Preliminary ICP stage for data-thinning based on object geometry

D Tihonkih¹, A Makovetskii¹, V Kober¹ and A Voronin¹

¹Chelyabinsk State University, Br. Kashirinikh, 129, Chelyabinsk, Russia, 454001

e-mail: sparsemind@gmail.com, artemmac@csu.ru, vkober@cicese.mx

Abstract. Basically, the ICP algorithm steps are as follows: for each point in the first set, match the closest point in the second set; estimate mapping parameters using the RMS cost function; transform points using estimated parameters; perform multiple iterations (reconnecting points, and so on). In this paper a preliminary data-thinning algorithm is proposed. It can help to speed up all ICP algorithm stages by reducing the amount of input data. The proposed algorithm is based on the human perception of objects geometry. A human often analyzes edges and corners of objects and does not pay attention to the inner parts of polygons when comparing two objects and looking for similar parts. The algorithm described in this article begins with a search of planes in point’s cloud. Next, the search for intersection of the found planes is performed in order to extract object edges. Finally, the intersection of the edges help us to get object corners. Further, all points not belonging to the edges and corners are removed from the point cloud. In real objects polygons most often occupy a large part of the object, therefore the proposed algorithm allows to get rid of a large number of insignificant points.

1. Introduction

The ICP (Iterative Closest Point) algorithm [1,2] has become the dominant method for aligning three dimensional models based purely on the geometry. The algorithm is widely used for registering the outputs of 3D scanners, which typically only scan an object from one direction at a time. For alignment it is necessary to find a geometric transformation that connects two point clouds in $\mathbb{R}^3$ in the best way with respect to the $L_2$ norm. The ICP algorithm consists of two main stages:

1. Searching of corresponding points (pairs) in two clouds;
2. Minimizing the error metric (variational subproblem of the ICP) [3-12].

The computational complexity of minimizing of error metric and, especially, searching of the correspondence points essentially depends on the number of points in the clouds [13-17]. In this article, we propose the method that need to use before the ICP algorithm. It serves to thin out point clouds. The main purpose of the ICP algorithm is to find a match between point clouds. To reach this goal a series of consecutive steps is performed. At each step, 2 point clouds are used. The more points there are in the clouds, the more time is spent at each stage. The proposed method allows to reduce the number of points due to the removal of points that lie...
inside the polygons. The idea is based on the human perception of objects. To implement this algorithm, you can use 2 approaches. The first of them is to look for the internal points of the polygons. After finding - remove them. The second approach is to search for edges of objects on point clouds. After finding - delete all points except them. This article offers a second approach.

2. Method description

Let P be a cloud of N points.

\[ x_i \in P, i = [1..N] \]  

Before starting the main steps of the algorithm it need to find some additional parameters:

a) For each point \( p_i \) take over all other points of the cloud \( p_j, j \neq i \)

b) Find maximum and minimum distances between x, y and z component

\[ p_{x, \text{min}} = \min(p_{xi}, p_{xj}), \quad p_{x, \text{max}} = \max(p_{xi}, p_{xj}) \]  
\[ p_{y, \text{min}} = \min(p_{yi}, p_{yj}), \quad p_{y, \text{max}} = \max(p_{yi}, p_{yj}) \]  
\[ p_{z, \text{min}} = \min(p_{zi}, p_{zj}), \quad p_{z, \text{max}} = \max(p_{zi}, p_{zj}) \]  

(2)

(3)

c) Find the minimum of the differences

\[ p_{min} = \min(\min(p_{x, \text{max}} - p_{x, \text{min}}, p_{y, \text{max}} - p_{y, \text{min}}, p_{z, \text{max}} - p_{z, \text{min}})) \]  

(4)

d) Find the subset radius

\[ r = 0.2 \times p_{\text{min}} \]  

(5)

The radius of the neighborhood of the subset is used to select a subset of points that will participate in the calculations at each iteration of the algorithm.

Let \( P_{\text{used}} \) be the set of points already used. Let \( Pl \) be the set of planes found. Let \( L \) is the set of found lines. Let \( Ph \) is the set of points in the neighborhood. At first, all these sets are empty. For each point \( p_i \in P, i = [1..N] \) we do the following:

a) If the point belongs to the set \( P_{\text{used}} \) then skip it.

\[ \text{if} \quad p_i \in P_{\text{used}} \quad \text{then} \quad i = i + 1 \]  

(6)

b) Find all points belonging to the neighborhood of the point \( p_i \) with radius \( r \)

1) Enumerate all points \( p_j \in P, j = [1..N] \)

2) Find the distance to the center point

\[ l = |p_i - p_j| \]  

(7)

3) If the distance is less than the radius of the neighborhood, add a point to the set \( Ph \)

\[ \text{if} \quad l > r \quad \text{then} \quad j = j + 1 \]  

(8)

c) Find the minimum distance between points \( l_{\text{min}} \)

\[ l_{\text{min}} = \min(|p_k - p_m|), p_k \in Ph, p_m \in Ph \]  

(9)

d) Find the median of the distances between points
1) Find all distances between points
\[ d_u = |p_k - p_m|, \quad p_k \in Ph, \quad p_m \in Ph \] (10)

2) Sort the resulting array
3) Take the center point of the array
\[ d_{med} = median(d_u) \] (11)

e) Find the tolerance distance
\[ d_{tlrnc} = 0.1 \times l_{min} \] (12)

f) Find the tolerance count point
\[ c_{tlrnc} = 0.4 \times count(Ph) \] (13)

g) Declare an array of points in planes \( P_{inPlanes} \)
h) Enumerate all possible triples of points from the neighborhood
\[ v_1, v_2, v_3 \in Ph, \quad v_1 \neq v_2 \neq v_3 \] (14)
i) Check that they do not belong to the set \( P_{inPlanes} \)
\[ v_1, v_2, v_3 \notin P_{inPlanes} \] (15)
j) Check that the points are not too far apart. If not, go to the next three points.
\[ |v_1 - v_2| > d_{tlrnc} \quad \text{or} \quad |v_2 - v_3| > d_{tlrnc} \quad \text{or} \quad |v_3 - v_1| > d_{tlrnc} \] (16)
k) If the points lie on a straight line with some error, skip this triple
\[ line = makeLine(v_1, v_2) \] (17)
\[ \text{if} \quad line \_distanceTo(v_3) > d_{tlrnc} \quad \text{then go to next 3 points} \] (18)
l) Build a plane \( Plane \) on these three points.
m) Find the points \( p_v \) of the original set \( P \) lying in this plane with some error
\[ p_v \in P, p_v \_distanceTo(Plane) < d_{tlrnc} \] (19)
n) Check if the number of points is greater than the threshold, save this plane to set of planes
\[ count(p_v) > c_{tlrnc} \Rightarrow \text{insert Plane to Pl} \] (20)
o) After finding all possible planes, we find the intersections of these planes and save to the set of lines \( L \)
\[ line = cross(Pl_k, Pl_l) \Rightarrow \text{insert line to L} \] (21)
p) Find the points from the initial set lying on the lines with some error
\[ line \in L, p_i \_distanceTo(line) \Rightarrow \text{insert } p_i \text{ to } P_{result} \] (22)

The proposed method is applied to each input point cloud separately. The algorithm does not require additional information about the geometry of objects in the point cloud.
3. Results

The described algorithm allows to reduce the number of points in the source cloud up to 2 times. In the Figures below number input points describes $N_{in}$ and number output points describes $N_{out}$. The first test (Figure 1 and Figure 2) was conducted on the icosahedron with an additional two notches of complex shape. The number of points has almost halved. A second test (Figure 3 and Figure 4) was conducted with the Stanford bunny model. With this input cloud, after executing the algorithm, the number of points decreased by almost 4 times. The third test (Figure 5 and Figure 6) was carried out on a parallelepiped with a cut of complex shape. On this input cloud, the number of points has decreased by about three times. All images show that the algorithm leaves edges of objects and removes points lying inside polygons.

Figure 1. Input cloud $N_{in} = 861$.
Figure 2. Output cloud $N_{out} = 461$.

Figure 3. Input cloud $N_{in} = 1889$.
Figure 4. Output cloud $N_{out} = 457$. 
4. Conclusion
The proposed method decimates the input point cloud and removes points from it from the inner parts of the polygons. This approach allows to obtain a cloud of points from which little significant points are excluded. This reduces the execution time of all stages of the ICP algorithm. Also, the resulting point cloud resulting from the operation of the algorithm gives a better visual perception of objects inside the clouds.

5. References
[1] Besl P and McKay N 1992 A Method for Registration of 3-D Shapes Trans. PAMI 14(2)
[2] Chen Y and Medioni G 1991 Object Modeling by Registration of Multiple Range Images Proc. IEEE Conf. on Robotics and Automation
[3] Picos K, Diaz-Ramirez V, Kober V, Montemayor A and Pantrigo J 2016 Accurate 3D pose recognition from monocular images using template matched filtering Optical Engineering 102-113
[4] Echeagaray-Patron B, Kober V, Karnaukhov V and Kuznetsov V 2017 A method of face recognition using 3D facial surfaces Journal of Communications Technology and Electronics 648-652
[5] Echeagaray-Patron B, Miramontes-Jaramillo D and Kober V 2015 Conformal parameterization and curvature analysis for 3D facial recognition Proc. IEEE International Conference on Computational Science and Computational Intelligence 843-844
[6] Tihonkih D, Makovetskii A and Kuznetsov V 2016 A modified iterative closest point algorithm for shape registration Proc. SPIE’s 61 Annual Meeting: Applications of Digital Image Processing XXXIX 9971 99712D-1
[7] Tihonkih D, Makovetskii A and Kuznetsov V 2016 The iterative closest points algorithm and affine transformations Proc. Int. Conference of Analysis of Images, Social Networks, and Texts (AIST 2016) 349-356
[8] Echeagaray-Patron B, Kober V, Karnaukhov V and Kuznetsov V 2017 A method of face recognition using 3D facial surfaces Journal of Communications Technology and Electronics 62(6) 648-652
[9] Echeagaray-Patron B, Miramontes-Jaramillo D and Kober V 2015 Conformal parameterization and curvature analysis for 3D facial recognition Proc. IEEE International Conference on Computational Science and Computational Intelligence 843-844
[10] Makovetskii A, Voronin S, Kober V and Tihonikh D 2017 An efficient point-to-plane registration algorithm for affine transformations Proc. SPIE’s Applications of Digital Image Processing XL 10396 103962J
[11] Makovetskii A, Voronin S, Kober V and Tihonikh D 2017 Affine registration of point clouds based on point-to-plane approach Procedia Engineering 201 322-330
[12] Faugeras O and Hebert M 1986 The Representation, Recognition, and Locating of 3-D Objects Int. J. Robotic Res 5(3)
[13] Stein F and Medioni G 1992 Structural Indexing: Efficient 3-D Object Recognition Trans. PAMI 14(2)
[14] Johnson A and Hebert M 1997 Surface Registration by Matching Oriented Points Proc. 3DIM
[15] Dorai C, Weng J and Jain A 1997 Optimal Registration of Object Views Using Range Data Trans. PAMI 19(10)
[16] Chen C, Hung Y and Cheng J 1998 A Fast Automatic Method for Registration of Partially-Overlapping Range Images Proc. ICCV
[17] Chen C, Hung Y and Cheng J 1999 RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images Trans. PAMI 21

Acknowledgments
The work was supported by the Ministry of Education and Science of Russian Federation (grant No 2.1743.2017) and by the RFBR (grant No 18-07-00963).