and a maximal time. This is to reflect the difference in the computation time depending on the distance between both data sets when using a k-D tree. Basically, the tree search is longer when data sets are farther from each other. The min / max values must be considered as a best case / worst case type of measure.

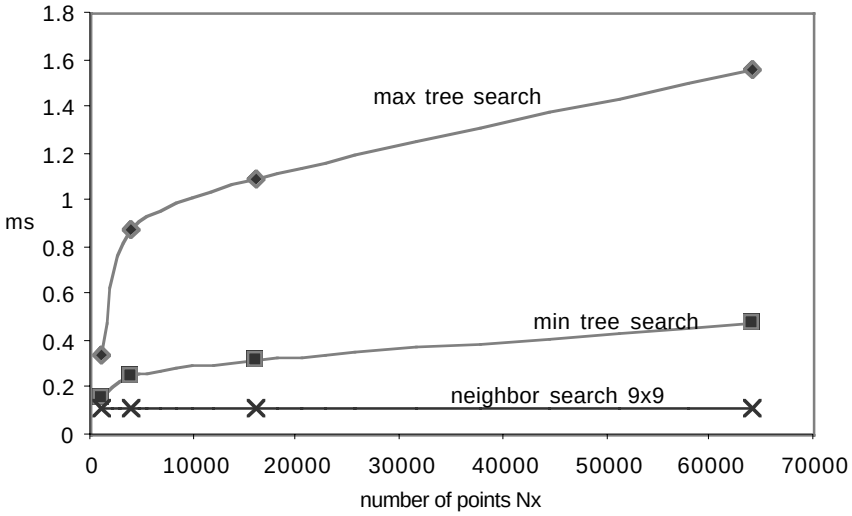

Fig. 5. Comparison of closest point computation time

One can first observe on figure 5 that the theoretical complexity of $O(N_p)$ was practically reached for the neighbor search. The gain in speed lies typically between 2 and 5 over a best case k-D tree. Considering the worst case tree search values, the gain can go up to 13 times!

A complementary aspect of the speedup emerges when measuring the actual acceleration gain with respect to a practical implemented exhaustive search of theoretical complexity $O(N^2)$. Table 1 reports these gains of the closest point computation for a k-D tree, in best and worst cases, and for the presented neighbor search method. These results were obtained by matching partially overlapping data sets, with the setup of section 3.2, using point and normal distance. One can see that the acceleration gain can go up to 650 over an exhaustive search.

Table 1. Closest points search time gain for the best and worst k-D tree cases and neighbor search with a 9x9 zone over an exhaustive search, at different resolutions

number of points	1000	4000	16000	64000
max tree search	3.1	5.2	16.7	47
min tree search	6.6	18.3	57.4	153
local search, 9x9	9.5	39.8	154	650

4. Conclusion and Future Work

This paper presented a comparison of existing methods to speed up the ICP algorithm and proposed and analyzed a new algorithm for very fast ICP registration. It permits

to reduce ICP complexity to $O(N_p)$ and differs from other approaches in that it uses a heuristic method to establish closest point correspondences. The method uses the assumption that two neighbors on a surface possess closest points that are neighbors on the other surface to easily obtain a first approximation of the closest point and then proceeds with a local search for refining the result.

Results from a series of registration experiments show that the proposed neighbor search algorithm performs significantly better than a tree search, the theoretical complexity of O(N) being practically reached. Basically, the neighbor search algorithm ranges from 2 to 13 times faster than a tree search. Also, the method improves the computation speed of the ICP algorithm, without altering the matching error and the domain of convergence. The results show that, in nearly all cases, the ICP registration that uses the neighbor search still converges toward the same position as when using an exact closest points search. This confirms the good potential of the proposed method.

Furthermore, there is still place for improvement, in particular by using multiresolution and other local search strategy.

With respect to multiresolution, the analysis of the results and literature suggest that using a multiresolution adaptation of the neighbor search closest point algorithm will have several very positive factor on ICP and that it can be easily implemented in the case of range images.

With respect to the search strategy, it is clear that a local search method smarter than the square zone search could be used. A steepest descent or a local 3 steps algorithm could both diminish the number of local search and augment their exactness.

References

1. R. Benjemaa, F. Schmitt, “Fast Global Registration of 3D sampled Surfaces Using a Multi-Z-Buffer Technique”, *Proceedings 3DIM,* Ottawa (1997) 113-119.
2. JL. Bentley, "Multidimensional Binary Search Tree Used for Associative Searching", *Communications of the ACM*, vol. 8(9) (1975) 509-517.
3. PJ. Besl, ND. McKay, "A Method for Registration of 3D Shapes*", IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14(2) (1992) 239-256.
4. AD. Brett, A. Hills, CJ. Taylor, "A Method of 3D Surface Correspondence and Interpolation for Merging Shape", *Image and Vision Computing*, vol. 17(8) (1999) 635-642.
5. Y. Chen, G. Medioni, "Object modeling by registration of multiple range images", *International Journal of Image and Vision Computing*, vol. 10(3) (1992) 145-155.
6. H. Hügli, C. Schütz, "Geometric Matching of 3D Objects: Assessing the Range of Successful Initial Configurations*", Proceedings 3DIM*, Ottawa (1997) 101-106.
7. S. Rusinkiewicz, M. Levoy, “Efficient Variants of the ICP Algorithm”, *Proc. 3DIM*, Quebec (2001) 145-152.
8. C. Schütz, T. Jost, H. Hügli, "Multi-Featured Matching Algorithm for Free-Form 3D Surface Registration*", Proceedings ICPR*, Brisbane (1998) 982-984.
9. DA. Simon, M. Hebert, T. Kanade, "Techniques for Fast and Accurate Intra Surgical Registration", *Journal of Image Guided Surgery*, vol. 1(1) (1995) 17-29.
10. G. Turk, M. Levoy, "Zippered Polygon Meshes from Range Images", *Proceedings of ACM Siggraph*, Orlando (1994) 311-318.
11. Z. Zhang, "Iterative Points Matching for Registration of Free Form Curves and Surfaces", *International Journal of Computer Vision*, vol. 13(2) (1994) 119-152.