Visual Computing
Summary

October 6, 2020

Chapter 1

The Digital Image

1.1 What is an image:

Signal: A function depending on some variable with physical meaning

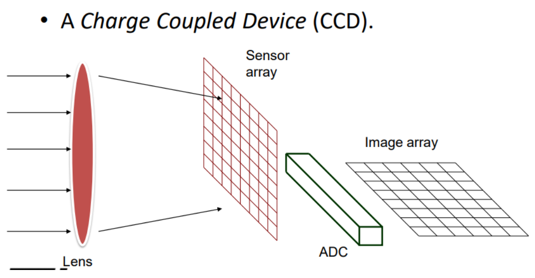
Image: A continuous function. The value can take on physical values. e.g Brightness, temperature, pressure, depth, etc. We distinguish images with:

- **2 variables:** xy- coordinates
- **3 variables:** xy + time (video)

Hence an image is a picture or pattern of a value varying in space and/or time.

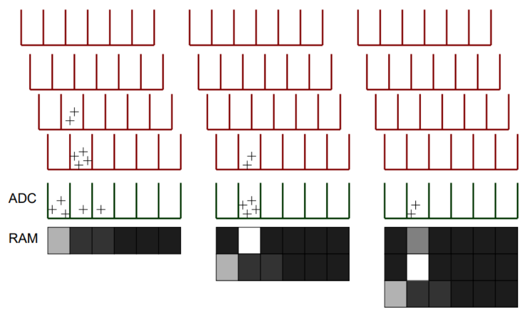
$$f : \mathbb{R}^n \to S$$

1.2 The digital camera



The sensor array: An array of photosites. Each photosite is a bucket of electrical charge and contain a charge proportional to the incident light intensity during exposure.

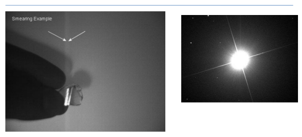
Analog to Digital Conversion: The ADC measures the charge and digitizes the result. The conversion happens line by line. The charges in each photosite move down through the sensor array.



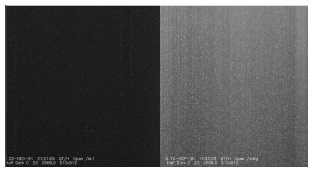
Various errors:

- **Blooming:** The buckets have finite capacity, saturation causes blooming. It happens when a large amount of light gets focused to a single point on your cameras image sensor. This can create so much charge that it actually bleeds from pixel to pixel until it eventually spreads out.

- **Bleeding or Smearing:** During transit buckets still accumulate some charges. The amount is influenced by the time ”in transit” versus the integration time



- **Dark Current:** A relatively small electric current that flows through photosensitive devices even when no photons are entering the device.

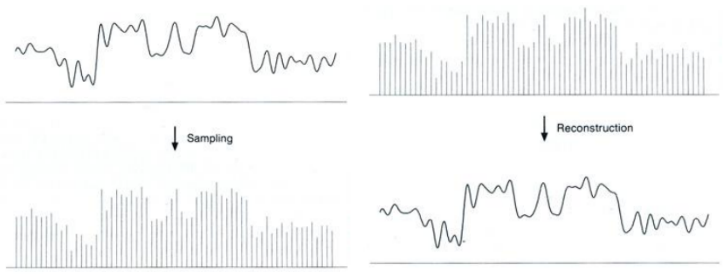


CMOS: Contains the same sensor elements as CCD but each photo sensor has its own amplifier. the benefits:

- Recent technology
- Standard IC technology
- Cheap
- Low Power
- Less sensitive
- Per pixel amplification
- Random pixel access
- Smart pixels
- On chip integration with other components

1.3 Sampling 1D:

Sampling in 1D takes a function, and returns a vector whose elements are values of that function at the sample points.

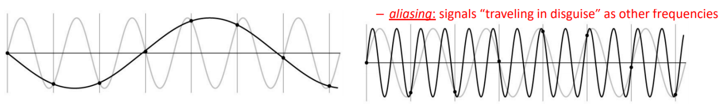


1.3.1 Reconstruction:

Making samples back into a continuous function. This amounts to guessing what the function did in between the individual samples

1.3.2 Undersampling:

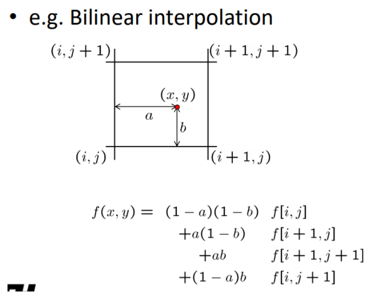
Occurs if not enough sampling points are available. It results in a loss of information and can be indistinguishable from other samples e.g undersampling a sin wave:


– aliasing: signals “traveling in disguise” as other frequencies

1.4 Sampling 2D:

Sampling in 2D takes a function and returns an array. The array can have infinite dimensions and have negative as well as positive indices.

1.4.1 Reconstruction continuous signals:



$$\begin{aligned} f(x,y) = \; & (1-a)(1-b) \quad f[i,j] \\ & +a(1-b) \quad f[i+1,j] \\ & +ab \quad f[i+1,j+1] \\ & +(1-a)b \quad f[i,j+1] \end{aligned}$$

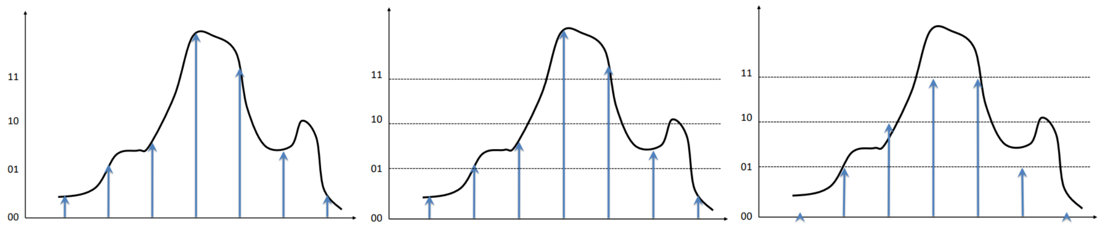
1.4.2 Nyquist Frequency:

Half the sampling frequency of a discrete signal processing system. The signals max frequency (bandwidth) must be smaller than this.

1.4.3 Quantization:

Real valued function gets digital values (integer values). Quantization is lossy i.e the orignial signal cannot be reconstructed anymore. Simple quantization uses equally spaced levels with k intervals:

$$k = 2^b$$



1.4.4 Image Properties:

Geometric Resolution: How many pixels per area

Radiometric Resolution: How many bits ber pixel

1.5 Image Noise

The different types of common Noise models:

- **Gaussian noise:**

$$I(x,y) = f(x,y) + c$$

  where $c \sim N(0, \sigma^2)$ so that

$$p(c) = (2\pi\sigma^2)^{-1} e^{\frac{-c^2}{2\sigma^2}}$$

- **Poisson noise:**

$$p(k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- **Rician noise:**

$$p(I) = \frac{I}{\sigma^2} e^{\frac{-(I^2+f^2)}{2\sigma^2}} I_0(\frac{If}{\sigma^2}$$

- **Multiplicative noise:**

$$I = f + fc$$

Signal to noise ration (SNR) An index of image quality:

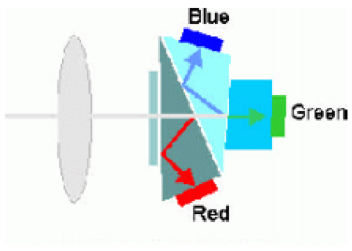
$$s = \frac{F}{\sigma} \text{ where } F = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1} Y f(x,y)$$

Peak Signal to Noise Ration (PSNR)

$$s_{peak} = \frac{F_{max}}{\sigma}$$

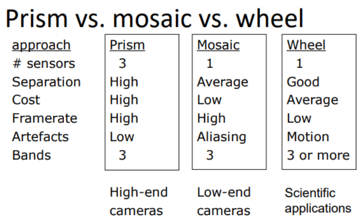
1.6 Color Cameras:

Prism Color Camera: Seperates the light into 3 beams using a dichroic prism



Filter mosaic: Filter is coated directly on sensor

Filter wheel: Rotate multiple filters in front of lens. This is only suitable for static scenes.



Prism vs. mosaic vs. wheel

approach	Prism	Mosaic	Wheel
# sensors	3	1	1
Separation	High	Average	Good
Cost	High	Low	Average
Framerate	High	High	Low
Artefacts	Low	Aliasing	Motion
Bands	3	3	3 or more
	High-end cameras	Low-end cameras	Scientific applications

Chapter 2

Image Segmentation:

Image segmentation is the concept of partitioning an image into regions of interest.

Complete Segmentation: A finite set of regions $R_1, ..., R_N$, such that

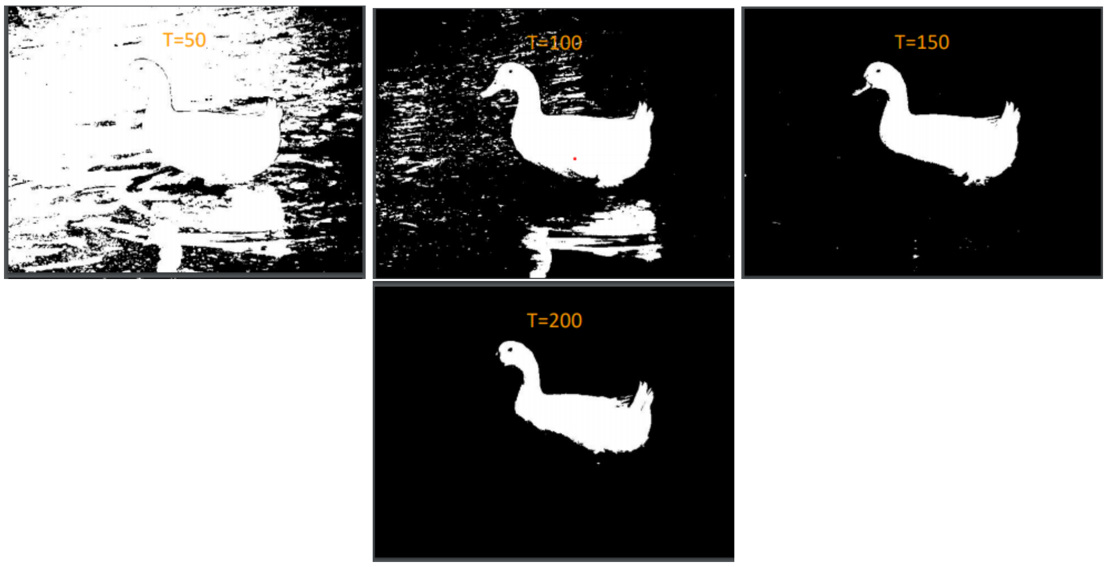
$$I = \bigcup_{i=1}^{N} R_i \text{ and } R_i \cap R_j = \phi \quad \forall i \neq j$$

Where I is an image.

2.1 Thresholding

Thresholding is a simple segmentation process which produces a binary image B. It labels each pixel "in" or "out" of the region of interest by comparison of the greylevel with a threshold T:

$$B(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{if } I(x,y) < T \end{cases}$$



2.2 ROC Analysis

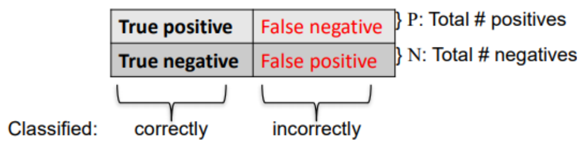
ROC = Reciever Operating Characteristic

An ROC curve characterizes the performance of a binary classifier. A binary classifier distinguishes between two different types of things e.g:

- Healthy/afflicted patients
- Pregnancy tests
- Object detection
- Foreground/background image pixels

2.2.1 Classification Error

Binary classifiers make errors. There are two inputs to a binary classifier i.e positives and negatives but there are four possible outcomes in any test:



2.2.2 ROC Curve

Characterizes the error trade-off in binary classification tasks. It plots the TP (True-positive) against the FP (False-positive) fraction

$$\text{TP fraction (sensitivity):} \frac{\text{True positive count}}{P}$$
$$\text{FP fraction (1-specificity):} \frac{\text{False positive count}}{N}$$

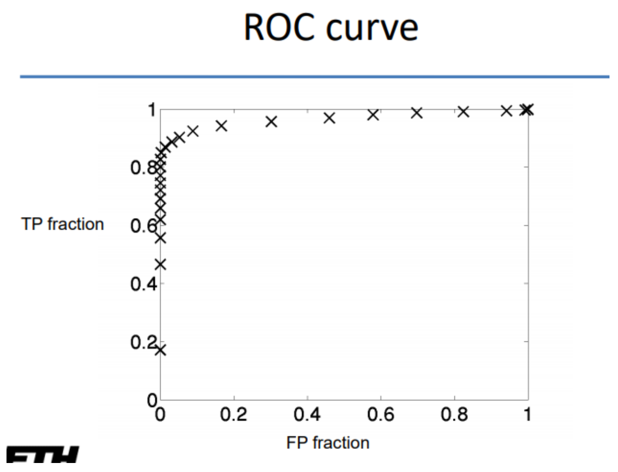
An ROC curve always passes through (0,0) and (1,1)
We choose an operating point by assigning a relative costs and values to each outcome:

- V_{TN} Value of true negative
- V_{TP} Value of a true positive
- C_{FN} Cost of a false negative
- C_{FP} Cost of a false positive

We choose the point on the ROC curve with gradient:

$$\beta = \frac{N}{P}\frac{V_{TN}+C_{FP}}{V_{TP}+C_{FN}}$$

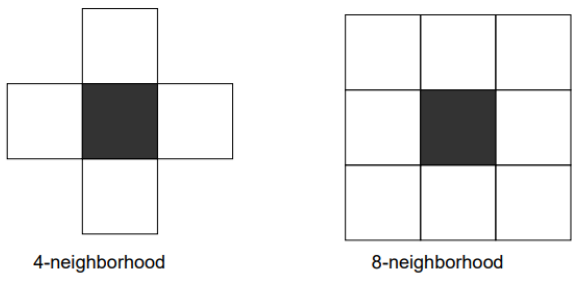
For simplicity we often set $V_{TN} = V_{TP} = 0$



In reality we use 2-3 seperate sets of test data:

1. **Training set:** For tuning the algorithm
2. **Validation set:** For tuning the performance score
3. **Test set:** To get a final performance score on the tuned algorithm

2.3 Pixel Connectivity:



Pixel Neighbourhood:

Pixel Paths:

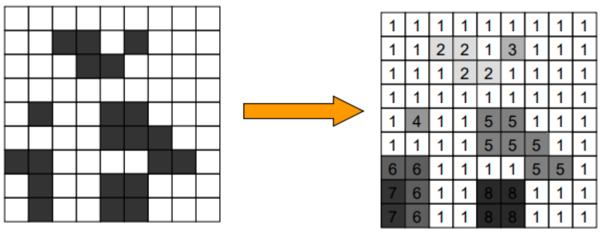
- A 4-connected path between pixels p_1 and p_n is a set of pixels $\{p_1, p_2, ..., p_n\}$ such that p_i is a 4-neighbour of p_{i+1} i=1,...,n-1
- A 8-connected path, p_i is an 8-neighbour of p_{i+1}

Connected Regions:

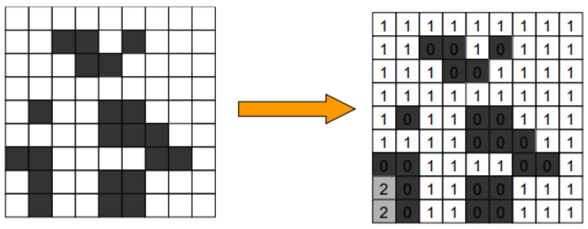
- A region is 4-connected if it contains a 4-connected path between any two of its pixels
- A region is 8-connected if it contains an 8-connected path between any two of its pixels

2.4 Image Labelling

Connected components Labelling: Labels each connected component of a binary image with a seperate number



Foreground Labelling: Only extract the connected components of the foreground



Region Growing: Start from a seed point or region and add neighbouring pixels that satisgy the criteria defining a region, we repeat until we can include no more pixels. There are 3 meaningful variations when considering region Growing:

- Seed Selection i.e from where do we start the region growing
- Inclusion criteria
- Boundary constraints and snakes i.e how can the boundary evolve.

Seed selection: We can select a seed point or region by hand or automatically e.g from a conservative thresholding. Multiple seeds can be chosen aswell.

Inclusion Criteria: Until now we have looked at Greylevel thresholding. Another possibility would be to use a Greylevel distribution model. We use a mean μ and a standard deviation σ in the seed region. We include the pixel if:

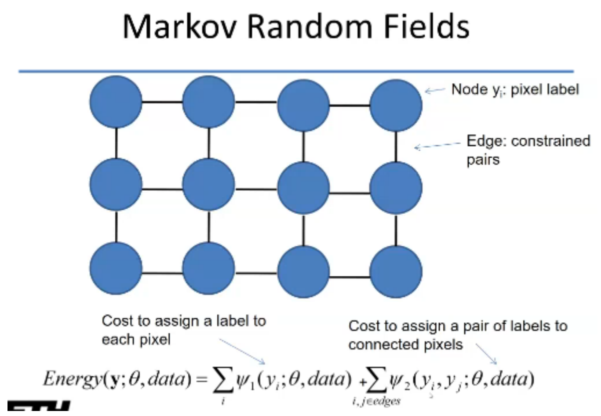
$$(I(x, y) - \mu)^2 < (n\sigma)^2$$

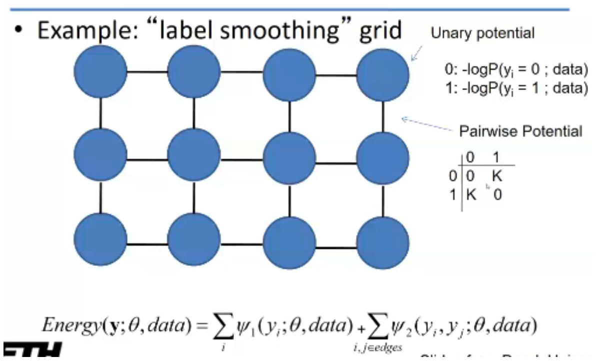
We can update the mean and standard deviation after every iteration.

snakes A snake is an active contour. It is a polygon i.e an ordered set of points joined up by lines. Each point on the contour moves away from the seed while its image neighborhood satisfies an inclusion criterion. contours usually have smoothness constraints. To avoid snakes the algorithm iteratively minimizes an energy function:

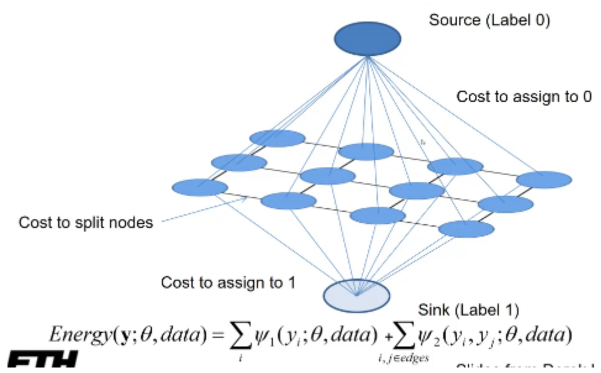
$$E = E_{tension} + E_{stiffness} + E_{image}$$

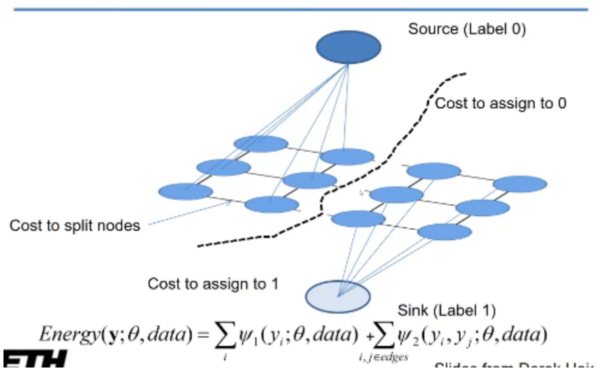
Stiffness and tension are used to keep the region from growing too much and grow in a regular way.


Markov Random Fields
Node y_i: pixel label
Edge: constrained pairs
Cost to assign a label to each pixel
Cost to assign a pair of labels to connected pixels
$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$


Example: "label smoothing" grid
Unary potential
0: -logP(y_i = 0 ; data)
1: -logP(y_i = 1 ; data)
Pairwise Potential
0 1
0 0 K
1 K 0
$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$

Markov Random Fields: The Energy Function consists of two terms. The first term is an unary term i.e only considering the current pixel e.g wether or not the pixel is foreground (F) or background (BG) and the second term is a pairwise term which has 4 possible states: F F, F BG, BG F, BG BG. Usually we favor pixels agreeing with eachother, hence these pixels will have a lower energy. Our goal is to minimize this energy function. One way to solve this is using graph cuts i.e Min-Cut Max Flow


Source (Label 0)
Cost to assign to 0
Cost to split nodes
Cost to assign to 1
Sink (Label 1)
$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$

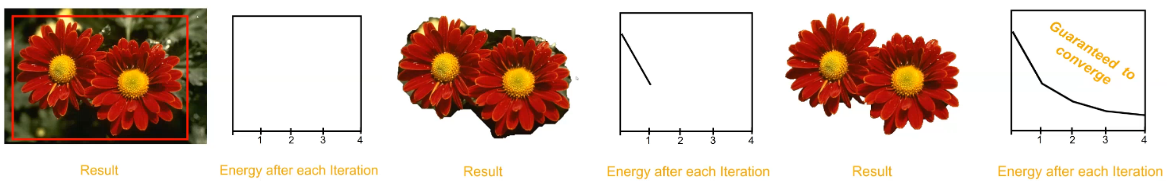

Source (Label 0)
Cost to assign to 0
Cost to split nodes
Cost to assign to 1
Sink (Label 1)
$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$

We want to find a cut such that all the foreground pixels are connected to one side and the background pixels connected to the other side while minimizing the separation of the two. The cost in the cut in the picture on the right is the pairwise cost of cutting the three boundary edges (3K) and the data cost for labelling each pixel to the source or the sink. Terminoligy:

- **Cut:** Separates source from sink
- **Energy:** Collection of edges
- **Min Cut:** Global minimal energy

The Min-Cut can be found in polynomial time.

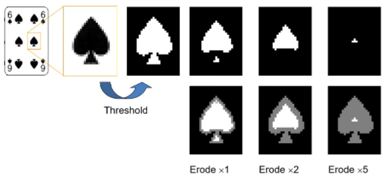
Iterated Graph Cut: Given the picture the user picks a rough boundary which contains all of the pixels which belong to the foreground. We look at the 2 distribution of colours inside and outside of the chosen boundary. We then apply **K-means clustering** which is a method in which we define K clusters and assign each pixel to one of the clusters. After assigning the clusters we check wether or not the cluster fits better to the distribution of the foreground or background and assign it to the corresponding match. The background model can then retrain itself with the newly assigned background pixels to achieve an even tighter foreground model. Using this method it is guaranteed that we saturate at a certain energy level and dont oscillate between values. The saturated energy value is not necissarily a local or global minimium.



2.5 Morphological Operations

Operations which are purely spatial. Hence we wont use data driven operations. They are local pixel transformations for processing region shapes. Most often used on binary images. Logical transformations based on comparison of pixel neighbourhoods with a pattern. Some examples:

- **8-neighbour erode (Minkowsky Subtraction)** if the current pixel has an 8-neighbour which belongs to the background add it to the background.



- **8-neighbour dilate (Minkowsky Addition)** Paint any background pixel that has one eight-connected neighbour that is foreground

Morphological Operation are used to smooth region boundaries for shape analysis and can be used to remove noise and artefacts from an imperfect segmentation.

Structuring Elements The structuring element is a binary array indicating pixels of interest in a region. A structuring element has an origin. Morphological operations take two arguments

- A binary image
- A structuring element

We compare the structuring element to the neighbourhood of each pixel, this determines the output of the morphological operation.
We define 3 operations (S is a structuring element) :

- **Fitting:** S fits I at $\underline{x}$ if

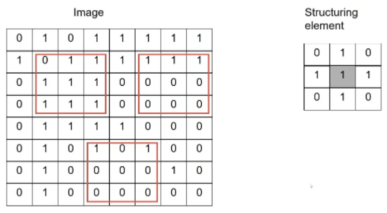
$$\{y : y = \underline{x} + \underline{s}, \underline{s} \in S\} \subset I$$

- **Hitting:**S hits I at $\underline{x}$ if

$$\{\underline{y} : \underline{y} = \underline{x} - \underline{s}, \underline{s} \in S\} \cap I \neq \emptyset$$

- **Missing:**S misses I at $\underline{x}$ if

$$\{\underline{y} : \underline{y} = \underline{x} - \underline{s}, \underline{s} \in S\} \cap I = \emptyset$$



Image

0	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1
0	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	1	0	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	0	0

Structuring element

0	1	0
1	1	1
0	1	0

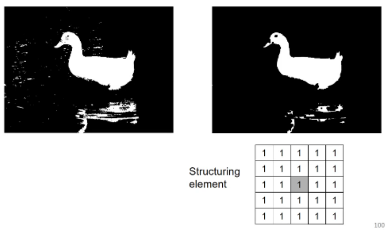
The above picture there are 3 regions of interest: upper left, upper right, down center:

- For the upper left region the structuring element origin placed in the center all the ones coincide with the ones in the region
- The upper right region demonstrates hits: We place the structuring element origin at all the squares with one e.g the middle square which is 0 is covered by a one in the structuring element and is hence a hit.
- The down center image demonstrates a miss. We place the structuring element origin at all the squares with a one but the middle 0 does not get covered with a one and hence its a miss.

Erosion: The image $E = I \ominus S$ is the erosion of image I by structuring element S:

$$E(\underline{x}) = \begin{cases} 1 & \text{if S fits I at } \underline{x} \\ 0 & \text{otherwise} \end{cases}$$

$$E = \{x : x + s \in I \text{ for every } s \in S\}$$

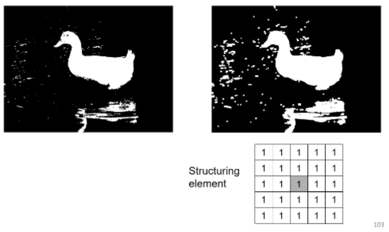

Structuring element

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Dilation: The image $D = I \oplus S$ is the dilation of image I by structuring element S:

$$D(\underline{x}) = \begin{cases} 1 & \text{if S hits I at } \underline{x} \\ 0 & \text{otherwise} \end{cases}$$

$$D = \{\underline{x} : \underline{x} - \underline{s}, \underline{y} \in I \text{ and } \underline{s} \in S\}$$



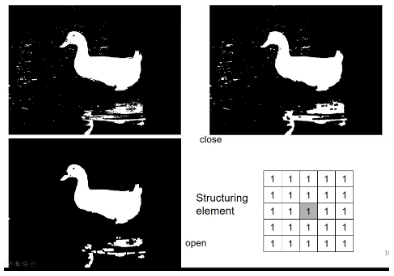
$$\begin{array}{|c|c|c|c|c|} \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline \end{array}$$

Opening The opening of I by S is

$$I \circ S = (I \ominus S) \oplus S$$

Closing The closing of I by S is

$$I \bullet S = (I \oplus S) \ominus$$

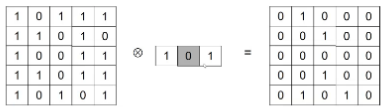


$$\begin{array}{|c|c|c|c|c|} \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline 1&1&1&1&1\\ \hline \end{array}$$

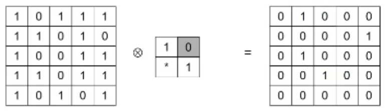
Morphological filtering uses both opening and closing to remove holes in the foreground and islands in the background. The size and shape of the structuring element determine which features survive. In the absence of knowledge about the shape of features to remove, use a circular structuring element.

Granulometry: Provides a size distribution of distinct regions or "granules" in the image. We open the image with increasing structuring element size and count the number of regions after each operation. This creates a "morphological sieve". (e.g counting red blood cells)

Hit-and-miss transform Searches for an exact match of the structuring element. $H = I \otimes S$ is the hit-and-miss transform of image I by structuring element S. It is a simple form of template matching



$$\begin{array}{|c|c|c|c|c|} \hline 1&0&1&1&1\\ \hline 1&1&0&1&0\\ \hline 1&0&0&1&1\\ \hline 1&1&0&1&1\\ \hline 1&0&1&0&1\\ \hline \end{array} \otimes \begin{array}{|c|c|c|} \hline 1&0&1\\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 0&1&0&0&0\\ \hline 0&0&1&0&0\\ \hline 0&0&0&0&0\\ \hline 0&0&1&0&0\\ \hline 0&1&0&1&0\\ \hline \end{array}$$



$$\begin{array}{|c|c|c|c|c|} \hline 1&0&1&1&1\\ \hline 1&1&0&1&0\\ \hline 1&0&0&1&1\\ \hline 1&1&0&1&1\\ \hline 1&0&1&0&1\\ \hline \end{array} \otimes \begin{array}{|c|c|} \hline 1&0\\ \hline *&1\\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline 0&1&0&0&0\\ \hline 0&0&0&0&1\\ \hline 0&1&0&0&0\\ \hline 0&0&1&0&0\\ \hline 0&0&0&0&0\\ \hline \end{array}$$

The $*$ indicates a dont care hence can either be zero or one

Thinning and Thickening Defined in terms of the hit-and-miss transform:

- The thinning of I by S is:

$$I \oslash S = I \backslash (I \otimes S)$$

- The thickening of I by S is:

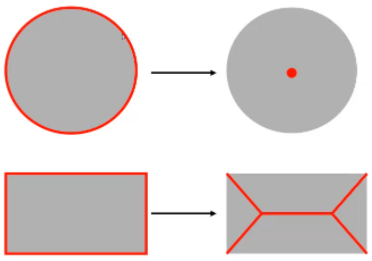
$$I \odot S = I \cup (I \otimes S)$$

We have the following equality:

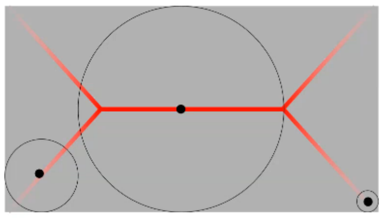
$$I \odot S)^C = I^C \oslash S$$

In contrast to Erosion and Dilation, Thinning and thickening will stop at some point. Erosion and Dilation will continue until the whole image is either background or forground only.

Skeletonization and the Medial Axis Transform (MAT): The MAT are stick-figure representations of a region $X \subset \mathbb{R}^2$. Essentially what it is, is starting a grassfire at the boundary of the region. The skeleton is the set of points at which two fire fronts meet



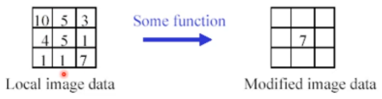
Medial Axis Transform (MAT): Provides an alternative skeleton definition. The skeleton is the union of cneteres of maximal discs within X. A maximal disc is a circular subset of X that touches the boundary in at least two places. The MAT is the skeleton with the maximal disc radius retained at each point.



Chapter 3

Convolution and Filtering

Image Filtering The modification of pixels in an image based on some function of a local neighborhood of the pixels



Linear Shift-invariant Filtering: Modification of pixels based on neighborhood using a linear combination of neighbors. Shift invariant means we do the same for each pixel. With these filters we can:

- Low level image processing operations
- smoothing and noise reduction
- sharpen
- detection and enhancement

Linear Filter: L is a linear operation if

$$L[\alpha l_1 + \beta l_2] = \alpha L[l_1] + \beta L[l_2]$$

The output l' of linear image operation is a weighted sum of each pixel in the input I.

$$I'(x,y) = \sum_{(i,j)\in N(x,y)} K(x,y;i,j)I(i,j)$$

Where:

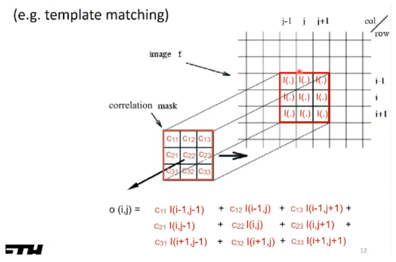
- I is the input image
- I' the output,
- K the kernel of the operation
- N(m,n) is the neighbourhood of (m,n)

If the operations are **shift-invariant** then K does not depend on (x,y) and we use the same weights everywhere

3.1 Correlation

Linear operation of correlation:

$$I' = K \circ I$$
$$I'(x,y) = \sum_{(i,j)\in N(x,y)} K(i,j)I(x+i,y+j)$$

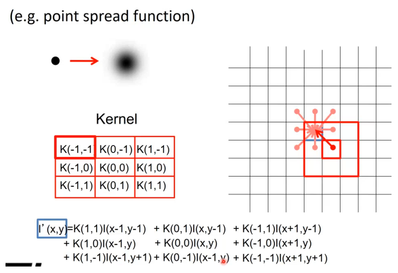

(e.g. template matching)
j-1
j
j+1
col
row
image f
correlation mask
i-1
i
i+1
I(.)
c11 c12 c13
c21 c22 c23
c31 c32 c33
o (i,j) = c11 I(i-1,j-1) + c12 I(i-1,j) + c13 I(i-1,j+1) +
c21 I(i,j-1) + c22 I(i,j) + c23 I(i,j+1) +
c31 I(i+1,j-1) + c32 I(i+1,j) + c33 I(i+1,j+1)
12

3.2 Convolution

Convolution is a blurring function. Convolution is a linear operation:

$$I' = K \circ I$$
$$I'(x, y) = \sum_{(i,j)\in N(x,y)} K(i, j)I(x - i, y - j)$$

The difference between Convolution and correlation is that the kernel is reversed. If your kernel is symmetric the Convolution is equivalent to Correlation.



K(-1,-1	K(0,-1)	K(1,-1)
K(-1,0)	K(0,0)	K(1,0)
K(-1,1)	K(0,1)	K(1,1)

I' (x,y)=K(1,1)I(x-1,y-1) + K(0,1)I(x,y-1) + K(-1,1)I(x+1,y-1)
+ K(1,0)I(x-1,y) + K(0,0)I(x,y) + K(-1,0)I(x+1,y)
+ K(1,-1)I(x-1,y+1) + K(0,-1)I(x-1,y) + K(-1,-1)I(x+1,y+1)

For the edge of the image we have no data, hence applying the filter around the edge will shrink the image. Applying many filters to an image would eventually cause the image to disappear. In order to prevent this we must make some assumptions of the what the filter should apply when at the edge of the image. Possible methods are :

- clip filter
- wrap around
- copy edge
- reflect across edge
- vary filter near edge

3.3 Separable Kernels

Have the following attribute:

$$K(m, n) = f(m)g(n)$$

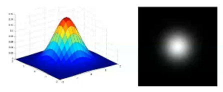
Hence the Kernel can be written as the product of 2 vectors. For a rectangular neighbourhood with size $(2M + 1) * (2N + 1)$

$$I'(m, n) = f \cdot (g \cdot I(N(m, n)))$$
$$I''(m, n) = \sum_{j-=M}^{M} g(j) \cdot I(m, n - j)$$

Separating the Kernels has a computational advantage because vector matrix multiplication is more efficient.

3.4 Gaussian Kernel

The idea is to have the weight contributions of neighboring pixels proportional to their nearness



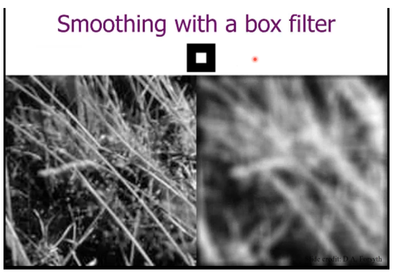
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

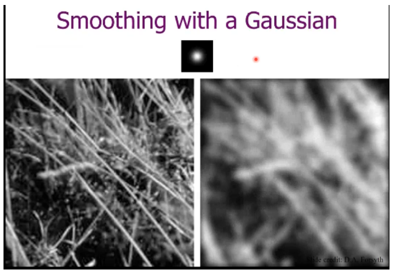
5 x 5, σ = 1

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

The constant factor at the front of G_σ makes the volume sum to 1

Smoothing with Gaussian vs Box Filter





Comparing the two filters, the box image still shows artifacts. The reason being that when applying the box filter convolution on the image we spread out the values such that they get fainter, yet it is still a box, hence we still get a strong edge effect. The fact that the filter itself has strong edges is a problem.

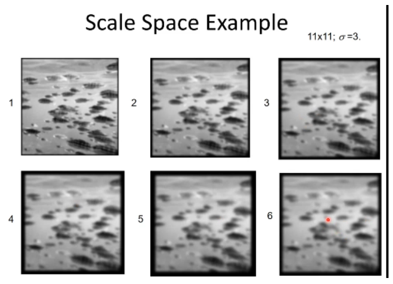
The gaussian kernel is separable and becomes the product of 2 one dimensional gaussians. i.e

$$g(x, y) = g(x)g(y)$$

The amount of smoothing depends on σ and the kernel size. The top benefits of the Gaussian smoothing kernel:

- Rotationally symmetric
- Has a single lobe (Neighbors influence decreases monotonically)
- Still one lobe in frequency domain (no corruption from high frequencies)
- Simple relationship to σ
- Easy to implement efficiently

Scale Space A Convolution of a Gaussian with standard deviation σ with itself is a Gaussian with standard deviation of $\sigma\sqrt{2}$. The repeated convolution by a Gaussian filter produces the scale space of an image.



Differential FIlters:

Prewitt operator:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel operator:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

We can see that both of these filters average out (sum of the elements = 0), hence in homogeneous parts of the image the response of the filter will be zero. If we sth that is brighter on one side than the other then we will have a larger response. (for vertical edges). The Sobel operator has more weight on the central pixel hence its a smoother response.

High-pass filters

High-pass filters

Laplacian operator:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

High-pass filter:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

For these the highest response is given from isolated pixels surrounded by pixels of high contrast

Filters are templates Filter at some point can be seen as taking a dot-product between the image and some vector. Filtering the image corresponds to a set of dot products. Filters look like the effects they are intended to find and will get the largest response to regions which look like the filter

Image Sharpening (Enhancement) Increases the high frequency components to enhance edges. Sharpening has the following structure:

$$I' = I + \alpha|k \cdot I|$$

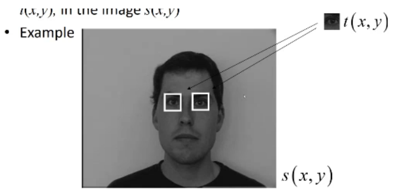
where k is a high-pass filter kernel and α is a scalar in $[0, 1]$

Chapter 4

Image Features

4.1 Template Matching:

Locate an onject, described by a template $t(x, y)$ in the image $s(x, y)$



We search for the best match by minimizing mean-squared error

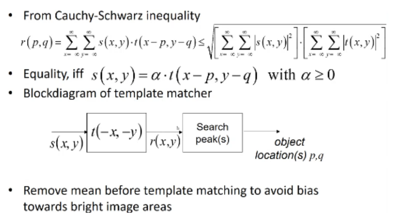
$$E(p, q) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} [s(x, y) - t(x - p, y - q)]^2$$

$$= \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} |s(x, y)|^2 + \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} |t(x, y)|^2 - 2 \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} s(x, y)t(x - p.y - q)$$

In the last equation the terms are constant with the exception of the last one, which has the form of the correlation operator. Hence the problem is equivalent too maximizing area correlation:

$$r(p, q) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} s(x, y)t(x - p, y - q) * t(-p, -q)$$

Area correlation is equivalent to convolution of image $s(x, y)$ with-impulse response $t(-x, -y)$.



- From Cauchy-Schwarz inequality

$$r(p,q) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} s(x,y) \cdot t(x-p, y-q) \leq \sqrt{\left[\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} |s(x,y)|^2\right] \cdot \left[\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} |t(x,y)|^2\right]}$$

- Equality, iff $s(x,y) = \alpha \cdot t(x-p, y-q)$ with $\alpha \geq 0$
- Blockdiagram of template matcher

$$\begin{tikzcd} {} \arrow[r, "s(x,y)"'] & \boxed{t(-x,-y)} \arrow[r, "r(x,y)"'] & \boxed{\text{Search peak(s)}} \arrow[r] & \text{object location(s) } p,q \end{tikzcd}$$

- Remove mean before template matching to avoid bias towards bright image areas

4.2 Edge detection

An Edge is a region of strong change in intensity, hence we look for a region with a large gradient. The gradient of an image is the local change of the image. The size of the gradient is given by:

$$|grad(f(x, y))| = \sqrt{\frac{\delta f}{\delta x}^2 + \frac{\delta f}{\delta y}^2}$$

Since we are only interested in finding the edge we do not need to worry about the direction. The following image shows examples of edge detection filters.

$$\text{Prewitt}\quad \begin{pmatrix} -1 & 0 & 1 \\ -1 & [0] & 1 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -1 & -1 \\ 0 & [0] & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\text{Sobel}\quad \begin{pmatrix} -1 & 0 & 1 \\ -2 & [0] & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 & -1 \\ 0 & [0] & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

$$\text{Roberts}\quad \begin{pmatrix} [0] & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} [1] & 0 \\ 0 & -1 \end{pmatrix}$$

The left matrix is the derivative of x and the right is the derivative of y. An edge has a sign. Either it can go from bright to dark or dark to bright which have the inverse gradient orientation.

Laplacian operator Another method to detect discontiuities i.e edges. We now consider the second derivative.

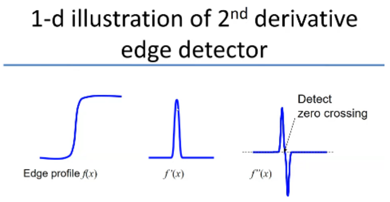
$$\nabla^2 f(x,y) = \frac{\delta^2 f(x,y)}{\delta x^2} + \frac{\delta^2 f(x,y)}{\delta y^2}$$

The laplacian operator has the following properties:

- Isotropic (rotationally invariant)
- Zero-crossings mark edge location

We can make a discrete space approximation by convoluting with the 3x3 impulse response:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & [-4] & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & [-8] & 1 \\ 1 & 1 & 1 \end{pmatrix}$$



The challenge with detecting zero crossing is that images have alot of noise and hence there could be many small oscillations causing zero crossings which we are not interested in. We can reduce some of the noise by blurring the image first (e.g with a gaussian filter). The Laplacian operator responds equally to strong and weak edges, hence we can also solve this problem by looking at the gradient magnitude (i.e the first derivative), if the magnitude is above a certain value we then mark the zero crossing as an edge.

Blurring of an image with Gaussian and Laplacian operator can be combined into a convolution with Laplacian of Gaussian **(LoG)** operator which has the following form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

4.3 Canny edge detector

Works in 5 steps:

1. Smooth image with a Gaussian filter
2. Compute gradient magnitude and direction of gradient

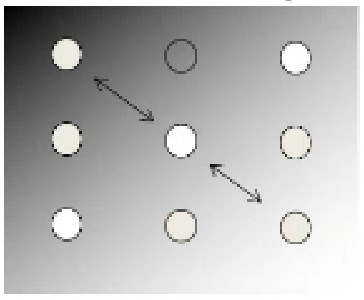
$$M(x,y) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \alpha(x,y) = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

3. Apply nonmaxima supression to gradient magnitude image
4. Double thresholding to detect strong and weak edge pixels
5. Reject weak edge pixels not connected with strong edge pixels

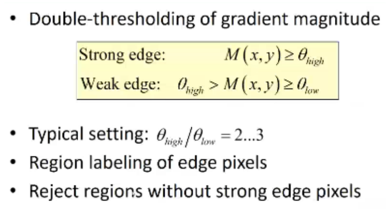
We Quantize the edge normal (direction of the edge) to one of four directions

- horizontal
- -45°
- vertical
- +45°

For nonmaxima suppression we find the direction of the largest gradient and if $M(x, y)$ is smaller than either of its neighbors in the edge normal direction then we suppress it otherwise we keep it.



Canny thresholding has the following form:



- Double-thresholding of gradient magnitude

Strong edge:	$M(x,y) \geq \theta_{high}$
Weak edge:	$\theta_{high} > M(x,y) \geq \theta_{low}$

- Typical setting: $\theta_{high}/\theta_{low} = 2...3$
- Region labeling of edge pixels
- Reject regions without strong edge pixels

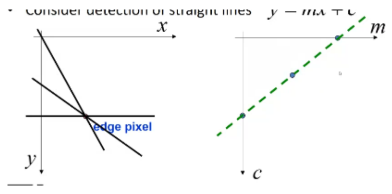
We discard all the weak edges which arent connected to strong edges.

4.4 Hough transform

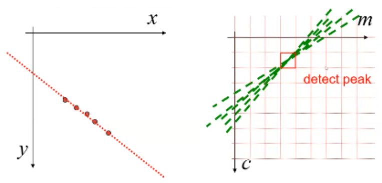
Our goal is to fit a straight line or curve to a set of edge pixels. Lines are defined by the following equation:

$$y = mx + c$$

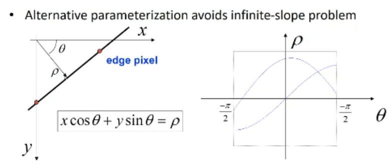
Given a pixel p=(x,y) we can find all parameters m and c such that the line goes through the pixel, we see that these two spaces are symmetric i.e (-c = mx - y) hence the same way a choice of m and c define a line in the x,y space, a choice of x and y will define a line in the parameter space



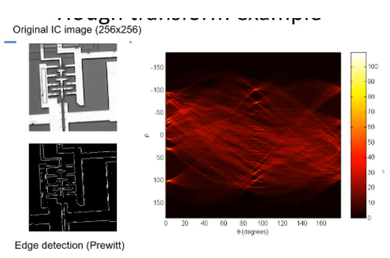
We now subdivide the (m,c) plane into discrete ”bins”, and initialize their count to 0. For each edge pixel x,y we draw a line in the parameter space and increment the bin counts along the line. Our goal is to find a line in the (x,y) space, which contains as many lines as possible in the (m,c) space, hence we need to find the bin in the (m,c) space which has the highest count (peak)



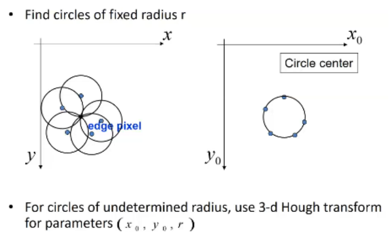
The downside to this choice of parameter space is, that if you have a vertical line then it cannot be expressed with this parameter space. Hence we use an alternative parameterization which avoids this infinite slope problem.


Alternative parameterization avoids infinite-slope problem
x
θ
ρ
edge pixel
$x\cos\theta + y\sin\theta = \rho$
y
ρ
$\frac{-\pi}{2}$
$\frac{-\pi}{2}$
θ

example: Given an image, we first run edge detection on it, then draw up the edges in the Hough space (seen on the right)



The peaks in the Hough space in the image above are the two sets of orthogonal lines. The length of the line (in pixels) is approximately the number of counts the peak has. **Circle detection by Hough transform**



- Find circles of fixed radius r

- For circles of undetermined radius, use 3-d Hough transform for parameters (x_0, y_0, r)

Chapter 5

Fourier Transform