Advanced Vision Assignment № 2

Todor Davchev & Ruslan Burakov (s1569105), The University of Edinburgh 17/03/2015

1 Overview

The main goal is to merge range data for cube like object obtained by Kinect 3d sensor. In order to align data for different frames into one coordinate system 3 marker balls of different colors are used. These 3 balls shape triangle which surrounds a cube like object in the middle. For all frames the objects must remain static and keep their relative distances and orientations.

1.1 General Algorithm

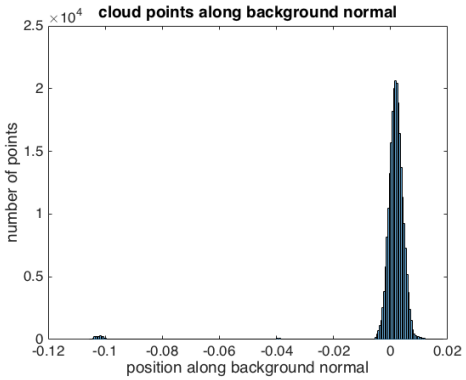
(adopted from description of main function)
For each frame:

- Extracting foreground using PCA and remove noise by using percentile statistics (relying on the fact that overwhelming majority of cloud points belongs to the background). Due to that most of outliers are removed on this stage
- Use Density-based spatial clustering of applications with noise (DBSCAN) clustering to allocate initial clusters based on points neighbouring and number of neighbours around them
- Remove outliers near border from clustering stage by using the fact that once background is subtracted most of the points must belong to cube and clusters which are too far away from cube are outliers
- Extract spheres clusters by using geometric information

After that, select baseline frame and compute colour histograms of its spheres and align spheres from all other frames accordingly. Find the rotation and translation of coordinate frame via PCA. Apply rotation and translation to the cube cluster.

Finally merge all cube cloud points from each 3D frame and utilise patch growing algorithm to select 9 planes. On this stage sampling may used to reduce the number of cloud points and reduce load on patch growing algorithm.

(a) Background points lie almost within plane



(b) Cloud points projected along background normal

Figure 1

2 Background plane extraction

2.1 Initial Subtraction

It relies on the observation that overwhelming majority of pixels belongs to the background. The background surface essentially can be approximated by plane as frames are recorded on smooth floor. It can be seen on Figure 1a where it is shown for the first frame. However, note that near the border cloud points of the background go slightly below the main plain. This happens presumably due to the imperfection of Kinect 3d sensor. To extract plane we need to estimate normal of this plane and plane distance from origin of coordinate frame.

First, we compute the normal to the background surface by applying PCA to centred cloud points. As we have 3D data we will receive 3 principal components. 1st and 2nd will be lying within surface of the background (because on the background surface the variety of coordinates is the greatest) and 3rd component will be orthogonal to them and, therefore it will be normal to the background surface. After that we project centred cloud points on background normal and get distribution where most of the points are concentrated on one level (background points near the surface of background plane). This distribution is shown on Figure 1b. As we can see most of the points are located near one place along background normal axis (high peak near 0) and only relatively small amount of cloud points are located above the plane (small peak near -0.1). This confirms our hypothesis that background forms a plane. By taking 50th percentile of such distribution (basically it will give us location of the biggest peak because it dominates the data) we are able to find distance from origin of global coordinate frame.

Finally, we determine the correct direction of normal vector (where the spheres and cube cloud point lie) and remove all points which are located not high enough from surface plane. By doing that we are removing background cloud points near the border which go slightly below the main background plain. The results on this stage contain almost no points which belong to background. This is demonstrated on Figure 2a for the first frame. The only exception occurs on 8th frame where we have a few border outliers which lie above the main background plane. It can be seen on Figure 2b where outliers are marked by red circle. But this is handled on the

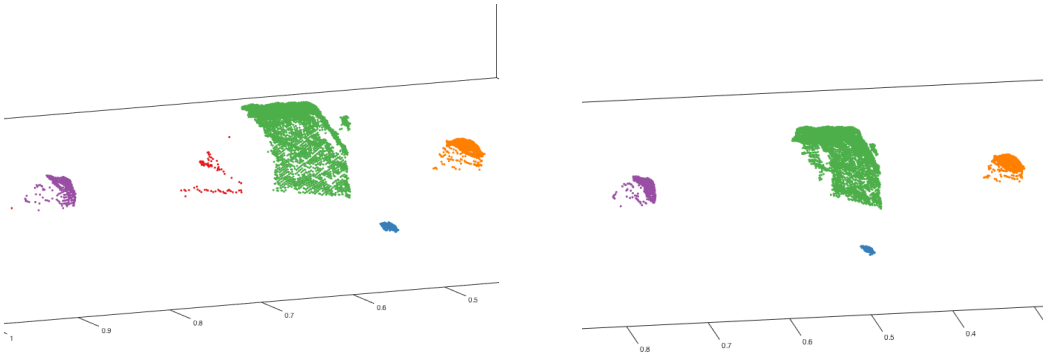
next stage.

(a) Subtracted background for frame 1

(b) Small number of border outliers on frame 8 marked by red circle

Figure 2

2.2 Clustering



(a) Clustering with some noise marked as red

(b) After removing noise

Figure 3

Here we determine the main clusters which potentially can be one of 4 objects we are interested in: 3 spheres or 1 cube. For that we are using Density-based spatial clustering of applications with noise (DBSCAN). The main advantage of this clustering algorithm to let's say kmeans is that it doesn't use explicit number of clusters and clusters are formed by connectivity to the nearest neighbours. Due to that it is possible to extract patches that forms very long shapes which allows to remove most of the noise (by offsetting minimal number points required to form cluster) located near the cube. This is demonstrated on Figures 3a and 3b.

The parameters used for scanning:

- number of objects in a neighbourhood of an object: **80**. Or in other words minimal size of the cluster. It was estimated empirically by printing number of points in located clusters which were matched to object of interests (on this stage by plotting the data).

- neighbourhood radius: **0.015**. It was also estimated empirically by plotting clustering results (similarly to what can be seen on Figures 3a and 3b

Finally, the biggest cluster is located which must belong to cube. For example, the number of cloud points in the extracted clusters for the first frame:

cluster1	137
cluster2	5050
cluster3	442
cluster4	796

As we can see we have a cluster which has sufficiently more points as others. The same sitiuation holds for the rest of the frames.

Next, the remaining clusters are checked by proximity to the cube cluster. If they are too far way then they are recognised as border outliers and removed. If outliers were detected then after their removal additional clustering is used to get final results. This allows to remove border outliers from Figure 2b. The implementation of DBscan is taken from [Das04].

3 Sphere extraction

Figure 4: Spheres fitted to ball patches for first frame

For all 16 frames once we located the biggest cluster (and we confirmed visually that indeed it corresponds to cube object) then we had only 3 other cluster which corresponded to the patches of ball cloud points. As extensions just in case we implemented searching of sphere clusters based on the idea that ball clusters will form triangle with the biggest area given that border outliers are successfully removed.

After that the spheres were fitted to the ball patches. This is shown on Figure 4. Approximation of sphere centres proved to be reliable in the further stage when the final merging of cube data was done.

4 Registration

4.1 Matching spheres to baseline frame

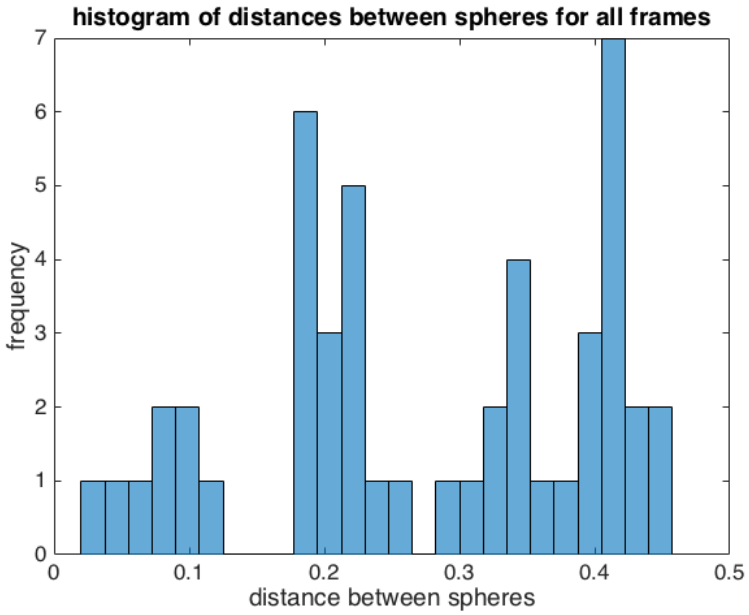
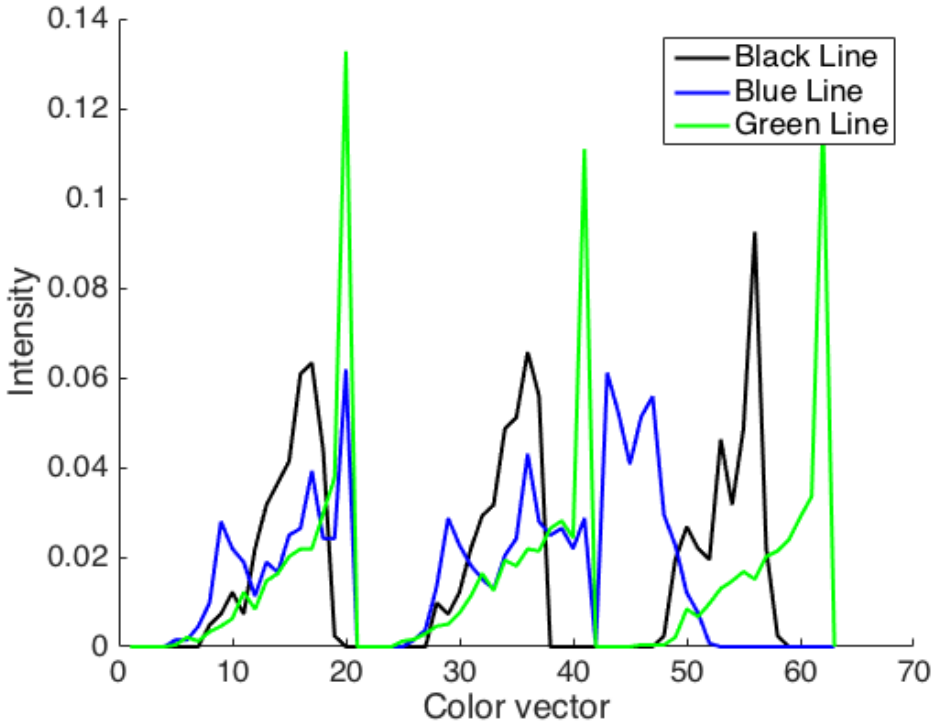


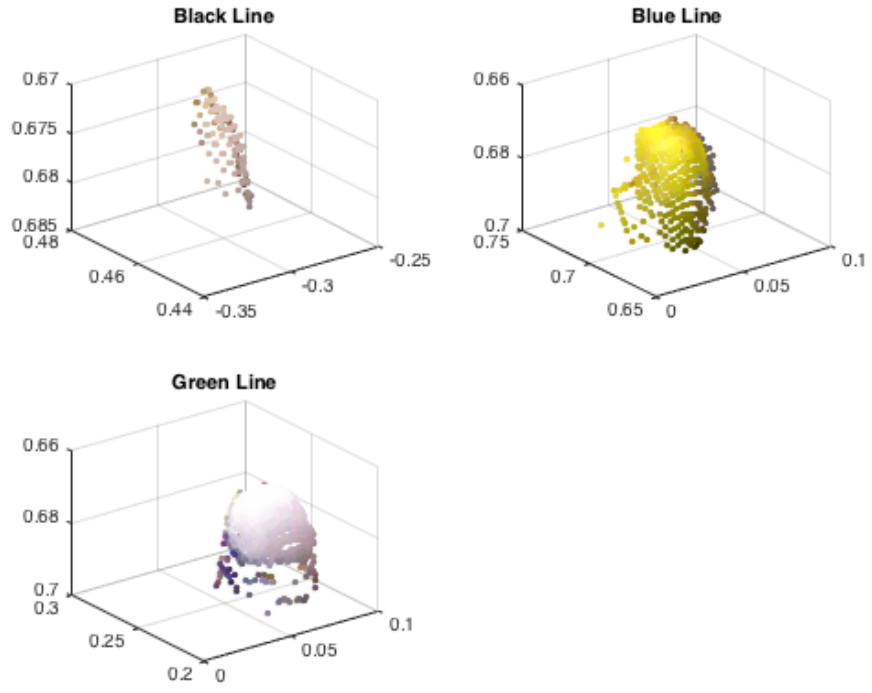
Figure 5: Distribution of distances between sphere for all frames

We had several ideas on that. First we thought about using the length of triangle sides which is formed by the balls. Indeed, as it can be seen on Figure 5 we can distinguish three different groups which corresponds to 3 different sides of triangle. Given that one of the balls is considerably smaller than others, potentially this method could be used. However, acknowledging the fact the last 2 groups can be very similar we decided to utilise colour information in order to distinguish between different balls. For that we computer colour histograms and calculated bhattacharyya distance between histograms for different frames in order to find right matches between balls. This can be done easily because in contrast to the first assignment colour histograms of each ball are very unique (there is no ambiguity between them). This is demonstrated on Figures 6a and 6b. Here colours of the lines mean only correspondence to the particular ball on Figure 6b because all colour channels (red, green, blue) are concatenated to form single vector and normalised.

Using colour histograms matching worked properly for all frames which was confirmed by next stage



(a) Colour histograms (all channels are concatenated to form single vector and normalised)



(b) Corresponding balls from the first frame

Figure 6

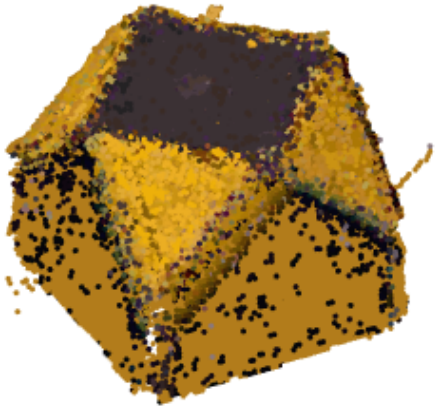


Figure 7: merged cube cloud points

4.2 Merge Cube Points

Using matches between different balls for different frames obtained from colour histograms the transformation to baseline frame was computing for all frames. Transformations were found with PCA to minimise mean square error between triangles positions and rotations formed by balls. The resulting merged cube is shown on Figure 7. Note, that due to the clustering stage the resulting merged Cube has relatively small amount of noise.

References

[Das04] Michal Daszykowski. *DBScan Matlab implementationk*. 2004. URL: http://www.chemometria.us.edu.pl.