Computer Vision Project
Winter 17/18
Franz Köferl, Peter Fürsattel, Vincent Christlein, André Aichert,
Tobias Geimer, Thomas Köhler, Schirrmacher Franziska

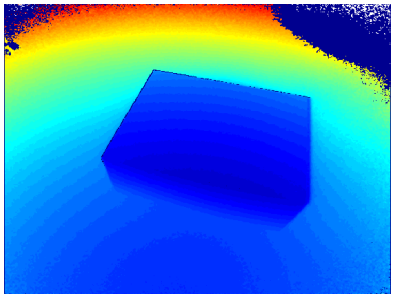
Sheet 1, starting from October 23rd, 2017, due December 4th, 2017, 20:00 Uhr

Exercise 1.1: Box Detection

In the first exercise you will implement a small program that estimates the size of a box from a distance image in Matlab. While the slides of the first lecture provide most of the tools you will need to do some research on your own on how to solve the task best. The following description will serve you as a guideline to complete the task.

Feel free to write your own helper functions and try to keep your code structured and easy to understand. You may include own ideas and extensions as you like.

(a) ToF amplitude image



(b) Distance image

Getting and reading the data

Download the example files from StudOn and extract them into your working directory. The `.dat`-files contain different examples that you can use for testing your implementation. Each example consists of an amplitude image, a distance image and a point cloud. Pixels can be accessed via `A(height,width)`. Begin with loading one of the examples and visualize its content, for example with the command `imagesc` or `scatter3`. Experiment with the filters introduced in the lecture and evaluate if one of them can be useful to improve the input data.

Note that it is valid to subsample the point cloud to improve the runtime (e.g. when using `scatter3`). However, you need to make sure that you do not loose detection accuracy.

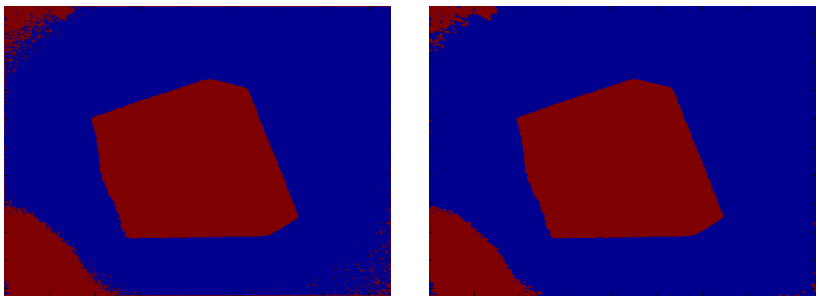
RANSAC

Subtracting single distance measurements from each other to calculate the height of the box is sensitive to noise. A better approach consists of finding two planes that approximate the floor and the top of the box and calculate the distance between the planes. A simple solution to finding dominant planes in a point cloud is to use RANSAC to find the plane models with the most inliers, in our example the floor and the top of the box.

Use the pseudo-code contained in the slides to implement RANSAC. Your implementation should take three parameters: a point cloud that contains 3D-vectors, a threshold which is required to evaluate the quality of a model and a parameter for the maximum number of iterations. It is advised to use a parameter/normal representation for the plane models.

$$n_x x + n_y y + n_z z = \mathbf{nx} = d \tag{1}$$

Your code should return the best fitting model if the maximum number of iterations have passed or if all candidate points are within the inlier set. Use your implementation of RANSAC to find all pixels that belong to the floor. Visualize all inliers with a mask image that represents inliers with a value of 1 and outliers with 0. Experiment with different parameters and find a parameter set that finds the floor plane reliably.



(a) Floor mask (b) Filtered floor mask

Some additional notes on the implementation:

- The point cloud may contain invalid measurements which can be identified by checking if the z-component of a vector is 0. These points should be ignored in your RANSAC implementation.
- Due to the amount of points, you should make sure that you use Matlab functions efficiently and avoid using loops. Useful functions could be `repmat`, `all` or `dot`'s capability of calculating the dot product of two matrices.

Filtering on the Mask Image

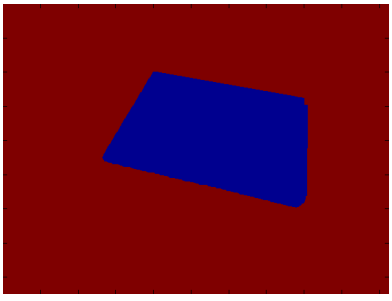
Use morphological operators to improve the quality of the floor mask. Evaluate which operators suit the task best. Use the resulting mask to find all pixels/3D points that do not belong the floor (e.g. the box, background objects, noise, ...).

Finding the Top Plane of the Box

Create a new matrix that contains all 3D-points which are not part of the floor. Use your implementation of RANSAC to find the dominant plane within this set. The example data is captured such that the second largest plane is the top of the box. Create a new mask image that represents the inliers of the box plane.

Pixels which do not truly belong to the box will later create errors in the size estimation of the box. It can be assumed that the largest connected component in the mask delimits the top of the box. Find the largest connected component in the mask with Matlab's `bwconncomp` command. Evaluate if additional preprocessing or filtering steps can improve your result.

Abbildung 3: Box top component



Measuring the Dimensions of the Box

At this point everything that is required to estimate the size of the box is known: a mask that delimits the top plane of the box and plane representations of the top and the floor. Calculate the height of the box by computing the distance between the two planes. One way to find the length and width of the box is subtract the 3D-coordinates of the corners from each other. Analyze the pixels of the box mask to determine the corners.

Exercise 1.2: Box Detection - Discussion

Create a simple visualization of your results, either text, figures or both. If you made any extensions or included own ideas in the algorithm visualize these as well.

This simple implementation comes with several weaknesses. Identify some of these drawbacks and make suggestions on how to make the algorithm more robust, more accurate or faster.

Show your implementation and your results to one of the advisors.

Abbildung 4: Visualization of floor, box and box corners.