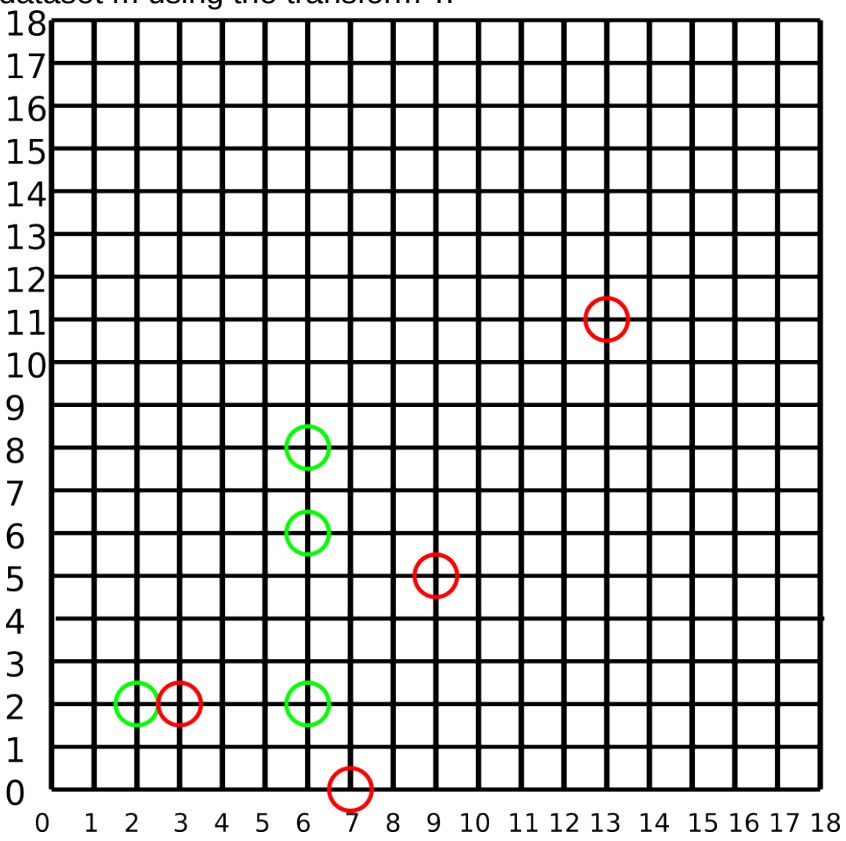
Exercise 1: Iterative Closest Point (ICP) Algorithm

In this exercise you will use a standard ICP algorithm with the point-to-point distance metric to estimate the transform between the 2D datasets (model - red and target - green) depicted in the below figure. For the correspondence estimation please use the nearest neighbor search with the maximum radius set to 4 grid units. For the rotation estimation use an SVD library of your choice or calculate it yourself. Recall the main steps in the ICP algorithm:

- find point pairs;
- compute centroids
- build the correlation matrix H;
- estimate rotation matrix R using SVD;
- estimate the translation vector t;
- calculate the transform T;
- transform dataset m using the transform T.



Solution:

The solutions will be following this tutorial:
http://webcourse.cs.technion.ac.il/236329/Spring2011/ho/WCFiles/10%20-%20Registration.pdf

1. Find point pairs: Since maximum radius for the nearest neighbor search is set to 4 grid units the following point pair becomes an outlier ($_m$ - model, $_t$ - target): P_{4t} = [6, 8] and P_{4m} = [13, 11] and we do **not** consider it anymore in the further calculations.

2. Let us calculate the centroid point for the target point cloud ($_c$ – centroid, $_m$ - model):
   P_{ct} = ([2,2] + [6,2] + [6,6])/3 = [4.66, 3.33]
   In the same fashion the centroid point for the model dataset equals: P_{cm} = [6.33, 2.33]

3. The correlation matrix H describes how much both, the model and the target datasets correlate with respect to each other. From H we will then estimate the rotation that minimizes the distance between the two centered dataset using the singular value decomposition algorithm. For H we first have to center/de-mean (move to the origin of the coordinate system) both datasets, that is subtract the respective centroid from every point in the dataset ($_m$ - model, $_d$ – de-meaned):
   Target dataset:
   $P_{1td} = P_{1t} – P_{ct} = [2,2] – P_{ct} = [-2.66, -1.33]$
   $P_{2td} = P_{2t} – P_{ct} = [6,2] – P_{ct} = [1.34, -1.33]$
   $P_{3td} = P_{3t} – P_{ct} = [6,6] – P_{ct} = [1.34, 2.67]$
   Model dataset:
   $P_{1md} = P_{1m} – P_{cm} = [3,2] – P_{cm} = [-3.33, -0.33]$
   $P_{2md} = P_{2m} – P_{cm} = [7,0] – P_{cm} = [0.67, -2.33]$
   $P_{3md} = P_{3m} – P_{cm} = [9,5] – P_{cm} = [2.67, 2.67]$
   Then we build 2 matrices, M_{md} and M_{td} from the respective dataset with the dimensions of nxm where n denotes the dimensionality of the point (2 in our case) and m the number of points (3 in our case):
   M_{td} =
   [[-2.66,1.34,1.34],
   [-1.33,-1.33,2.67]]
   M_{md} =
   [[-3.33 ,0.67,2.67],
   [-0.33,-2.33,2.67]]
   Finally, H is computed as follows:
   $H = M_{md} * M_{td}^T$ =
   [[13.3334, 10.6667],
   [1.3334, 10.6667]]

4. Estimate rotation matrix R using SVD: For this we will use the numpy library and decompose the H matrix using an SVD algorithm:
   [U, D, V] = numpy.linalg.svd(H)
   Finally R can be computed as following:
   R = V * U.transpose() =
   matrix([[0.93200562, -0.36244382],
   [0.36244382, 0.93200562]])

5. Estimate the translation vector: **t** is defined as a difference between the centroids of the remaining inliers of both, model and target datasets (with the rotation matrix applied to the model one). The translation vector **t** is thus:
   $t = P_{ct} - R * P_{cm} = [-0.3903, -1.1368]$

6. Build the final transformation matrix T (it will have 3x3 dimension in our case):
   T =
   [R t'
   0 1]

7. Transform dataset m into dataset t using the transform T (add a 1 as the last element to the coordinate vectors):
   $t_i' = T * m_i'$

Exercise 2: Generate Scale Invariant Feature Transform (SIFT) Feature

In this exercise you will generate the SIFT histogram from the below window in blue). We will assume that the scale spaces were already computed and the keypoint already found (in red). Your task is a) to calculate gradient magnitudes and orientations according to the following formulas:

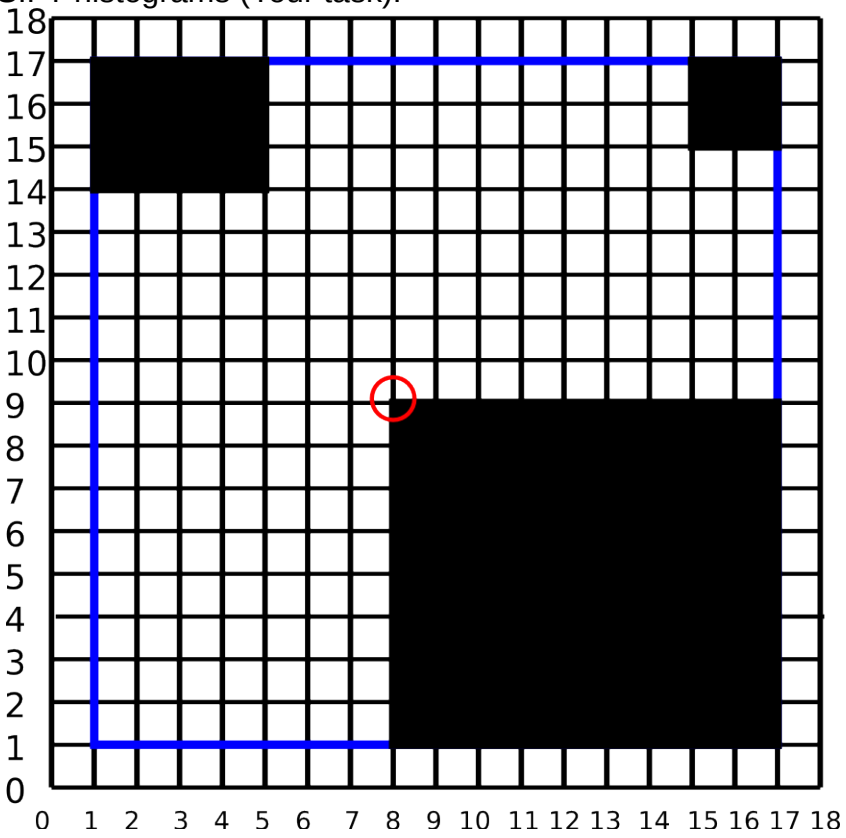
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

and b) to generate the SIFT histogram for the found keypoint (in red below). Please use the atan2() function for the gradient orientation.

Recall the main steps in SIFT estimation:

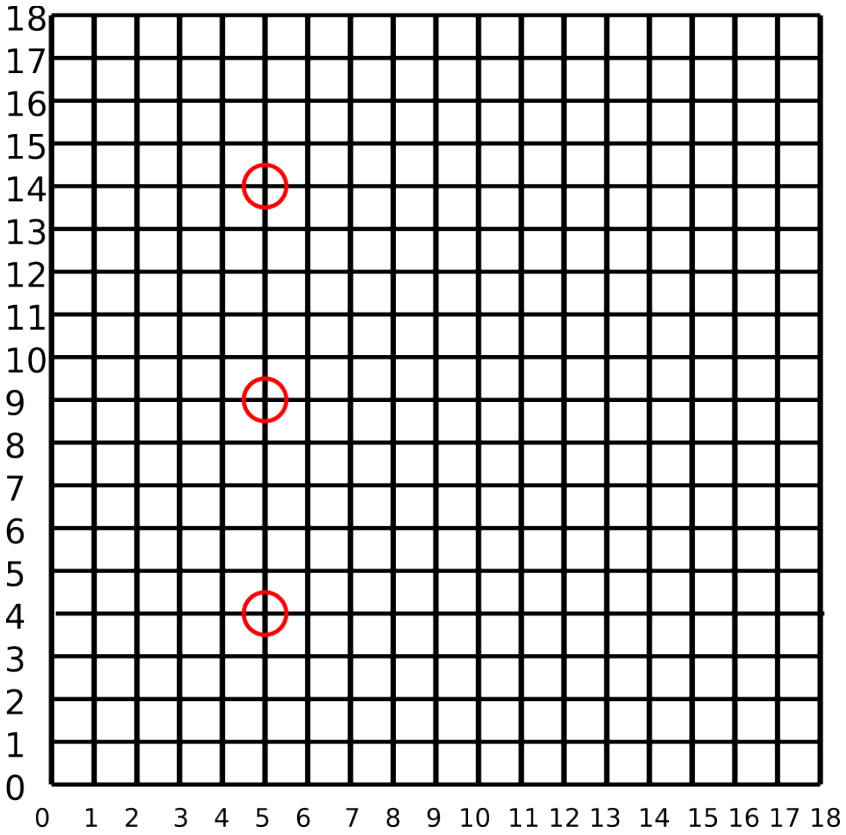
- Construct a scale space (Done);
- Laplacian of Gaussian approximation (Done);
- Find keypoints (Done);
- Eliminate edges and low contrast regions (Done);
- Assign Keypoint Orientation (Your task);
- Generate SIFT histograms (Your task).



Solution:

Exercise 3: Hough Transform

Consider below 3 points in the image and using the Hough Transform cast the votes for them in the parameter space for finding the lines with the 0, 45, 90 and 135 degrees inclination angle. Select an appropriate parameter space which shall be a) uniquely defined for the given model and b) as low dimensional as possible. Which parameter space would be particularly bad for this example? Plot the Hough space graph and prove that the vertical line is the best fit.



Solution:

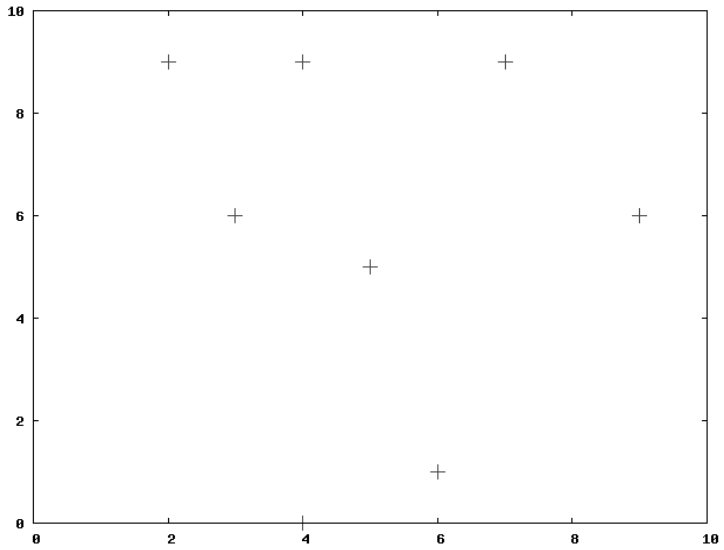
See: http://en.wikipedia.org/wiki/Hough_transform#Example. The only cell in the parameter space (http://en.wikipedia.org/wiki/Line_equation#Normal_form) receiving more votes is the vertical line (angle: 90, distance: 5), it receives one vote from each of the points.

Exercise 4: KD-Tree

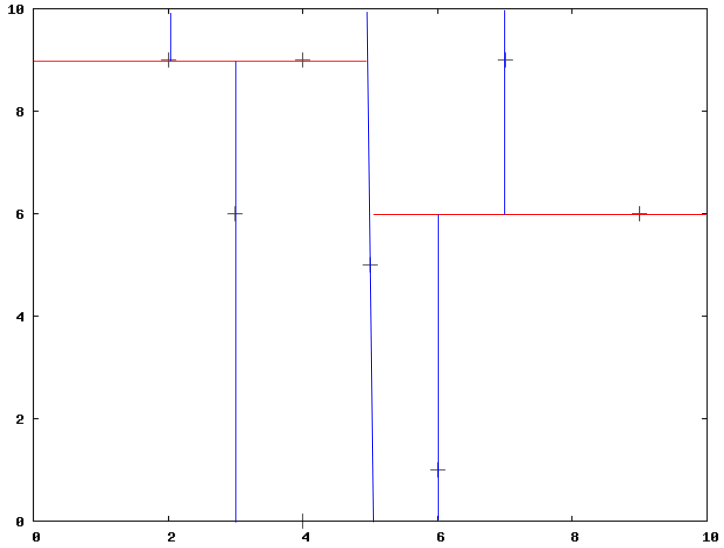
Using a KD Tree algorithm decompose the below 2D dataset of 8 points into 3 steps or until the bucket size of 1. Provide the KD Tree decomposition in the below figure and the resulting KD Tree graph.

For the kd-tree construction follow the following heuristics:

- Start with x axis and then alternate between x and y axes.
- Points are inserted by selecting the median of the points being put into the subtree, with respect to their coordinates in the axis being used to create the splitting plane.
- What is the complexity of the search in O notation?



Solution:



There are two solutions in the upper left corner (the same Y coordinate for the two points).

Since this is a binary tree which contains all points as separate nodes, level *m* contains 2^m nodes, with the total number of nodes being *n*. The total number of levels (l) satisfies then:

$\Sigma_{m=0}^{l-1} 2^m = 2^l - 1 < n \Rightarrow l < \log_2(n+1)$ which is the max number of steps for a search: O(log n).

Exercise 5: RANSAC

- How many points with normals need to be sampled to identify the equation of a cylinder?
- What is the effect of $P_{success}$ set to 1? What about having a thousand samples?

Solution:

a) 2 points with normals:

- each of them defines a line, which are potentially skew (see interactive 3D representation at: http://mathworld.wolfram.com/SkewLines.html)
- the closest points on these lines are on the axis of the cylinder, or if the lines a co-planar the intersection point and the axis direction can be identified as the cross product (vector product) of the two normals
- the radius of the cylinder is the average distance

b) The the number of iterations, can be determined as:

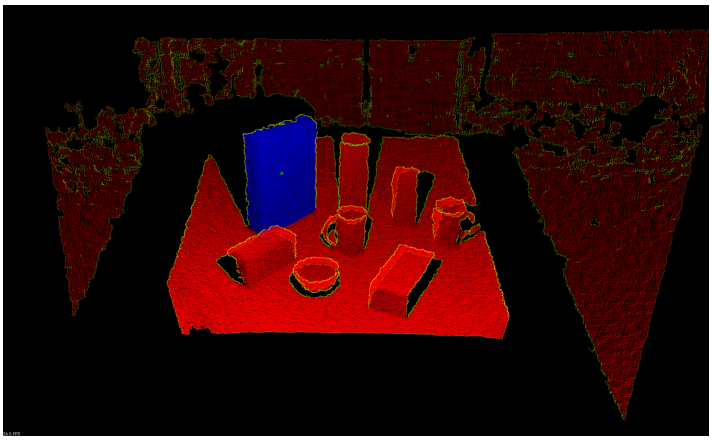
$$k = log(1 - P_{success}) / log(1 - P_{inlier}^{n})$$

in both cases k tends to infinity

Exercise 6: Region Growing

Given a point belonging to a convex object, P(x,y,z), segment this object from the scene using region growing.

- Check if a point belongs to the same convex object as the original point
- Don't grow from points that are near a boundary (given that you find a neighborhood that contains a point which does not satisfy convexity, make sure you don't grow further)
- Don't process the same point twice



Solution:
Complete pseudo code on slide no. 59. of http://ias.cs.tum.edu/~pangerci/04-perception.pdf with:

...in the inner loop

```
if ( qj  is processed )
        continue;
if ( qj is boundary )
        stop_growing =true;
        Ri ← Ri ∪ {qj }
        P (qj ) =true;
        break;
//check angle between vector and normals...see figure
// (points belonging to the object enclose an angle > 90^0)
if (isconvex)
        Ri ← Ri ∪ {qj }
        P (qj ) =true;
else
        break;
```

….at the end of the outer loop

```
if ( !stop_growing && q  has no boundary)
        add points in q as seed
```