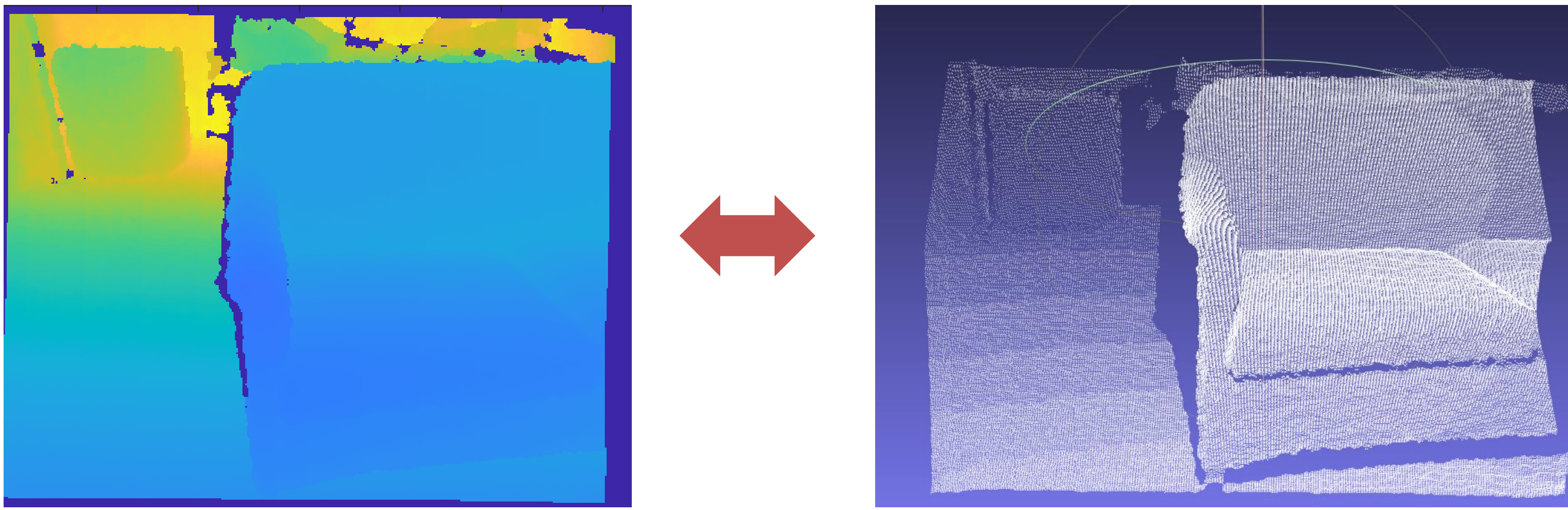
3D Analysis

Umberto Castellani
Robotics, vision and control

3D analysis pipeline

- Data structure,
- Range image analysis,
- Point cloud analysis,
- Primitive fitting and analysis

Data structure



Range image **3D point cloud**

Key aspect: the image structure (i.e., the connectivity) is usefull to process the cloud of point

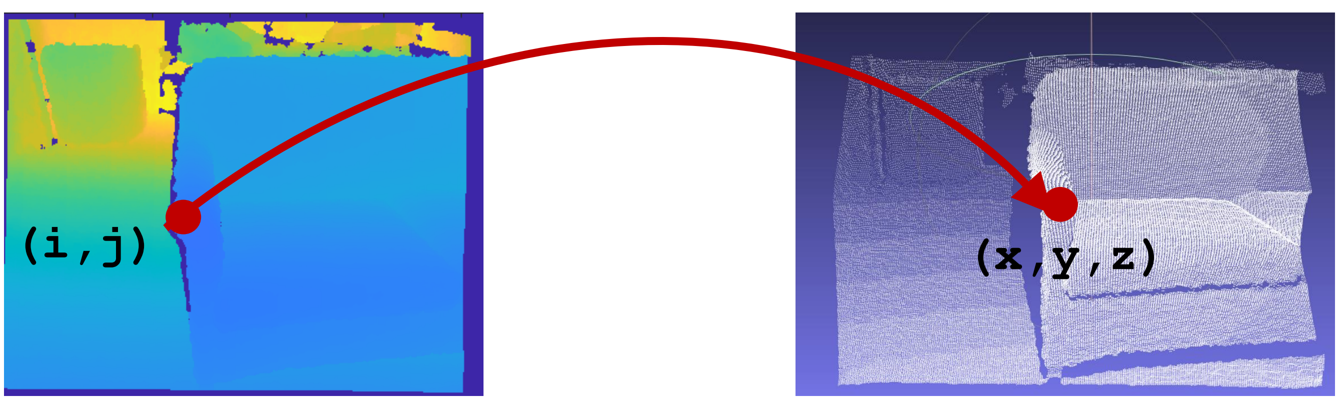
Data structure

- Structure to move from range image to point-cloud:

```
3DPoint(i,j)=[x,y,z]
```

Or…

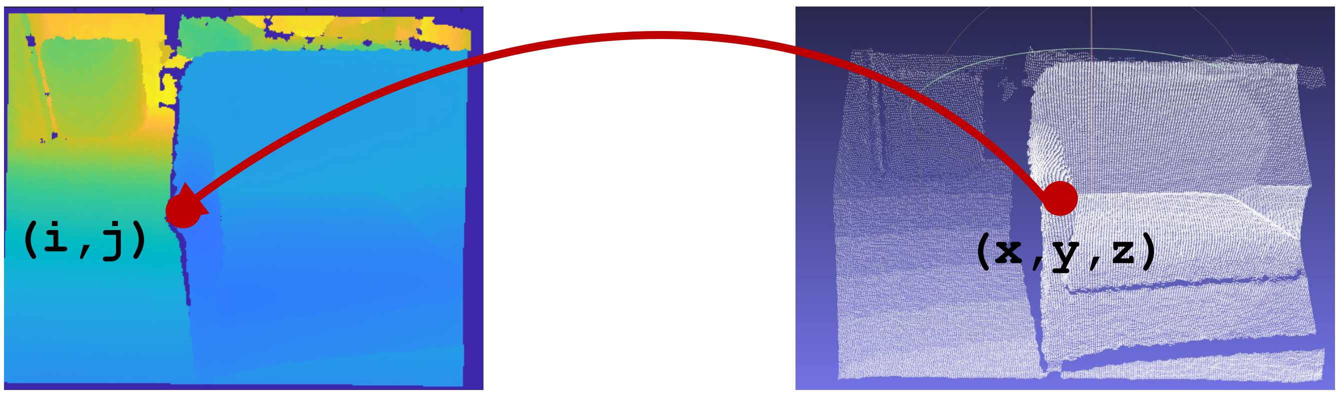
```
X(i,j)=x
Y(i,j)=y
Z(i,j)=z
```



Data structure

- Structure to move point-cloud to range image:

```
3DPoint(k)=[x,y,z,i,j]
```



Switch domain functions

- It is possible to set some **functions**:

```
[i,j]=point2range(x,y,z)
```

Use projecton equations

```
[x,y,z]=range2point(i,j)
```

Use the given range image z=Z(i,j) and then invert the projection equations for x and y

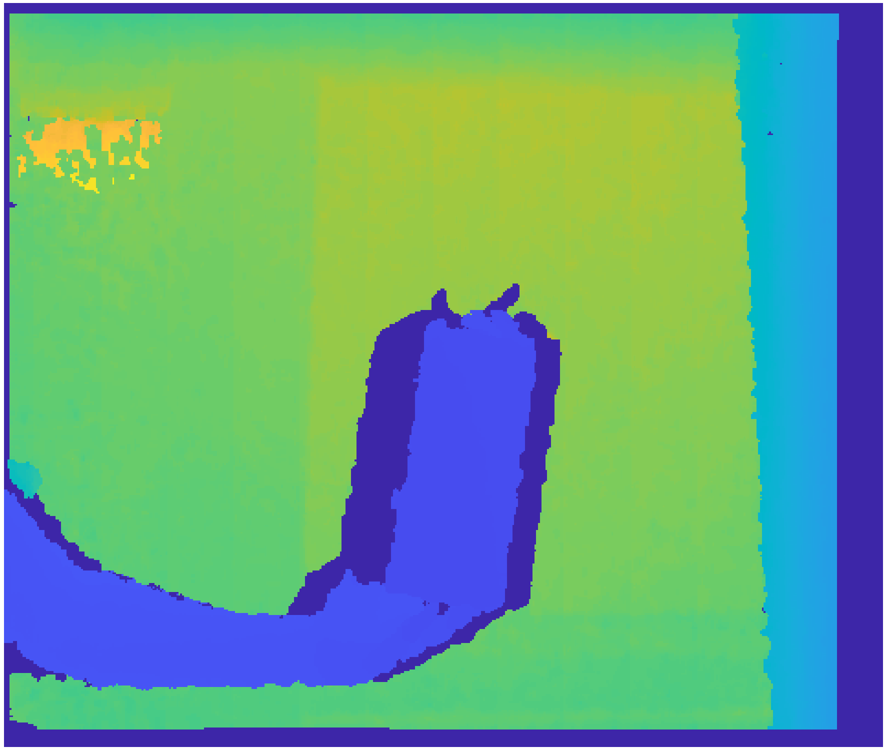
Range data analysis

NOTE: on the rage image is possible to employ standard image processing techniques!

1. Depth-based image thresholding,
2. Image binarizaton,
3. Boundary extraction,
4. Region charaterization,

Depth thresholding

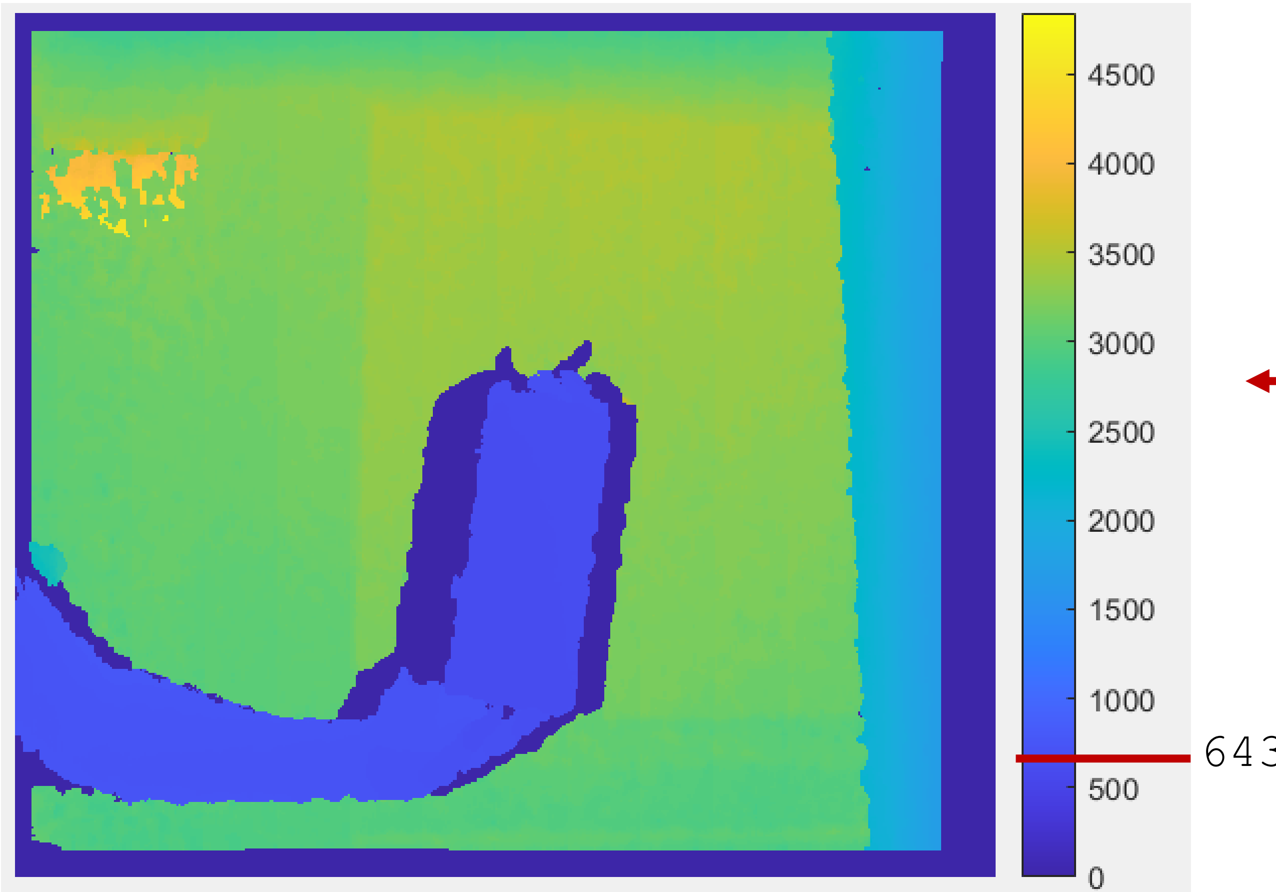
Color image



Range image

We want the closest region, suggestions?

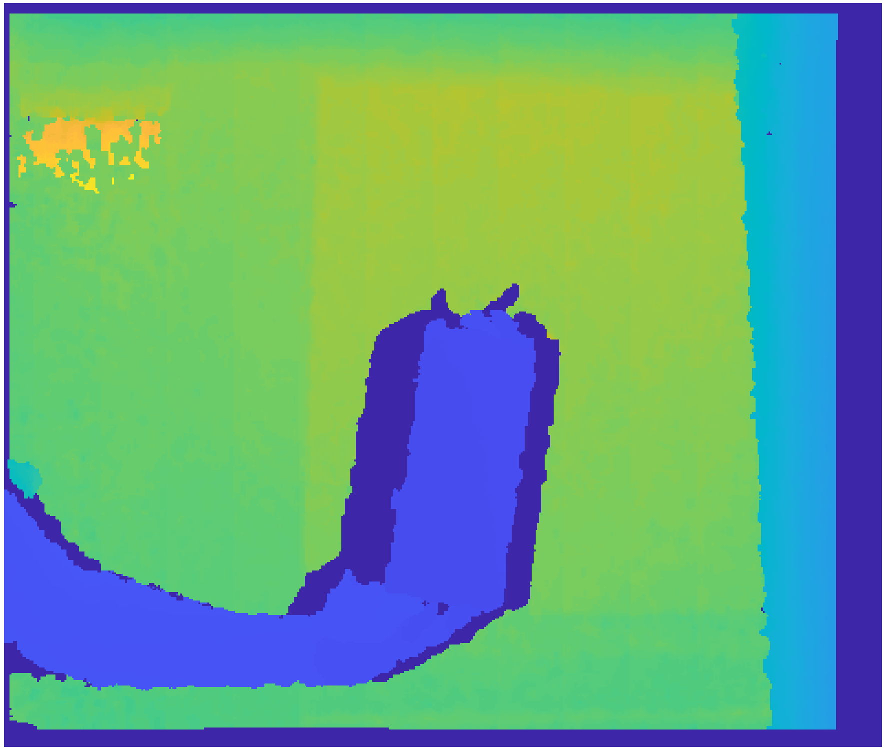
Depth thresholding



Check the depth range

A reasonable choice is `DephTH=643`

Depth thresholding

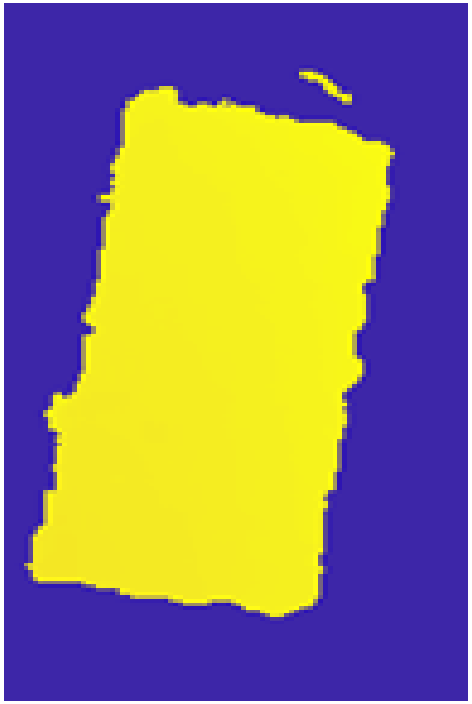


Range image

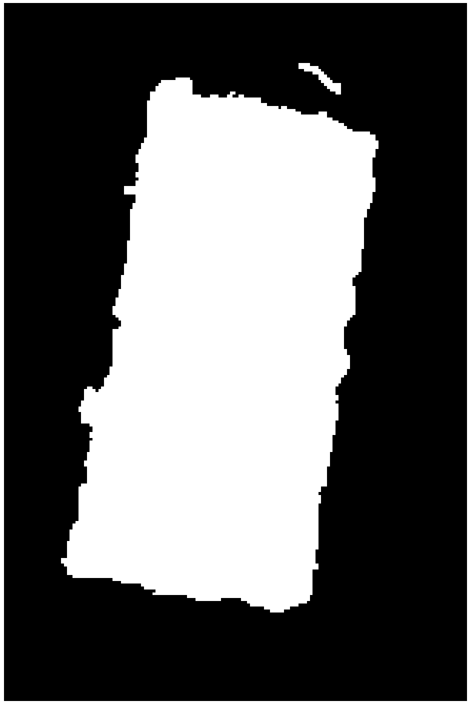
Extracted region

Note that the region is still not a binary image

Image binarization

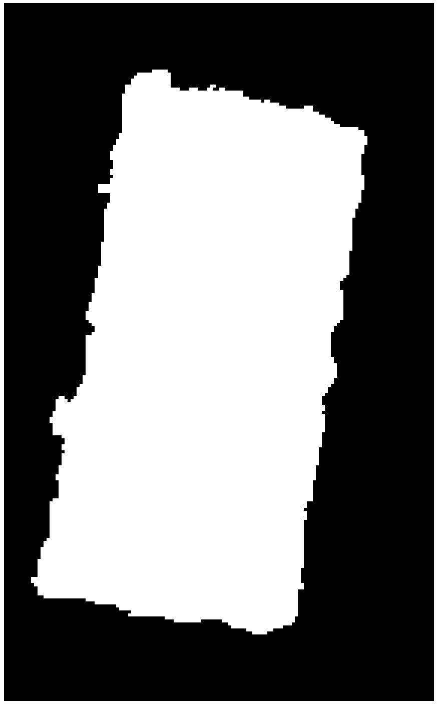


Region of the range image



Region binarization

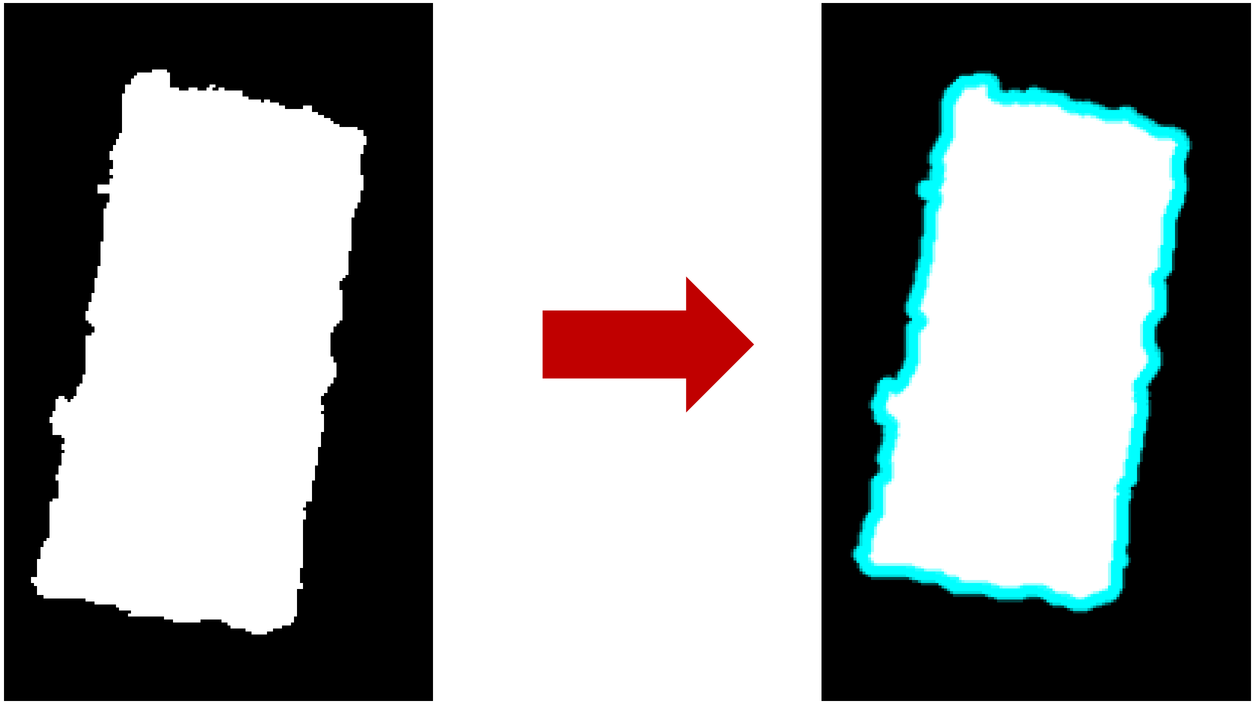
[BW = imbinarize(I)]



Small unconnected region removal

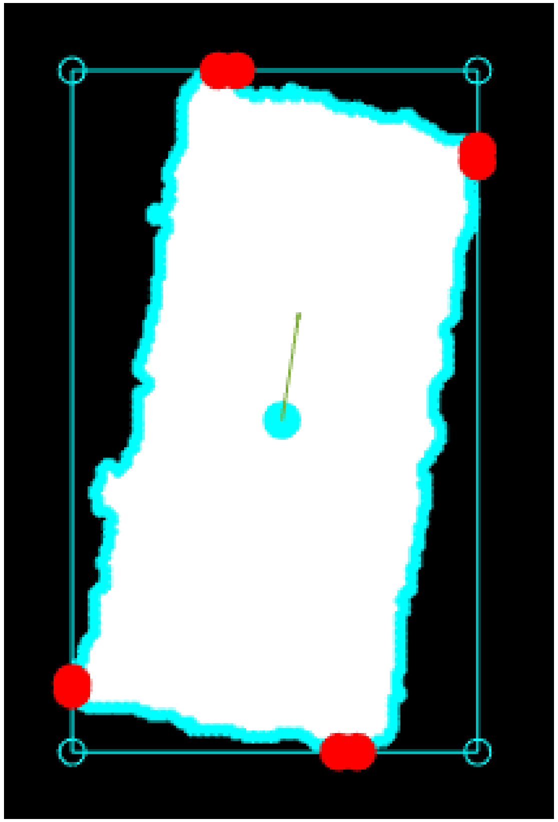
[BW2 = bwareaopen(BW,P)]

Boundary extraction



imcontour(I)

Region characterization



- -Centroid,
- -Bounding Box,
- -Orientation α,
- -Extrema

NOTE: to obtain the orientation vector from the angle we should compute:

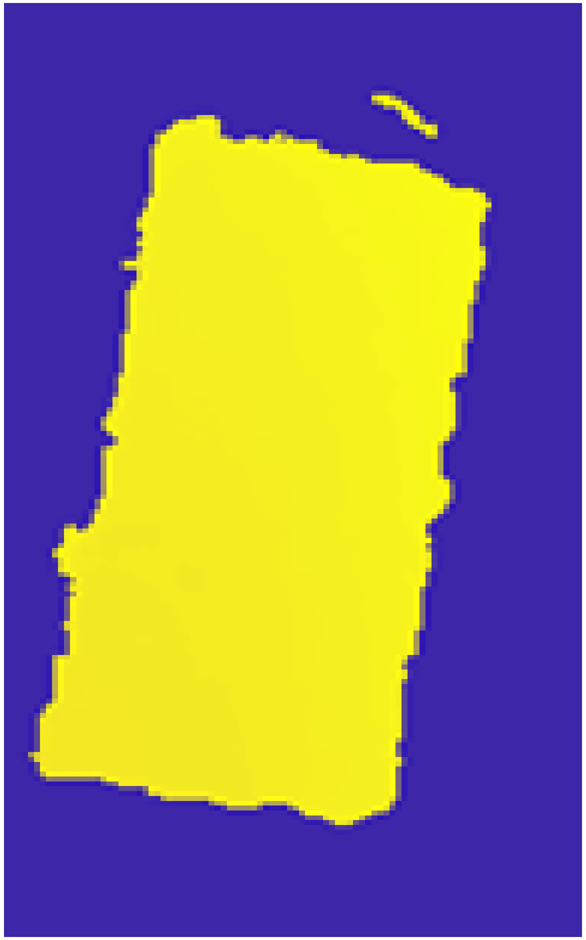
$$n_i = \cos(\alpha)$$
$$n_j = -\sin(\alpha)$$

See function [regionprops(BW,properties)]()

Point cloud analysis

- Object point cloud,
- Point cloud boundaries,
- Centroid,
- Object orientation

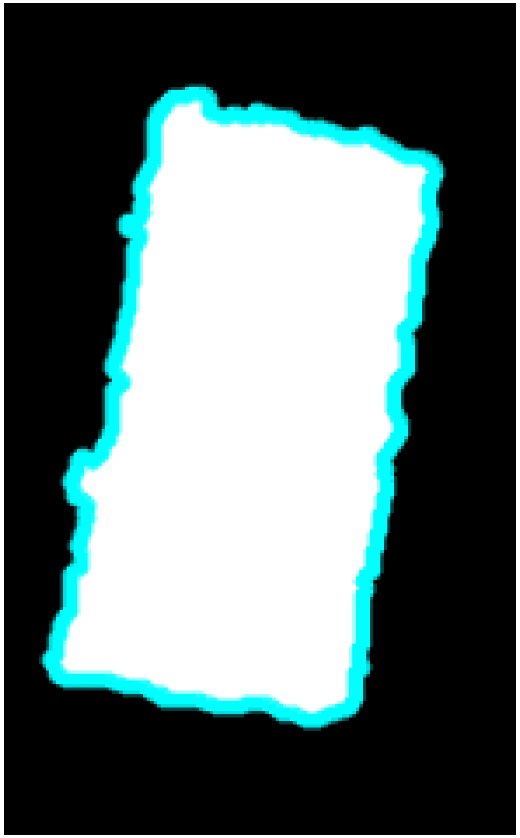
Object point cloud

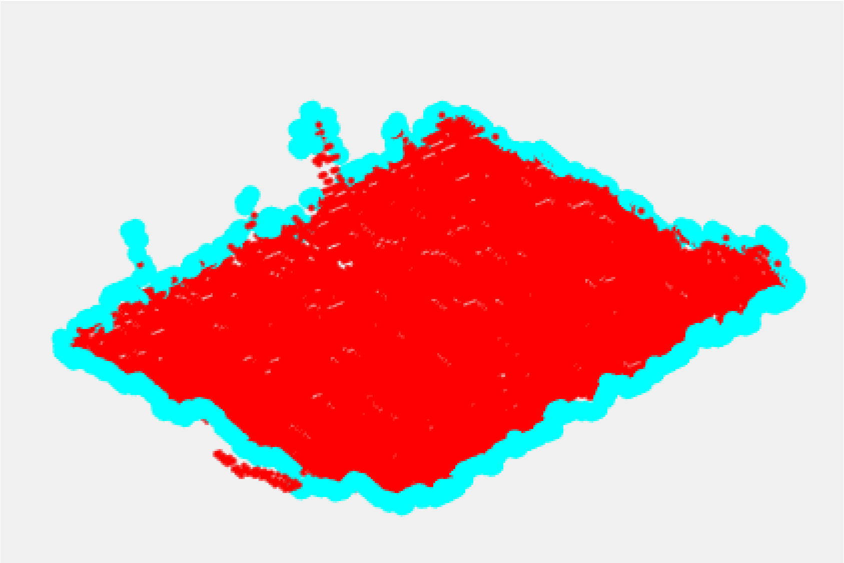


Region of the range image

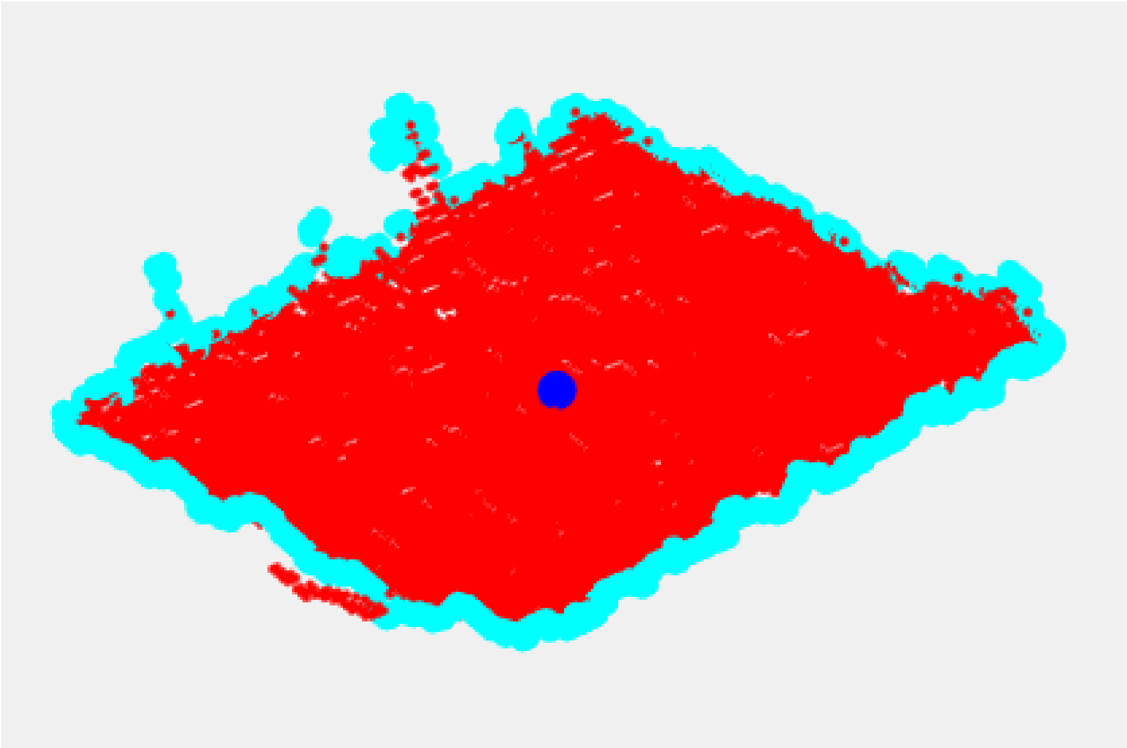
3D cloud point

2D to 3D boundary



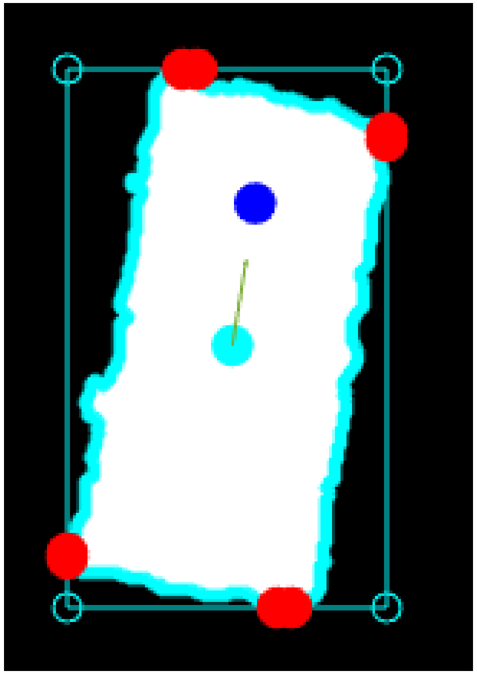


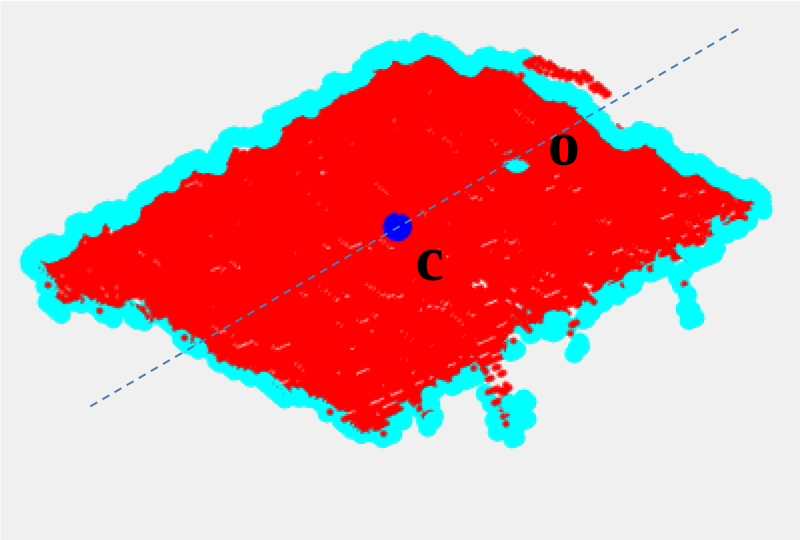
Centroid



Here we compute the **centroid** from 3D coordinates

Object orientation

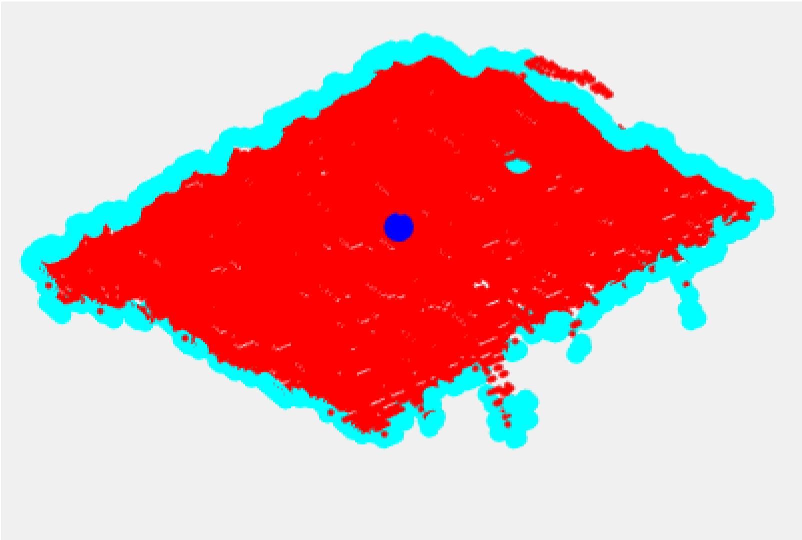


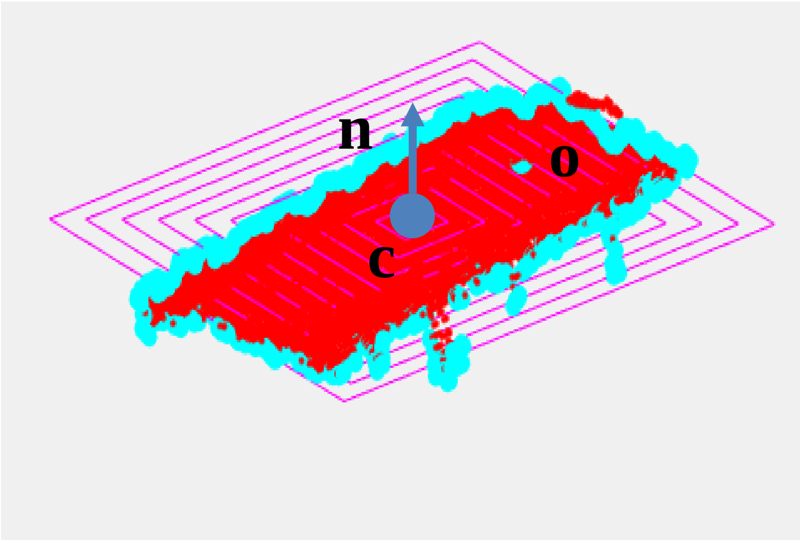


We pick a pixel along the main orientation (the blue point)

We project that pixel to the 3D space obtaing the point **o** and connect it with the centroid **c**

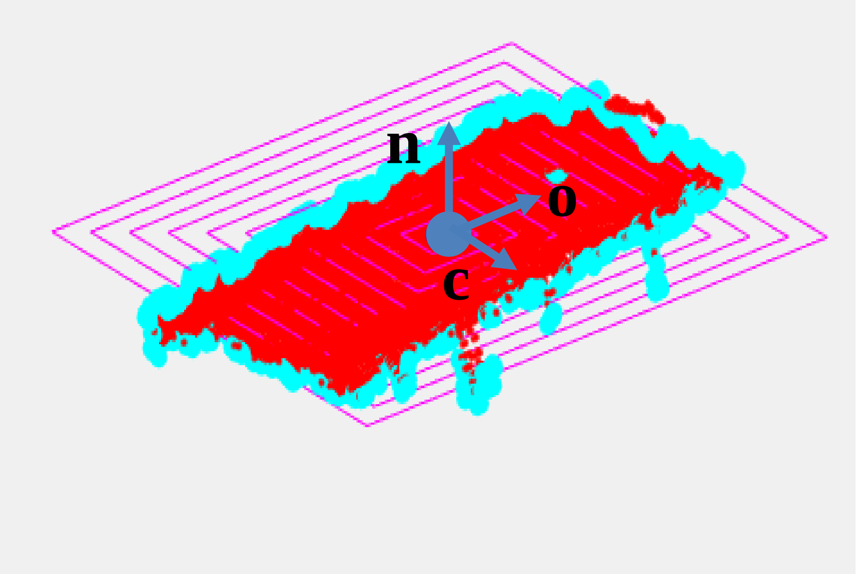
Plane fitting





From plane fitting we get the **surface normal n**

Object orientation



- We can define the local frame for the object as

n, (co), n×(co)

3D line fitting

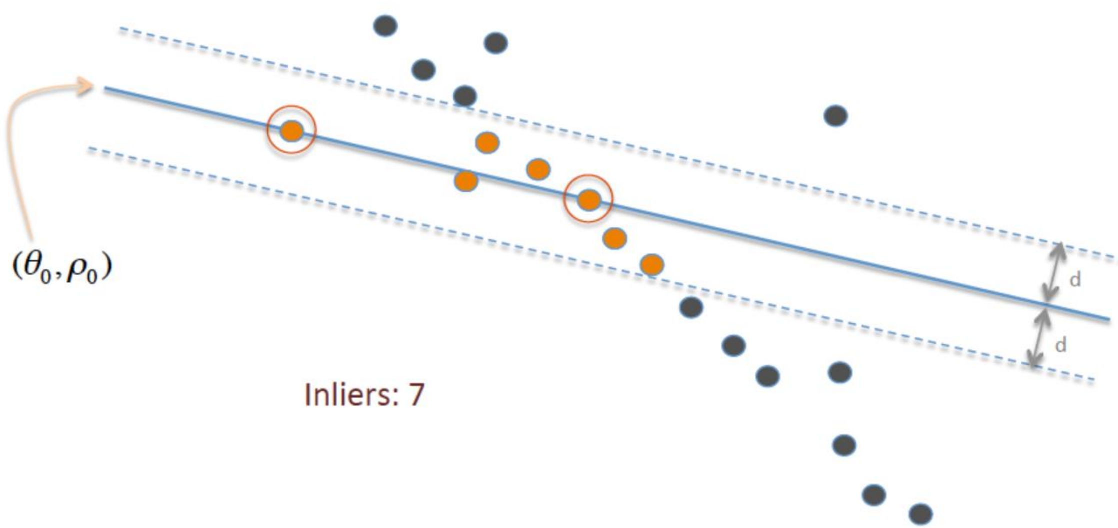
- We need to extract 4 lines,
- We need to adopt a robust approach

We can use RANSAC iteratively!

1. Fit one line,
2. Discard inlier and keep outliers,
3. GOTO 1 (4 times)

Ransac method

- **Random sample consensus** (**RANSAC**) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates.



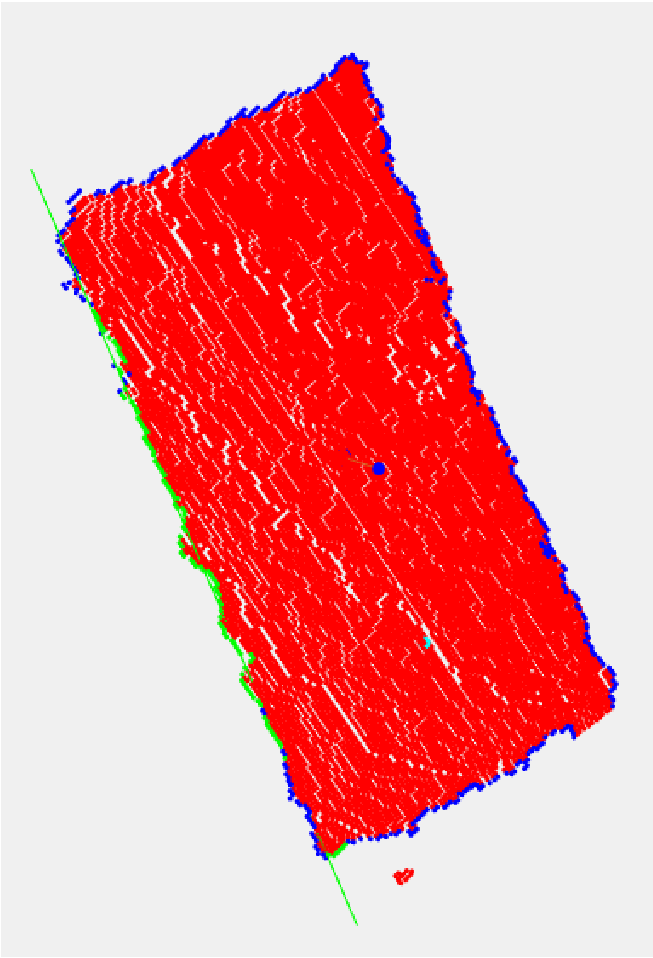
https://en.wikipedia.org/wiki/Random_sample_consensus

Ransac algorithm

```
Given:
    data – A set of observations.
    model – A model to explain observed data points.
    n – Minimum number of data points required to estimate model parameters.
    k – Maximum number of iterations allowed in the algorithm.
    t – Threshold value to determine data points that are fit well by model.
    d – Number of close data points required to assert that a model fits well to data.

Return:
    bestFit – model parameters which best fit the data (or null if no good model is found)

iterations = 0
bestFit = null
bestErr = something really large

while iterations < k do
    maybeInliers := n randomly selected values from data
    maybeModel := model parameters fitted to maybeInliers
    alsoInliers := empty set
    for every point in data not in maybeInliers do
        if point fits maybeModel with an error smaller than t
             add point to alsoInliers
    end for
    if the number of elements in alsoInliers is > d then
        // This implies that we may have found a good model
        // now test how good it is.
        betterModel := model parameters fitted to all points in maybeInliers and alsoInliers
        thisErr := a measure of how well betterModel fits these points
        if thisErr < bestErr then
            bestFit := betterModel
            bestErr := thisErr
        end if
    end if
    increment iterations
end while

return bestFit
```

https://en.wikipedia.org/wiki/Random_sample_consensus

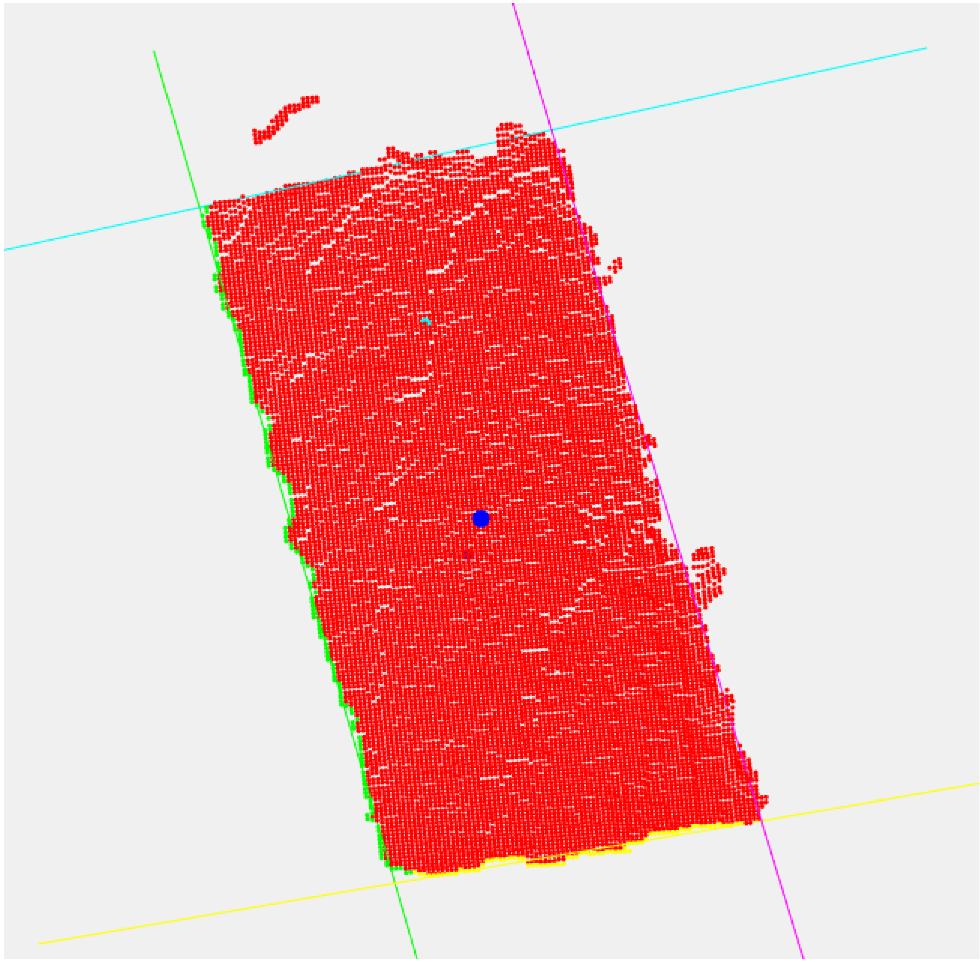
3D line fitting



First line

°Inlier

°Outlier



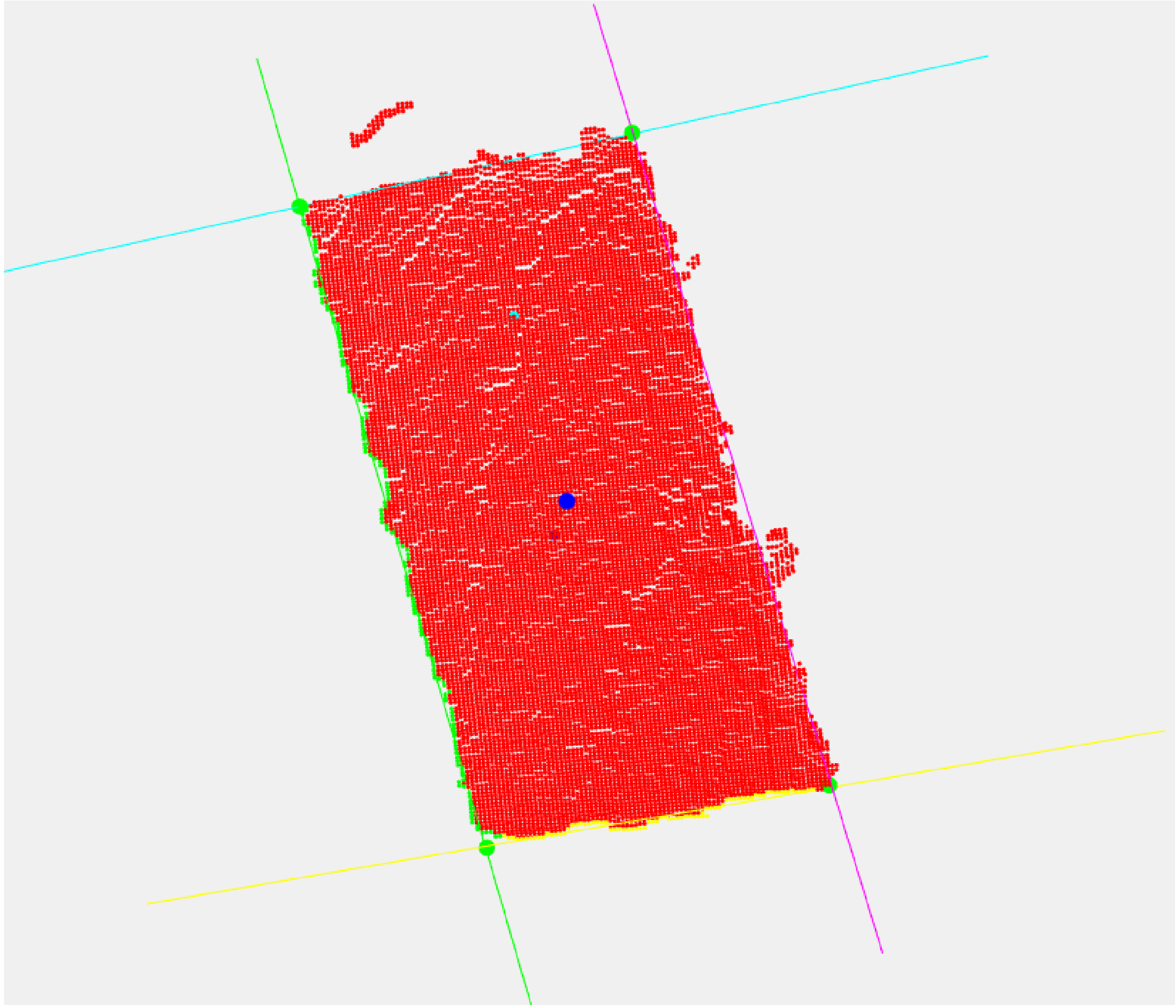
Four line

Line pairing

- We need to analyse the line orientation to avoid the intersection between parallel line

```
For all pair lines(l_i,l_j)
      if(not(l_i||l_j))
            compute inter(l_i,l_j)
      End
end
```

Line intersection



We get 4 interesting 3D point to pick the object!

highlights

- Standard image processing techniques can be employed to range images to detect and analyse 3D regions,
- Combination of 3D features and 3D primitives enable us to estimate the object location and orientation.

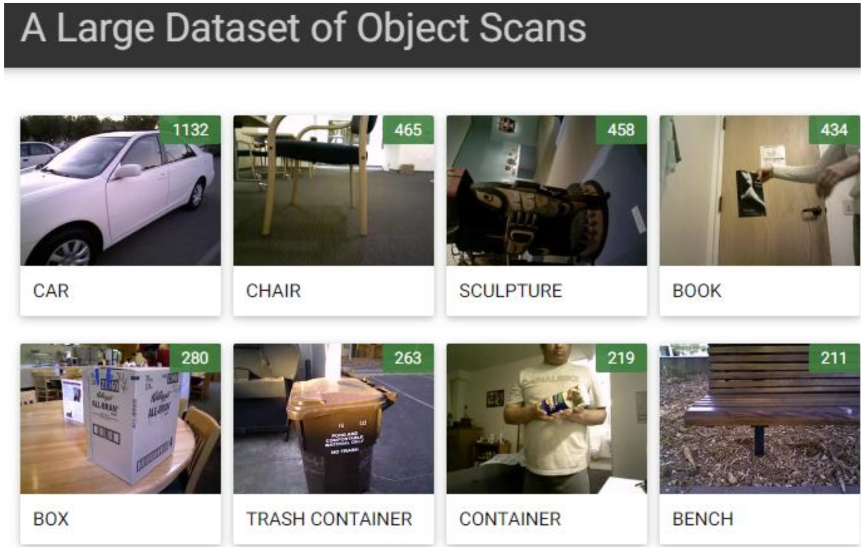
Next step: find object location and orientation in the robot reference system, we need **hand-eye calibration**!

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