Comparison of ICP-Algorithms: First Results

S. Seeger, B. Glomann, X. Laboureux

In our previous report [1] we introduced a framework for the comparison of ICP-algorithms. Now we present first interesting results of this work.

An ICP-algorithm is the standard way to solve a registration task that is formulated as an optimization problem. In our framework we distinguished the factors that influence the function we have to optimize from the strategies to find the optimum. We implemented three different optimization strategies and found six reasonable influence factors that can be hierarchically arranged in the following way (number of implemented methods in brackets):

- operation[1] for finding corresponding points
  - operation for finding the *closest index* (4)
  - operation for finding the *closest point* (2)
  - treatment of outliers (operations for handling boundary points (2), for handling usual outliers (2) and for handling closest points with a distance above a given threshold (2))
- cost function (1).

Until now the only cost function we used is a least squares sum of distances of corresponding points.

The four methods we have implemented for finding the *closest index* are

1. a straightforward search by testing each of the n_1 points in data set 1 with each of the n_2 points in data set 2 (complexity of $O(n_1 n_2)$) and doing this in each of the l iterations ($O(n_1 n_2 l)$)
2. performing the $O(n_1 n_2)$ search only in the first iteration; in all further iterations doing only a local search over a given neighborhood of the last iteration closest index point ($O(nl)$)
3. building up in a preprocessing step a so called k-D tree ($O(n \log n)$ for $n = n_1 = n_2$) [2] that enables a closest index point search with worst case complexity of $O(n^{5/3})$; using the k-D tree in each of the l iterations (worst case $O(n^{5/3} l)$)
4. building up a k-D tree but using it only in the first iteration (worst case $O(n^{5/3})$); in all further iterations performing a local search over a given neighborhood ($O(nl)$).

Not surprisingly, the fourth method is faster than the other three ones: While using the first method needs 17.10 sec on the average per iteration, it needs 1.48 sec with the second, 1.30 sec using the third and 0.99 sec with the fourth method[2]. The size of the neighborhood in methods 2 and 4 has been arbitrarily chosen to be a level 4 ring (i.e. all points in the neighborhood are not farther away than 4 edges from the closest index point computed in the last iteration) which gives reasonable results in all our test cases. However, since the optimal size of the neighborhood as a tradeoff between time and accuracy depends on the given data sets, we started to implement a method for finding the optimal neighborhood automatically.

As methods for finding the *closest point* we compared simply taking the closest point that belongs to the closest index (see above) with taking the closest linear interpolated point on the triangles. The closest point search using interpolation provides remarkably better results with only neglectable additional time costs. Certainly there are other than linear interpolation schemes that could give motivation for future work.

We would like to emphasize that all the methods to handle outliers are of great importance. Boundary points or usual outliers make the results extremely dependent on the distance threshold that is used to reject closest points as corresponding points. However, a method for automatically finding an optimal threshold is in progress.

As mentioned earlier we have implemented three strategies to optimize the cost function in each iteration step of the ICP algorithm: Firstly, by using a least squares sum of distances of corresponding points as the cost function the transformation parameters could be analytically solved with the help of a singular value decomposition (SVD) of a 3×3 matrix [3]. Secondly we implemented an approximate solution by linearizing the transformation parameters in the least squares sum and solving the resulting system of linear equations by an LU decomposition. Finally we used the Levenberg-Marquardt algorithm for finding the optimum.

It is remarkable that in all of our test cases the linearization strategy exhibits the highest performance, e.g. for two data sets with about 10000 points the ICP with linearization needs 20.7 sec, with SVD[3] 38.3 sec and with Levenberg-Marquardt 33.6 sec. This result is in contradiction to [3] where it is claimed that the analytical solutions are significantly faster.

[1] S. Seeger, B. Glomann, X. Laboureux, *A Framework for a Comparison of ICP-Algorithms*, Tech. Rep.
[2] Z. Zhang, *Iterative point matching for registration of free-form curves and surfaces.*, IJCV, 13(1), 1994, pp. 119 – 152.
[3] K.S. Arun et al., *Least-squares fitting of two 3-d*

[1] Operation is the object oriented term for an abstract function that can be implemented in several ways.

[2] Measured on Pentium II 300 MHz for 10000 data points.

[3] SVD algorithm was not optimized for 3×3 matrices!