컴퓨터 비전 세미나

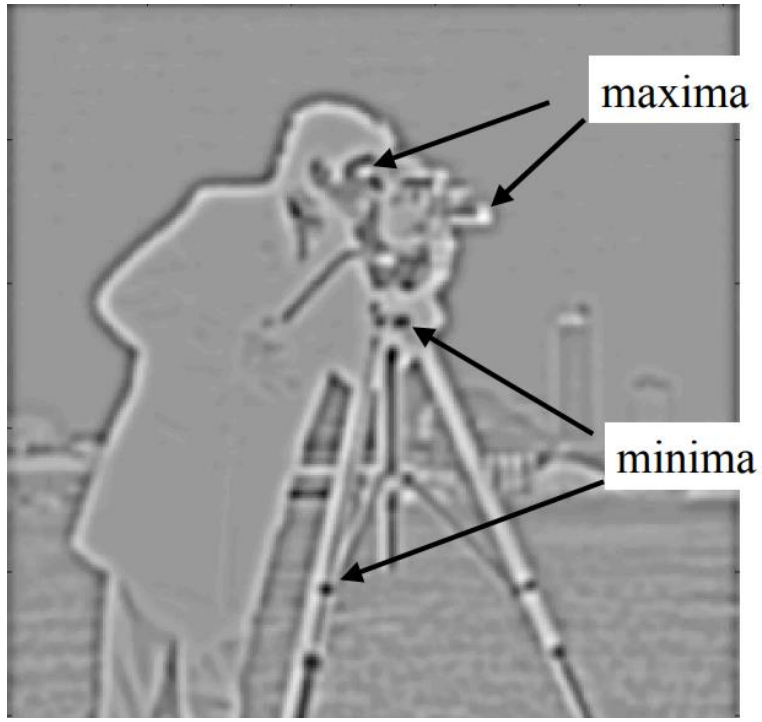
12191584 김창수

Review

Why do we use Laplacian filters?

- useful for finding edges and blobs

edges

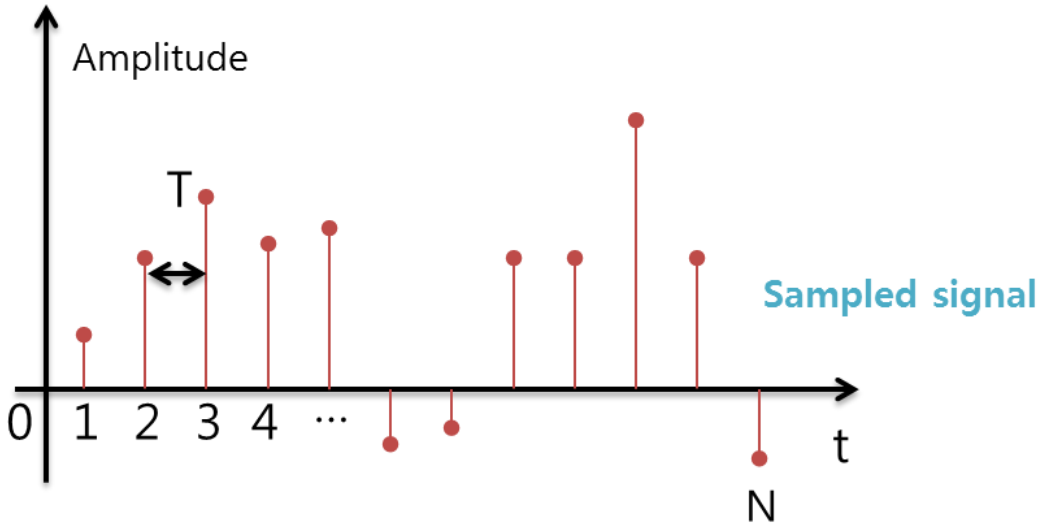


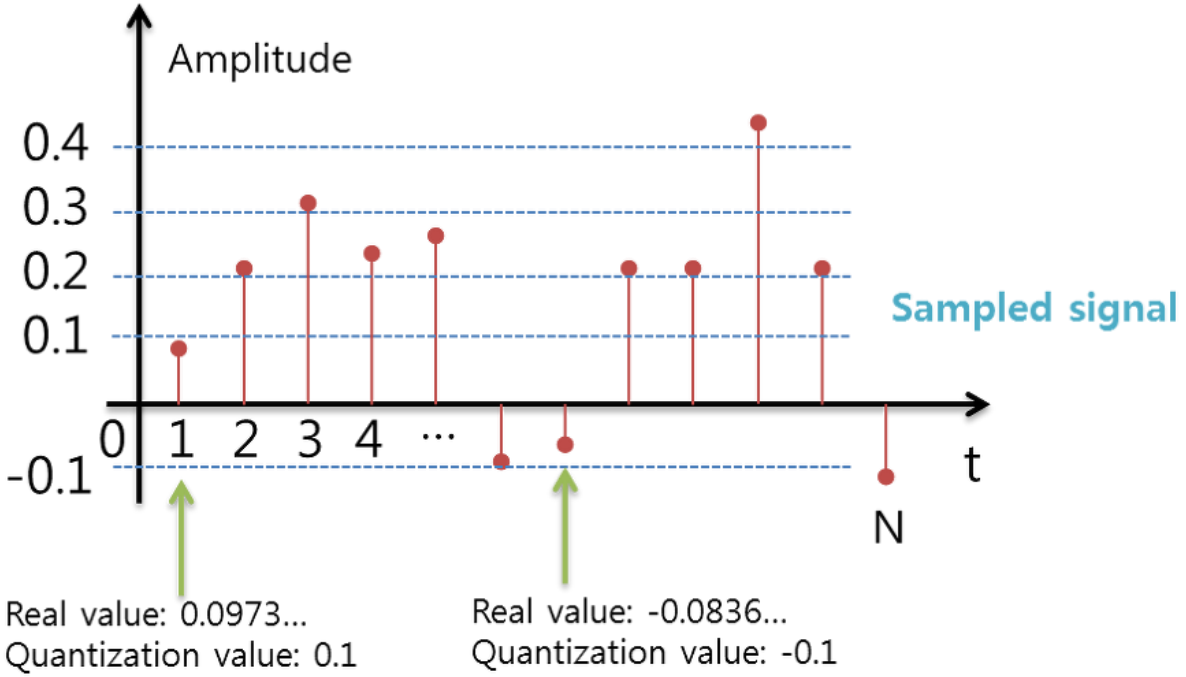
blobs

Review

Sampling & Quantization

- Sampling is the reduction of a continuous-time signal to a discrete-time signal
- Quantization is the process of mapping input values from a large set to output values in a smaller set, often with a finite number of elements, digitizing





Review

Why do we scale images?

- next 2 weeks

Index

Edge detector
- Canny edge
- Color edge

Lines
- Hough transforms
- RANSAC

Canny Edge Detection

Canny has found the general criteria for edge detection.

1. Low error rate: There should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points.
2. Good localization: The points marked as edge points by the operator should be as close as possible to the center of the true edge.
3. Minimal response: Only one detector response to a single edge.

The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.

Canny Edge Detection

Canny edge detection is a multi-step algorithm that can detect edges.

Process of Canny edge detection algorithm

1. Noise Reduction
2. Finding Intensity Gradient of the Image
3. Non-maximum Suppression
4. Hysteresis Thresholding

1. Noise Reduction

Since all edge detection results are easily affected by the noise in the image, it is essential to filter out the noise.

To smooth the image, a Gaussian filter kernel is convolved with the image.

It is important to understand that the selection of the size of the Gaussian kernel σ will affect the performance of the detector.

- The larger the size is, the lower the detector's sensitivity to noise.
- The larger the size is, the higher the localization error to detect the edge.

2. Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction and vertical direction.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

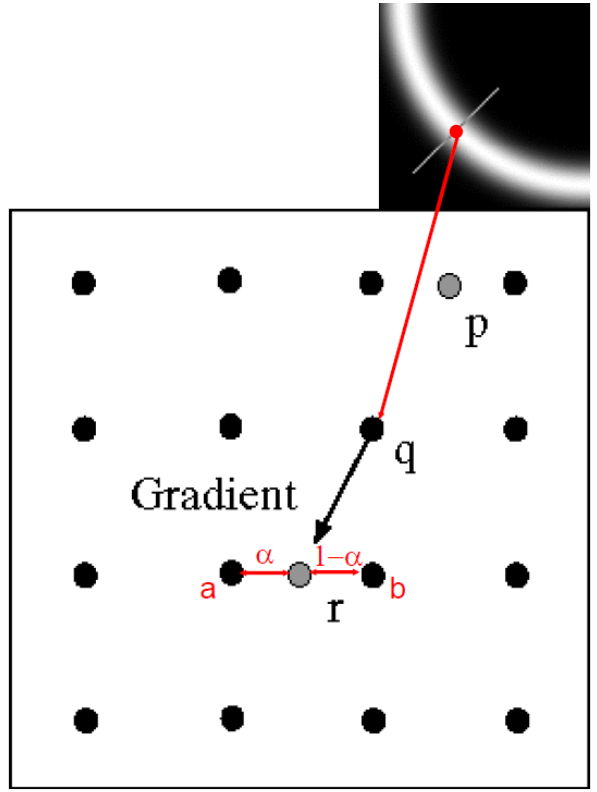
$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3. Non-maximum Suppression

Gradient direction is normal to the edge.

When pixel *q* has an intensity that is larger than both *p* and *r* where pixels *p* and *r* are the pixels in the gradient direction of *q*.

If this condition is true, then we keep the pixel,
otherwise we set the pixel to zero.



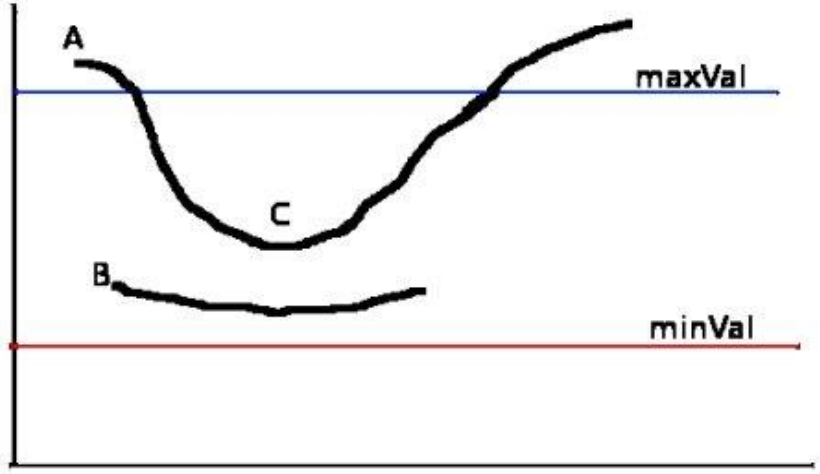
4. Hysteresis Thresholding

we choose two type of threshold, high and low threshold value(maxVal and minVal).

Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges.

Those who lie between these two thresholds are classified edges or non-edges based on their connectivity.

If they are connected to "sure-edge" pixels, they are considered to be part of edges.



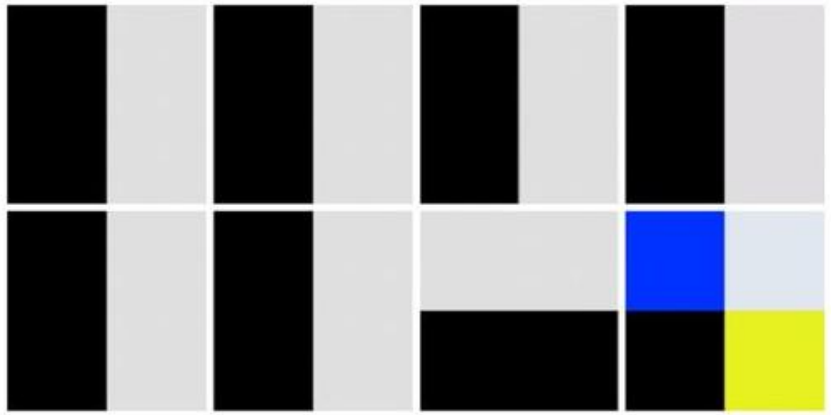
Canny Edge Detection

Edge detection by Canny method (σ = 1, 2, 3, maxVal = 0.7, minVal = 0.3)

Color Edge Detection

Difference operators can be applied to each component of a multi-image, and the results can be combined by taking the RMS, or the sum, or the maximum of their absolute values.

In all of these approaches the image-components do not actually cooperate with one another, i.e., edge evidence along a given direction in one component does not reinforce edge evidence along the same direction in other components.



Color Edge Detection

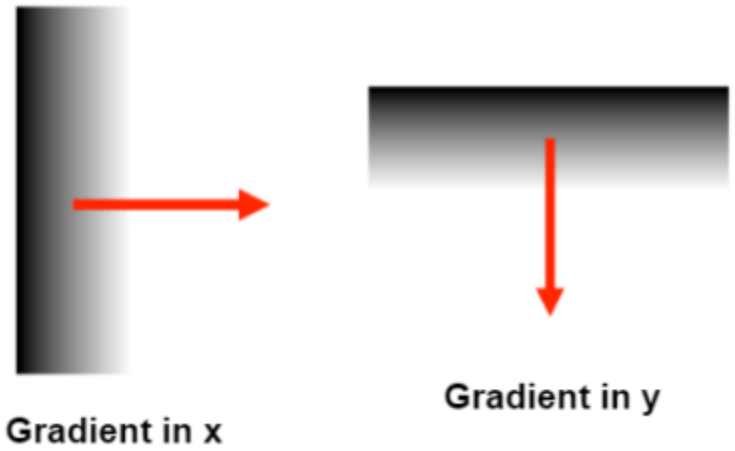
Definition of the gradient applicable to vector quantities : method by Di Zenzo

The gradient(magnitude and direction) of the vector c

$$g_{xx} = \mathrm{u} \cdot \mathrm{u} = \mathrm{u}^T \mathrm{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2$$

$$g_{yy} = \mathrm{v} \cdot \mathrm{v} = \mathrm{v}^T \mathrm{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2$$

$$g_{xy} = \mathrm{u} \cdot \mathrm{v} = \mathrm{u}^T \mathrm{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y}$$



Color Edge Detection

The direction of maximum rate of change of

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2g_{xy}}{(g_{xx} - g_{yy})}\right]$$

The value of the rate of change at *(x, y)*

$$F(\theta) = \left\{\frac{1}{2}[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})\cos 2\theta + 2g_{xy}\sin 2\theta]\right\}^{\frac{1}{2}}$$

Edge Thinning

Skeletonization consists of iterative deletions of the dark points along the edges of a pattern until the pattern is thinned to a line drawing.

SPTA is wide known skeletonization algorithm that is fast, that prevent excessive erosion, and one that give good skeletons.

SPTA(Safe Point Thinning Algorithm)

Consider 4 types of edge points:

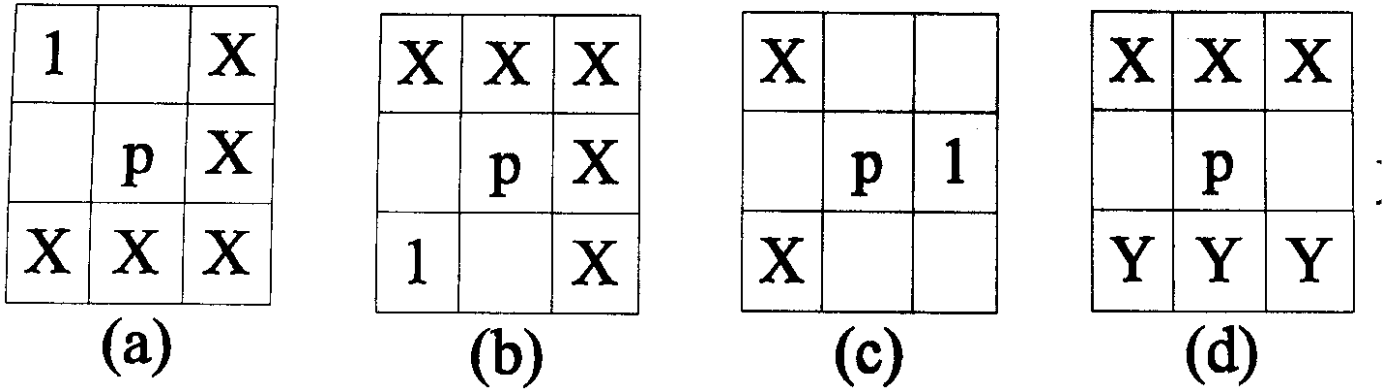
- Left edge point : n4 = white(0)
- Right edge point: n0 = white(0)
- Top edge point: n2 = white(0)
- Left edge point: n6 =white(0)

n_3	n_2	n_1
n_4	p	n_0
n_5	n_6	n_7

SPTA

Consider left edge point (WLOG):

- If a pixel matches one of the following, the pixel not flagged. (X,Y = don't care)



- If following expression s4 is false, the pixel not flagged.

$$s_4 = n_0(n_1 + n_2 + n_6 + n_7)(n_2 + \overline{n}_3)(n_6 + \overline{n}_5)$$

SPTA

Similarly,

A right safe point: $s_0 = n_4(n_5 + n_6 + n_2 + n_3)(n_6 + \bar{n}_7)(n_2 + \bar{n}_1)$

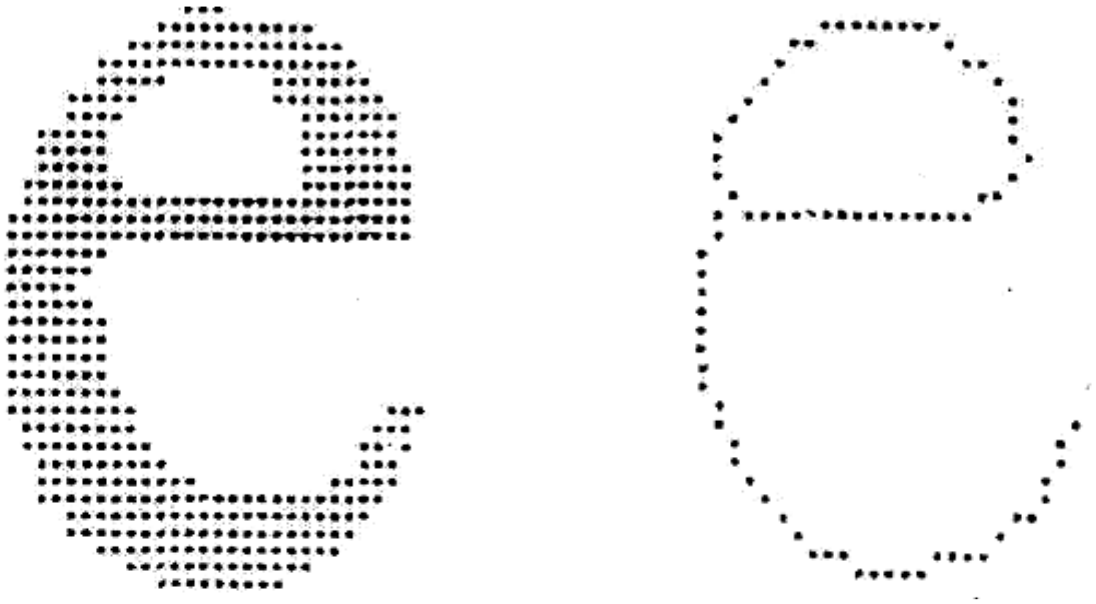
Top safe point: $s_2 = n_6(n_7 + n_0 + n_4 + n_5)(n_0 + \bar{n}_1)(n_4 + \bar{n}_3)$

Bottom safe point: $s_6 = n_2 + (n_3 + n_4 + n_0 + n_1)(n_4 + \bar{n}_5)(n_6 + \bar{n}_7)$

- During each iteration, “flag” those edge points that are not end-point or break-point.
- At the end of iteration, all flagged point removed.

SPTA

- One advantage of skeletonization is to reduce the memory space required for storing the essential structural information present in the patterns.



The skeleton produced by SPTA.

Edge Detection

- After implementing SPTA, choose starting pixels in binary image and detect edge segments.

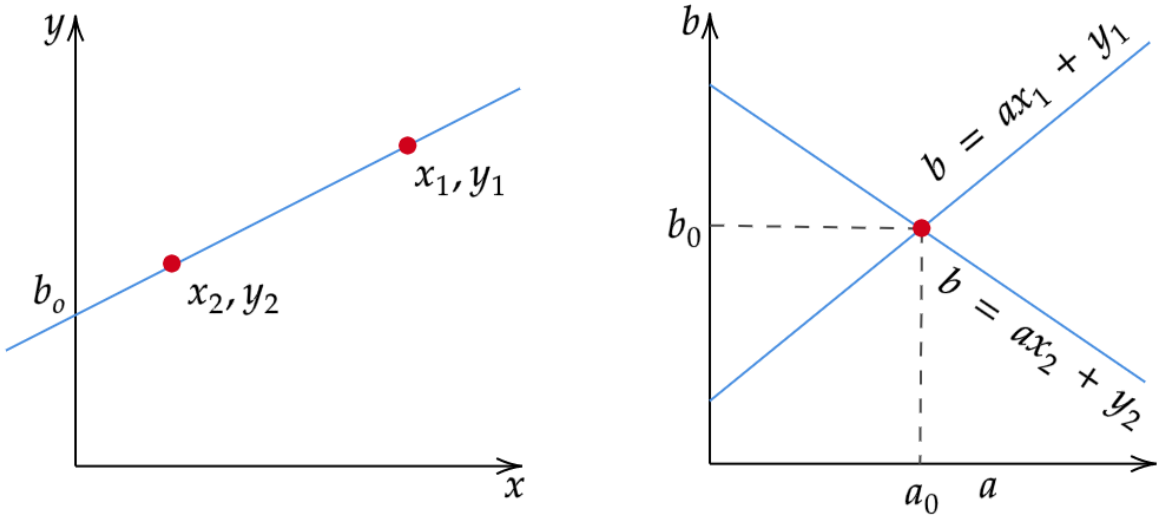
- Each edge segments are transformed into Line segments.

- Hough transform can be used to detect straight lines in an image.

Hough Transform

- The Hough Space is a 2D plane that has a horizontal axis representing the slope and the vertical axis representing the intercept of a line on the edge image.

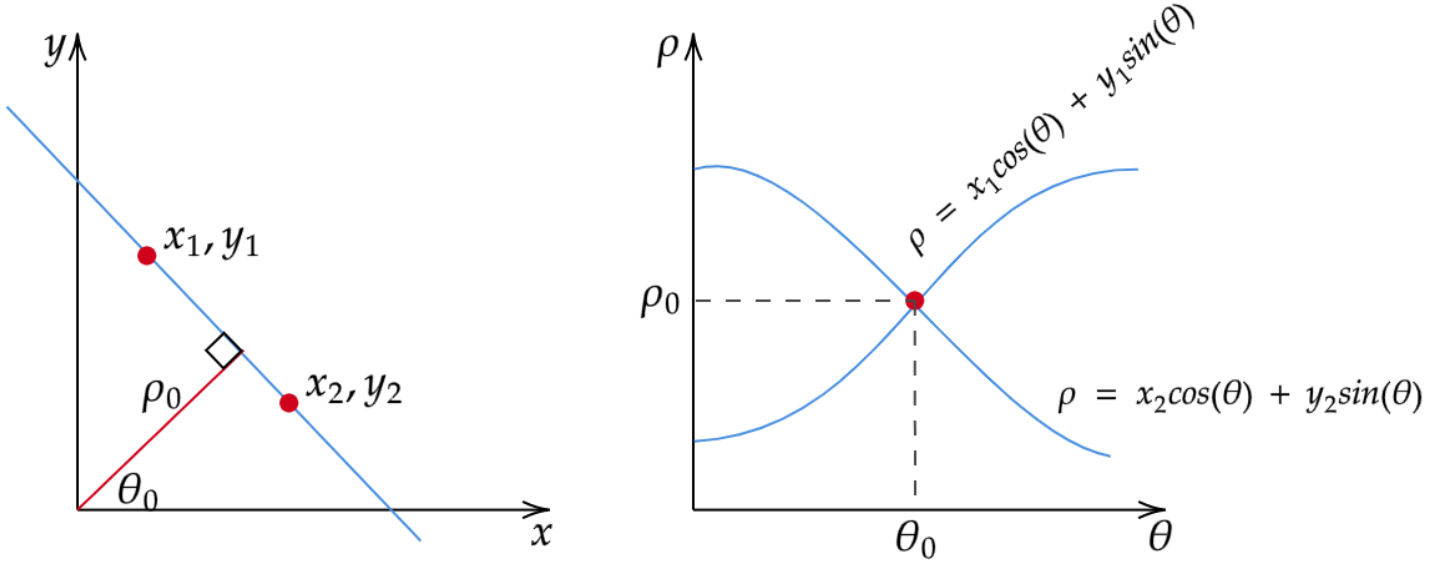
- Therefore, an edge point produces a line in the Hough Space in the form of $b = ax_i + y_i$



Hough Transform

- In this form, the algorithm won't be able to detect vertical lines because the slope *a* is undefined/infinity for vertical lines

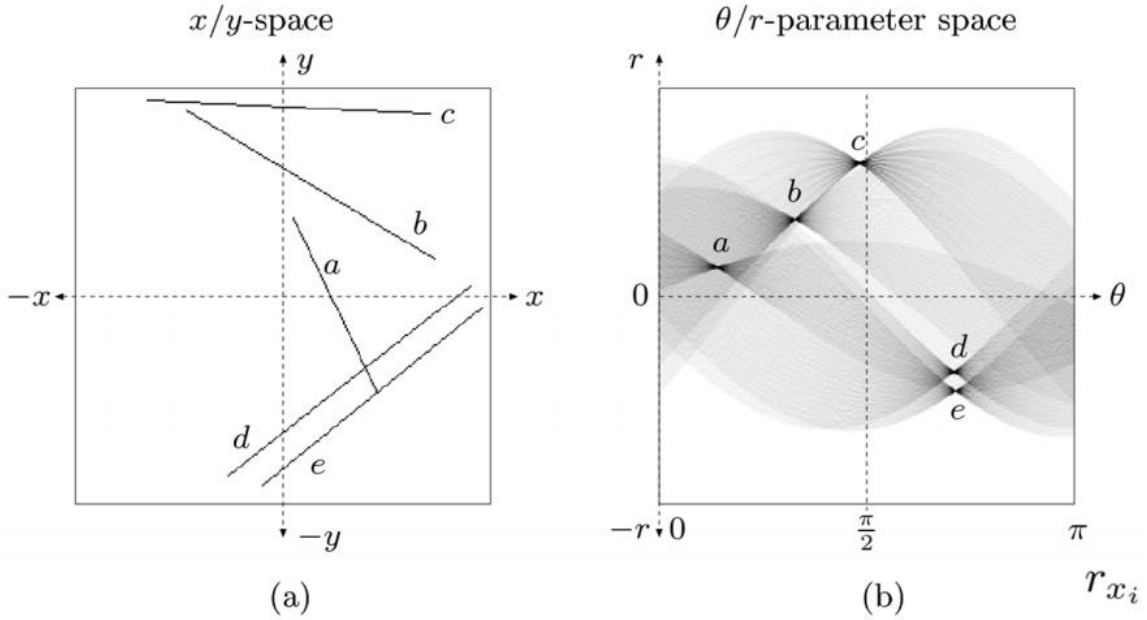
- The form of the normal line is ρ = x cos(θ) + y sin(θ) where ρ is the length of the normal line and θ is the angle between the normal line and the x axis.



Hough Transform

- If we draw points which form a line in the image space, we will obtain a bunch of sinusoids in the Hough space. They are intersecting at exactly one point

- It means that, to identify candidates for being a straight line, we should seek for intersections in Hough space.



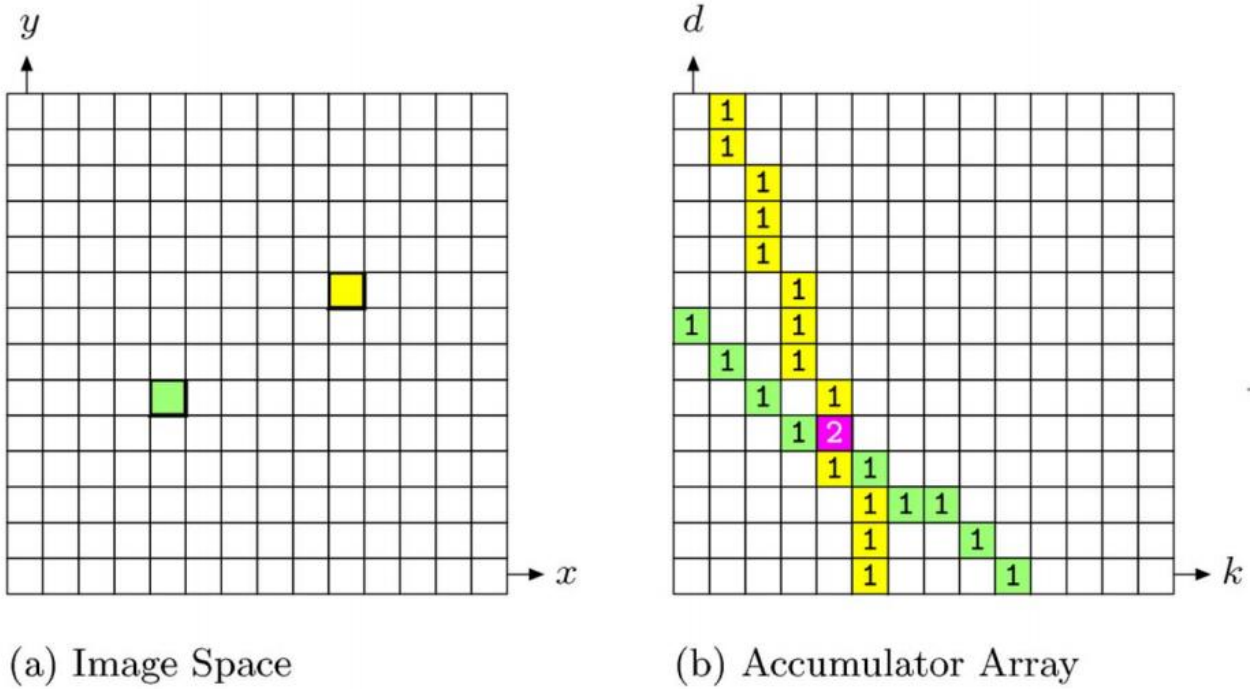
Hough Transform

1. Decide on the range of ρ and θ. It is important to quantize the range of ρ and θ meaning there should be a finite number of possible values.

2. Create a 2D array called the accumulator representing the Hough Space and initialize all its values to zero.

Hough Transform

3. For every edge pixel on the edge image, calculate the all possible values (ρ, θ) and increment the accumulator base on those index pairs.

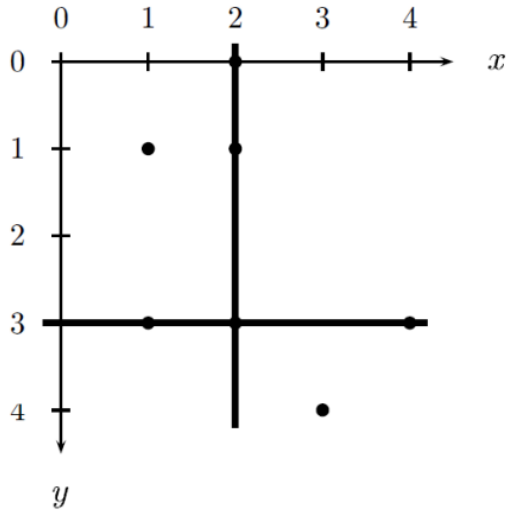


(a) Image Space (b) Accumulator Array

Hough Transform

4. Loop through all the values in the accumulator. If the value is larger than a certain threshold, get the (ρ, θ) and convert back to the form of y = ax + b.

$(r, \theta) = (2, 0°)$ and $(r, \theta) = (3, 90°)$



	−1.4	−0.7	0	0.7	1	1.4	2	2.1	2.8	3	3.5	4	4.9
−45°	1		2	1	2		1						
0°					2		3			1		1	
45°						2		1	1		1		2
90°			1		2					3		2	

Hough Transform

The same idea is applied for other shapes.

Once you have parametric equation that describes the shape you can build parameter space and detect that shape. For the circle

$$r^2 = (x - x_0)^2 + (y - y_0)^2$$

Parameter space now is 3D parameter space.

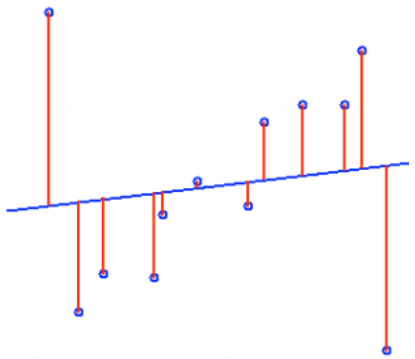
RANSAC(RANdom SAmple Consensus)

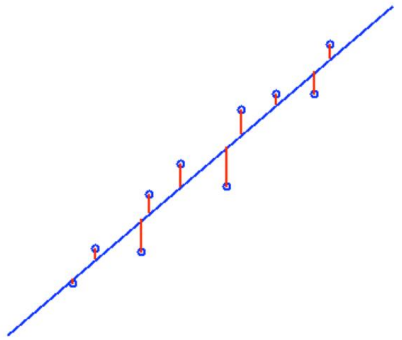
Method to estimate parameters of a mathematical model from a set of observed data that contains outliers.

RANSAC assumes that, given a set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.

Outliers

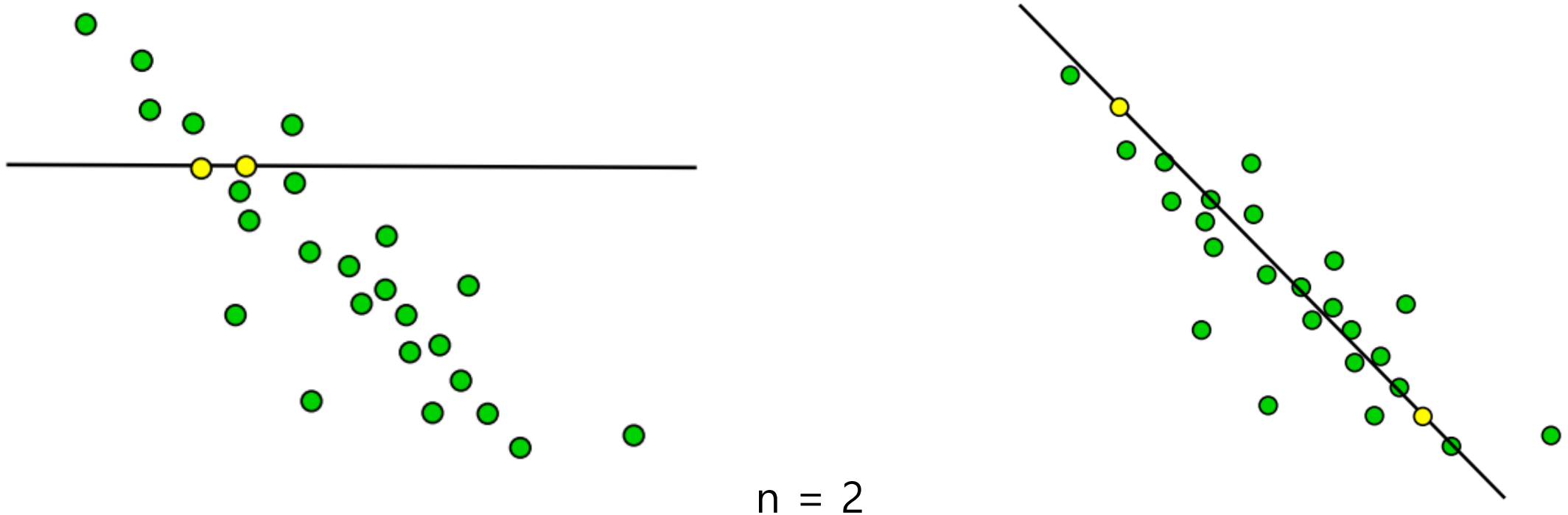
- Data points that are far from other data points.
- Least squares estimation is sensitive to outliers, so a few outliers can skew the result.





RANSAC

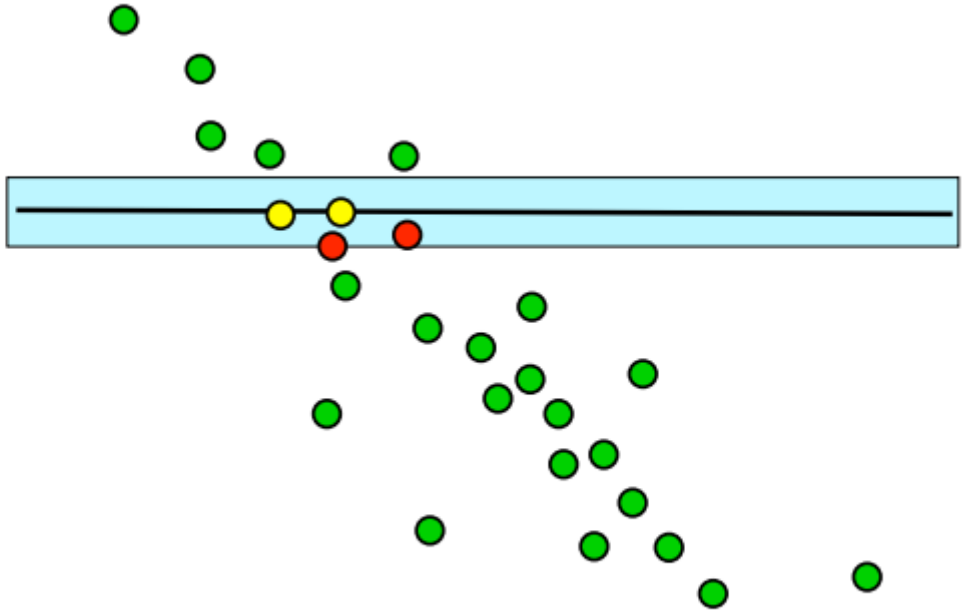
1. Select randomly the minimum number of points(n) required to determine the model parameters. And solve for the parameters of the model.



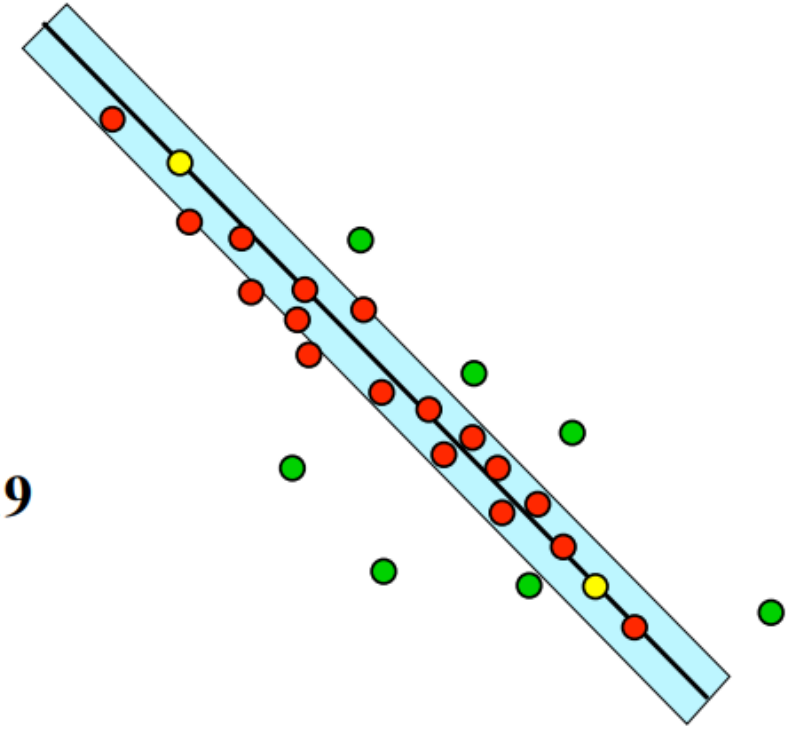
n = 2

RANSAC

2. Determine how many points from the set of all points fit with a predefined tolerance(ε)



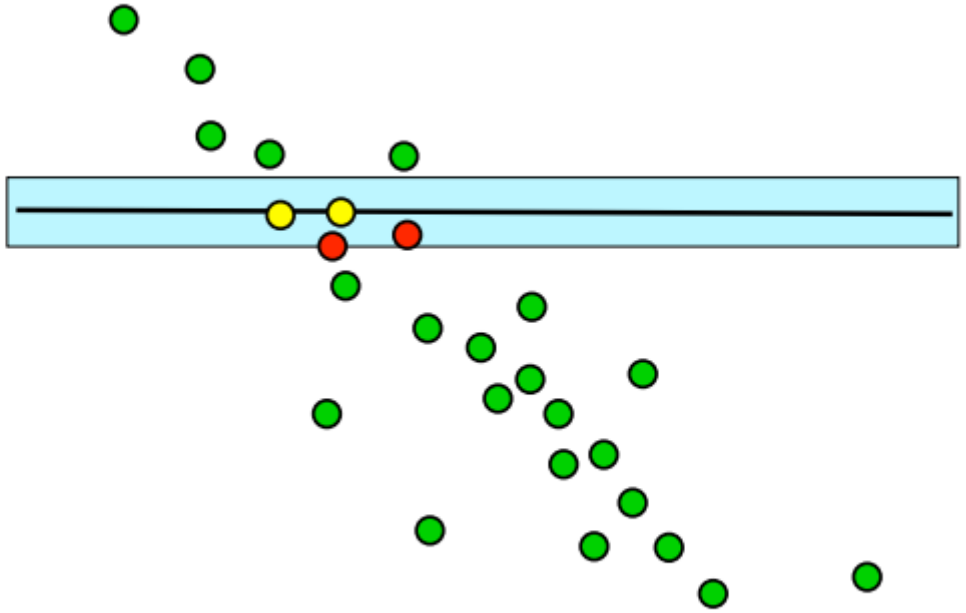
Count = 4



Count = 19

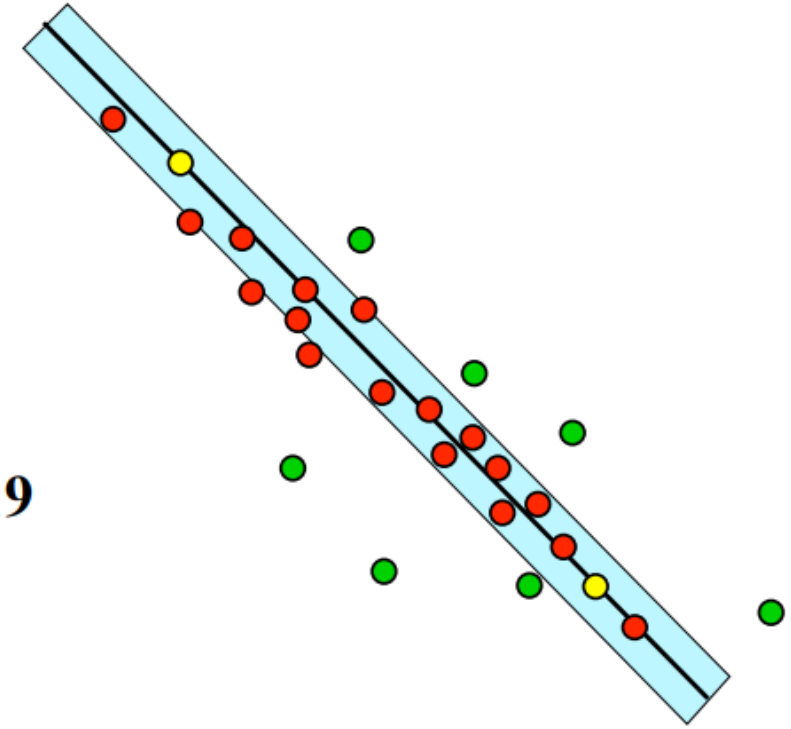
RANSAC

3. If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold(τ), re-estimate the model parameters using all the identified inliers and terminate.



Count = 4

4 / 26

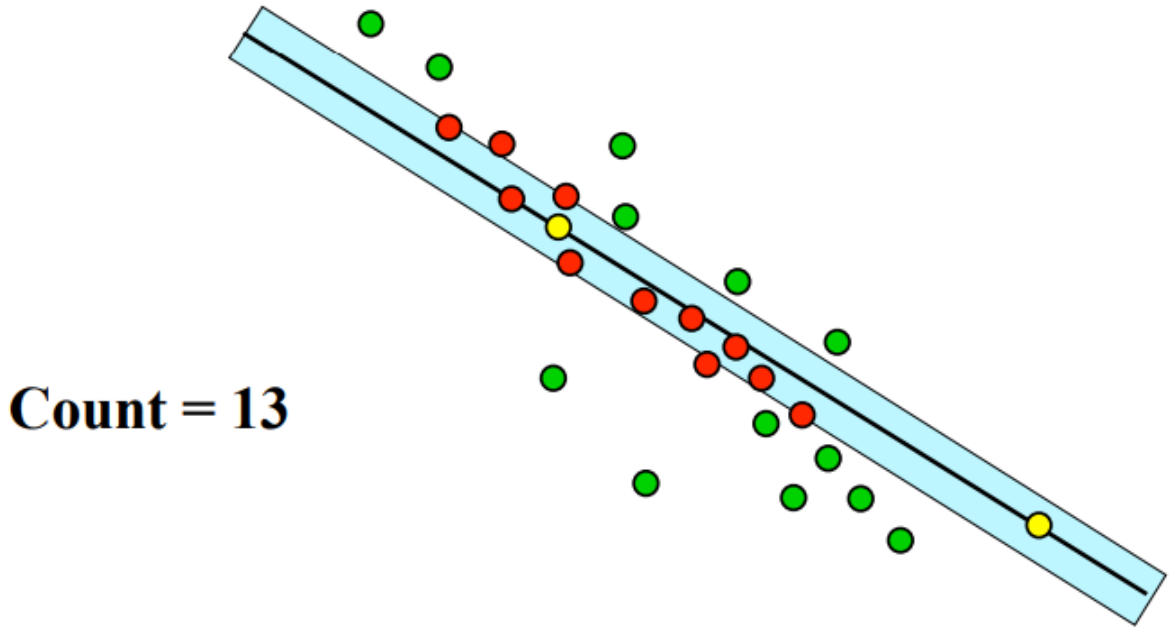


Count = 19

19 / 26

RANSAC

4. Otherwise, repeat steps 1 through 4 (maximum of k times)



RANSAC

k – Maximum number of iterations allowed in the algorithm

n – Minimum number of data points required to estimate model parameters.

ε – Predefined closeness tolerance to assert that a model fits well to data.

τ – Threshold value to determine data points that are fit well by model.

RANSAC

The number of iterations, N, is chosen high enough to ensure that the probability p that at least one of the sets of random samples does not include an outlier.

Let n be the probability of choosing an inlier.

$1 - w^n$ is the probability that at least one of the n points is an outlier.

That probability to the power of k is the probability that the algorithm never selects a set of n points which all are inliers and this must be the same as $1 - p$.

$$1 - p = (1 - w^n)^k$$

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}$$

RANSAC

Advantages

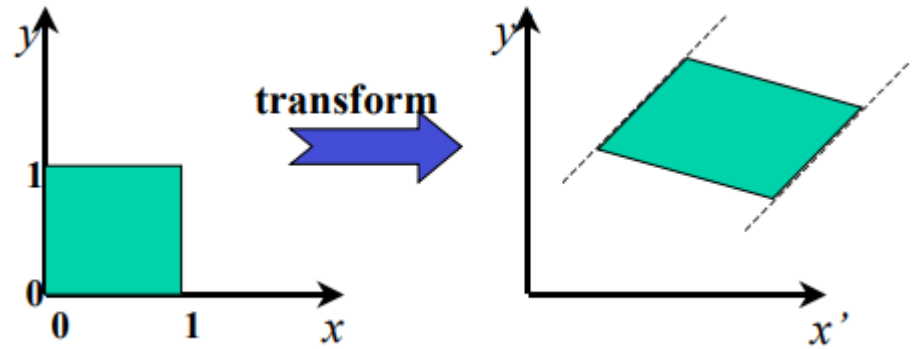
- It can estimate the parameters with a high degree of accuracy even when a significant number of outliers are present in the data set.

Disadvantages

- There is no upper bound on the time it takes to compute these parameters.
- It is not always able to find the optimal set.
- It can only estimate one model for a particular data set.

RANSAC

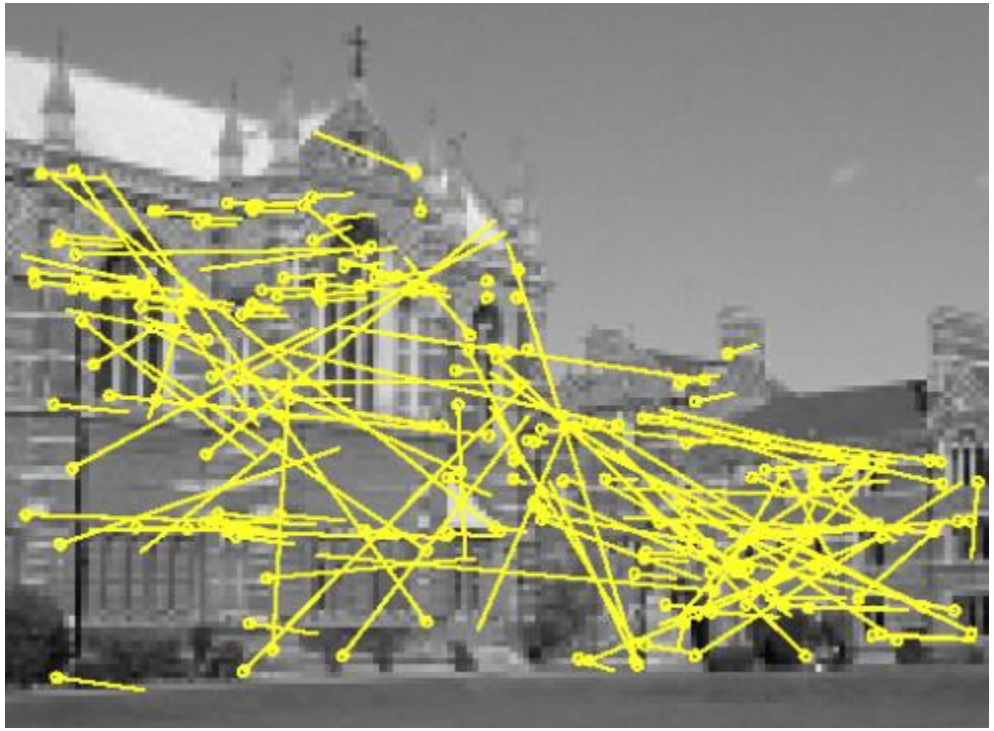
Use RANSAC to robustly fit best affine transformation to the set of point matches.



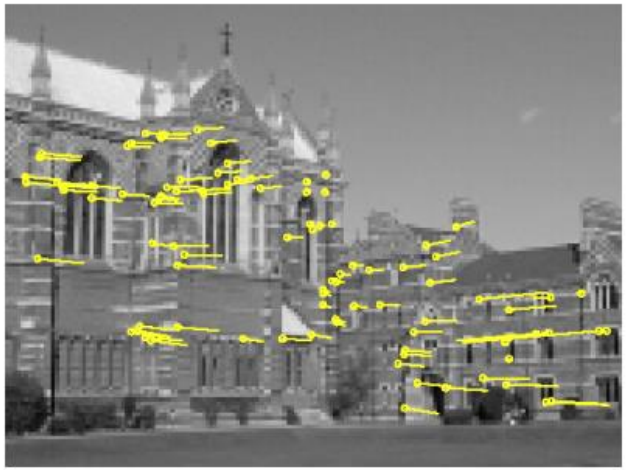
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Affine transformation has 6 degrees of freedom. We therefore need 3 point matches

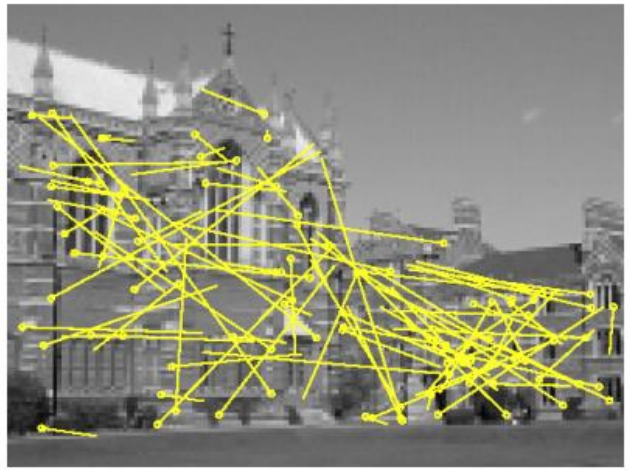
RANSAC



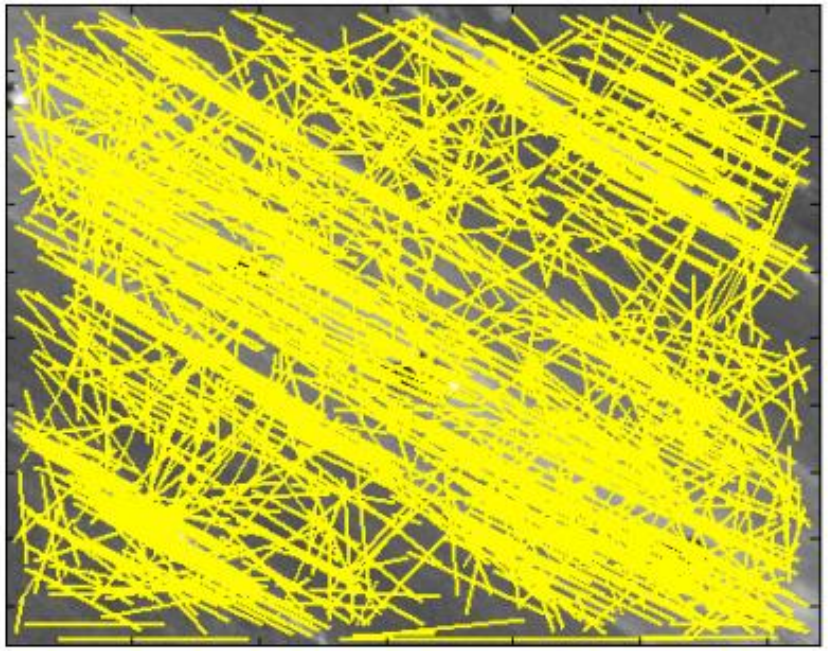
Inliers (99)



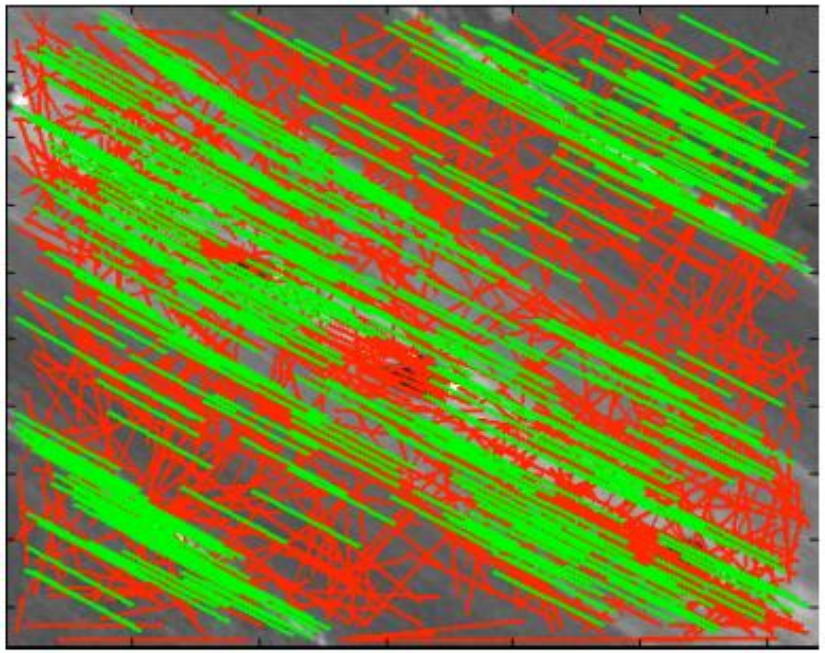
Outliers (89)



RANSAC



original point matches



labels from RANSAC
green: inliers
red: outliers