Comparing algorithms for point cloud registration for accurately locating rod intersections for robot action

Author: Aske Bach Lassen, Teknologisk Institut, Denmark

Abstract (150-250 words, currently 143)

In this paper we test 15 point-cloud registration algorithms (including four iterative closest point (ICP) and three coherent point drift (CPD) variants) to find which ones give least registration error and are fastest for our use case: detecting the intersection point between two rods for an industrial robot application. The algorithms are all in common use. The data comprises both real and simulation-adjusted images of approximately perpendicular metal rods taken from different angles.

The most accurate algorithms of those tested were CPD-affine and CPD-rigid, but they were both rather slow. SVR-rigid (support vector regression) was almost as accurate and considerably faster. Accuracy is much more important than speed for our application, as the time taken for matching is always significantly shorter than the time required for the robot to act. Further investigation is needed before the final "best" algorithm can be chosen.

Keywords: 3D point cloud, registration algorithms, localization, robot action

Declarations:

This project was funded as a Financial Support to Third Parties (FSTP) award under the European Union's Horizon 2020 project Robot Union, Grant Agreement no 779967. There are no conflicts of interest. All code is already in the public domain. All the experimental data is confidential except for the data in this document.

1. Introduction

There are many scenarios that require a real-life image to be matched with a template image that is annotated to give key points for robot action. Matching the images allows offsets to be calculated and robot parameters to be adjusted to make the action more likely to succeed. The camera is placed at a known location relative to (often: fixed to) the robot system, so that the conversion from camera-relative coordinates to robot-relative coordinates is fixed or at least easily calculable without the need for recalibration. For robot-guidance applications like this, relative location needs to be as accurate as possible, but we are not interested in any absolute locations.

Finding a robot action point in the real world requires images that give information in 3D. Normally "3D" sensors (e.g. stereo camera or cameras that include a range finder) collect a 2.5D image which is converted to 3D point cloud information. 2.5D is viewpoint-dependent and cannot see the back of anything. Nevertheless, 2.5D images contain sufficient information to expand each pixel to give 3D world coordinates – but only on a per-pixel basis, and therefore only as individual points. There is no information about what exists between the points (hence the interest in high-density point clouds). Where the scene contains identified objects, sometimes models can be examined to infer some of the missing data e.g. the "back" side of cylinders or other precisely-described shapes.

Robot action requires not only accurate 3D coordinates for the place where the action is to occur, but also an assurance that the path from "here" to that point is unobstructed. Point clouds give some of that information, as each point represents the *nearest* obstruction along the direct line from the sensor optical centre to that point – the space in between is known to have been unobstructed when the image was taken. Robot action also takes place in real time, meaning that algorithms that

require a lot of processing time can cause the robot to move slowly or stop and wait until the algorithm has its finished output. Most industrial applications want the robot to work as fast as possible, so it is wise to avoid anything that makes the robot slower.

3D point cloud registration (PCR) algorithms work by finding the transformations required to change the template image into the real-world image. The purpose of matching the real-world image with the template image is that the key points annotated for action are moved and/or deformed along with the template, so their positions in the real world are known. Then the robot can go to the designated point and orientation and perform its action with a good chance of success.

Rigid transformations such as translation and rotation (also mirroring) do not change the distance between any points, so are fairly fast to calculate. The registration problem for rigid transformations is restricted to 6 degrees of freedom. Affine transformations such as shearing and scaling preserve lines and parallelism but not necessarily angles or distances. Other non-rigid transformations may be based on thin plate splines or the eigenmodes of variation of the point set. Non-rigid transformations require more computation time as they are less constrained.

The most popular PCR algorithms in robotics are simultaneous pose and correspondence methods, notably Iterative Closest Point (ICP) [1,2]. Local registration methods such as ICP assume that similar point clouds are already roughly aligned and output the best match – even if the images are, in fact, completely different (which is not the case for our data). Many variants of each method exist and new methods are frequently developed.

2. Related work

Many people have compared different variants of ICP using various datasets and various sensors to collect the data. Pomerleau et al [3] compared ICP variants on real-world datasets, stating: "Because ICP has many variants whose performances depend on the environment and the sensor, hundreds of [ICP] variations have been published. However, no comparison frameworks are available." Bellekens et al [4] used the mathematically deterministic methods principle component analysis (PCA) and singular value decomposition (SVD) to produce a benchmark and compare the precision and robustness of different ICP variants. They concluded that "The choice of an algorithm generally depends on several important characteristics such as accuracy, computational complexity, and convergence rate, each of which depends on the application of interest. Moreover, the characteristics of most registration algorithms heavily depend on the data used, and thus on the environment itself." Donoso et al [5] explored the performance of 20,736 ICP variants applied to the registration of point clouds for the purpose of terrain mapping, using data obtained from a mobile platform. They concluded that no single ICP algorithm is better than the others in every scenario.

Cheng et al [6] compiled an extensive review of different types of PCR algorithms for LiDAR datasets but had to conclude that no real evaluation can be done based on the existing literature.

Since there is no overall "best" algorithm for matching 3D point cloud data, investigations have to be made for each type of case. For our crossover-detection case, we chose to investigate 13 of the most popular algorithms, see the method section below.

3. Method

3.1 Data used

The dataset contained 1024x1024 images of pairs of 12mm diameter cylindrical rods lying approximately at right angles against a plain workbench. All images were taken with a Kinect, which means the point clouds are not very dense. An open3d algorithm was used to crop the data in a

sphere around the crossover point to reduce the number of 3D points further and speed up the matching process. This process also means that an initial "coarse" matching can be assumed and only local registration methods are needed.

For the purposes of the work reported in this paper, we took three images taken from different angles plus one "template" image where the position of the intersection was already precisely known. Each of the three test images were augmented as described below until we had 100 test images in total.

3.2 Image manipulation

The chosen images were manipulated at three different levels:

Variation	Low	Medium	High
Noise (m^{-3})	+/- 0.02	+/- 0.05	+/- 0.2
Translation (m, each axis)	0 – 0.003	0.003 – 0.01	0.01 – 0.05
Rotation (degrees, each axis)	0 – 2	2 – 5	5 – 10
Shear	0 – 0.02	0.02 – 0.1	0.1 – 0.2

3.3 Algorithms chosen

The algorithms compared in this paper are:

A. Iterative closest point (ICP, variants: point to point, point to plane, color and global)
B. Bayesian coherent point drift (BCPD), both rigid, affine and non-rigid
C. Support vector regression (SVR), both rigid and non-rigid
D. Gaussian mixture model (GMMReg) – a form of kernel correlation
E. Gaussian mixture model combined with decision tree (GMMTree)
F. FilterReg - point to point and feature

The ICP algorithms were taken from open3d version 0.9.0 [7]. The other algorithms were taken from the ProgReg library version 0.1.6 [8].

3.4 Choosing parameters / pre-experiments

ICP variants: A threshold value of 0.02 m was set for point-to-point and point-to-plane, otherwise the default initialization values were used. For ICP color, parameters chosen were: radius 0.05 m, relative fitness of 1e-6, relative root mean square error (RMSE) of 1e-6 and maximum allowed iterations set to 100. For ICP global we used feature matching with a distance threshold of 0.4 m and RANSAC convergence criteria. Fast point feature histograms (FPFH) features were calculated for each point with a radius of 0.05 m and a max nearest neighbors of 100.

The other algorithms were taken straight from the ProgReg 0.1.6 library that implements PCR algorithms with probabilistic (not deterministic) models which should be more robust than ICP. Default configurations were used. GMMTree had max iteration set to 20 and tolerance set to 0.0001. FilterReg - feat had sigma2 set to zero for the point-to-point and 1000 for "feature", with feature=FPFH.

3.5 Procedure

Two tests were carried out as follows:

1. A visual inspection test, where the registered image pairs were examined by eye to assess the match. Four result categories were used: unusable, bad, good, very good.
2. Investigation of mean point-to-point error distance and speed for each algorithm.

4. Results

4.1 Visual inspection of 100 matches

Algorithm	Type	Match quality			
		Very good	Good	Bad	Unusable
1. ICP - p2p	Rigid	45	35	12	8
2. ICP - p2pl	Rigid	40	29	11	20
3. ICP - color	Rigid	9	33	22	36
4. ICP - global	Rigid	40	37	16	7
5. CPD	Rigid	40	46	11	3
6. SVR	Rigid	22	42	31	5
7. GMM	Rigid	40	35	18	7
8. GMMTree	Rigid	15	24	21	41
9. FilterReg - p2p	Rigid	18	30	29	23
10. FilterReg - feat	Rigid	0	0	7	93
11. CPD	Non- Rigid	0	0	0	100
12. CPD	Affine	52	12	18	17
13. SVR	Non- Rigid	39	4	2	56

These qualitative results show that algorithms **10** (FilterReg - feat) and **11** (CPD non-rigid) cannot be used, as the matches that they produce are mostly unusable. **13, 8, 3** and **9** also give over 20% unusable matches.

The algorithms giving the highest number of very good matches are **12** (CPD-affine) and **1** (ICP-p2p). Algorithms **2** (ICP-p2pl), **4** (ICP-global), **5** (CPD-rigid) and **7** (GMM-rigid) all produce matches where 40% are assessed as "very good". Algorithms **1** and **5** are assessed as best as they both produce over 80 "good" plus "very good" matches and under 10 "unusable" matches. **4** and **7** are also highly rated as they both produce 75+ workable matches and under 10 unusable.

4.2 Low variation tests

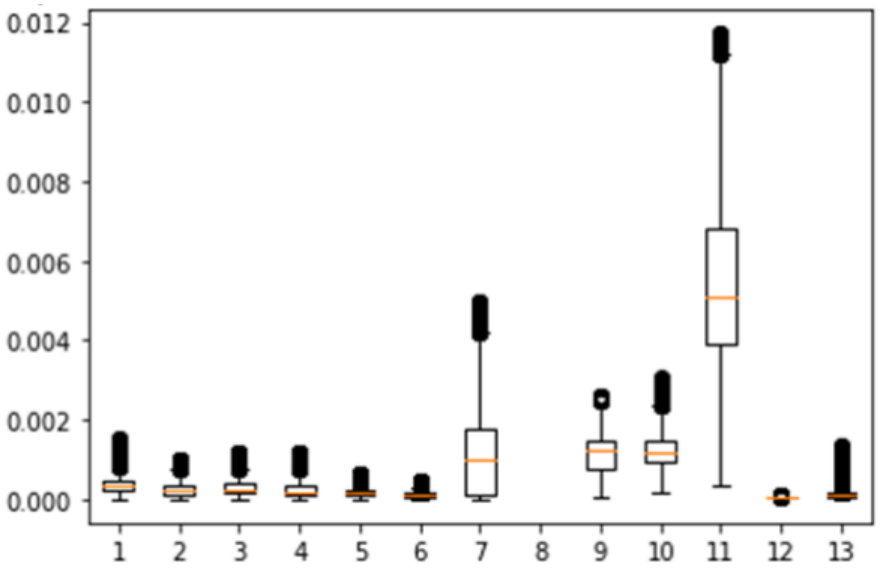

Figure 1: Boxplot of point-to-point error distance (mm) with low variation images, 8 GMMTree removed. Orange lines show the mean, open boxes the quartiles, black blobs are outliers.

Testing the algorithms on image pairs with low variation, the mean point-to-point error distance for **8** (GMMTree) was 51,000 mm (to 2 significant figures). Results from the remaining algorithms are illustrated in Fig.1. Algorithm **6** gave the best results with a mean error of only 0.11 mm. **13, 5** and **12** also gave mean errors of under 0.2 mm. As can be seen from Fig.1, **12** gave the tightest spread of results with a standard deviation of only 0.017 mm.
12 and **5** were the slowest algorithms, taking 33 and 25 seconds per match, respectively. **1, 2, 3, 4, 6, 8** and **9** took under 1 s per match on average. Algorithm **6** (SVR-rigid) therefore looks best here.

4.3 Medium variation tests

Using images with medium variation, algorithm **8** again gave an error measured in meters and not mm. Algorithm **3** is also missed out of Fig.2, with its mean error of 0.013 mm and a standard

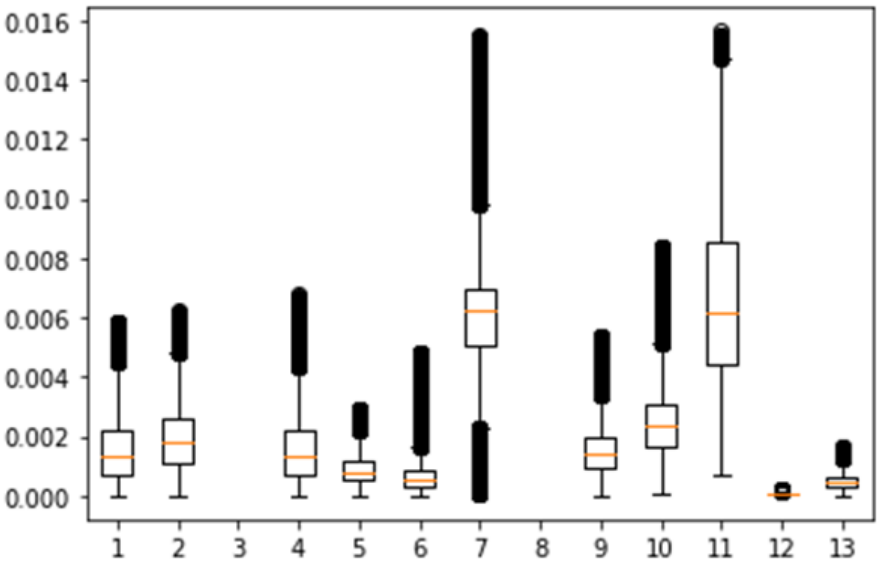

*Figure 2: Boxplot of point-to-point error with medium variation images, **8** GMMTree and **3** ICP color removed. Orange lines show the mean, open boxes the quartiles, black blobs are outliers.*

deviation of 0.044. Algorithm **12** was most accurate with a mean error of 0.083 mm and a standard deviation of 0.035. **13, 5** and **6** also gave errors with a mean of under 1 mm.

Again, **5** and **12** took the longest time, both around 35 s per match. **1, 2, 3, 4, 6** and **9** took under 1 s.

4.4 High variation tests

Algorithm **8** again failed completely. Algorithms **2, 3** and **13** also gave errors too high to be fitted into a plot with the other algorithms, which are illustrated in Fig.3.

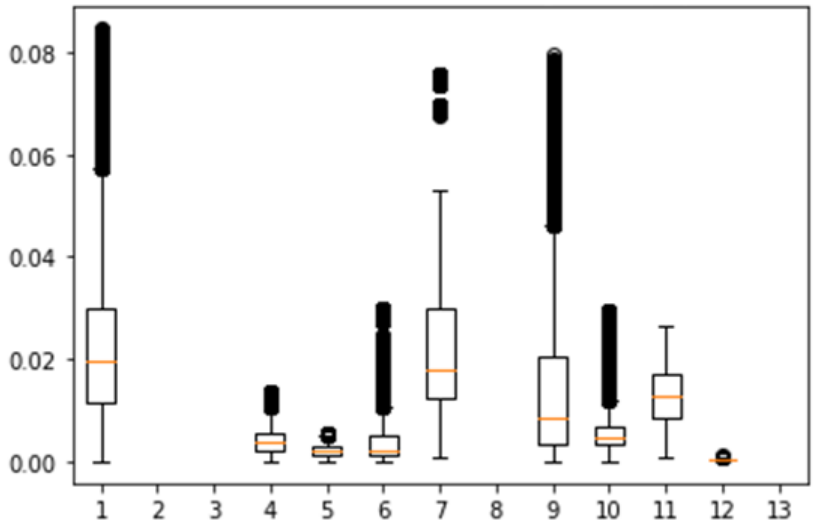

Figure 3: Boxplot of point-to-point error distances on images with high variation. Results from algorithms 2, 3, 8, and 13 have been removed due to high error values: 0.97 with S.D. 4.3, 0.4 with S.D. 2.17, 38000 with S.D 150,000 and 48 with S.D. 130 respectively. Orange lines show the mean, open boxes the quartiles, black blobs are outliers.

Algorithm **12** gave the most accurate results, with a mean error of 0.320 despite the high variation in the images. Algorithms **5, 4, 6** and **10** were also good. **13, 12** and **5** were the slowest on average, taking a mean of 58, 32 and 31 s respectively, though **10** was also slow and both **10** and **13** had outliers taking over 400 s! **2** was fastest, closely followed by **9, 1** and **3**. Algorithms **6** and **8** also came in under 1 s.

5. Conclusions

As could be expected, the ranking of the individual algorithms is similar with all three levels of variation -- the highest variation mostly just separates them out, making the ranking more obvious. Therefore the results from the low and medium variance tests were ignored, being considered as already included in the high variance test results.

The accumulated results from all the tests are gathered in the table below.

Algorithm score		Results from visual test	Results from high variance test		Overall
			Mean error mm	Speed s	
1.	ICP – p2p	√√	X	0.44	Bad
2.	ICP – p2pl	OK	X	0.044	Bad
3.	ICP – color	X	X	0.051	Very bad
4.	ICP – global	√	√	1.7	Good
5.	CPD – rigid	√√	√√	31.0	Second best

					accuracy but slow
6.	SVR – rigid	OK	√	0.35	Good
7.	GMM	√	OK	7.0	OK
8.	GMMTree	X	X	0.89	Bad
9.	FilterReg – p2p	X	OK	0.014	Bad
10.	FilterReg – feature	XX	√	58.0	Very bad
11.	CPD – non-rigid	XX	OK	2.6	Very bad
12.	CPD – affine	√√	√√	32.0	Best accuracy but slow
13.	SVR – non-rigid	X	X	58.0	Very bad

Algorithm **12** (CPD-affine) gave minimum error and was slowest with all amounts of variation, with algorithm **5** (CPD-rigid) close behind. Algorithm **6** (SVR-rigid) gave almost equally low mean point-to-point error distance as the CPD algorithms and was considerably faster.

The ICP variants (algorithms **1-4**) gave acceptable results even when using images with high variation, always giving errors under 1 mm.

6. Discussion

It can be seen that in general the best results come from the slowest algorithms, and there is no algorithm that outperforms the others in all areas. Therefore the "best" algorithm depends on the nature of the task and its priorities. Here, we assume that the match is being made so that a robot can perform some task at the rod intersection. The robot action will be considerably slower than all but the worst outliers, so the speed of the match is not important for our task. By contrast, a robot's success rate at any given task depends strongly on how well the robot can find and access the correct position. The more accurately we can locate the intersection, the greater the probability of the robot task succeeding. Therefore the best algorithms for our case appear to be **5** (CPD-rigid) and **12** (CPD-affine).

The difference between visual inspection results for **11** (CPD non-rigid) and the other CPD algorithms tested (**5** and **12**) shows how important it is to choose not only the right approach (CPD) but also the right version of the chosen approach.

Pomerleau et al [3] suggests that ICP point-to-plane is generally better than ICP point-to-point. Our results agree but the difference is rather small.

One interesting result is the mismatch between the results from the visual inspection and the mean error measurements. One problem with PCR algorithms is that they contain no semantics and give a result anyway, meaning that sometimes they make matches where the point pairs are reasonably close together, but visual inspection shows that the "matches" are of points that are not paired in the real world.

Generally, PCR algorithms work poorly if the template and scene differ a lot. A bad match could either be considered an error or an indication that the scene crossover differs too much from

expected. Since all the scenes and the templates were very similar, being images of two rods of the same size crossing at approximately right angles, the extent of bad matches is surprising.

Although the robot action is automated via the point cloud registration, for our case it does not matter is a few of the cross-overs are missed out. The current process is manual and it is possible for one of the current workers to deal with the trickier crossovers.

6.1 Future work

We intend investigating algorithms **5, 12, 6** and **4** further. Running these algorithms on the real robot will indicate if the slow speed of the two CPD algorithms is problematic in our case or not and if the slightly reduced accuracy of the faster **4 (**ICP-global) and **6** (SVR-rigid) matters for our case. If necessary, i.e. if the given algorithms cannot be made to work well enough, then we may continue this work by investigating other PCR algorithms and also non-PCR methods of finding the intersection point e.g. finding the pose of each of the two rods separately and using our knowledge that they are approximately straight to identify the intersection in 2D, and extending this to 3D by using the average distance to the center of each rod as calculated from the point cloud data. No template and no matching would be required.

Very recently, Yang et al. [10] have developed the first certifiably robust registration algorithm, named *Truncated least squares Estimation And SEmidefinite Relaxation* (TEASER). For point cloud registration, TEASER not only outputs an estimate of the transformation, but also quantifies the optimality of the given estimate. This could provide another interesting approach.

7. Acknowledgements

This project was funded as a Financial Support to Third Parties (FSTP) award under the European Union's Horizon 2020 project Robot Union, Grant Agreement no 779967.

8. References

[1]. Chen, Y. and Medioni G.: Object modelling by registration of multiple range images. *Image Vision Comput*. **10**(3): 145–155 (1991). doi:10.1016/0262-8856(92)90066-C.

[2]. Besl, P.J. and McKay, N.D.: A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **14** (2): 239–256 (1992). doi:10.1109/34.121791.

[3]. Pomerleau, F., Colas, F., Siegwart, R. and Magnenat, S.: Comparing ICP variants on real-world data sets. *Autonomous Robots* 34, no. 3: 133-148 (2013). https://link.springer.com/article/10.1007/s10514-013-9327-2

[4]. Bellekens, B., Spruyt, V., Berkvens, R:, Penne, R. and Weyn, M.: A benchmark survey of rigid 3D point cloud registration algorithms. *Int. J. Adv. Intell. Syst* 8: 118-127 (2015).

[5]. Donoso, F. A., Austin, K.J. and McAree, P.R.: How do ICP variants perform when used for scan matching terrain point clouds?. *Robotics and Autonomous Systems* 87: 147-161 (2017).

[6]. Cheng, L., Chen, S., Liu, X., Xu,H., Wu, Y., Li, M. and Chen, Y.: Registration of laser scanning point clouds: A review, *Sensors* 18 1641 (2018).

[7]. Zhou, Q.-Y., Park, J. and Koltun, V.: Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018). http://www.open3d.org/

[8]. Kenta-Tanaka et al. https://probreg.readthedocs.io/en/latest/ v. 0.1.6 date: 2019-9-29. Also available via GitHub https://github.com/neka-nat/probreg

[9]. Yang, H., Shi, J. and Carlone, L.: TEASER: Fast and Certifiable Point Cloud Registration (2020-01-21). *arXiv:2001.07715 [cs.RO].*