COMPM080: Coursework #1

Due on Friday, February 12, 2016

Maria Ruxandra Robu- 14042500

This coursework covers the Iterative Closest Point algorithm, which was first introduced in [1]. Task 2 describes the structure of the point-to-point approach and discusses the strategies chosen for the implementation. Task 3 analyses the convergence of the algorithm under varying degrees of rotation for the initial alignment. Task 4 analyses the performance under Gaussian noise. Task 5 illustrates how subsampling impacts the results of ICP. Task 6 explores several strategies for multi-body registration. Finally, Task 7 computed the normals for each vertex and uses them in the ICP formulation.

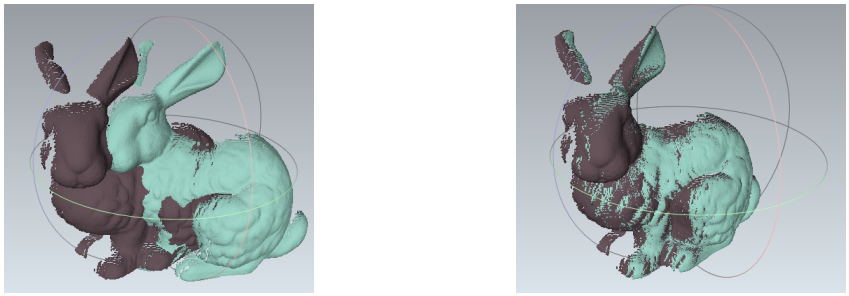
Task 2

Iterative Closest Point (ICP) algorithm

In this section, the point-to-point ICP algorithm was implemented. Given two meshes with a good initial alignment, the algorithm iteratively finds an optimal rigid transformation that best aligns them. ICP is proven to converge to a good solution. However, without a good initialization, it will fall into local minima.

Given two point clouds p and q:

- for each $q \in \{q\}_M$ find the closest match in $\{p\}_N$
- reject bad pairs
- center the point clouds at their origin by subtracting their means
- compute the covariance matrix C of the centered $\tilde{p}$ and $\tilde{q}$
- extract R from the singular value decomposition of C ($R = VU'$)
- the translation vector is $t = \bar{p} - R\bar{q}$
- apply the current transformation $q = Rq + t$ and start from the beginning



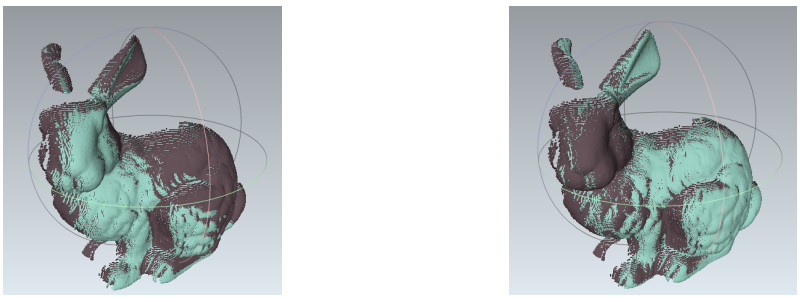
(a) (b)

Figure 1: Task 2 - a) two meshes with no initial alignment; b) registration of two point clouds (27 iterations; error = 0.118)

Stopping conditions must be set in order to stop the iterations. In this implementation, a maximum number of iterations is set. Furthermore, if two consecutive errors do not change by more than a threshold, the algorithm is considered to have converged. This additional condition shortens the computation time by stopping unnecessary iterations.

The most important step of the algorithm is establishing the correspondences between the two point clouds. In the point-to-point approach, the algorithm finds the closest point to the query point by computing the

Task 2 continued on next page...



(a) (b)

Figure 2: Task 2 - a) two meshes with a good initial alignment; b) registration of two point clouds (11 iterations; error = 0.116)

minimum euclidean distance. In this implementation, the bad pairs are rejected through a threshold. For example, if the distance between a current pair of points is more than 50% of the maximum distance, the pair is discarded.

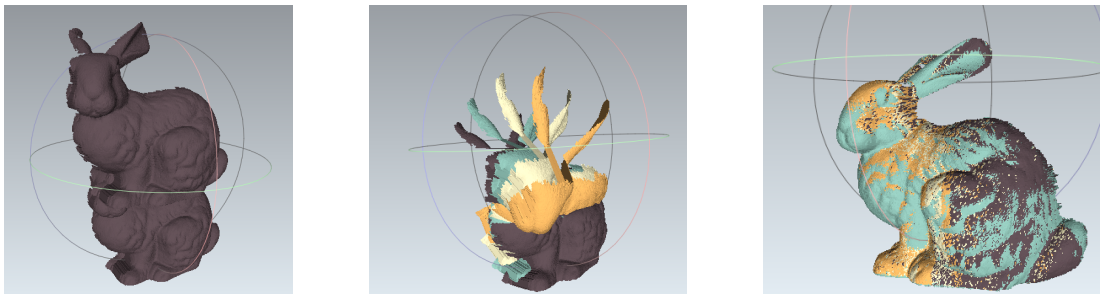
The algorithm was implemented in C++ with the use of the libraries: OpenMesh, Eigen and ANN. The .ply objects were loaded into memory with the OpenMesh mesh reader. Eigen was used for the matrix decompositions and operations. ANN was used for finding the correspondences and computing the normals in Task 7. All of the meshes were coloured differently to easily observe the overlapping regions. The meshes are visualized using MeshLab. The matrices used for initial alignment have been obtained from rough manual alignment in MeshLab.

The figures 1 and 2 illustrate the point-to-point registration for two points clouds. They show two cases which work even without a very good initial alignment, due tothe big overlapping areas. however, in most situations, the algorithm would get stuck into a local minima.

Task 3

ICP convergence analysis for different rotations

The mesh M1 was centered at its origin by subtracting the mean $\bar{p}$ from all its points p, as shown in Figure 3a. Once its position is set, incremental rotations are applied to the mesh in the x, y and z axes. Figure 3b shows an example for 3 rotations along the x axis. Once we have the initial M1 and the rotated M3, ICP is ran for each rotation. Figure 3c illustrates the results for the 3 rotations in the example.

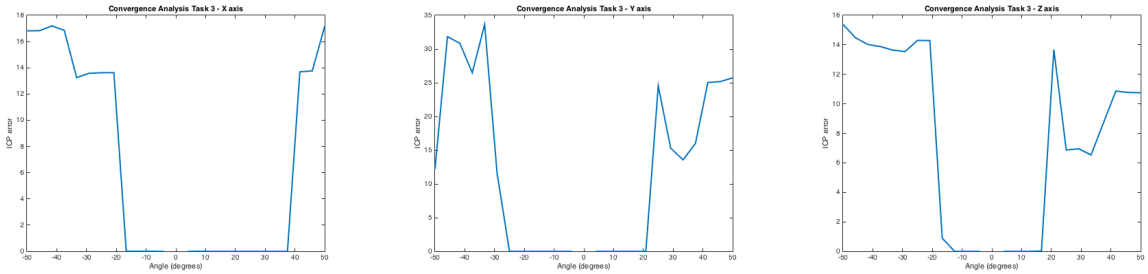


(a) (b) (c)

Figure 3: Task 3 - a) M1 centered at its origin; b) 3 Rotations along the x axis; c) the final registration after ICP is ran for case b)

The parameters used for this section are:

- axes: x, y, z
- number of different rotations = 25
- interval of degrees [-50, 50]
- maximum number of iterations for ICP = 50
- error threshold for ICP = 1e-04
- threshold for the bad points rejection = 70%



(a) (b) (c)

Figure 4: Task 3 - Convergence Analysis for the 3 axes of rotation

For each iteration, the final error and number of iterations were saved in a file and plotted with Matlab. The figures 4 show the basins of convergence for the ICP algorithm by varying the rotation angle on the 3 axes.

Task 3 continued on next page...

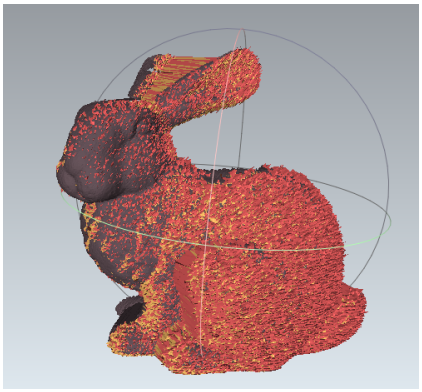
The minimum error obtained (1.48e-11) was for the z axis at -4 degrees and the maximum (33.64) on the y axis for -33 degrees.

Figures 4 clearly show how ICP can get stuck into local minima. For example, if an initial alignment is made with a rotation on the z axis, at around 30 degrees, the algorithm would find close solutions to the starting point. however, it will not jump and find the global solution like in the basin of convergence.

Task 4

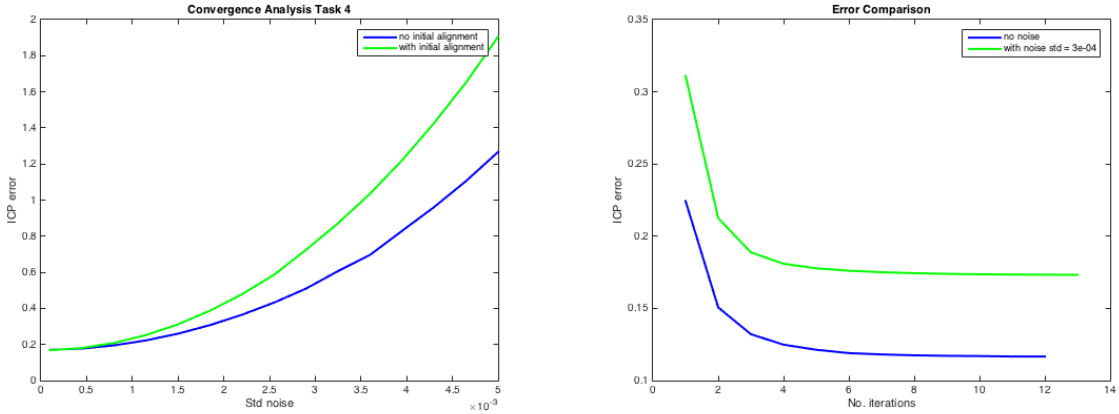
ICP convergence analysis with noisy data

In this section, increasing amounts of noise are added to mesh M2 and ICP is computed for each case. In order to generate the graphs in 6, noise was added in 15 steps between 1e-04 and 5e-3 standard deviations.



(a)

Figure 5: Task 4 - Example of bunny with different levels of noise. Orange corresponds to 55e-04 standard deviation and red to 1e-03 standard deviation



(a) (b)

Figure 6: Task 4 - Noise analysis: a) error comparison for increasing amounts of noise; b) shows the error against the number of iterations needed for convergence

Figure 6 shows increasing errors for noisy data. however, the algorithm still manages to find a good registration between the two meshes. In this case, the final values of errors cannot really be trusted given the amount of noise that was added as a perturbation on the vertices.

Task 5

ICP convergence analysis with subsampling

The mesh M2 was subsampled by uniformly sampling the point cloud. In the first implementation, every n-th point would be selected for the subsampled mesh. However, the resulting point cloud was truncated or with missing regions. In this implementation, the whole bunny is well represented for percentages of subsampling between 1 and 100 (for example, figure 7). The M1 mesh remains fixed the whole time and all its points are used in the computation. This allows for good and fast registration even for high subsampling rates. Figure 8 shows the analysis of the algorithm for this approach. ICP manages to correctly align the two point clouds faster, as long as one mesh is kept intact and only one is subsampled.

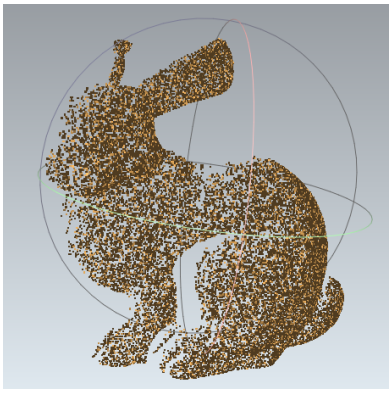
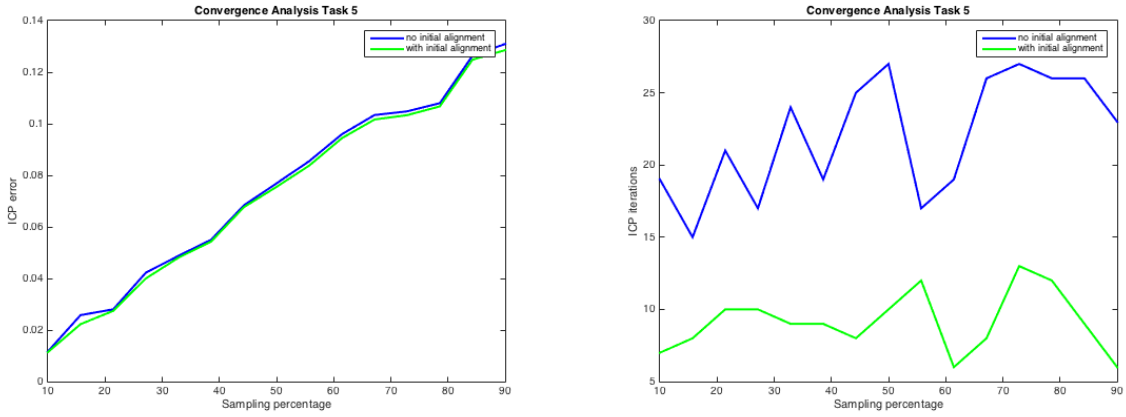


Figure 7: Task 5 - Example of subsampling - 35%

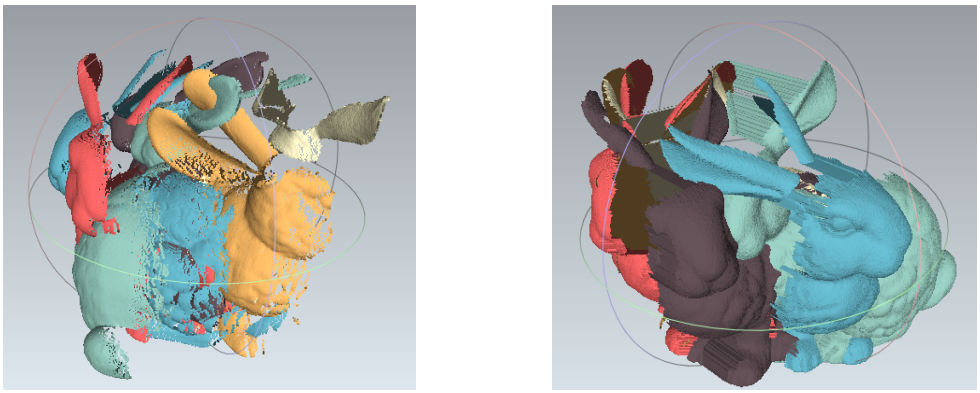


(a) (b)

Figure 8: Task 5 - Error analysis for alignment with increasing subsampling rates

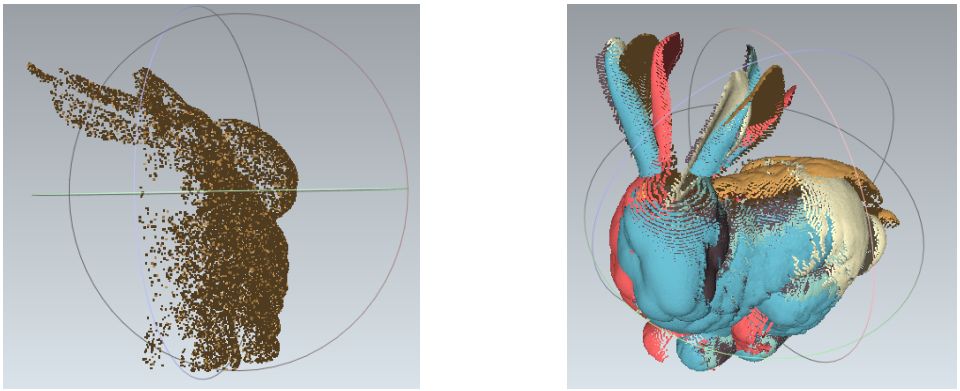
Task 6

Multi-body Registration



(a) (b)

Figure 9: Task 6 - a) no initial alignment; b) the meshes with initial alignment



(a) (b)

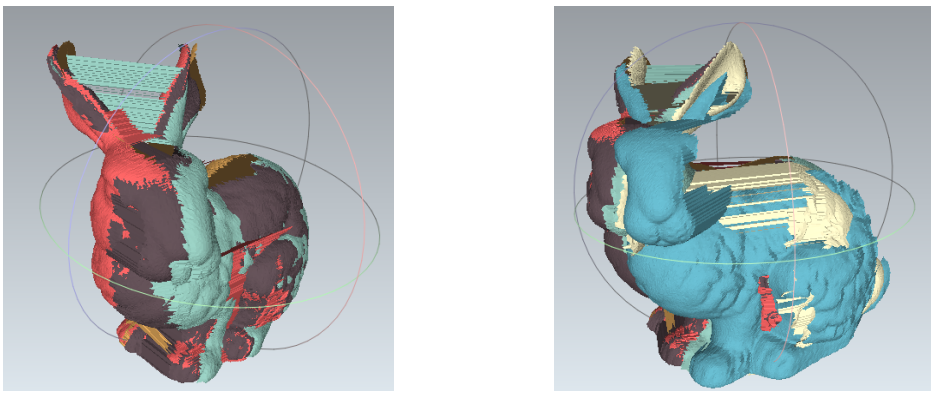
Figure 10: Task 6 - a) multi-body ICP with subsampling (Strategy 1); b) multi-body ICP in pairs (Strategy 2)

Several strategies were tried for this task. Initially, I planned on using the functions developed in the previous tasks to make the algorithm faster.

Strategy 1:

- run ICP on the first 2 meshes
- merge the resulting alignment
- subsample mesh M3
- run ICP between the merged mesh and the subsampled version
- go to step 3 and repeat for all the other meshes

Task 6 continued on next page...



(a) (b)

Figure 11: Task 6 - Multi-body alignment with normals (Strategy 3). a) good alignment between most of the meshes; b) issues with meshes that did not have a good initial alignment or that did not have enough overlapping regions

Several issues were observed. Firstly, the merged mesh gathers a high number of unnecessary points in overlapping regions. So, the computation time increases very fast, despite the subsampling used in the new meshes. I tried to uniformly subsampling the merged mesh as well. however, the results were not promising, as discussed in Task 5. In this situation, an adaptive algorithm to merge the mesh into a global model should provide better results. Another downside of this approach is that any errors obtained in the first few steps get carried throughout all the alignment steps.

Strategy 2:

- keep mesh M1 fixed
- align mesh M2 to M1
- align mesh M3 to the new position of M2
- continue until all the meshes are aligned

This pair-alignment strategy would get rid of the drifting problem caused by the propagation of errors. Better results were obtained than in the first strategy. however, there were still high discontinuities in the final mesh of the bunny.

Strategy 3:

- keep mesh M1 fixed
- compute normals of all the meshes
- include normals in the ICP
- align meshes pair by pair

The third approach to multi-body alignment uses normals to restrain the ICP formulation. The method to compute the normals is described in Task 7. This step is included in te algorithm as a way to reject bad correspondences. If the dot product of the normals is shown to be lower than a threshold, the pair of points is rejected. The aim would be for their dot product to be as close to 1 as possible (parallel normals). The choice of angle for the threshold is detailed in Task 7. In this implementation, normals that were further

Task 6 continued on next page...

apart then 5 degrees are discarded.

Strategy 3 proved to show the best results with faster computation times, as shown in figure 11 (the strict threshold on the normals rejects a high number of points). However, not all of the meshes could be aligned perfectly, due to bad initial alignment or insufficient overlapping areas. The errors for the multi-body alignment of the bunny dataset are:

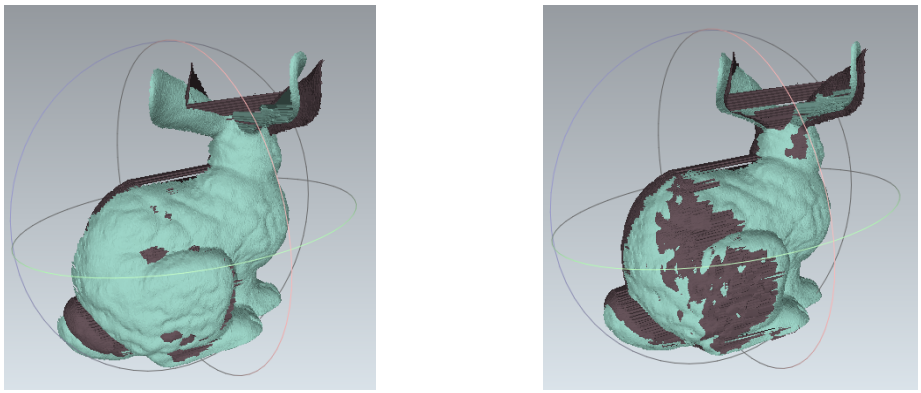
- mesh45 - 14 iterations, error = 0.0042
- mesh90 - 13 iterations, error = 0.243
- mesh180 - 7 iterations, error = 0.018
- mesh 270 - 9 iterations, error = 0.035
- mesh315 - 19 iterations, error = 0.124

Task 7

ICP with normals

Normals were estimated at every point in the mesh by following the next steps:

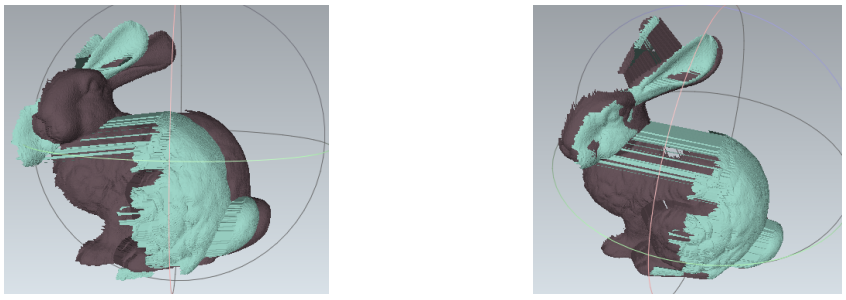
- for each point p on a mesh
- find its k nearest neighbours
- compute the covariance matrix and find its eigenvectors
- the normal at point p is the eigenvector corresponding to the smallest eigenvalue



(a) (b)

Figure 12: Task 7 - Threshold for the angle between the normals - comparison: a) angle = 30 degrees, error = 1.24, iterations = 13; b) angle = 5 degrees, error = 0.018, iterations = 7

In this implementation, k was chosen to be 10 in all the iterations. The closest neighbours are found by using kdtrees with the ANN library. A search radius of 5e-05 is used. This value was set empirically because it returns around 250-350 neighbours for a mesh of $\tilde{4}0000$ vertices.

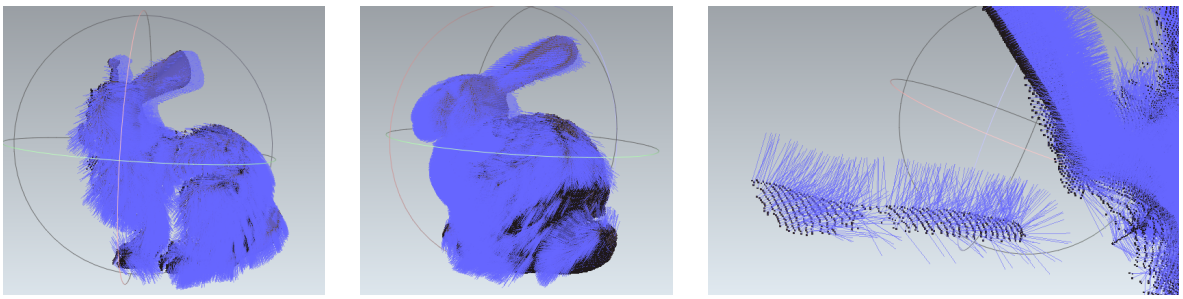


(a) (b)

Figure 13: Task 7 - Alignment bunny45-bunny90: a) bad alignment (error = 0.139) - parameters used: angle for the normal criteria: 5 degrees, distance rejection threshold = 50%; b) good alignment (error = 0.02) - parameters used: angle for the normal criteria: 5 degrees, distance rejection threshold = 30%

Figure 13 illustrates a difficult case of registration. The overlap between bunny45 and bunny90 is not sufficient and leads to erroneous solutions for the final alignment. However, with strict values for the rejection

of bad correspondences, a good alignment can be obtained.



(a) (b) (c)

Figure 14: Task 7 - a) bunny with normals; b) the orientation of several normals is flipped; c) problems with normals at the boundaries

So, the smallest eigenvector returns the direction of the normal, but not the orientation. However, for this task, orientation is not necessary. Several other problems appear at the boundaries of the mesh where the plane fit to the k neighbours does not represent the surface well, as shown in figure 14.

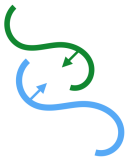
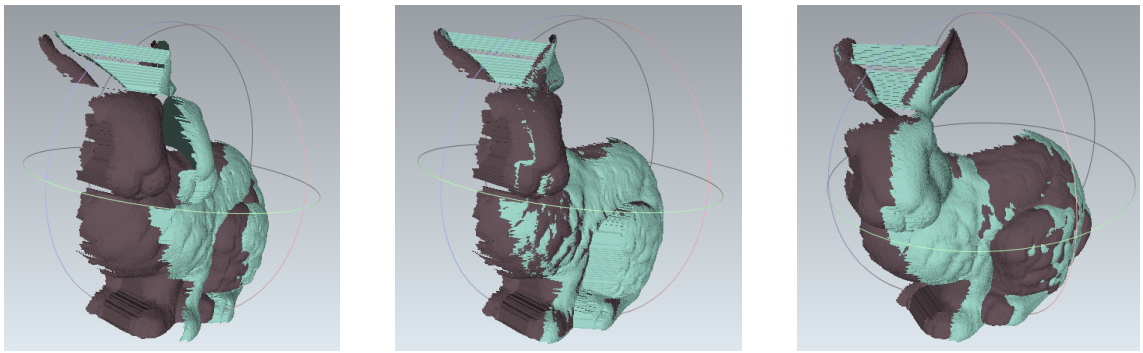


Figure 15: Task 7 - Limitation of the current algorithm. If we don't take into consideration the orientation of the normals, this situation will result in a good correspondence between the selected points

The normals were incorporated in ICP as a criterion for the rejection of bad correspondences. For each pair, the dot product of the two normals is computed to obtain the angle between them. Tests were conducted to find an optimal angle for which the convergence is faster and closer to the optimal transformation. This approach improves the accuracy of the final results, given that the remaining correspondences (after the rejection of bad points) are more likely to be correct. Consequently, given that less points will be taken into consideration, the algorithm performs slightly faster. Figure 15 illustrates a situation in which this implementation fails. A more robust formulation could be made by including the orientations of the normals. For the moment, a strong assumption is a good initial alignment between the meshes.

Figure 17 shows that ICP with normals converges faster and with lower errors for the alignment between bunny00 and bunny45.



(a) (b) (c)

Figure 16: Task 7 - Alignment bunny00-bunny45: a) initial alignment; b) alignment without normals (error = 0.0042, iterations = 14); c) alignment with normals (error = 0.116, iterations = 32)

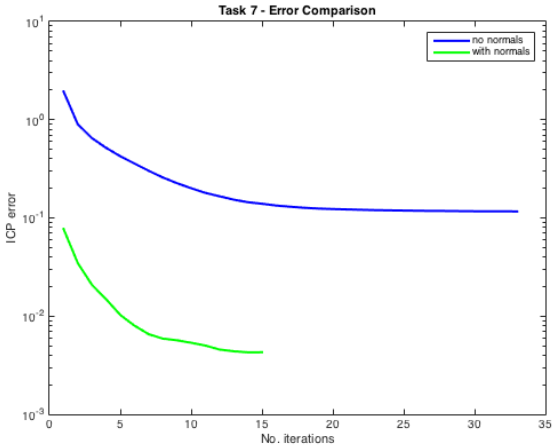


Figure 17: Task 7 - Error comparison (ICP - bunny00 and bunny45) between ICP with normals and without

References

[1] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239–256, Feb. 1992.