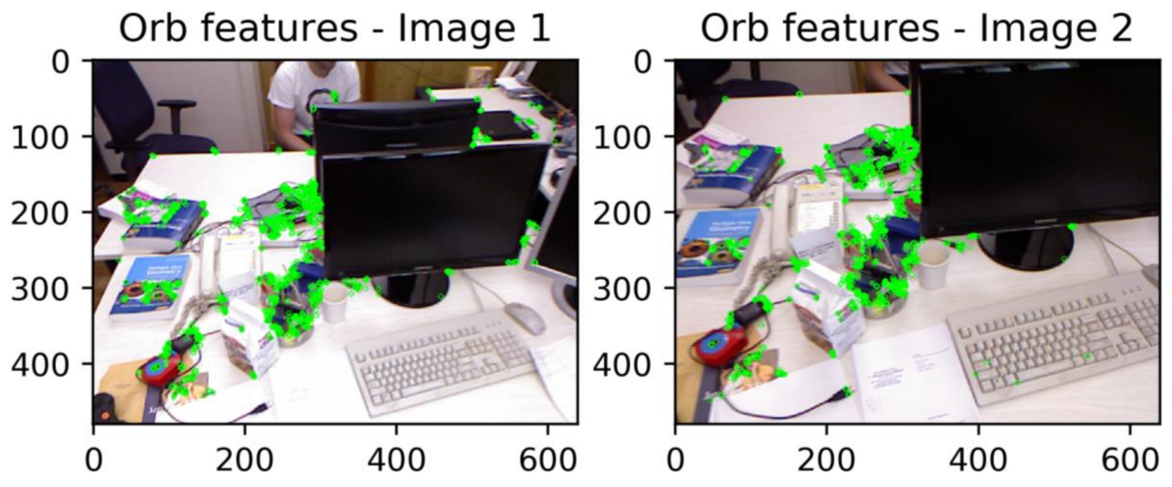
Visual SLAM - Assignment 1

Rui Oliveira

In this work we will study the problem of indirect pose estimation, where the objective is to understand how an RGB-D camera move in between two frames, i.e., the transform between the two cameras. Once this transformation matrix is obtained, we use both images to create a unified point cloud.

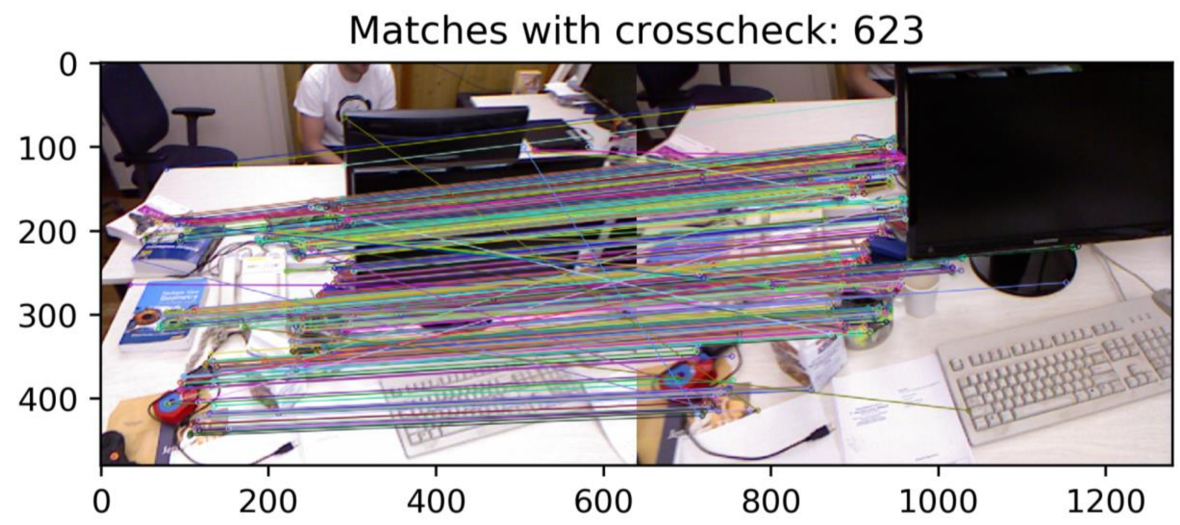
ORB descriptors and matching between them

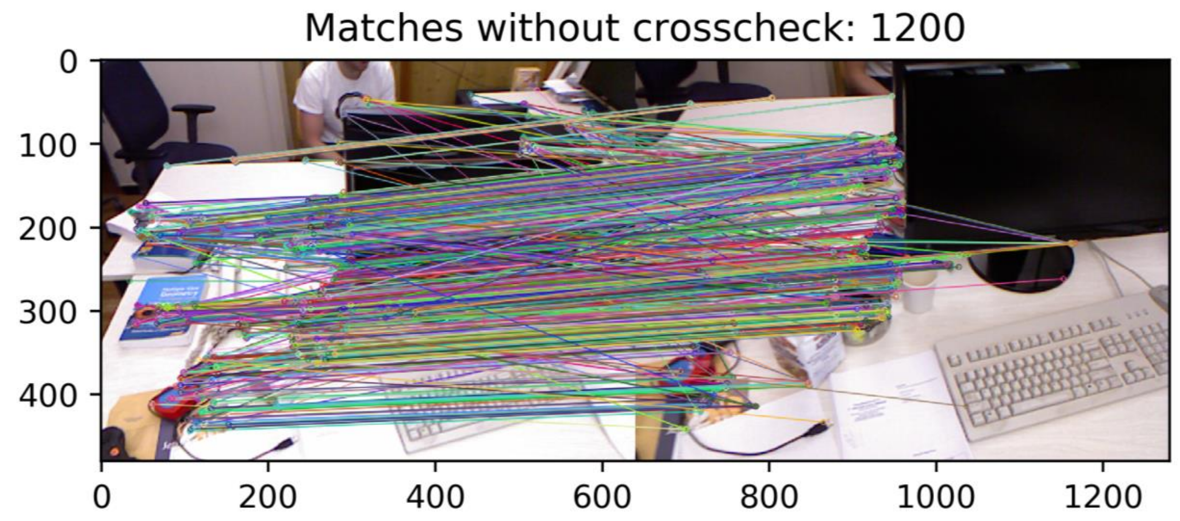
To obtain the pixel level correspondence between the two images we make use of ORB descriptors. The obtained descriptors in each image frame are shown below:



The descriptors in itself are of little value, unless they can be matched within the two images. We therefore perform a matching to find out how do the descriptors relate to each other in both images. The match can be performed with cross-check or without. Using cross-check will usually yield better matches, however they will be fewer. In these two frames we have a good amount of cross-check matches, and therefore we will only use those to ensure that we are using the matches of high quality.

The figures below show the matches using cross-check and not using cross-check.

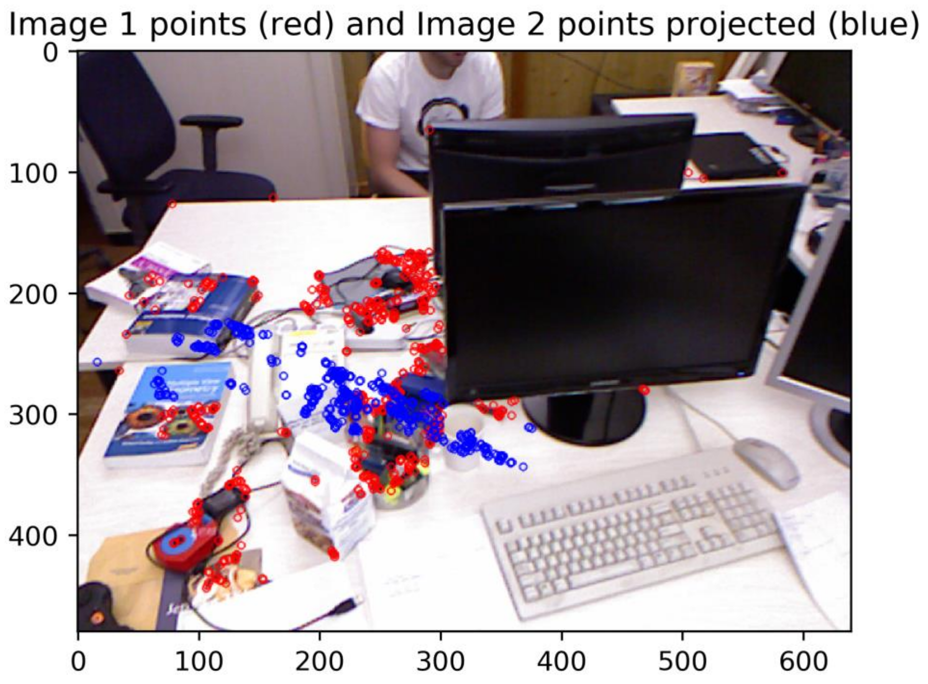




Finding the transform between cameras and RANSAC algorithm

Now the objective is to understand how points in the 3D space of image 2 project into image 1. To do so we can use function solvePnP(). We start by feeding solvePnP() with the 3D space points of image 2 and the corresponding 2D image points of image 1.

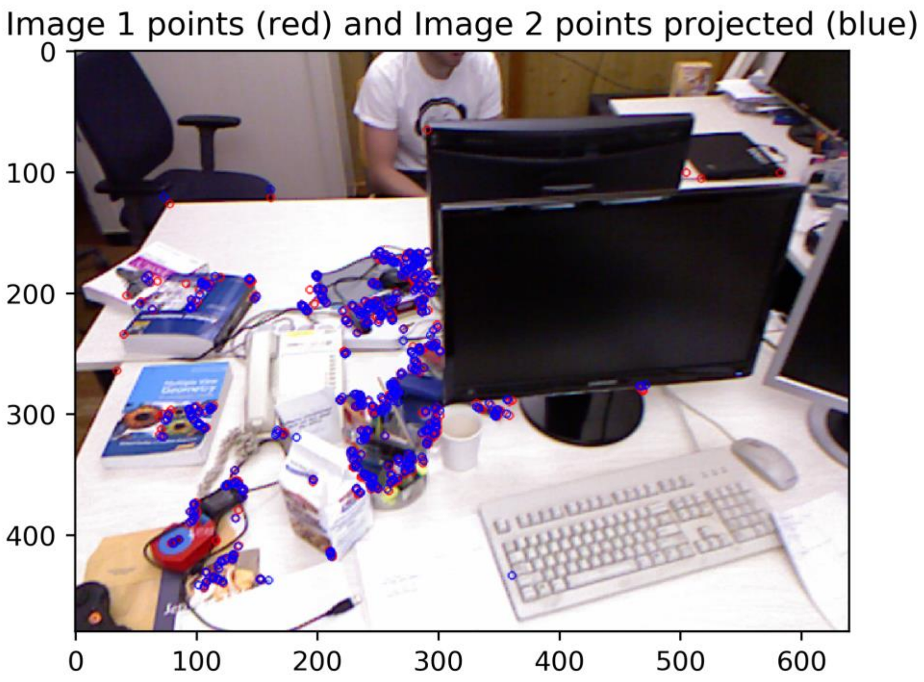
The correspondence between 3D space points of image 2 and 2D image points of image 1 is done by the descriptor matches previously found. The result is seen below:



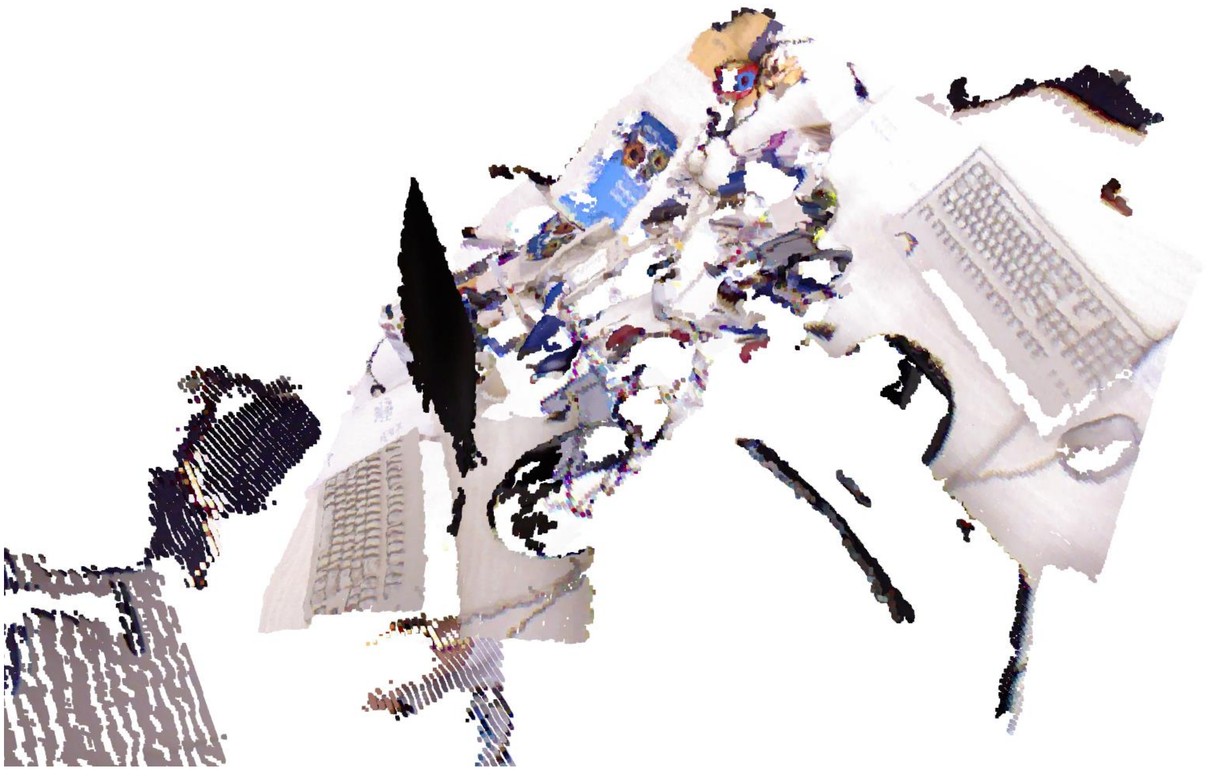
The figure shows that something is wrong. The matching is not properly done. This happens because some of the matches between the ORB descriptors are outlies, and they were used to compute the transformation between cameras. To deal with the problem of outliers we use the RANSAC algorithm. At each iteration RANSAC will select a handful of randomly chosen matches and compute a transform out of it. Using that transform it will compute the projection of all points and measure a residual to understand how good the fit is.

The residual we chose in this work is simply sum of Euclidean image coordinate distances between the projected points and their expected position (based on the descriptor matches).

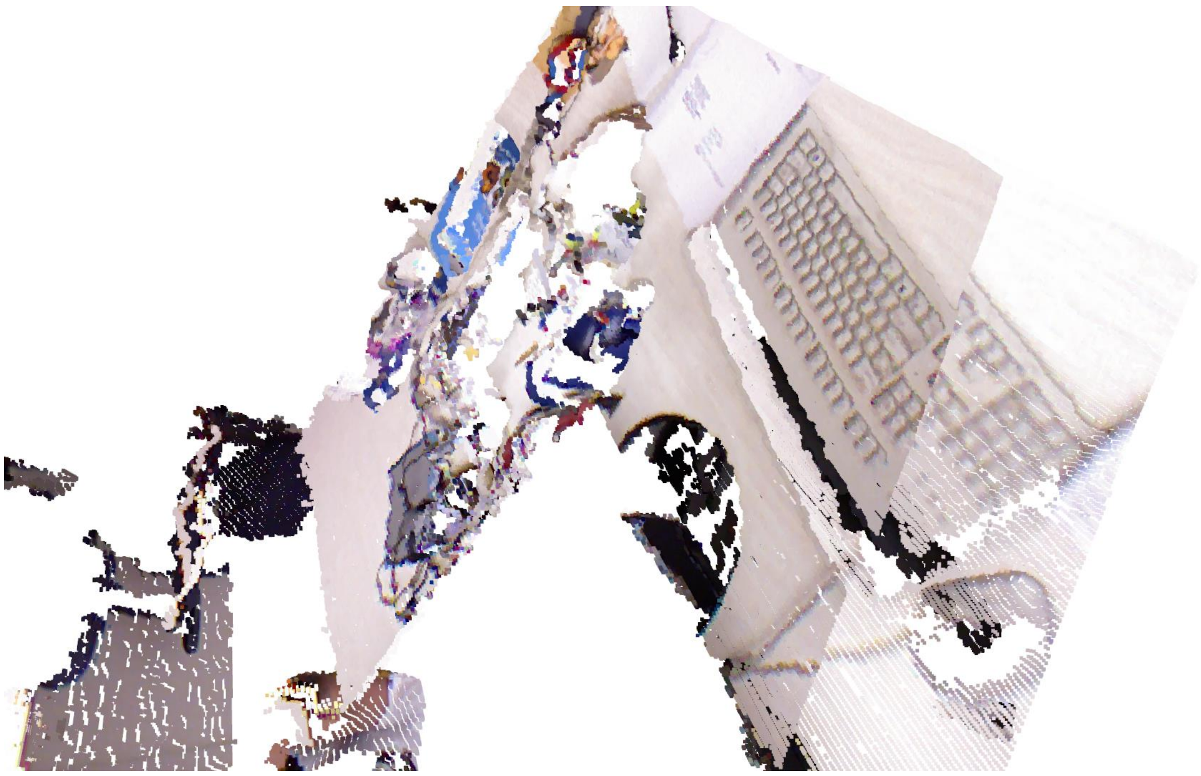
Using the RANSAC algorithm the transformation now becomes:



Now we see that the projected points seem to align quite well with the matched points on the original image. This can also be seen in the registration result of the point clouds. We first show the bad transform, where we can notice that the same keyboard appears in two distinct place, indicating that the transform is wrong.



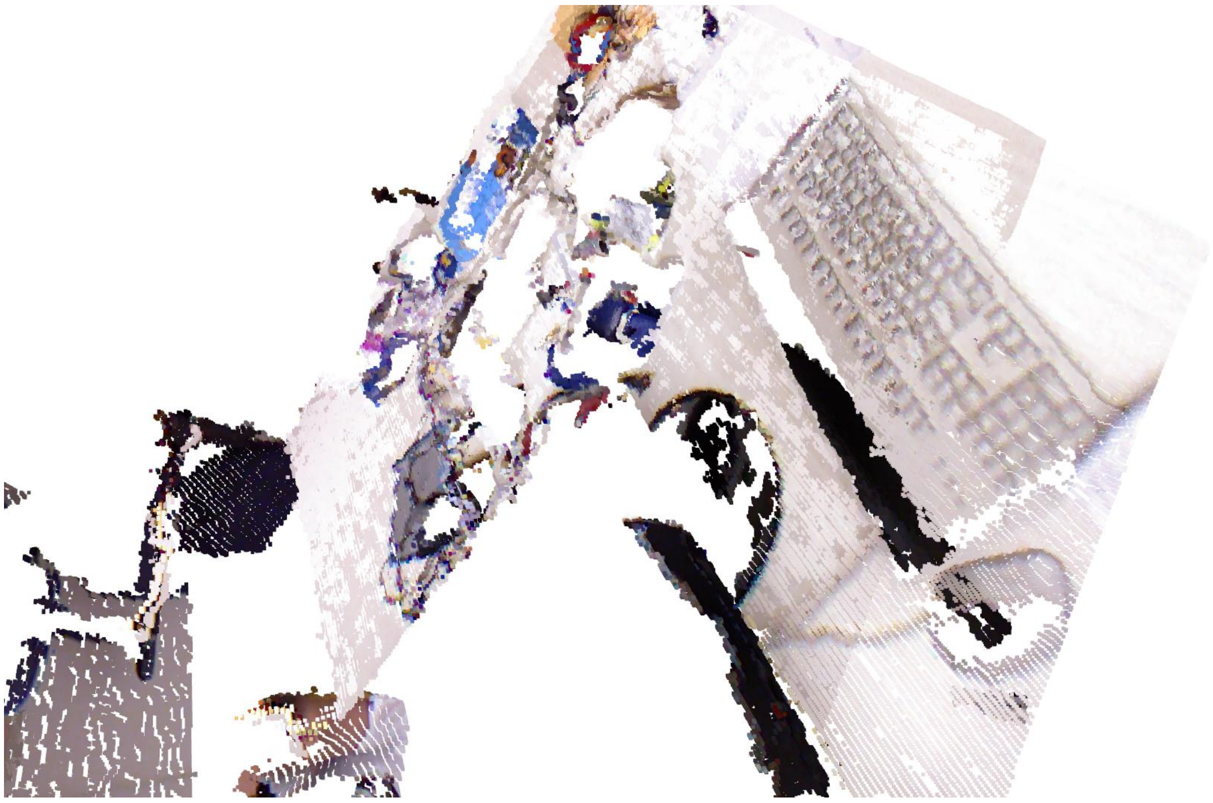
When using RANSAC the results improve significantly:



Now at least the keyboards are somewhat aligned. However the result could still be further improved.

Dense Iterative Closest Point

The last step of this assignment consists in using dense ICP (Iterative Closest Point). ICP in itself is not capable of determining the transform between two point clouds, however it is cabable of refining an already existing guess of a transform between two point clouds. The ICP algorithm works by trying to tightly align two point clouds. Running this algorithm we obtained the following result:



We note now that the keyboard is perfectly aligned. Thus we have managed to arrive at a very precise and accurate transformation between the two camera frames. We note however that the ICP takes a considerable amount of time to run. The RANSAC algorithm however is quite fast, and seems to arrive at a reasonable transform between the two cameras.

Depending on the application one might use one algorithm or another, or combinations of both.