Stefano Binotto 2052421

ITERATIVE CLOSEST POINT CLOUD REGISTRATION

In order to successfully perform ICP I completed the following methods:

- **find_closest_point(…)**: for each point in the transformed source point cloud I found the nearest neighbor in the target point cloud. In order to compute this as efficiently as possible I exploited the KD-Tree approach explained during the lectures. In addition, I discarded the correspondences if their distance was bigger than a threshold. Finally, returned the matched pairs and the final RMSE.

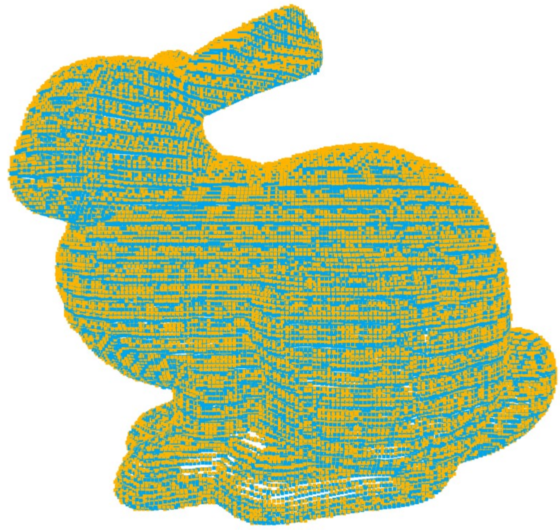
- **get_svd_icp_registration(…)**: initially I computed the centroids of the transformed source point cloud d_c and the target point cloud m_c. Then I implemented the correspondence-based registration by decoupling R and t.
  To find R, I computed the W matrix in this way:
  $$W = \sum_{i=1}^{n^\circ\, of\, match} m_i' d_i'^T$$
  with $d_i' = d_i - d_c$ and $m_i' = m_i - m_c$.
  Then, I decomposed it using SVD and found the rotation matrix R considering only the obtained U and V matrices:
  $$R = UV^T$$
  In case I was dealing with a reflection, so $\det(UV^T) = -1$, I computed the rotation matrix by negating the last column of the V matrix:
  $$R = Udiag(1,1,-1)V^T$$
  Finally, I found the translation t in closed form:
  $$t = m_c - Rd_c$$

- **get_lm_icp_registration(…)**: as done in **get_svd_icp_registration(),** this method wants to return the transformation for the allignment between the transformed source point cloud and the target point cloud. But this time it uses the Levenberg-Marquardt algorithm, which is a general-purpose nonlinear optimization of a registration error in order to find the optimal rotation and translation parameters (6 in total). Using Ceres I defined a Cost Function in which I manually implemented the registration error by computing the difference between the transformed source point and the target matched point.

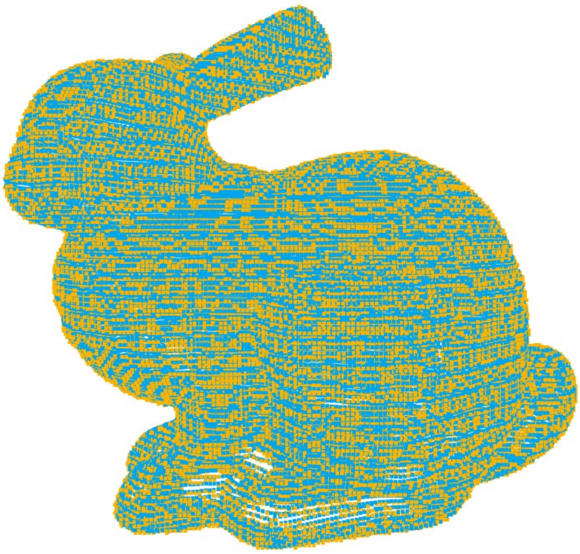
- **execute_icp_registration(…)**: it groups together all the methods above implementing the ICP loops until convergency or until a predefined number of iterations is reached. At first, in each iteration, I find the correspondences between the source and target clouds using **find_closest_point()**. If the RMSE returned is bigger than the RMSE of the previous iteration or the improvement is not big enough with respect to a threshold, interrupt the ICP loop and consider the current transformation as the best one. Otherwise, depending on the chosen method (svd or lm), call **get_svd_icp_registration()** or **get_lm_icp_registration()** and update the transformation matrix (transformation_ variable).

RESULTS:

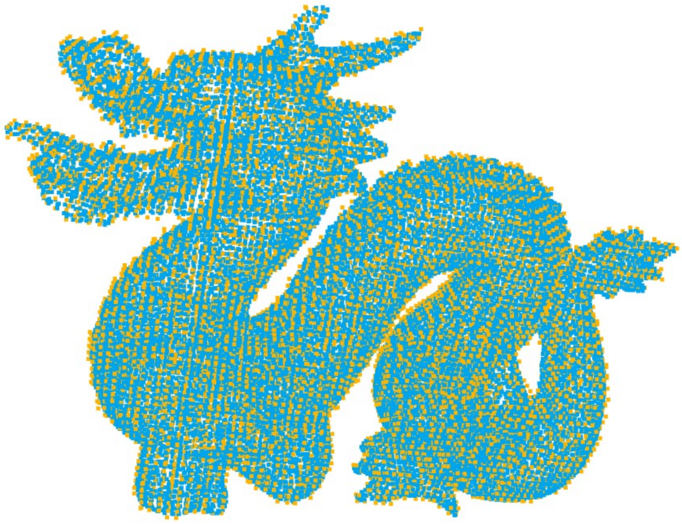
1) Bunny (SVD) → RMSE = 0.00401692



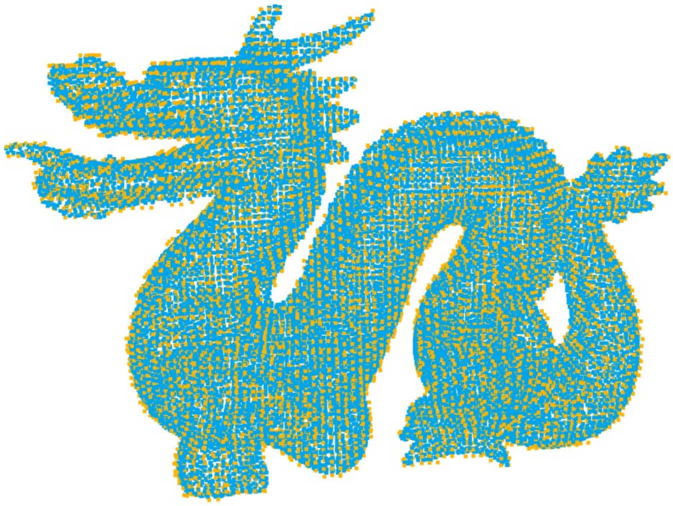
2) Bunny (lm) → RMSE = 0.00341361



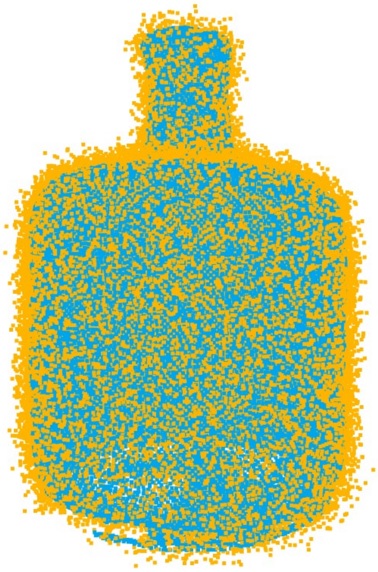
3) Dragon (SVD) → RMSE = 0.00568868



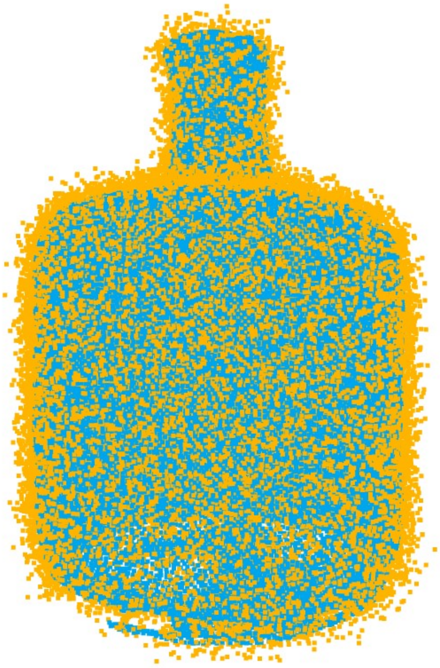
4) Dragon (lm) → RMSE = 0.0056415



5) Vase (SVD) → RMSE = 0.0162231



6) Vase (lm) → RMSE = 0.016221



Note: As expected, the Levenberg-Marquardt approach provides always better results than the standard SVD, since the ICP with LM has a wider convergence basin and more accurate solutions (usually). In addition, it should be noticed that the vase registration is a bit noisier than the other ones, maybe because the vase initial allignment was much worse.