

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

[Link to TISS Lecture](#)

[TUWEL](#) / [My courses](#) / [376.054-2018W](#) / [10. 3D Object Recognition](#) / [Open Challenge: 3D Object Recognition](#)

Open Challenge: 3D Object Recognition

In the final exercise you will plug different modules and functions you programmed in the previous exercises together to form a complete 3D object recognition pipeline. As this exercise is designed as an *open challenge*, you will not be provided with a code construct or function headers and it's up to you how to implement the algorithm, which will be explained in the following sections.

Dataset

Download files

The 7 objects represented in the dataset are:















Book

Cup

Cookiebox

Ketchup

Sugar bowl

Sweets

Tea

The dataset of 3D point clouds is divided into three sets, namely **training**, **test** and **groundtruth**. The training set contains point cloud files in which the respective object has already been cut out by the clustering algorithm. For each object, many different point clouds are available, such that almost all viewpoints of the objects should be captured. The test set consists of 10 different point clouds containing a setup of the objects. In these point clouds the object recognition algorithm has to correctly detect all objects. The groundtruth set is actually not necessary for this exercise, but it might be used to check the recognition results, as it contains the correctly recognized and cut out objects for each test image.

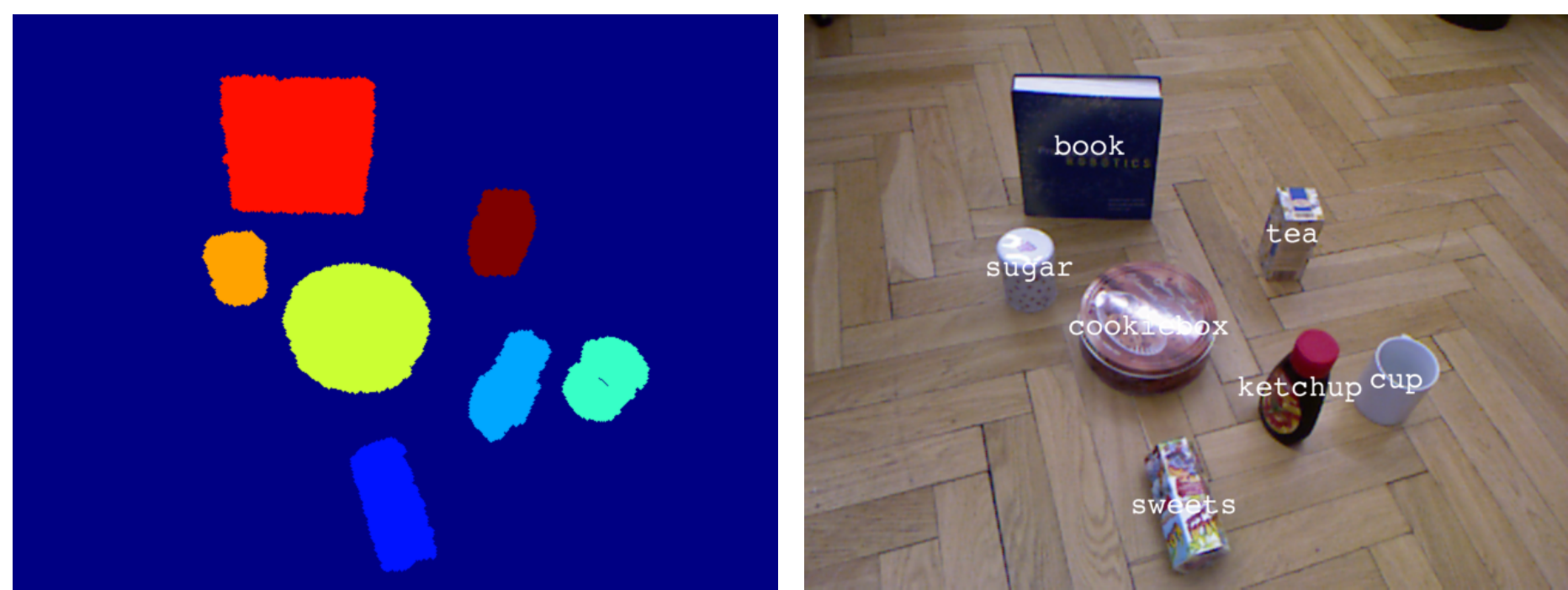
Examples of how to load a point cloud from a file can be seen in the main scripts of the previous exercises.

Useful files

The file download contains a few files you can use if you prefer to use MATLAB builtins to do plane fitting and clustering. This may be useful if your solutions do not give good results. Also included is a file with parameters for the camera used to generate the images of objects on the floor (you only need the focal lengths `fx_rgb`, `fy_rgb`, and the coordinates of the optical center `cx_rgb`, `cy_rgb`).

Algorithm (12 points)

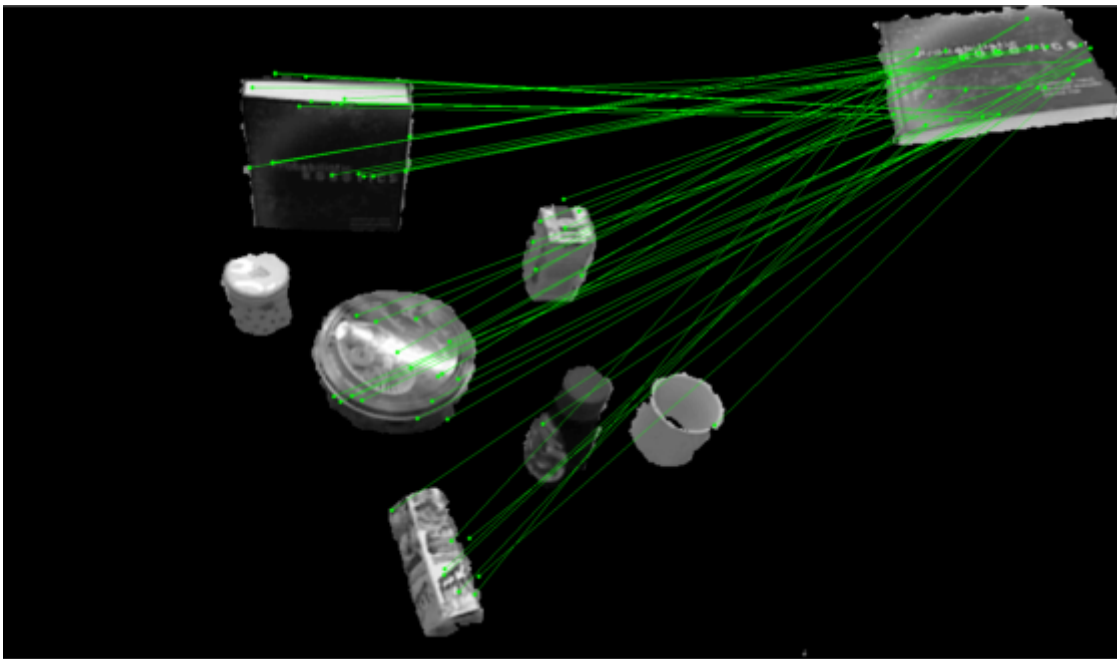
The object recognition algorithm works on the result of the clustering method of the previous exercise. The goal of this final step in the pipeline is to be able to assign each cluster of the input point cloud to the corresponding object class. A correct result could look like this:





The general idea of the method is again based on the principle of matching SIFT descriptors. Matches will be calculated between a test point cloud and all of the separate training point clouds containing an object. As SIFT is a 2D image feature and cannot directly be used on 3D point clouds, point clouds have to be projected to the 2D image plane first. After the matches to all training point clouds have been calculated, the classification of each object cluster is decided according to which training point cloud had the most matches with the respective cluster.

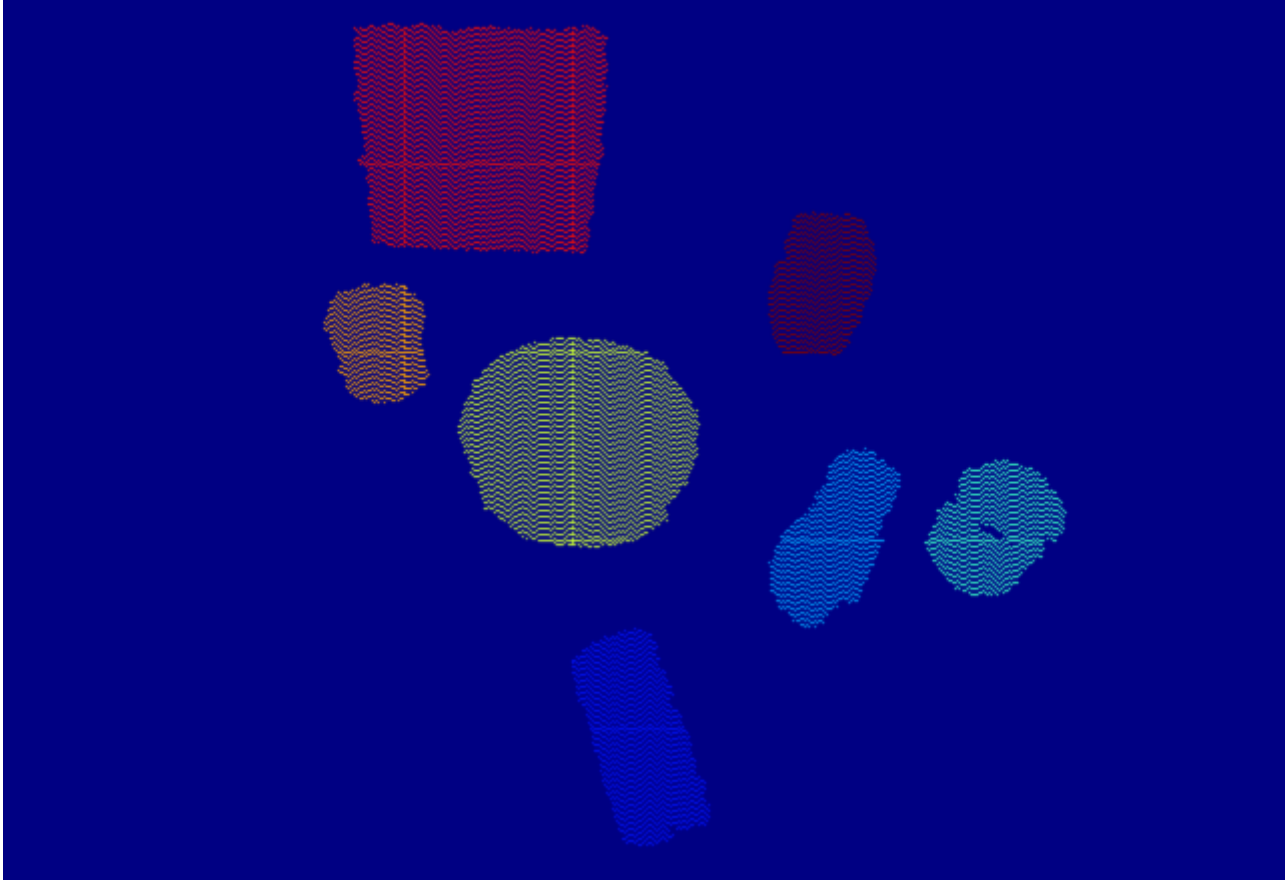
Here is an example of the matches of a book training point cloud to the backprojected clusters of an input point cloud:



To calculate the SIFT feature points and corresponding descriptors, the same code package as in exercise 3 can be used. For the necessary pre-processing steps required by the SIFT calculation, refer to exercise 3 as well.

Here is the (suggested) basic sequence of the algorithm:

1. Apply your RANSAC algorithm from exercise 4 on the test point cloud to eliminate all points corresponding to the ground or the table plane.
2. Project all remaining 3D points to the 2D image space using the given camera parameters (focal length and the coordinates of the optical center).
3. Calculate the SIFT descriptors for the projected image.
4. In 3D space, cluster the remaining points from step 1 as you did in exercise 5. You should probably subsample the point cloud before to speed up the process. An additional post-processing step after the clustering could be the elimination of clusters which do not contain many points, because these clusters might not correspond to objects but to the ground or table.
5. Project the resulting **cluster numbers** of the 3D points to the 2D image space. This will result in an "image" like the left one above (displayed with colormap jet). If you subsampled before clustering you might end up with holes in the projected objects (like in the image below). These holes can be "filled" by applying a simple dilate filter on the image: `se = strel('disk',round(subsampling/2)); projected_img = imdilate(projected_img,se);` where `subsampling` is the subsampling rate for the clustering.
6. Start a "voting" process with the clusters and training point clouds: For each training point cloud:
 - Project it to 2D image space
 - (Maybe crop the projected training image to the object pixels)
 - Calculate SIFT descriptors
 - Match the descriptors with the ones calculated for the test image in step 3.
 - Count how many matches fall into each cluster. One way to check in which cluster a match fell is to look at the cluster numbers of the adjacent pixels of the matched 2D coordinates in the "label image" you created in step 5.
 - Assign each cluster the object class of the training image which had the most matches in the cluster. You might have to introduce some kind of normalization here, otherwise object classes of large objects like the cookiebox will get assigned most of the time because they get many false matches.



The projected clusters contain "holes" due to the subsampling before the clustering.

Bonus points

You can make improvements and add additional functionality to get up to 10 bonus points for this exercise:

- Calculate the object's pose. RANSAC could be used on the counted SIFT matches for the recognized cluster to retrieve the object's correct geometric pose in the scene.
- Improve the recognition rate. You can think of additional or better features (e.g. use color information as well) to also be able to reliably recognize the cup which doesn't have a lot of texture.
- Improve the clustering. Also the clustering algorithm could benefit from using the color information as an additional cue to separate objects. One approach could be to incorporate an additional "color distance" next to the spatial distance term which is used to determine how "similar" points are.
- Anything else you can think of which would be a logical extension. If something you do does not work, you can still include it and discuss its failure in the documentation.

You may receive some bonus points for attempts at implementing these the above, or any other ideas you have, even if they do not perform well. **You must explain extensions in your documentation to receive bonus points.**

Documentation (8 points)

- 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 a screenshot of the result for each test point cloud (like the "recognition result"b image). You can add text to a MATLAB plot using the `text` command. This is required so we can see all your results. (no points for this)
2. Describe your approach. Have a separate section for each step, explaining what you do in that step and what its intended purpose is (e.g. apply clustering so we can create a set of points which we assume are all part of the same object). In each section, list all additional ideas you tried to improve that part of the process, even if they didn't work. Explain why you tried each, and summarise the results. Mention the files and line number ranges where the extensions can be found. You should include images showing what the extension does, if relevant. (3 points)
3. Evaluation of the result and discussion. How many correct detections can you get for each class and overall? You must present these results quantitatively. Include tables and/or graphs. What could be the reasons why some classes are recognised better than others? What parameters have been used and how do they influence the results? If you implemented some extensions, you should compare detection performance before and after adding them. You should also include details about the runtime of each stage (excluding RANSAC and clustering, unless you have implemented something different to previous exercises) (5 points)

You must specify any parameter settings you used to generate images that you include.

There is no word limit for this exercise.

You must include instructions for how to run your code. This can be in a separate readme file submitted in the .zip, or as part of your documentation.

Assistance


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

1. Forum: Use the TUWEL forum to discuss your problems.

Upload

Please upload all files required to make your system work, any code for extensions (even if it does not work well), the helper files from previous exercises, if you made use of them, and your documentation (pdf-format) **as one ZIP-file** via TUWEL.

Submission status

Submission status	Submitted for grading	
Grading status	Released	
Due date	Tuesday, 15 January 2019, 12:00 PM	
Time remaining	Assignment was submitted 8 hours 5 mins early	
Last modified	Tuesday, 15 January 2019, 3:54 AM	
File submissions	<div><div>-</div><div> exercise6.zip</div></div>	15 January 2019, 3:54 AM

Submission comments


+

[Comments \(0\)](#)

Edit submission

You can still make changes to your submission

Feedback

Grade	11.5 / 20.0	
Graded on	Friday, 1 February 2019, 2:26 PM	
Graded by	<div></div>	Kessler Markus

Feedback comments

+

Implementation

Extension:
- (+1 point) Attempts to use colour information

(-1 point) There is a bug at the end of your main script (line ...

◀ Lecture slides 10

Jump to...

Files for open challenge ▶



[My courses](#)

[All my courses](#)

[Course overview](#)

[Search courses](#)

[New course](#)

Help

[TUWEL Tutorials \(Students\)](#)

[TUWEL Tutorials \(Teachers\)](#)

[TUWEL scenarios](#)

[E-Mail Ticket Support](#)

[General Information](#)

[Terms of Use](#)

[Teaching Support Center](#)