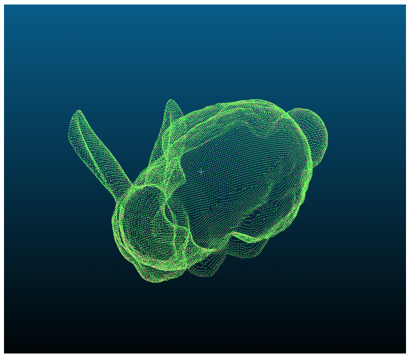
NPM3D - TP2

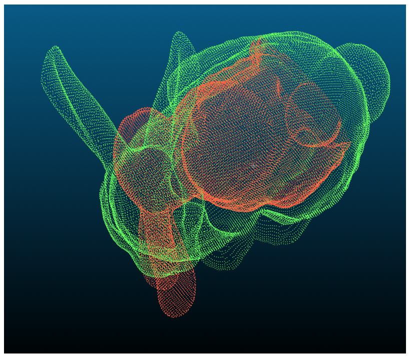
Aubin TCHOÏ

January 2023

Question 1

ICP performs well between bunny_original.ply and bunny_perturbed.ply, with a final RMS close to 0 (10^{-8}) and an almost-instant computation. However it does not succeed in matching bunny_original.ply and bunny_returned.ply as the method seems to get stuck in a local minima where the two point clouds are somewhat close but not matched at all (one bunny is upside down). In the exemple of Notre Dame, the two point clouds are correctly matched, however the RMS is equal to 1.32, which is far from zero and comes from the fact that there are points from the aligned cloud (Notre_Dame_Des_Champs_2.ply) that cannot be matched with any point in the reference cloud (Notre_Dame_Des_Champs_1.ply), mostly around the smaller roads perpendicular to the boulevard (rue Péguy and end of rue Montparnasse).


a)


b)

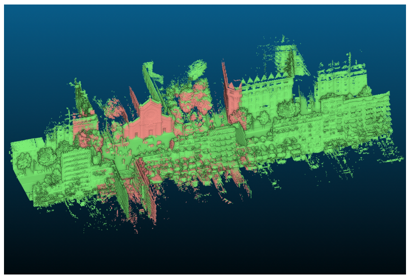

c)

Figure 1: Results obtained with CloudCompare ICP when aligning a) bunny_perturbed.ply b) bunny_returned.ply c) Notre_Dame_Des_Champs_2.ply

ICP is not symmetric in the roles of the aligned cloud and the reference cloud because it matches each point of the aligned cloud to its closet point in the reference cloud. If you take a minimal example with one point in one cloud and two points in the other cloud, if you take the single-point cloud as a reference ICP will place the two points such that the reference point is in their middle, and if you take it as the aligned cloud the point will be matched on its closest neighbor among the two points.

In the last example of Notre Dame des Champs the largest cloud Notre_Dame_Des_Champs_1.ply should be the reference and Notre_Dame_Des_Champs_2.ply the aligned cloud because the first one contains points that should not be matched to any point in the second one as it covers a larger portion of a scene.

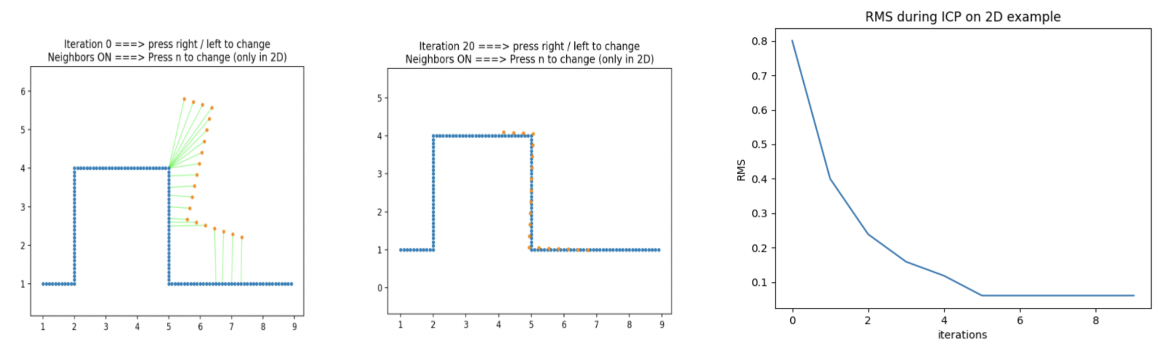
Question 2

We obtain a RMS of 0, which means that we found a rigid transformation that perfectly matches the two point clouds. The alignment worked here even though CloudCompare ICP failed because we know beforehand the matching between the two point clouds. The matching in question is given by the order in the arrays: the ith point of the is matched with the ith point of . CloucCompare ICP does not make use of this prior and has to find this matching, it does so by performing a closest neighbor search. If we were to shuffle one of the point cloud using `np.random.shuffle` we can observe that the alignment fails, as it tries to match points that have nothing to do with one another.

This function would not align the 3D scans of ”Notre Dame des Champs” as the two point clouds do not have the same number of points. To align them we need to come up with some sort of matching strategy, which is what is done in the ICP method.

Question 3

With the implementation for ICP we obtain the following results:



a) b) c)

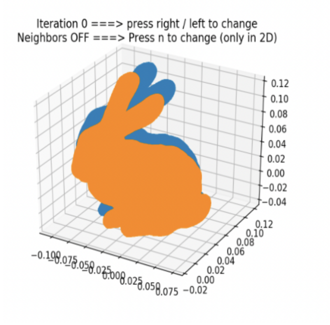
Figure 2: ICP on 2D example a) Original point clouds
b) Reference and aligned point cloud c) RMS convergence

Different parameters are used for the maximum iterations number (10 and 25 respectively), the same RMS threshold of 10^{-4} is used between the two examples. When applied to the matching of bunny_original.ply and bunny_perturbed.ply, we get the following figures:

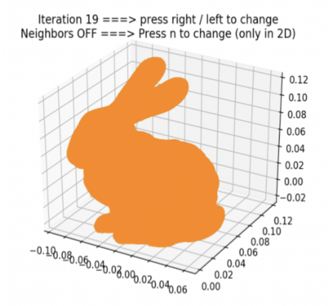
Question 4

We observe that in the first example the method does not converge to a almost-null RMS, it finds its final value at the fifth iteration and no matter the number of iterations it will stay at this value. This means that in figure b) for the closest neighbors matching the solution displayed is optimal. Visually we can tell that any solution that does not follow the same matching would be sub-optimal, therefore ICP here finds a global minimum. CloudCompare ICP finds the same result.

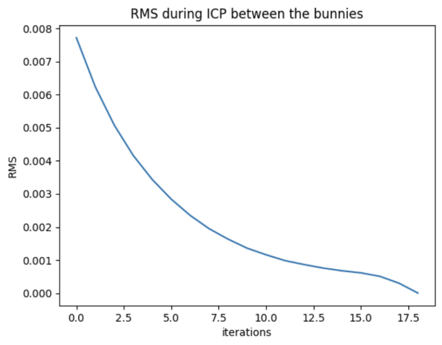
On the second example with bunnies, convergence is slower and requires 18 iterations, however it reached the RMS threshold, which means that the convergence leads to a null RMS with perfectly matched point


Iteration 0 ===> press right / left to change
Neighbors OFF ===> Press n to change (only in 2D)

a)


Iteration 19 ===> press right / left to change
Neighbors OFF ===> Press n to change (only in 2D)

b)



c)

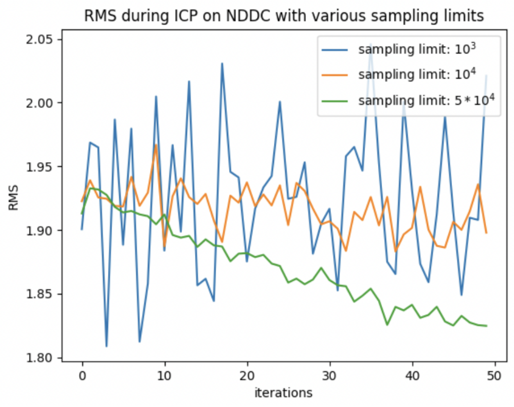
Figure 3: ICP on bunny example a) Original point clouds
b) Reference and aligned point cloud c) RMS convergence

clouds. The same result was achieved previously with CloudCompare ICP.

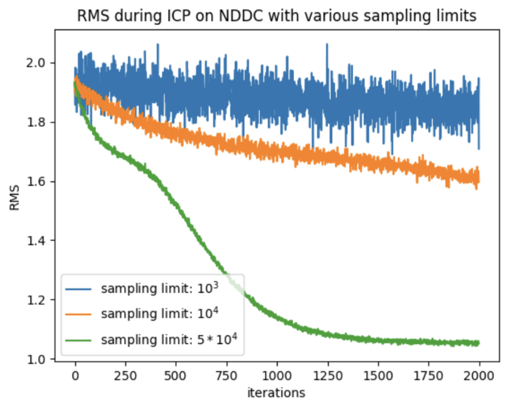
In both cases the ICP algorithm converges monotonically to a local minimum with respect to the RMS cost function, which comes from the fact that in each iteration, the algorithm performs two actions that each attempt to reduce the value of its cost function, but in any case cannot increase it. This observation follows the result in the case from Besl and McKay where the number of points is constant seen during the course.

Bonus question

The figures below show the RMS convergence during ICP for various sampling limits (10^3, 10^4, $5*10^4$). The figure on the left shows the convergence on 50 iterations only, whereas the one on the right displays the evolution for 2000 iterations.



a)



b)

Figure 4: RMS convergence during ICP on NDDC example a) 50 iterations b) 2000 iterations

The method never meets the stopping criterion on the RMS error, which means that we never end up matching the two point clouds perfectly. This is expected as the two point cloud do not cover the same portion of the space.

When we start the experiment with a low maximum number of iterations for sampling rates equal to either 10^3 or 10^4 (in blue and in orange in the figures), we observe that the observations are very noisy, and the sampling rate seems to impact the standard deviation of the noise but not necessarily its mean. Furthermore, the method does not seem to converge. However when we increase the sampling rate to $5 * 10^4$ the RMS notably decreases during ICP and the observations seem even less noisy. By increasing the number of iterations even more we get a proper convergence as displayed in **??** b).

The noise observed comes from the fact that the aligned point cloud is not exactly a subset of the reference point cloud, as mentioned in the first question there are parts of the point cloud that cannot be matched with any portion of the reference. Therefore in an iteration if we choose in our batch points that fall into one of these part we might drift off from the target solution because the rigid transformation will attempt to match these points with neighbors that are very far from them in the target solution.

One thing we could do would consist in dropping these points and never choosing them in our batches. To discriminate them we could only choose points that are close to the barycenter of the cloud.