

Morphological Techniques in the Iterative Closest Point Algorithm

C.A. Kapoutsis, C.P. Vavoulidis, I. Pitas

Department of Informatics, University of Thessaloniki, Thessaloniki 540 06, GREECE
pitas@zeus.csd.auth.gr

Abstract

This paper describes a method for the accurate and computationally efficient registration of 3-D shapes. The method is based on the iterative closest point (ICP) algorithm and improves it by dramatically decreasing the computational cost of the algorithm's most inefficient step, namely the implementation of the closest point operator. The decrease is achieved with the help of a 3-D volume containing the points to be registered. Prior to the implementation of the ICP algorithm, the Voronoi diagram of the "model" points is constructed in the volume, by means of the morphological Voronoi tessellation method with respect to the Euclidean distance metric. The use of the tessellated volume renders the calculation of the closest point operator extremely fast and speeds up the ICP algorithm tremendously.

1 Introduction

The ICP algorithm proposes a solution to a key registration problem in computer vision: given a "model" 3-D shape and a "data" 3-D shape, estimate the optimal translation and rotation that register the two shapes by minimizing the mean square distance between them. An important application of this algorithm is to register actual data sensed from a 3-D object with an ideal 3-D model. It is also useful in multimodal 3-D image registration in medical imaging (e.g. between NMR and CT volumes), in the shape equivalence problem as well as in estimating the motion between point sets where the correspondences are not known.

A crucial drawback of the algorithm is the high computational complexity of the closest point operator. The morphological ICP algorithm solves this problem by using a 3-D array representing a volume in \mathcal{Z}^3 and by constructing the Voronoi diagram of the model points within this volume, by using the morphological Voronoi tessellation method. Then, the ICP algorithm is employed, with distance calculations substituted by simple array references.

The ICP algorithm is described in detail in [1].

Algorithms from Mathematical Morphology can be found in [2]–[6]. A region growing Voronoi tessellation method with respect to non-Euclidean distance metrics is described in [7]–[9], while [10]–[12] propose ways to incorporate the Euclidean distance metric into it. Algorithms for the analytical construction of the Voronoi diagram of a set of points are given in [13]–[15].

1.1 The Iterative Closest Point Algorithm

In the ICP algorithm, a data shape P is registered to be in best alignment with a model shape X . Both shapes must be decomposed first into point sets, if they are not already in this form. The number of 3-D points in the data shape will be denoted N_p and the number of points in the model shape will be denoted N_x . Thus, shapes P and X are respectively the N_p -tuple $P = (\vec{p}_1, \vec{p}_2, \dots, \vec{p}_{N_p})$ and the N_x -tuple $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{N_x})$.

The closest point operator is denoted by C . Equation $Y = C(P, X)$ means Y is an N_p -tuple $(\vec{y}_1, \vec{y}_2, \dots, \vec{y}_{N_p})$ of points of X , such that \vec{y}_i is the point of X closest to \vec{p}_i .

Given P and Y , the registration vector \vec{q} that best aligns P with Y is given by the least squares quaternion operation [1] $(\vec{q}, d_{ms}) = Q(P, Y)$, where d_{ms} is the point matching mean square error. Notation $\vec{q}(P)$ denotes P after transformation with \vec{q} .

The ICP algorithm follows:

1. Shapes X , P , a tolerance $\tau > 0$ and an initial registration \vec{q}_i are given.
2. Initialize by setting $P_0 = P$, $\vec{q}_0 = \vec{q}_i$ and $k = 0$.
3. (a) Compute closest points: $Y_k = C(P_k, X)$.
(b) Compute registration: $(\vec{q}_k, d_k) = Q(P, Y_k)$.
(c) Apply registration: $P_{k+1} = \vec{q}_k(P)$.
(d) If the change $(d_{k-1} - d_k)$ in mean square error is less than τ , terminate. Else, increase k and go back to Step 3a.

If a dimensionless threshold is desired, we can replace τ with $\tau\sqrt{\text{tr}(\Sigma_x)}$, where the square root of the trace of the covariance of the model shape indicates the rough size of the shape.

The algorithm guarantees that a local minimum of a mean square objective function [1] is found. This minimum depends on the initial registration \vec{q}_i and may not be the global one. In the typical case, many runs of the algorithm are needed, each one with an appropriately chosen different initial registration. Finally, the registration vector that corresponds to the smallest of all local minima is selected.

1.2 The Morphological Voronoi Tessellation Algorithm

Voronoi tessellation is a classical topic in computational geometry. For the purpose of improving the ICP algorithm, the Voronoi diagram has to be constructed in a volume in \mathcal{Z}^3 . Furthermore, the *seeds* of the tessellation of this volume must be the points $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{N_x}$ of the model shape X .

In the following, let $T_{N_x} = \{1, 2, \dots, N_x\}$. For all $i \in T_{N_x}$, $V(i)$ will denote the Voronoi region of point x_i . And, for any point \vec{q} ,

$$M(\vec{q}) = \{\vec{q}' \mid (\vec{q}' - \vec{q}) \in \{-1, 0, +1\}^3 - \{(0, 0, 0)\}\}$$

will be the neighborhood of \vec{q} in \mathcal{Z}^3 .

The algorithm is based on Mathematical Morphology methods. The basic data structure used is a 3-D integer array representing the volume to be tessellated. Each item in the array corresponds to a voxel of the volume and contains the index of the Voronoi region to which the voxel has been assigned. Initially, all voxels (i.e. the respective array items) are set to zero (meaning that they have not been assigned to any Voronoi region yet), except for those that correspond to the seeds of the tessellation, which contain the proper index. Then, the algorithm begins to iterate. At each iteration, every region grows by 1 voxel towards all directions and the newly appended voxels accept the correct indices.

The growth of each region is supported by a queue, much like the way a queue supports a breadth first traversal of a graph: Initially, the queue contains only one item, the seed of the region. Immediately before the k -th iteration, it contains all points that belong to the region and are a distance of $k - 1$ away from the seed. During the k -th iteration, the points are progressively removed and replaced by those that also belong to the region and their distance from the seed is k . We say that the contents of the queue constitute the *propagation front* of the region.

During this growing process, collisions of neighboring regions occur at voxels that reside close to the region borders. They are resolved by explicitly calculating the distances from those voxels to the corresponding seeds.

The algorithm terminates when all queues have got empty. Then, all voxels in the volume have been assigned to a Voronoi region.

The pseudocode of the algorithm follows:

1. The set of seeds $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{N_x})$ and a 3-D array d , representing a portion D of the \mathcal{Z}^3 volume that contains all the seeds, are given.
2. An array of queues $Q = (Q_1, Q_2, \dots, Q_{N_x})$ is created. For all $i \in T_{N_x}$, Q_i supports the growing of the Voronoi region $V(i)$.
3. Initialize the volume:
 - for all $\vec{q} \in D - X : d[\vec{q}] \leftarrow 0$, and
 - for all $i \in T_{N_x} : d[\vec{x}_i] \leftarrow i$,
 and the queues:
 - for all $i \in T_{N_x} : Q_i \leftarrow (\vec{x}_i)$.
4. Till all queues get empty, let i scan T_{N_x} cyclically. For all points \vec{q} currently in Q_i :
 - (a) for every $\vec{q}' \in M(\vec{q})$ that is not yet assigned to $V(i)$ ($d[\vec{q}'] \neq i$):
 - i. if \vec{q}' has been assigned to no region yet ($d[\vec{q}'] = 0$), assign it to $V(i)$ ($d[\vec{q}'] \leftarrow i$) and append it to Q_i .
 - ii. else, \vec{q}' has been assigned to $V(j)$ ($d[\vec{q}'] = j$), for some $j \in T_{N_x} - \{i\}$. So, if $\text{dist}(\vec{q}', \vec{x}_i) < \text{dist}(\vec{q}', \vec{x}_j)$, assign \vec{q}' to $V(i)$ ($d[\vec{q}'] \leftarrow i$), remove it from Q_j , if there, and call *correct*(\vec{q}', i).
 - (b) remove \vec{q} from Q_i .

Due to the nature of the region growing mechanism, it is possible that the first region to “reach” a voxel will not necessarily be the correct one. Step 4(a)ii of the above algorithm detects and corrects such erroneous voxel assignments. Function *correct* does even more: it searches in the neighborhood of the corrected voxel for more misassigned voxels and applies the proper corrections. Its pseudocode follows:

1. The coordinates of the corrected voxel are given in parameter \vec{q}' . The index of the region to which it has been assigned is given in parameter i . Model shape X , volume d and array Q are also visible.

2. A queue Q_0 is created and initialized to hold only \vec{q}' . That is, $Q_0 \leftarrow (\vec{q}')$.
3. Repeat until Q_0 gets empty.
Let \vec{r} be the first element in Q_0 .
 - (a) for each $\vec{r}' \in M(\vec{r})$ that has been assigned to $V(j)$ ($d[\vec{r}'] = j$), for some $j \in T_{N_x} - \{i\}$:
if $\text{dist}(\vec{r}', \vec{x}_i) < \text{dist}(\vec{r}', \vec{x}_j)$, then
assign \vec{r}' to $V(i)$ ($d[\vec{r}'] \leftarrow i$) and
append it to Q_0 .
 - (b) if at least one neighbor of \vec{r} has not been assigned to a Voronoi region yet, append \vec{r} to Q_i .
 - (c) remove \vec{r} from Q_0 .

2 The Morphological ICP Algorithm

The computation of the closest point operator in the ICP algorithm is performed by means of the brute force method: for each data point calculate the distances from it to all model points and select the model point that corresponds to the smallest distance. Clearly, this is an $O(N_p N_x)$ procedure, with Euclidean distance calculations as the elementary operations.

Now suppose that a portion of the \mathcal{Z}^3 volume that contains both the model shape and the data shape is available and that it has already been tessellated with respect to the model shape. Finding the model point closest to a given data point would be a matter of simple reference: an access to an item of the 3-D array that represents the volume. Therefore, the complexity of the closest point operator would decline to $O(N_p)$, with the elementary operations now being table look-ups, which are much faster than Euclidean distance calculations.

This is exactly the novel approach of the morphological ICP algorithm, compared to the classical ICP algorithm. Provided that the proper portion of the \mathcal{Z}^3 is selected, the algorithm first tessellates the volume using the morphological Voronoi tessellation method, then employs the ICP algorithm using the efficient version of the closest point operator implementation.

To establish the efficiency of this approach, we have to argue on three issues.

The first one is whether the integer arithmetic introduced by the \mathcal{Z}^3 volume affects the validity of the algorithm. The answer, given by the experimental results (see Section 3), is negative. It is though possible that the same initial registration will lead the morphological and the classical ICP algorithms to different local minima. However, this is a rather unusual effect and, should adequately many initial registrations are

tested, it is unlikely to prevent the discovery of the global minimum.

The second issue regards the memory requirements of the method. If w , h and d are the volume dimensions in \mathcal{Z}^3 , the method needs at least $\lceil whd \log_2(N_x + 1)/8 \rceil$ bytes of RAM. For the typical case where $w = h = d = 100$ and $N_x = 1000$, this is approximately 1.2 MBytes, a significant but tolerable requirement. The additional amount of memory needed by the queues that support the growing of the regions can be kept relatively low, especially if a dynamic queue implementation is selected.

Finally, there is a question on the time needed for the volume tessellation. Indeed, this is a significantly time consuming procedure that, according to the experimental measurements, renders the morphological ICP algorithm a poor choice for all one-run cases where $N_x < 0.001whd$. Nevertheless, it is most likely that

- the ICP algorithm will have to run repeatedly for the same model and data shapes (with a different initial registration each time) till the global minimum is detected, and/or
- many data shapes will have to be matched with the same model shape.

In these cases the tessellation can be performed only once and the same tessellated volume may be used by all runs of the algorithm. Taking also into consideration that typically N_x will be greater than 0.1% of the total number of volume voxels, we conclude that the morphological ICP algorithm is expected to outperform the classical one in most practical situations.

3 Experimental Results

In the experiments described in this section we have used a volume of $100 \times 100 \times 100$ voxels centered at the origin. The number of data points has been equal to that of model points, namely $N_p = N_x = N$. Five cases for N have been examined: $N = 10, 10^2, 10^3, 10^4$ and 10^5 . That is, 0.001%, 0.01%, 0.1%, 1% and 10% of the number of voxels in the volume, respectively.

For each test case, N model points have been generated and were uniformly distributed within the volume. The data point set was then produced from the model point set via successive rotations of the latter around the x -axis (111°), the y -axis (-37°) and the z -axis (-69°). No translation has been employed. Care has been taken so that all points fall within the volume of size $100 \times 100 \times 100$. The same was true for all intermediate point sets produced by the registration of the initial data point set. The classical and the

morphological ICP algorithms were tested on a Silicon Graphics Indy MIPS R4400 200MHz workstation running IRIX 5.3 .

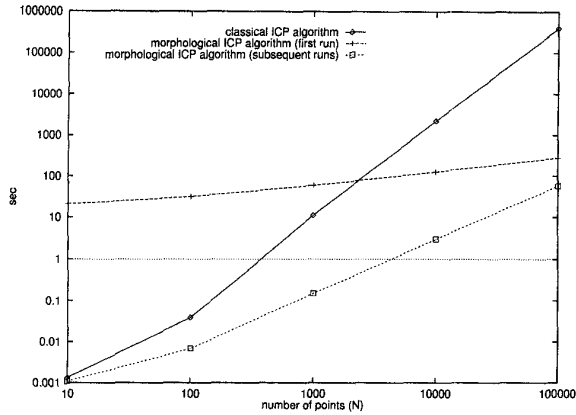


Figure 1: Time performance of the classical and morphological ICP algorithms.

Fig. 1 displays the results of the time measurements. Both axes are logarithmic. The continuous line shows the performance of the classical ICP algorithm. The dashed ones refer to the morphological ICP algorithm. The upper dashed curve shows the total time needed for the tessellation and the ICP algorithm. The lower dashed curve shows the computation time for the ICP algorithm only. Notice that tessellation time is much larger than the time needed for the ICP part.

We see that, if only one run is needed, the morphological ICP algorithm is preferable only in cases having many model/data points ($N > 1000$). For $N = 10000$, the proposed algorithm is close to 3 orders of magnitude faster than the classical one. However, if the tessellated volume is available, the morphological ICP algorithm is superior in all cases: it is 1 to 3 orders of magnitude faster than the classical algorithm.

For each test case there is a minimum number of runs over which the morphological ICP algorithm becomes the best choice. Fig. 2 presents this number for all cases. When $N = 1000$, 6 runs are enough to make the morphological ICP algorithm outperform its classical counterpart.

Finally, in order to convince about the insignificant effect of the integer arithmetic on the validity of the algorithm, we display in Fig. 3 an example of the progress of the point matching mean square error during the iterations of the algorithm. The two lines almost coincide with each other. The example is taken from the $N = 1000$ test case.

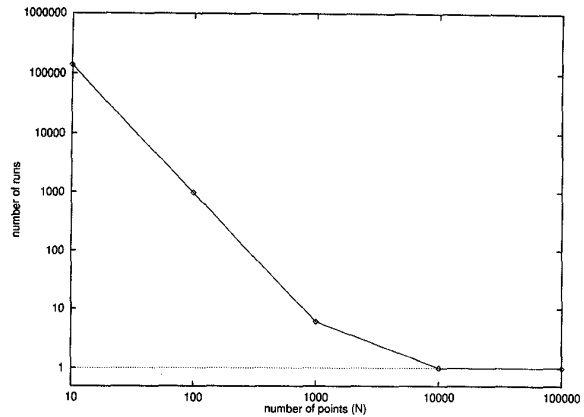


Figure 2: Minimum number of runs for preferring the morphological ICP algorithm.

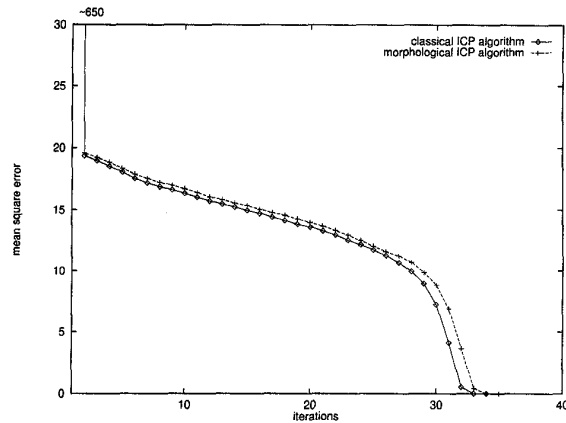


Figure 3: Progress of the point matching mean square error for the test case $N = 1000$.

4 Conclusions

The morphological ICP algorithm is a strong and fast method for the registration of actual data sensed from a 3-D object with an ideal 3-D model. It is faster than the classical ICP algorithm, especially in cases where multiple runs of the algorithm are required and/or a lot of data/model points are involved. Since these cases are the most likely to occur, the morphological ICP algorithm is expected to be much more useful than the classical one.

References

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

- [2] I. Pitas and A.N. Venetsanopoulos, *Nonlinear Digital Filters: Principles and Applications*, Kluwer Academic, Boston, 1990.
- [3] I. Pitas, *Digital Image Processing Algorithms*, Prentice Hall, 1993.
- [4] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, 1992.
- [5] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, New York, 1982.
- [6] L. Vincent, "Graphs and Mathematical Morphology", *Signal Processing*, vol. 16, 1989, pp. 365–388.
- [7] I. Pitas, "Performance Analysis and Parallel Implementation of Voronoi Tessellation Algorithms Based on Mathematical Morphology", in *Advances in Digital and Computational Geometry*, A. Rosenfeld, R. Klette, F. Sloboda editors, Springer Verlag, 1998.
- [8] G. Borgefors, "Distance Transformations in Digital Images", *Comput. Vision, Graphics, Image Proc.*, vol. 34, 1986, pp. 344–371.
- [9] J.E. Mazille, "Mathematical Morphology and Convolutions", *Journal of Microscopy*, vol. 156, 1989, pp. 3–13.
- [10] C. Kotropoulos, I. Pitas and A. Maglara, "Voronoi Tessellation and Delaunay Triangulation Using Euclidean Disk Growing in Z^2 ", *Int. Conf. on Acoust. Speech and Signal Processing*, 1993.
- [11] F.Y.-C. Shin and O.R. Mitchell, "A Mathematical Morphology Approach to Euclidean Distance Transformation", *IEEE Trans. on Image Processing*, vol. 1, no 2, April 1992, pp. 197–204.
- [12] Q.Z. Ye, "The Signed Euclidean Distance Transform and Its Applications", *Proc. 9th Int. Conf. on Pattern Recognition*, 1988, pp. 495–499.
- [13] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.
- [14] G.T. Toussaint, "Pattern Recognition and Geometrical Complexity", *Proc. 5th Int. Conf. on Pattern Recognition*, 1980, pp. 1324–1347.
- [15] D.T. Lee, "Two Dimensional Voronoi Diagram in the L_p -metric", *J.ACM* 27, 1980a, pp. 604–618.
- [16] B.J.H. Verwer, P.W. Verbeek and S.T. Dekker, "An Efficient Uniform Cost Algorithm Applied to Distance Transforms", *IEEE Trans. Pattern Anal. Machine Intel.*, vol. 11, no 4, April 1989, pp. 425–429.