Assignments

Robotics, Vision and Control

Umberto Castellani

Homework1

Play with Zephyr:

- **TUTORIAL 1**: https://www.3dflow.net/technology/documents/3df-zephyr-tutorials/convert-photos-3d-models-3df-zephyr/
- **VIDEO-TUTORAL**: https://www.3dflow.net/it/tutorial-per-3df-zephyr/
- **VIDEO-TUTORIAL WITH DATA**: https://www.3dflow.net/it/community-fotogrammetria/3df-zephyr-vetrina-di-ricostruzioni/

Cherub Statue

Clicca qui per scaricare il progetto .zep

Clicca qui per scaricare il dataset completo

Ricostruzione di una statuetta con 65 fotografie.

Storico dataset utilizzato nel primo tutorial di 3DF Zephyr nel quale viene mostrata la classica modalità di acquisizione per piccoli oggetti.

Homework2

Create your 3D model of your physical object:

- This model will be used in our robotics, vision and control pipeline

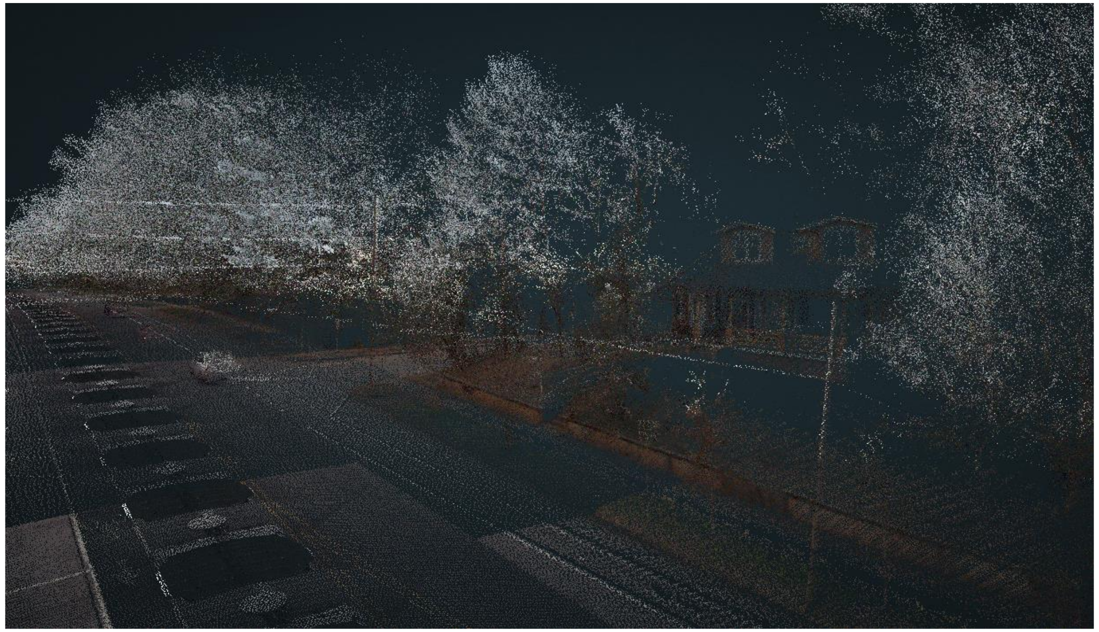
The object should be grabbed by the robot

12 cm

4 cm

Homework1

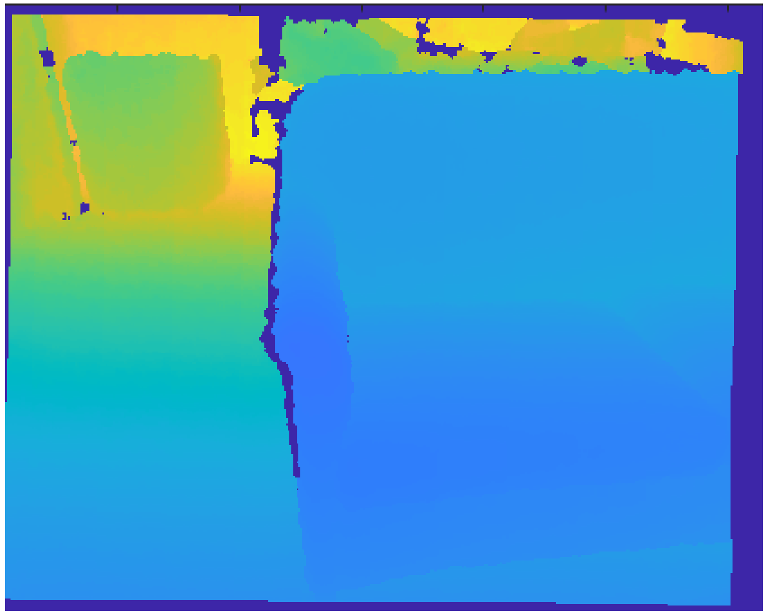
1. Find public dataset of range images from different acquisition systems
   - E.g. https://www.lidarusa.com/sample-data.html#



Homework2

2. Create a 3D cloud of points from a range image

Color image

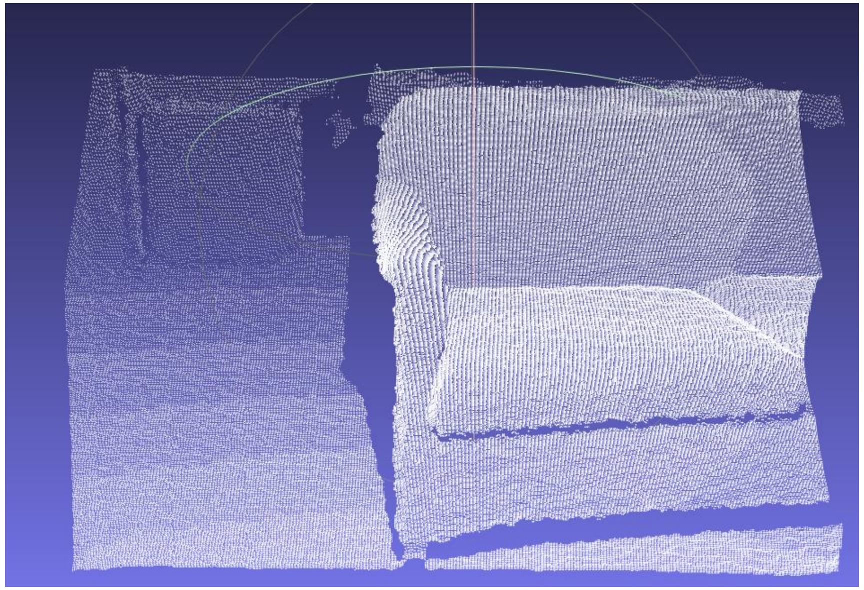


Range image

Homework2

2. Create a 3D cloud of points from a range image

Color image

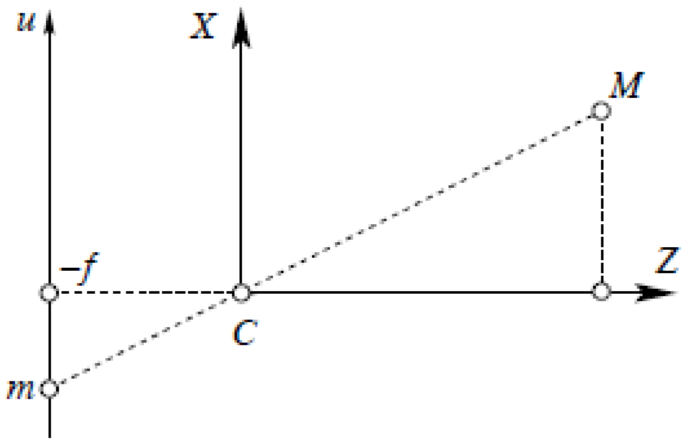


3D point cloud

Homework2

2. Create a 3 coud of points from a range image

Suggestions: z is encoded in the range image, find the x and y coordinates from the projection equations



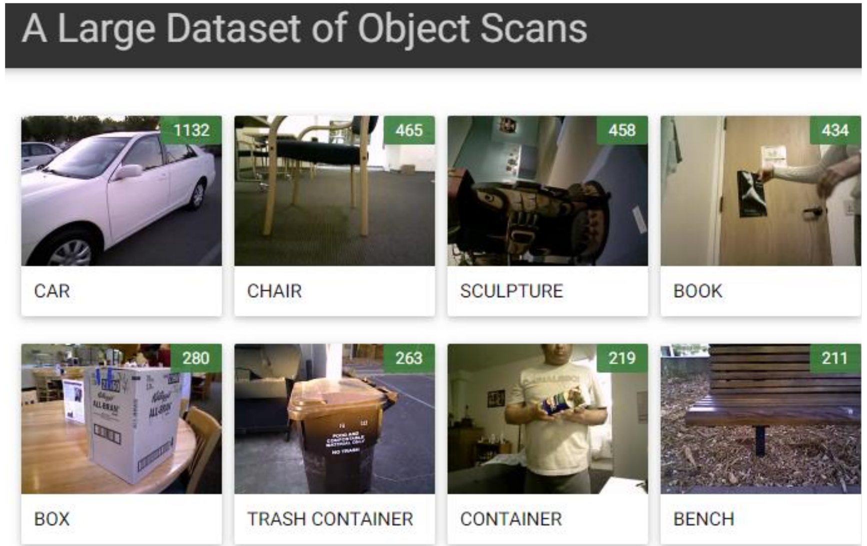
$$\begin{cases} u = k_u \dfrac{-f}{z} x + u_0 \\ v = k_v \dfrac{-f}{z} y + v_0 \quad , \end{cases}$$

The global reference system is on the camera, therefore only the main intrinsic parameters are required: (u_0, v_0) and $f_u = fk_u$, $f_v = fk_v$

Homework2

2. Create a 3 coud of points from a range image

E.g.: take some samples from http://redwood-data.org/3dscan/



The RGB-D sequences were acquired with **PrimeSense Carmine** cameras
The focal length is 525 for both axes (f_u=525, f_v=525) and the principal point is (u_0 =319.5, v_0=239.5).

Homework2

Export the cloud of point with this script and visualize it with Meshlab

```
function exportMeshToPly(vertices, faces, vertex_color, name)
if(max(max(vertex_color))<= 1.0)
   vertex_color = vertex_color.*256;
end
if(size(vertex_color,2) == 1)
    vertex_color = repmat(vertex_color,1,3);
end
vertex_color = uint8(vertex_color);
fidply = fopen([name '.ply'],'w');

fprintf(fidply, 'ply\n');
fprintf(fidply, 'format ascii 1.0\n');
fprintf(fidply, 'element vertex %d\n', size(vertices,1));
fprintf(fidply, 'property float x\n');
fprintf(fidply, 'property float y\n');
fprintf(fidply, 'property float z\n');
fprintf(fidply, 'property uchar red\n');
fprintf(fidply, 'property uchar green\n');
fprintf(fidply, 'property uchar blue\n');
fprintf(fidply, 'element face %d\n', size(faces,1));
fprintf(fidply, 'property list uchar int vertex_index\n');
fprintf(fidply, 'end_header\n');

for i=1:size(vertices,1)
      fprintf(fidply, '%f %f %f %d %d %d\n',vertices(i,1), vertices(i,2), vertices(i,3), vertex_color(i,1), vertex_color(i,2), vertex_color(i,3));
end

for i=1:size(faces,1)
    fprintf(fidply, '3 %d %d %d\n',faces(i,1)-1, faces(i,2)-1, faces(i,3)-1);
end

fclose(fidply);


end
```

Homework2

- Try other public available datasets like:

https://rgbd-dataset.cs.washington.edu/index.html

RGB-D Object Dataset

Home | People | Dataset | Results | Software | Demos

News

- April 5, 2014 - The RGB-D Scenes Dataset v.2 is now available here! It contains 14 new scenes reconstructed from RGB-D videos with furniture and tabletop objects, as well as Trimble 3D Warehouse objects use
- July 14, 2013 - Code for HMP features now available here. It achieves state-of-the-art results on the RGB-D Object Dataset!
- December 13, 2012 - Software and data for detection-based object labeling in Kinect videos now available here.
- October 3, 2012 - The dataset is now available for download directly from the website! No more sending emails necessary (questions and suggestions are, of course, still welcomed!).
- April 6, 2012 - RGB-D kernel descriptors are now available.
- March 22, 2012 - 3D reconstructions created by aligning video frames of all 8 scenes in the RGB-D Scenes Dataset are now available.
- June 20, 2011 - Pose annotations for all 300 objects in the RGB-D Object Dataset are now available.

Overview

The RGB-D Object Dataset is a large dataset of 300 common household objects. The objects are organized into 51 categories arranged using WordNet hypernym-hyponym relationships (similar to ImageNet). This datas and aligned 640x480 RGB and depth images at 30 Hz. Each object was placed on a turntable and video sequences were captured for one whole rotation. For each object, there are 3 video sequences, each recorded with t different angles with the horizon.

Unlike many existing datasets,such as Caltech 101 and ImageNet, objects in this dataset are organized into both *categories* and *instances*. In these datasets, the class *dog* contains images from many different dogs and the RGB-D Object Dataset the category *soda can* is divided into physically unique instances like *Pepsi Can* and *Mountain Dew Can*. The dataset also provides ground truth pose information for all 300 objects.

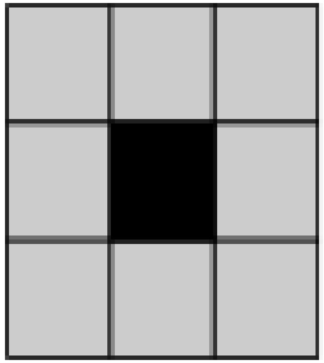
Here are some example objects that have been segmented from the background.

Use (and understand) the available code !

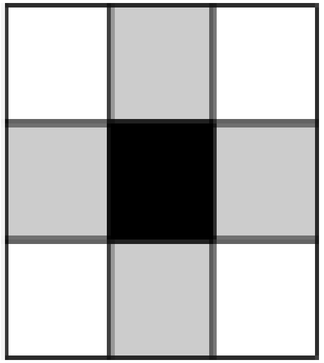
Homework 3

Mesh reconstruction from **range image**

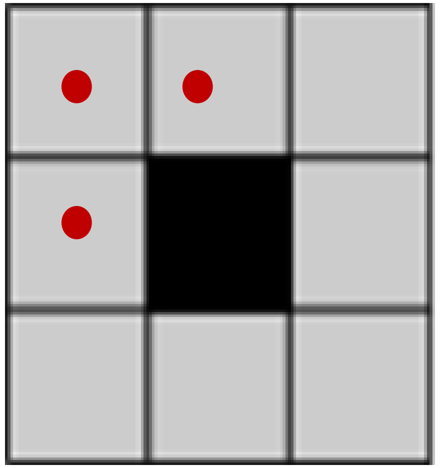
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



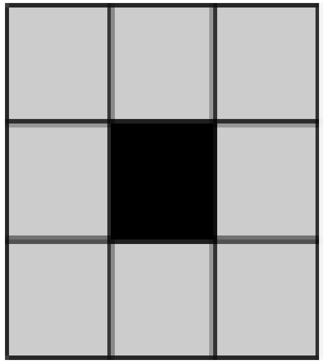
4-neigh



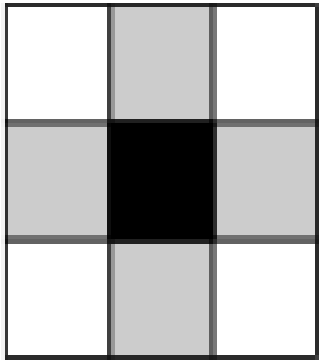
Homework 3

Mesh reconstruction from **range image**

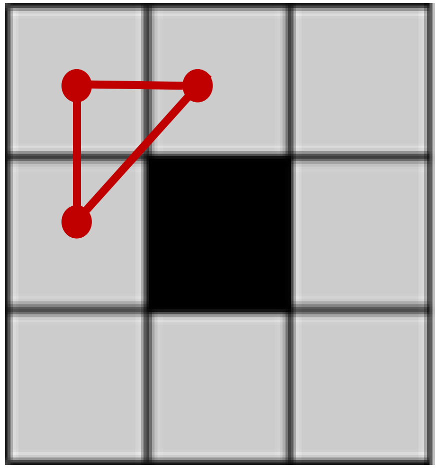
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



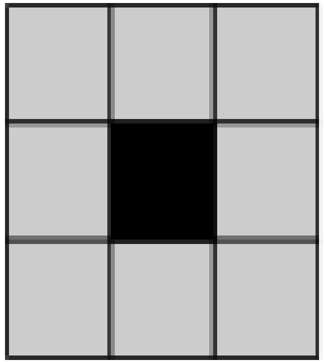
4-neigh



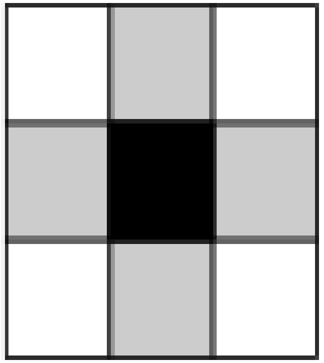
Homework 3

Mesh reconstruction from **range image**

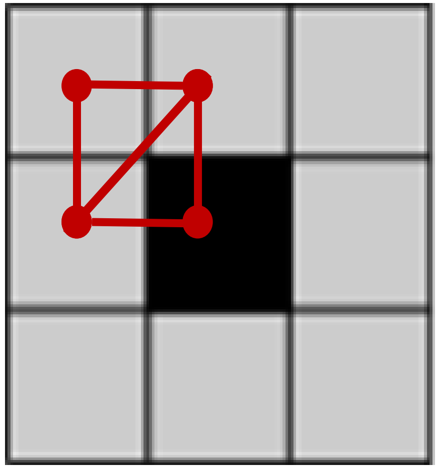
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



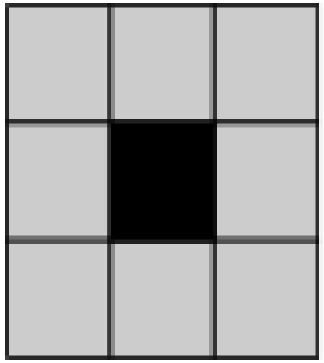
4-neigh



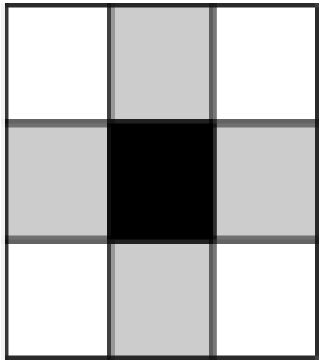
Homework 3

Mesh reconstruction from **range image**

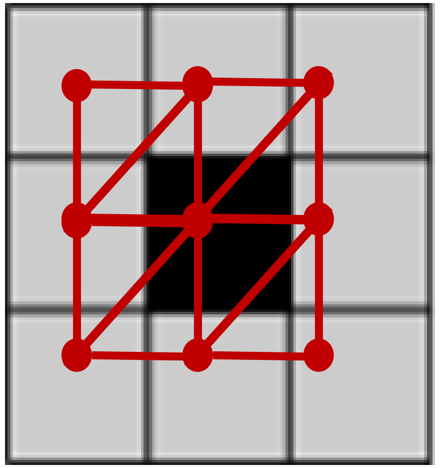
- **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



8-neigh



4-neigh



Homework 3

- Points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood,

BUT…

1) Not all the pixels on the range image are the projection of a point on the 3D space!

➡ a binary mask can be used to define valid points,

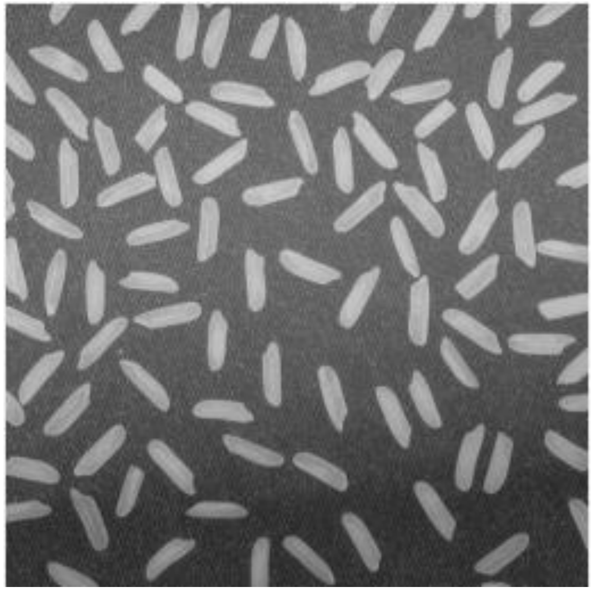
2) (optional) Nearby pixels should not correspond to nearby points on the 3D space!

➡ a robust strategy can be used to remove long edges.

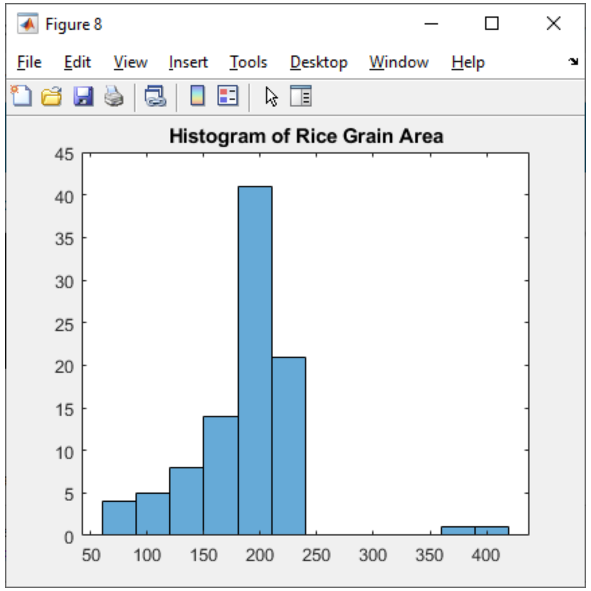
Ecersise 1

- **Run the Matlab demo available on image analysis:**

`https://it.mathworks.com/help/images/correcting-nonuniform-illumination.html`



Input



Output

Homework1

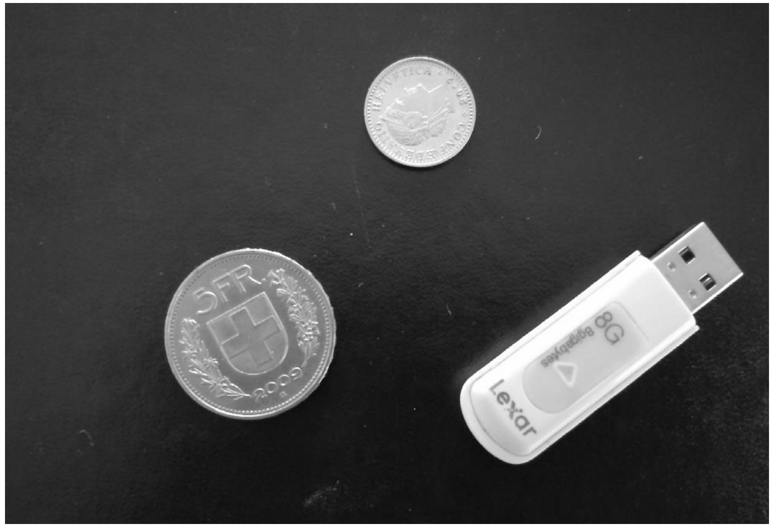
- Try to obtain the same results with coins image:

```
I = imread('eight.tif');
figure(1);
imshow(I);
```

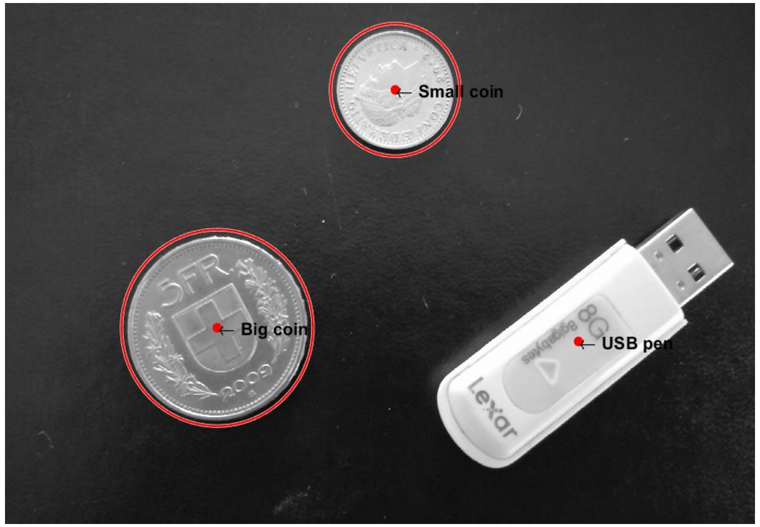
Suggestion: change the parameters and the combination of morphological operator to obtain a reliable binary image.

Homework2

- Use moprhological operators and region properties to infer the following information from this image:



Input image

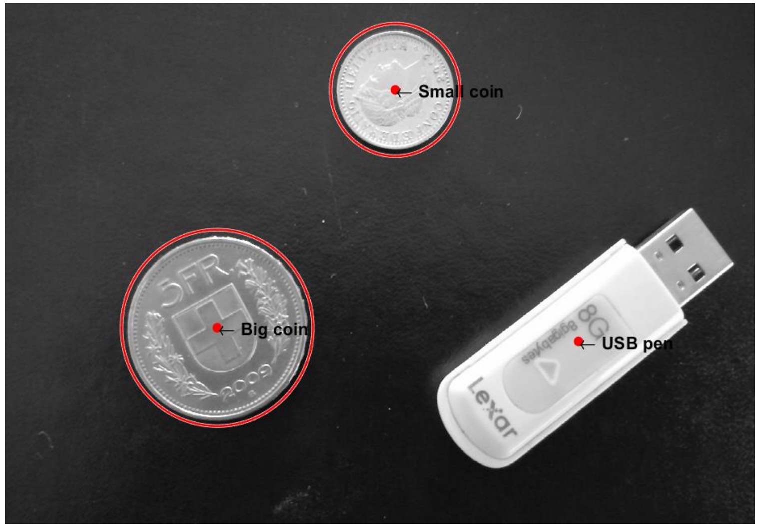


Output analysis

Homework3

- Use moprhological operators and region properties to infer the following information from a given image:

Suggestion: exploit the funcion '`regionprops`' to detect circular objects and compute the size of regions.



Output analysis

Homework4

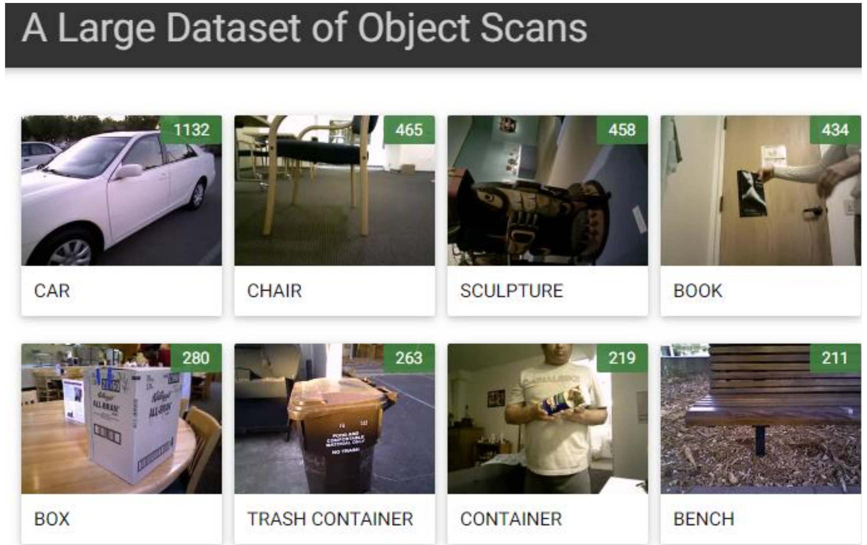
- You decide the scene and take a picture of that. You should take object with different characteristics and implement the right combination of morhological operations and region propertis to recognize and localize them.

Homework

- Study and implement the following methods:
  - 3D Plane fitting,
  - 3D Point-to-plane distance computation,
  - 3D Point-to-plane projection,
  - 3D Line fitting,
  - 3D Point-to-line projection,
  - Angle between two 3D lines,
  - Two lines (3D) intersection,
  - Robust line fitting using RANSAC.

Homework

- Implement the proposed 3D analysis pipeline to range images from `http://redwood-data.org/3dscan/`



Try to select images with a planar rectangular shaped object

Homework on Hand-eye calibration

- Use this code to evaluate the generic hand-eye calibration (i.e. case 1):

https://github.com/ZacharyTaylor/Camera-to-Arm-Calibration

  - (The most important part is the 'ProjectError' function).

- Use the code available on modle to evaluate the Tsai method for hand-eye calibration.
  - Modify the script to plot all the elements (i.e., robot base, gripper positions, camera positions, and calibration object) to the reference system of the calibration object.