NPM3D - Project Report

Rithy SOCHET (M2 IASD) Marc KASPAR (M2 IASD)

1 Paper summary

The research paper "IMPROVING 3D LIDAR POINT CLOUD REGISTRATION USING OPTIMAL NEIGHBORHOOD KNOWLEDGE" written by Adrien Gressin, Clément Mallet, Nicolas David and published in 2012 is about the ICP (Iterative Closest Point) algorithm used for automatic 3D point cloud registration. More specifically, it presents a way to improve this method by using local descriptors around each point. The paper additionally compares the result of different ICP variants (including this one) on three different point clouds.

The authors start by describing the registration problem. Essentially, when an area is captured twice or more, there's a need to fuse the multiple point clouds. While no optimal solution exist for this problem, the ICP method, introduced in 1992, is considered to be the most effective one. It is a very simple iterative method however it can suffer from noise and bad iterations and therefore multiple variants have been introduced like the ones proposed in (Bae and Lichti, 2008, [1]), (Sharp et al., 2002, [4]) and (Rabbani et al., 2007, [3]). The paper then introduces a new variant using the work of Demantké et al. ([2]) who worked on multi-scale analysis of lidar points, based solely on the 3D information.

1.1 Geometric features

The method starts by determining the geometric features for each point calculating an entropy function for different radius sizes, and picking the one that minimizes it. It then chooses the dimensionality features associated to that radius. To do that, for a given radius r, and it's neighborhood $\mathcal{V}^r$ a Principal Component Analysis is performed and gives the first 3 eigenvalues and 3 associated vectors. The linearity, planarity and scatter features are then computed respectively by the 3 formulas:

$$a_{1D} = \frac{\sigma_1 - \sigma_2}{\sigma_1}, \quad a_{2D} = \frac{\sigma_2 - \sigma_3}{\sigma_1}, \quad a_{3D} = \frac{\sigma_3}{\sigma_1}.$$

with $\forall i \in [1,3], \quad \sigma_i = \sqrt{\lambda_i}$.

After normalizing these values to ensure $a_{1D} + a_{2D} + a_{3D} = 1$, the Shannon entropy is then calculated as follow:

$$E_f(V_p^r) = -a_{1D}\ln(a_{1D}) - a_{2D}\ln(a_{2D}) - a_{3D}\ln(a_{3D}).$$

After choosing the optimal radius r^* between r_min and r_max, the associated dimension is determined as follow

$$d^*(V^r) = \arg\max_{d\in[1,3]} \left[a_{dD}\right].$$

1.2 ICP steps and improvements

The paper then presents the 5 steps of the ICP algorithm: Selection, Matching, Weighting, Rejecting and Minimizing.

The selection step samples the initial point cloud in order to reduce computation time. The authors present previous techniques like choosing points randomly, keeping the ones with a high intensity gradient or even selecting points so as to preserve a distribution of normals as large as possible. They then propose two new methods, choosing points with high entropy, since it corresponds to a prominent dimension that makes it easier to take a decision, and excluding points with dimensionality 1 as they correspond to border between surfaces and thin objects and dimensionality 3 as they won't have the same sampling across different view points.

The matching corresponds to associating of each point of the cloud another one in the reference point cloud by taking the on with the lowest distance or by using surfaces/meshes for example. The author don't propose any new method for this step.

The third step consists in weighting each pair. Multiple methods exist like giving the same weight for each pair or giving a weight equal to

$$w_D(P_1, P_2) = 1 - \frac{d(P_1, P_2)}{d_{\max}}$$

So a higher value for closer points. The paper presents 2 new ones, the first being based on the difference of the ellipsoid volumes. For each pair, they assign a value equal to :

$$w_V(P_1 P_2) = 1 - \frac{d_V(P_1 P_2)}{d_{\max}}$$

where :

$$d_V(P_1, P_2) = |V_{P_1} - V_{P_2}|$$

The omnivariance V being equal to the product of the three σ_i.

The second one simply defined as the product of the normals of the pair of points:

$$w_N(P_1, P_2) = \vec{n}_1 \cdot \vec{n}_2$$

The fourth step "Rejecting" removes the pair that are considered bad. It can be done by removing pairs with the highest distance, the ones that have a distance more than 2.5 times the standard deviation of distances of pairs or by removing the top n pairs in terms of distance. This paper proposes to use the last one but with the distance d_V defined above.

The last step "Minimizing" finds the optimal transformation that minimizes the sum of the squared distances between each couple of points. The two main distances used are the point-to-point distance, and the point-to-plane distance using the normal of each point. No new method is presented.

1.3 Dataset

The authors then present the 3 point clouds that they will test their algorithms on : airborne (ALS), terrestrial static (TLS), and mobile mapping systems (MMS).

The first one ALS consists of an urban area captured by two different airborne lidar scanners, one in 2003 with a Toposys fiber scanner, and the other in 2008 with an Optech 3100 EA device. The first one contains 400000 points, with 5 points per squared meter and was , while the second one contains 90000 points with a point density of 2 pts/m^2.

For TLS, the two scans were done with a Trimble system and represent 2 different point of views of an office desk with various objects. The point density is 0.3 pts/cm^2 and there are 20000 points in total.

The third dataset MMS represents a building taken twice at different parts of day, however not exactly the same parts of the buildings are sampled and a 3D shift between both point clouds naturally exists, due to georeferencing process. The point cloud density is variable but is typically of 100 pts/m^2. There around 200000 points per point cloud.

1.4 Experiments and results

The last part of the paper focuses on the experiments. A distance threshold equal to $t = 10R_n$ is introduced with R_n the mean distance of the n closest points. In this experiment, $n = 5$. The mean of the distances smaller than t is computed and allows the authors to compare the performance of the different methods. Finally, they also compare the results to a default method using all 3D points for election, closest point for matching, constant weighting, no rejection and point-to-point distance, without normal computation for the Minimizing step.

They first start by comparing the different selection techniques. 4 methods are used other than the default one:

- Random sampling with 10% of the points are conserved
- Keeping the points with an entropy of at least 0.6 or 0.7
- Keeping the points with a dimensionality feature of 2.

The last methods gives the lowest error while the random sampling always gives the worst ones. The entropy methods speed up the convergence time by 5 to 7 times but sometimes have worst accuracy than the base method.

Then the two new weighting methods are compared to the default one but they have worst performance both in terms of accuracy and speed.

Finally, they compare different rejection techniques :
- Rejecting the worst 70% pairs with regards to the d_2 distance.
- Rejecting the worst 50%, 70% and 90% pairs with regards to the d_V distance.
- Rejecting the pairs with a distance higher 2.5 times the standard deviation of the pair distances.

The omnivariance-based method gives the same results as the default method for the ALS dataset with 90% of rejection but but worst results for a smaller percentage of rejection, however these method seem to perform well on the other 2 datasets.

The two other methods perform worse than the default for TLS and MMS but give good results for ALS.

The authors finally present an optimal variant for each point cloud. For TLS and MMS, they recommend selecting points with entropy higher than 0.7, constant weighting, and keeping only the 70% best matches using d_V. For ALS, they recommend using the $d^* = 2$ selection and d_2^{70} rejection.

2 Paper criticism

Among the strengths of this paper we can notice :

- The paper presents interesting ideas on how to include geometrical features to improve ICP algorithm.
- The authors experimented on different type of point cloud, showing the efficiency of the proposed methods for real case applications.
- The mathematical computations to get the features are clearly shown. This makes implementation easy to reproduce and test as well as adding small variants.

However we can observe a few problems:

- The methods does not seem to improve ICP for ALS point cloud which is not a good thing considering the wide application of airborne lidar. We can then wonder if geometrical features are adapted for ALS or if it should only be used for TLS and MMS.
- All the geometrical features are based on eigenvalues. While it is simple and efficient, we could wonder if adding other features could be good. For example we could use the normals since we should have a good estimation with the optimal radius. This reliance on eigenvalues could be a problem if there are areas with not enough points.
- While there are multiple experiments with the different variants of ICP, these were done only on three point clouds. We can wonder if the method would work on more problematic dataset. For example noisy datasets or datasets with low point density.
- While there are some discussion on the time needed for the computation there are no discussion concerning the computation's complexity of the variant.
- About the writing of the paper, it is mentioned that for the minimizing step of ICP they use normal computation due to the fact that we have a good estimation with the optimal radius and that they have propose no improvement. However in the default setting used, it is stated that for this step there is no normal computation but they offer no explanation for this choice.
- While they showed the computation times of ICP in the plots. The computation time needed to get all the features can be really high. This makes testing take more time needed than what is actually shown. It also requires more computational power.

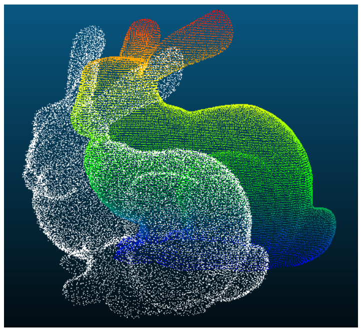
3 Our implementation

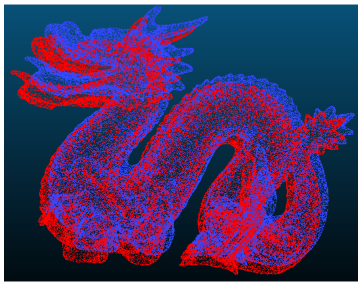
3.1 Dataset

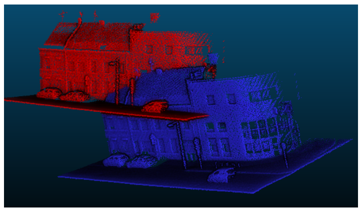
Ideally we would have used the point clouds used in the paper, however they are not available online so we used a few different point clouds used in the practical sessions and available online.

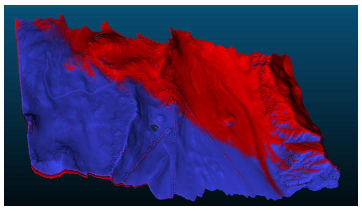
In particular we used:

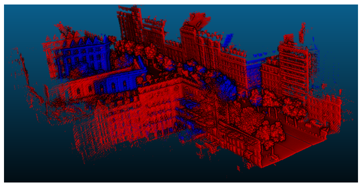
- The bunny and a slightly modified bunny.
- The Lille street point cloud from the third practical session and a slightly modified version.
- An eastern dragon point cloud from the simpleICP github repository.
- An airborne lidar point cloud from the simpleICP github repository.
- The Notre-Dame Des Champs street point cloud from the second practical session.


(a) Bunny and transformed bunny


(b) Eastern Dragon 1 and 2


(c) Lille street and transformed Lille street


(d) Airborne Lidar 1 and 2


(e) Notre-Dame Des Champs

We use the implementation on small objects like the bunny and dragon, but also on larger point cloud like the Notre-Dame Des Champs and Lille. And finally we used it on an airborne lidar point cloud.

3.2 Project file description

The paper is proposing many improvement to the ICP's different steps, in our project we tried implementing these improvements and tested them on the previously describes point clouds. As a baseline we used the implementation used in TP2. We then compared the performance of this baseline with the one with our implementation of the paper's method.

Following the description of the ICP algorithm, we created a python file containing all the variant improving each step. For example, the file `ICP_selection.py` contains functions that can be used in the selection step of ICP. The `ICP.py` contains the ICP algorithm and finally `main.py` contains the program to start ICP on a point cloud. `ICP_features.py` contains all the functions that will compute the different geometrical features of the paper.

We also have a small code in `Transform.py` to create an artificial point cloud to register. It is done by first subsampling the point cloud to keep around 90% of the point cloud, then applying a small rotation and translation. At the end, for each point we add a really small Gaussian noise.

4 Results

The performance of our implementation is measured visually with CloudCompare. We also used the metric used in the paper. This metric consists in computing a statistic called the $n-$resolution R_n of a point cloud to obtain a threshold which will allow us to reject some pairs of points when computing the error. The intuition is that during registration, there are points that should not be paired with a point from the other point cloud. Thus we should not consider it as it would add errors in the measurement.

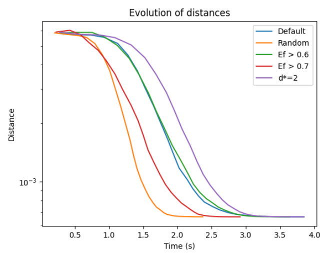
To compute R_n for a point cloud we first compute for each point, the average of distance between its n-nearest neighbors, then we average this information over all points of the point cloud. After obtaining R_n, we set the threshold $t = 10 \times R_n$. After each iteration of ICP, every point whose distance with its matching point in the reference point cloud is higher than t will not be considered when computing the errors.

For each point cloud and each ICP step, we tested different settings and plot the computed distance with respect to time. When we step different method of ICP step, the other steps uses the same default variant as the paper.

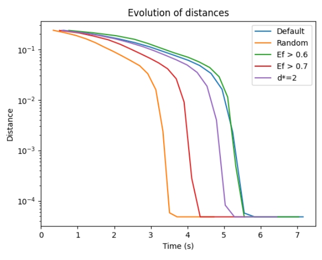
4.1 Selection step

For the selection step we tested multiple methods:

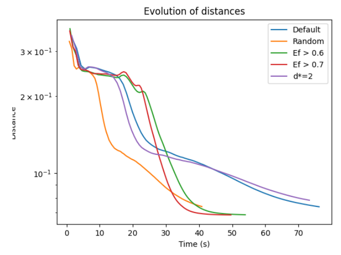
- Default : We select every point.
- Random : We select 10% of the points randomly.
- $E_f > x$: We select point whose entropy is higher than x.
- $d^* = 2$: We select point whose highest geometrical feature is 2 (planarity).



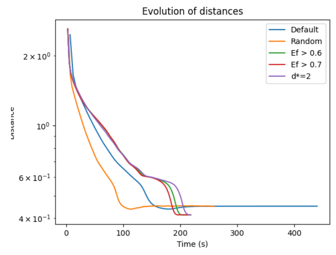
(a) Bunny



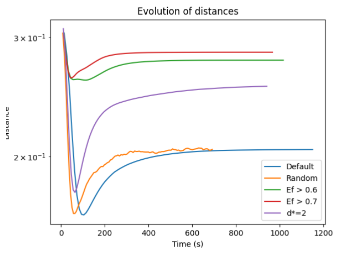
(b) Eastern Dragon



(c) Lille street



(d) Airborne Lidar



(e) Notre-Dame Des Champs

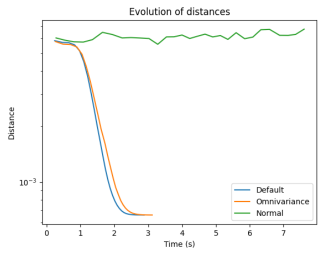
Figure 2: Plots for the selection step

Selecting points with high entropy seems to give better result than the default setting. In particular, taking $E_f > 0.7$ seems to give a faster convergence. $E_f > 0.6$ seems to give similar result to the default setting for the smaller point cloud. For the airborne dataset, all methods (besides randomly selection) seems to perform equally well. While for the Lille dataset, selection based on entropy performs the best. For the Notre-Dame dataset, selection based on the entropy performs the worst. The default and random setting performs well and the selection based on d^* does almost as well. From these plots we can observe that randomly selecting points seems to give us the best results. This contradicts the results obtained in the paper where randomly selecting points gives bad results. This difference could be explained by the way we transformed the dataset to register, which makes the problem easy. This is confirmed by the fact, that without adding the final Gaussian noise, using the entropy gives better results.

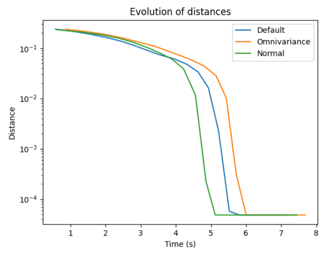
4.2 Weighting step

For the weighting step we tested multiples methods:

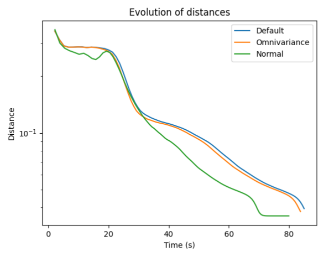
- Default : Every point have the same weight.
- Omnivariance : This weighting method consist in using the volume of the ellipsoid, so we have $w_V(P_1, P_2) = 1 - \frac{d_V(P1,P2)}{d_{max}}$ where $d_V(P_1, P_2) = |V_{P_1} - V_{P_2}|$.
- Normal : This weighting method consist in using the point's normal, so we have $w_N(P_1, P_2) = \vec{n_1}.\vec{n_2}$



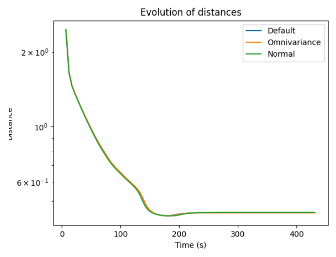
(a) Bunny



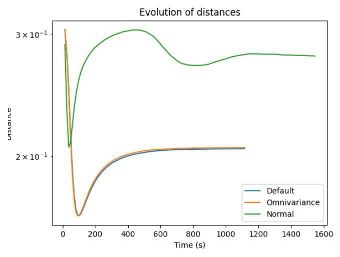
(b) Eastern Dragon



(c) Lille street



(d) Airborne Lidar



(e) Notre-Dame Des Champs

Figure 3: Plots for the weighting step

For the airborne dataset, the three methods performs equally well. In the other datasets, the omnivariance method and default one seems to performs similarly well, there is no major differences between the two. The normal method does not work for the bunny and Notre-Dame Des Champs but seems to performs better for the eastern dragon and Lille street, but the improvement is not that major.

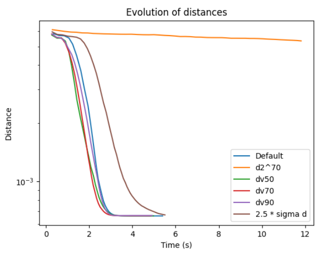
From these plots it seems that using a variant will not necessarily improve ICP. Especially considering that we would need to compute features beforehand, using the constant weighting seems to be the easier and better thing to do.

4.3 Rejection step

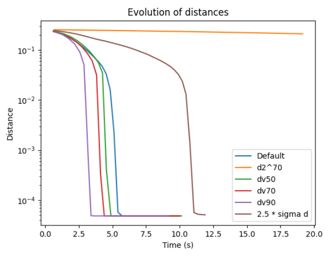
For the rejection step we tested multiple methods:

- Default : We reject no pairs.
- d_2^{70}: We reject 70% of the worst pairs using the Euclidian distance.
- $2.5 \times \sigma(distance)$: We reject points whose distance is higher than 2.5 times the standard deviation of all pairs distance.

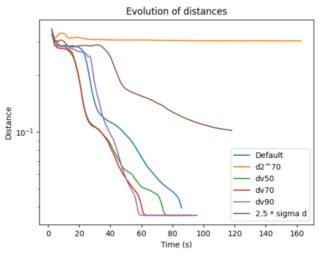
- d_V^X : We reject $X\%$ of the worst pairs using the omnivariance based distance.



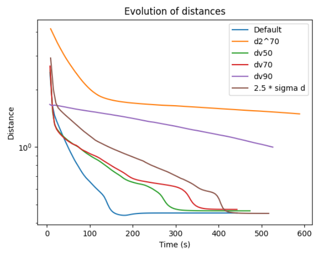
(a) Bunny



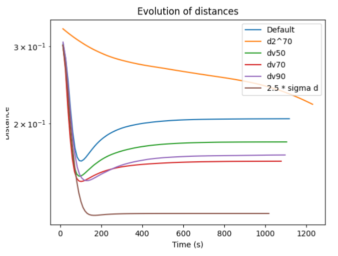
(b) Eastern Dragon



(c) Lille street



(d) Airborne Lidar



(e) Notre-Dame Des Champs

Figure 4: Plots for the rejection step

From these plots we can observe that rejecting based on the Euclidian distance doesn't converge in any dataset. We can remark that even in our airborne dataset this variant gives better result than with other point clouds but doesn't performs as well as what the paper results shows.

The rejection based on the standard deviation never does not gives satisfying results, it seems to almost always be the second worst method except for Notre-Dame where it performs the best.

The omnivariance based distance gives good results, often better than the default setting. In some dataset rejecting 90% of the points seems to better while in other rejecting 50% is better. Selecting to reject 70% of the points based on the omnivariance distance seems to be a good compromise. (This coincides with the paper's proposal for the optimal ICP)

We can notice that with the airborne dataset, the default method performs the best.

With these experiments, we can observe similar results to what was obtained in the paper. However there is still some difference like in the selection step

5 Future work

In our project we implemented and tested the proposed method on different point cloud. For future work we could try adding different geometrical descriptors like curvature, we could also consider adding colors. Finally, we could consider using machine learning to learn geometrical features and use these to improve ICP. Since the paper uses the point-to-point version of ICP, we can wonder if the method works as well for point-to-surface ICP.

Concerning the ALS problem, it could be interesting to try the method again for different kind of ALS point clouds, more precisely, test the hypothesis about the density of points being a cause for the failure of the method. Finally we could also try MMS and TLS point clouds that are more "problematic" due to noise or low density and see if the method holds.

6 Contribution to the project

In this project Marc KASPAR wrote section 1 of the report. He also wrote the code for the ICP step improvements. Finally he ran experiments to get the plots.

In this project Rithy SOCHET wrote section 2, 3 and 4 of the report. He also wrote the code to compute some features and the optimal radius for each point of the point clouds as well as run the code to get the list of these optimal radius.

References

[1] Kwang-Ho Bae and Derek D Lichti. “A method for automated registration of unorganised point clouds”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63.1 (2008), pp. 36–54.

[2] Jérôme Demantké et al. “Dimensionality based scale selection in 3D lidar point clouds”. In: *Laserscanning*. 2011.

[3] Tahir Rabbani et al. “An integrated approach for modelling and global registration of point clouds”. In: *ISPRS journal of Photogrammetry and Remote Sensing* 61.6 (2007), pp. 355–370.

[4] Gregory C Sharp, Sang W Lee, and David K Wehe. “ICP registration using invariant features”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.1 (2002), pp. 90–102.