376.054 Machine Vision and Cognitive Robotics (VU 4,0) 2018W

Link to TISS Lecture

Exercise 5: Clustering

Download files for exercise 5

Part 1: Implementation (10 points)

After the dominant plane in the pointcloud has been found with RANSAC in the previous exercise, in all points of the plane can be removed in order to isolate all points corresponding to potential objects. In this exercise, these remaining points should be clustered, such that in the ideal case all points of an object are associated to the same cluster. A good clustering result is crucial for a reliable object recognition, which you will implement as the final challenge of the course. An example of a good clustering result can be seen in figure 1.

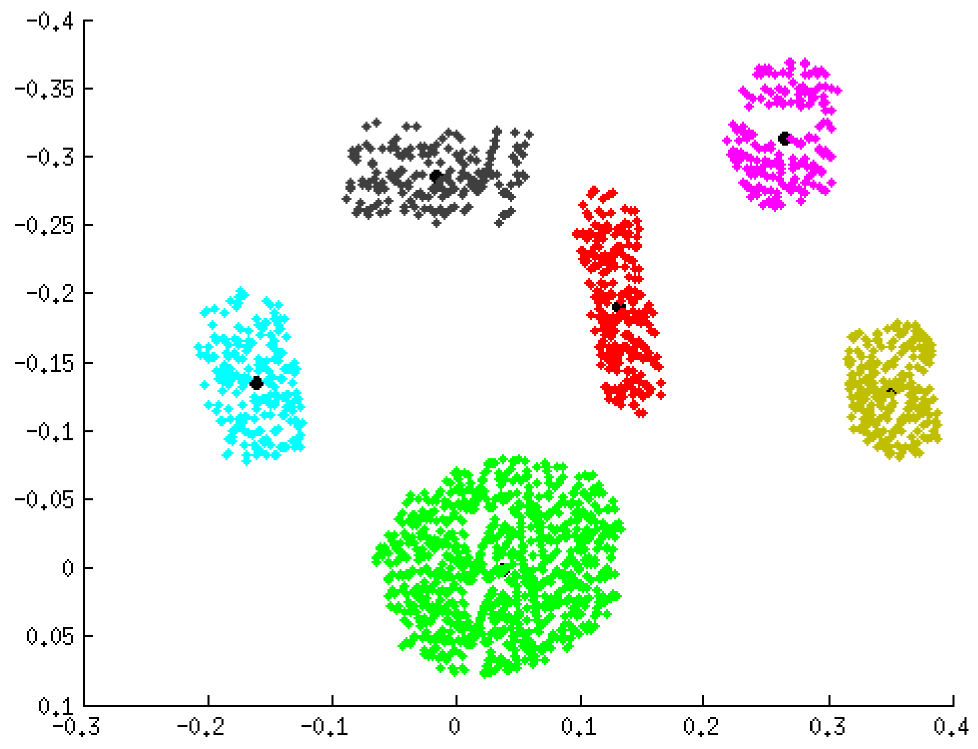


Figure 1: Perfect clustering result for 6 objects. Different colors represent different cluster numbers.

There are many different approaches to clustering and, all have their advantages and disadvantages. In this exercise you will implement hierarchical clustering algorithms. These methods have the advantage that the number of clusters does not need to be known in advance (as opposed to k-means). Furthermore hierarchical clustering can cope with different cluster sizes and is reasonably fast (in contrast to mean shift, for example).

1. Single linkage clustering (4 points)

The main idea of the algorithm is to cluster the points of the given pointcloud `p` considering the pairwise distances of the points to each other. In the beginning, every point can be seen as its own cluster. As the algorithm progresses, clusters which are similar enough are merged. In our case the similarity is defined as the euclidean distance between the clusters. **Thus, clusters which are close enough to each other will be merged.** For the distance measurement you should implement the *Single-Linkage method*, which defines the distance between two clusters as the **shortest distance between two points of the clusters**. The merging process stops if the distances between all clusters are larger as the allowed distance `maxdist`.

To calculate the pairwise distances of all points to each other, you can use the function `ipdm` written by John d'Errico. It provides many nice features which can save you a lot of time. Refer to the accompanying HTML documentation file to figure out which options are most suitable for an memory-efficient and fast implementation. Alternatively, you can use the matlab builtin `pdist`.

If you do not get satisfying plane fitting results with your RANSAC implementation of the previous exercise, you can use the built in plane-fitting method from the computer vision system toolbox by setting the `use_builtin` variable.

Complete **clustering_single.m**.

2. K-d tree clustering (4 points)

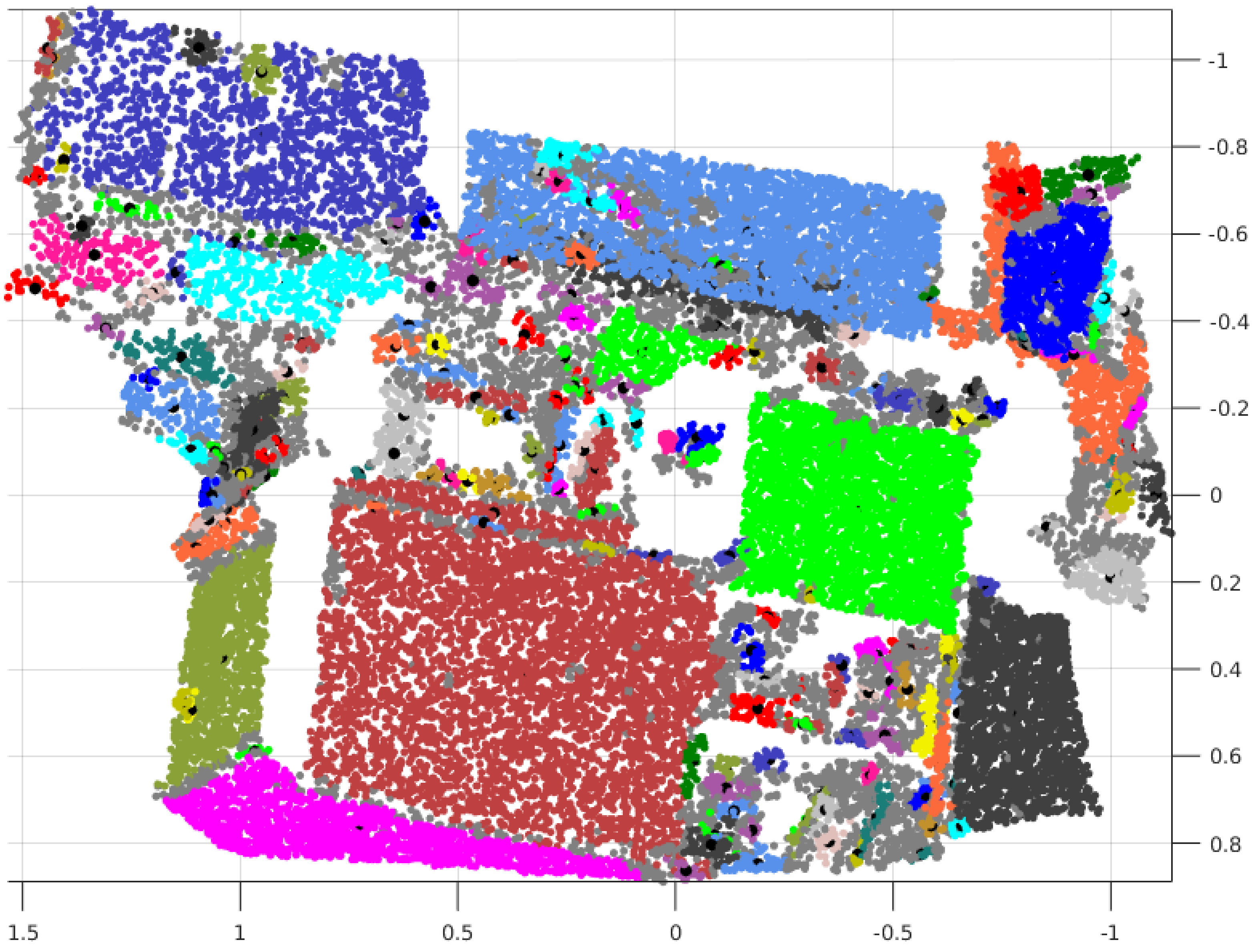
In the approach above for single linkage clustering we must at some point store the matrix of inter-point distances. For small clouds this is generally not an issue, but the raw clouds from an average depth sensor easily exceed 300,000 points. To store a 2D matrix of floats this width and height would require 36GB of memory, which, while in reach these days, is still rather a lot of memory to use. Downsampling the clouds to a tenth of the original size would cut this to 3.6GB, which most systems can handle, but is still rather a lot. In practice, we do not actually need the distances from each point to all other points - most points only need to compare distances to neighbours in a relatively small radius.

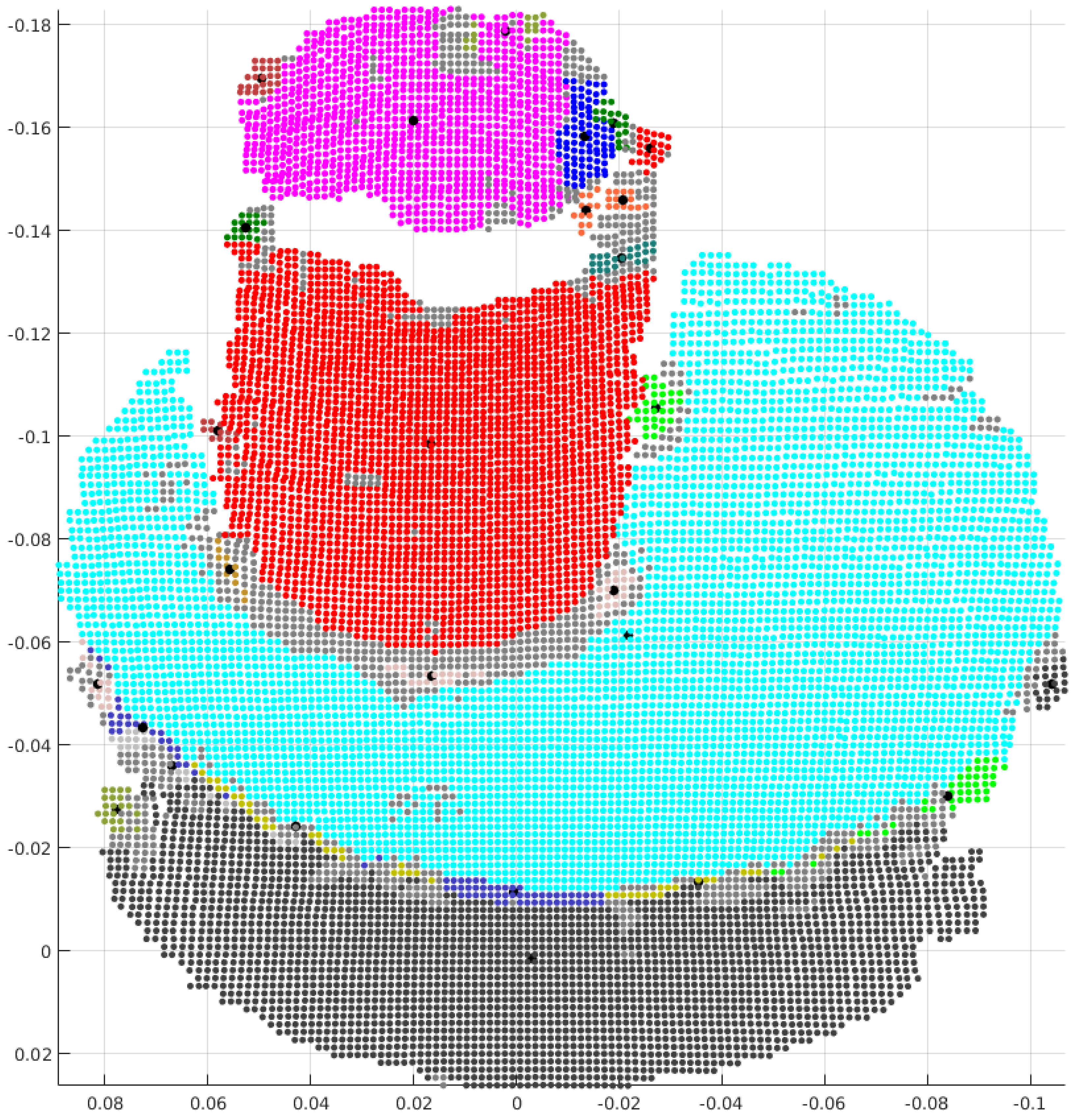
The k-d tree is a data structure which partitions a space based on the positions of points in that space. It allows for efficient range and nearest neighbour search due to this partitioning, and is much more memory efficient than using a matrix for storing all distances. It is available as `KDTreeSearcher` from MATLAB's statistics and machine learning toolbox.

Using the k-d tree, implement another clustering method. You should start the clustering with a single point, and look for its neighbours within `maxdist`, using `rangesearch`. Each neighbour is added to the cluster. The neighbours of each of those points is added (excluding points already in the cluster), until there are no longer any more points to add for that cluster. Repeat this process until all points in the cloud are assigned to a cluster. Clusters can be of size 1 if a point has no neighbours within `maxdist`.

Complete **clustering_kdtree.m**.

3. Adding normals (2 points)


-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1.5
1
0.5
0
-0.5
-1


-0.18
-0.16
-0.14
-0.12
-0.1
-0.08
-0.06
-0.04
-0.02
0
0.02
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1

If you apply the clustering implemented so far to the more complex scenes in the pointcloud set, you will likely receive a result which is not very useful. To mitigate this problem, we can add another constraint on what it means to be part of a cluster. There are many different constraints which can be used, but we will consider the normals. It is possible to estimate a normal vector at each point, which gives an indication of the orientation of the surface. We can thus differentiate, for example, objects on surfaces, or objects on other objects (such as in `image009.pcd`), as the normals of adjacent points from different objects will likely be quite different to each other.

Add an additional test on the neighbours in your k-d tree implementation so that neighbours with a normal which differs by more than `ang_thresh` radians of the normal of the current point are not included in the cluster.

You can use the small `tin.pcd` cloud to test your implementation more quickly than using full clouds.

Complete **clustering_kdtree_norm.m**.

Performance

The number of points to be clustered varies from scene to scene. Pointclouds can contain large numbers of points. Standard depth sensors produce more than 300,000 points per frame, which makes a fast clustering very difficult and requires a lot of memory, as for fast implementations many distance values have to be kept in memory. **To overcome this issue, the pointcloud is subsampled before your clustering function is called from the main script** (e.g. only every 10th point will be considered). This step drastically decreases the processing time (which grows, depending on the implementation, quadratically or even cubically with the number of points), while there are still enough points left to achieve a reasonable result. If later all points need to get an associated cluster for further processing, every point not having been assigned a cluster due to subsampling can be assigned the cluster number of its nearest neighbor (which has an assigned cluster number). Figure 1 shows a clustering result for a pointcloud for which only every 10th point has been processed (1928 instead of 19280 points, processing time 0.15 instead of 10 seconds).

Forbidden functions

Please note that the built-in `linkage` function must not be used to solve this exercise.

Remarks

In this exercise we cluster depending on the euclidean distance of points, thus we only consider the spatial coordinates of the points and ignore the colour information. However, the results could perhaps be improved if colour were also taken into account as a criterion for the clustering (which is done in many clustering algorithms). However, this would also raise some more questions: higher dimensionality means decreased performance? How to appropriately calculate distances in colour space? What happens with objects with strong colour transitions? As such, we focus only on the spatial channels in this exercise.

Part 2: Documentation (6 points)

- Answer each numbered question separately, with the answer labelled according to the experiment and question number. For example, experiment 1 question 2 should be numbered 1.2. You do not need to include the question text.
- Include images to support your answers. These should be referenced in the answers. The document should include information about what parameter settings were used to generate images so that they can be reproduced..
- Your answers should attempt an explanation of why an effect occurs, not just describe what you see in the supporting images.
- Be precise. "It is better" or similarly vague statements without any additional information are not sufficient.

Points may be deducted if your report does not satisfy the above points.

1. Show your results for each of the algorithms for pointclouds 5 and 9, and two of the more complex clouds. (1 point)
2. How does the choice of `maxdist` affect results? Give examples. (0.5 points)
3. Plot the runtimes of the different clustering algorithms on at least two clouds, with at least three different levels of downsampling. How do the methods compare? (1.5 points)
4. What effect does downsampling have on clustering, other than runtime benefits? (0.5 points)
5. How does the number of points used to estimate the normal at each point affect the results of the normal-based clustering? (0.5 points)
6. Explain how (if at all) the choice of a random starting point affects k-d clustering, as compared to a fixed starting point (e.g. always starting with the point at index 1). (0.5 points)
7. In the implementation, you used `rangesearch`, which returns points within a certain distance. Another option is to use `knnsearch`, which will return the `k` nearest neighbours of a given point. Describe how this might affect the clustering results. (0.5 points)
8. Without running SAC, compare the results of k-d tree and k-d with normal clustering on some of the numbered pointclouds. Are the results satisfactory compared to those with the plane removed? (0.5 points)
9. You may notice that the normal based clustering has some artifacts or regions without consistent clusters around the "seams" between objects or surfaces. Why is this? Suggest a step after the main clustering completes, or something you might be able to do while the clustering is running to mitigate the issue. (0.5 points)

You must specify the parameters used in all of your screenshots.

Maximum 1500 words, arbitrary number of pages for result images (referenced in the text). Violating the text limit will lead to point deductions!

Assistance

Please keep to the following hierarchy whenever there are questions or problems:

1. Forum: Use the TUWEL forum to discuss your problems.
2. Email for questions or in-person help requests: machinevision@acin.tuwien.ac.at

Upload

Please upload the Matlab files **cluster_single.m**, **cluster_kdtree.m**, and **cluster_kdtree_norm.m**, as well as your documentation (pdf-format) **as one ZIP-file** via TUWEL.

Submission status

Submission status	No attempt
Grading status	Not marked
Due date	Wednesday, 19 December 2018, 12:00 PM
Time remaining	16 days
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet

◀ Lecture Slides 9

Jump to...

Files for exercise 5 ▶