Pattern recognition by transformation

Laura Fernández Robles
University of León, Spain
l.fernandez@unileon.es

Contents

Fourier transform

FT: Fourier series

Jean Baptiste Joseph Fourier (1768-1830):

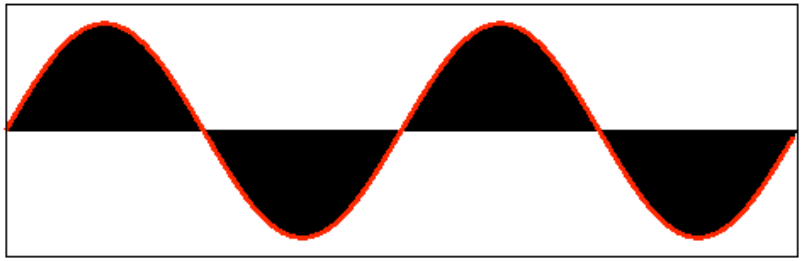
Any periodic function can be expressed as a sum of sinus and cosinus functions (of varying amplitudes and frequencies)

FT: Fourier series

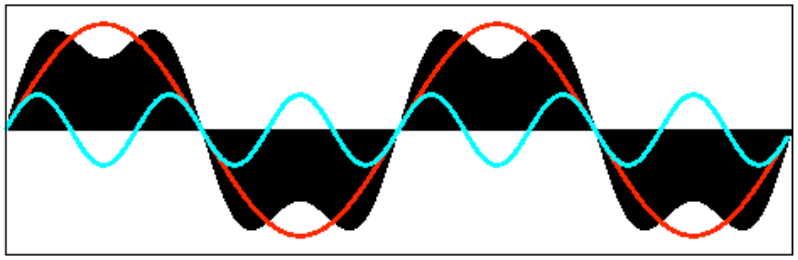
[M. Handley]

Representing a square wave as a sum of sinus functions

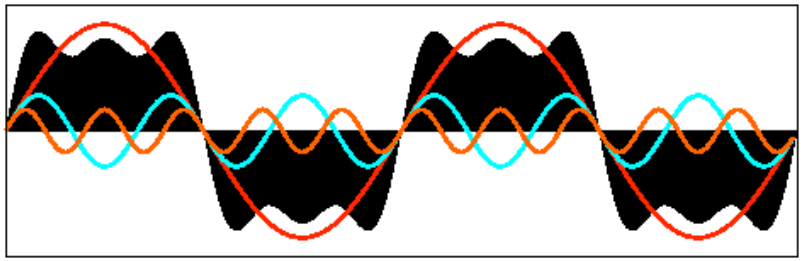
Frequencies: ω



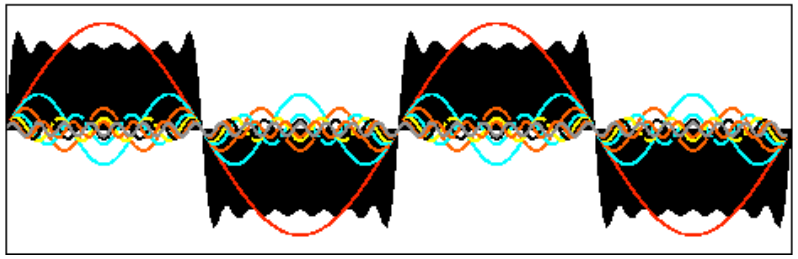
Frequencies: ω + *3*ω



Frequencies: ω + *3*ω + *5*ω



Frequencies: ω + *3*ω + *5*ω +…+ *15*ω

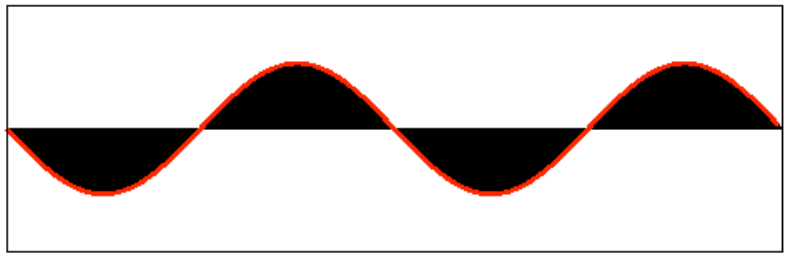


FT: Fourier series

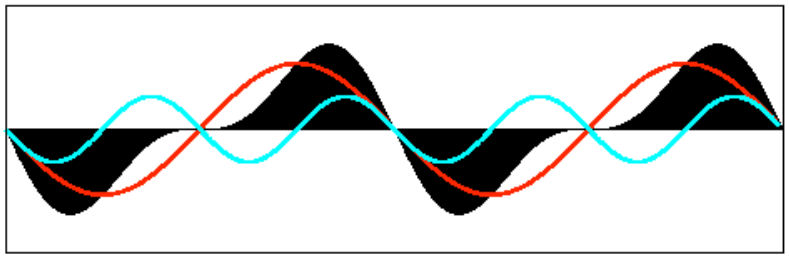
[M. Handley]

Representing a sawtooth wave as a sum of sinus functions

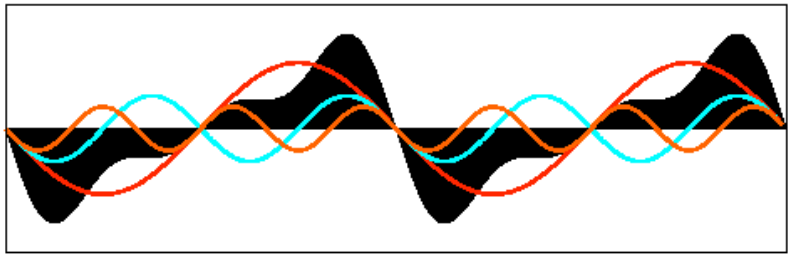
Frecuencies: ω



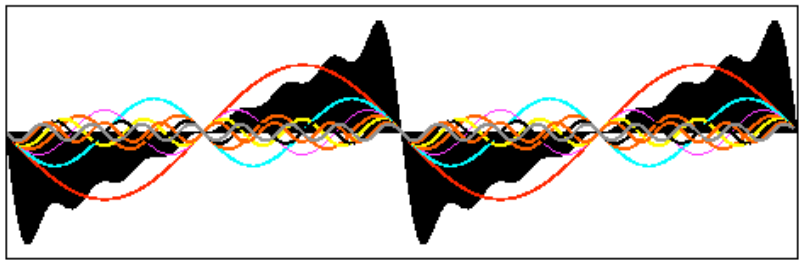
Frecuencies: ω + *2*ω



Frecuencies: ω + *2*ω + *3*ω



Frecuencies: ω + *2*ω + *3*ω +…+ *8*ω



FT: Fourier series

A function **f(x)** can be expressed as a sum of sinus and cosinus functions:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx),$$

Where the coefficients are computed as follows:

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$

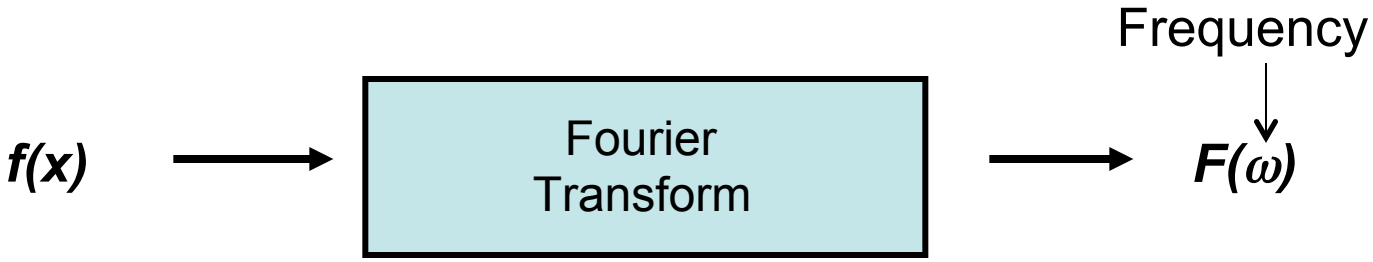
$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)dx$$

$$n = 1, 2, 3, ...$$

FT: Fourier transform

Fourier Series can be generalized to complex numbers, and further generalized to derive the Fourier Transform.



For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sinus function -> Complex numbers!

$$A\sin(\omega x + \phi)$$

$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

FT: Fourier transform

The purpose of the Fourier transform is to represent a signal as a linear combination of sinus signals of various frequencies.

Even functions that are not periodic (but whose area under the curve is finite) can be expressed as a weighted sum (integral) of sinus and/or cosinus functions

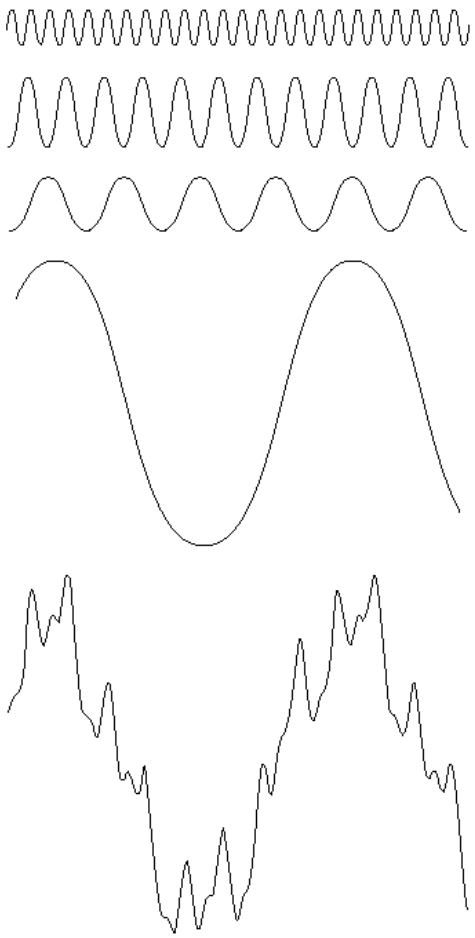


Figure: The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 was met with skepticism.

FT: Fourier transform

Fourier transform: maps a function of time (eg. signal, audio) into the frequency domain

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dk$$

Inverse Fourier transform: maps a function in the frequency domain back into the time domain.

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk$$

Note: $e^{xi} = \cos(x) + i \sin(x)$

FT: Discrete Fourier transform

When we wish to find the frequency spectrum of a discrete time series that we have *sampled*

Discrete Fourier Transform (DFT)

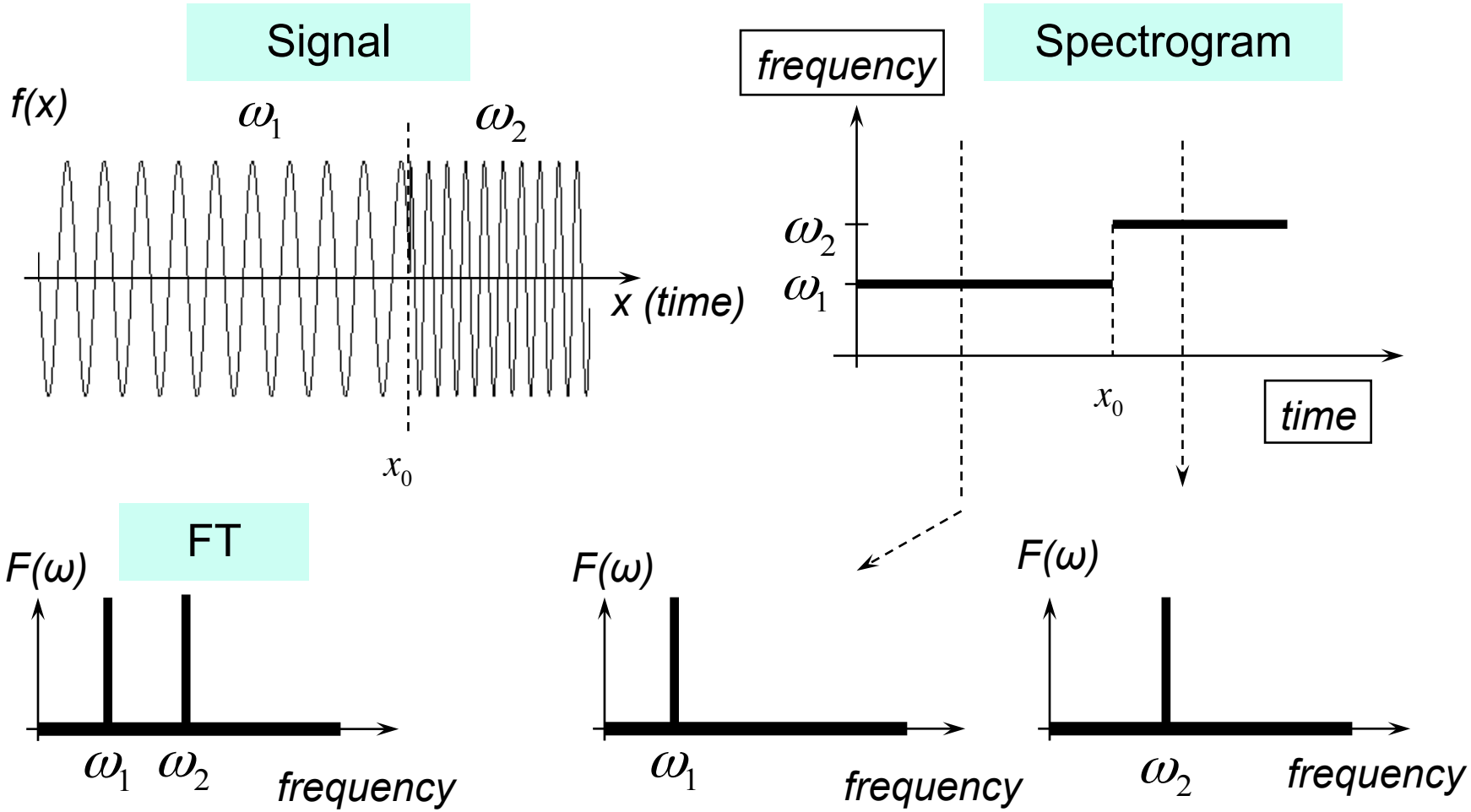
$$F_n = \sum_{k=0}^{N-1} f_k e^{-2\pi i n k/N}$$

Inverse Discrete Fourier Transform (IDFT)

$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{-2\pi i k n/N}$$

FT: Identification of musical notes [R. Cristi]

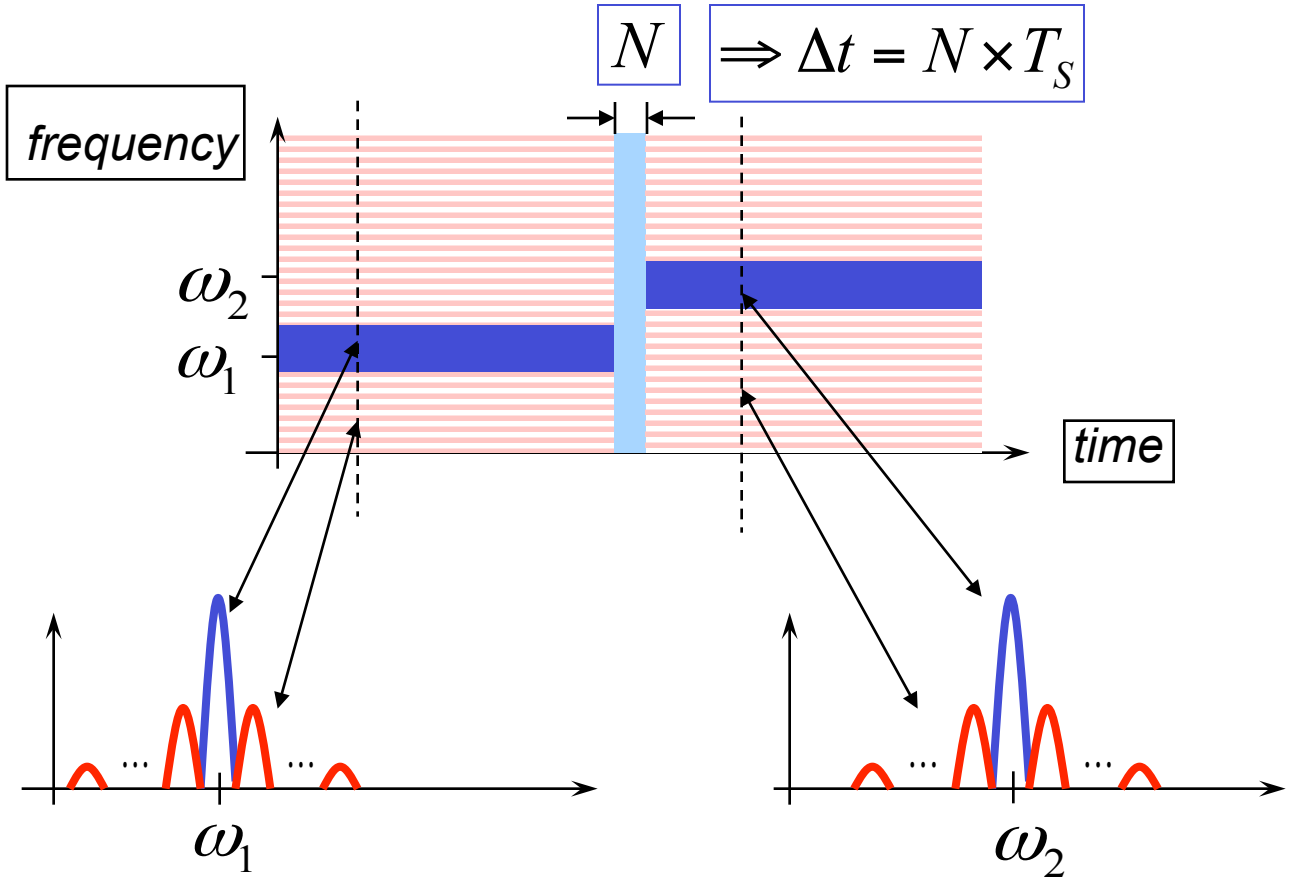
Ideally we would like to have this:



FT: identification of musical notes

[R. Cristi]

In a less ideal situation:

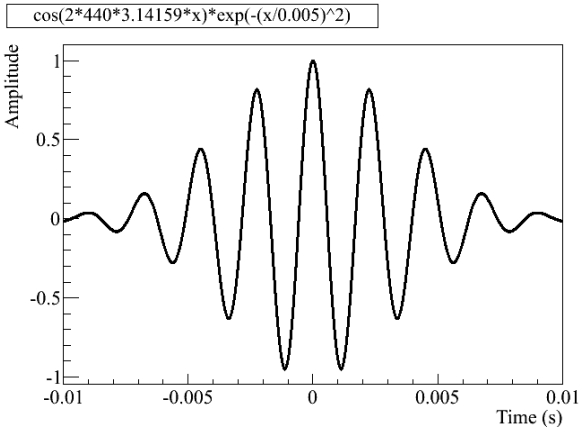


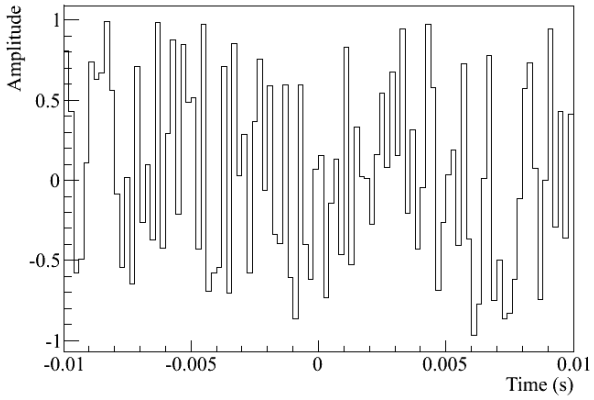
FT: Identification of musical notes

The root **note A** is a sound wave with a frequency of 440 Hz.

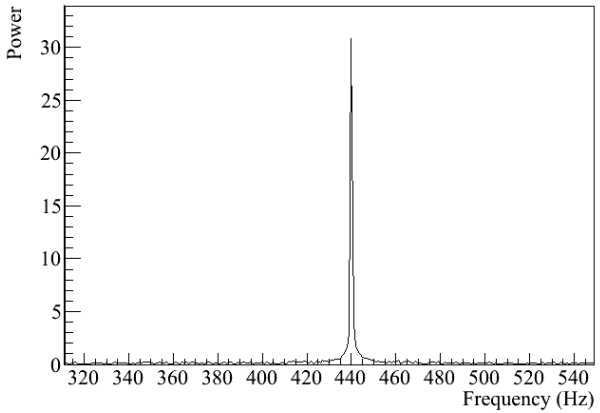
White noise is a random signal with a constant power spectral density (combination of all frequencies).

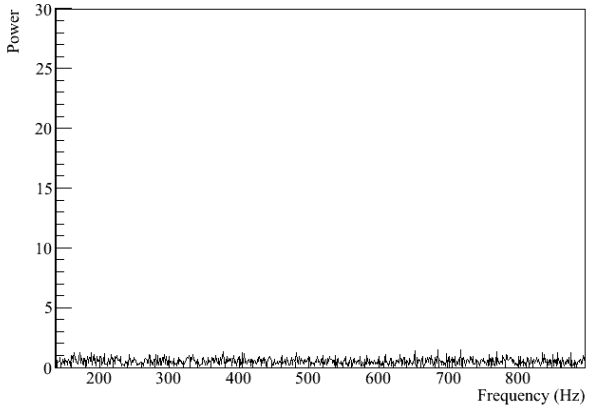
Signal





Fourier transform



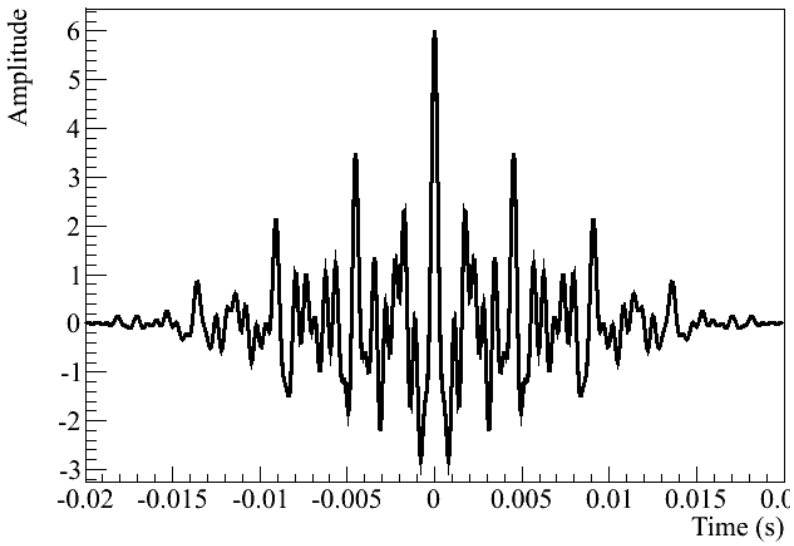


http://physicsformom.blogspot.com.es/2009/11/fourier-analysis-2-more-complicated.html

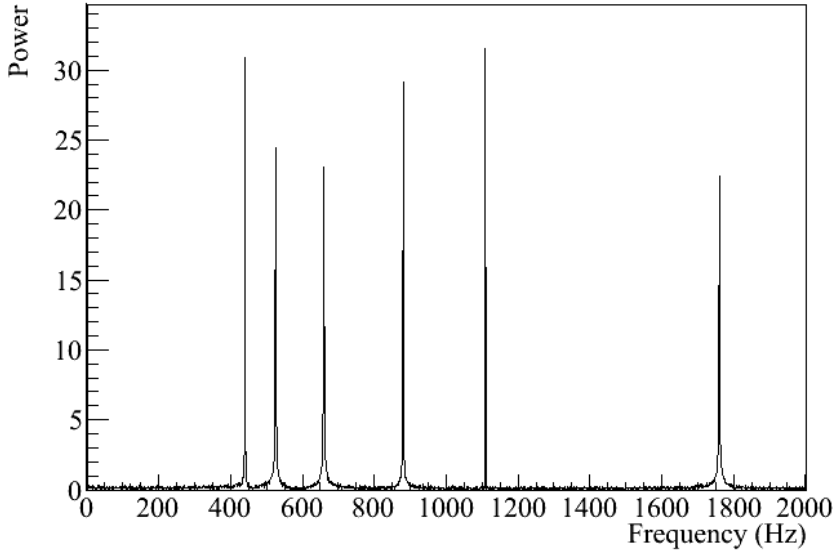
FT: Identification of musical notes

The full **A chord** (as produced by the string of guitar).

Signal

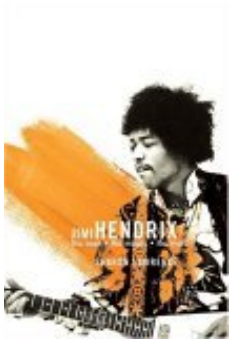


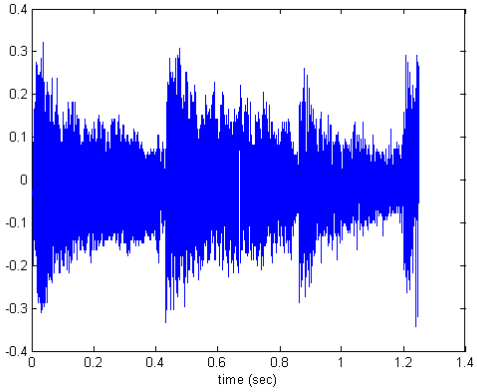
Fourier transform



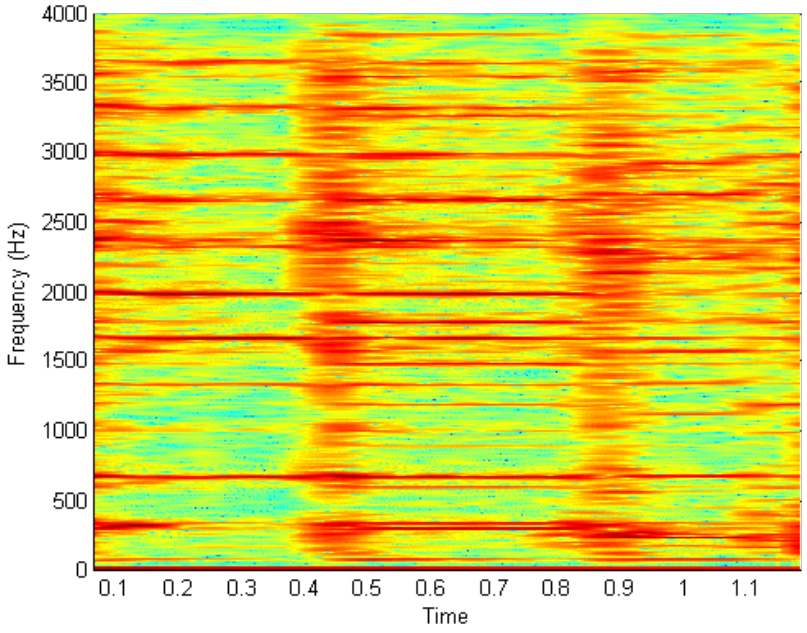
http://physicsformom.blogspot.com.es/2009/11/fourier-analysis-2-more-complicated.html

FT: identification of musical notes [R. Cristi]





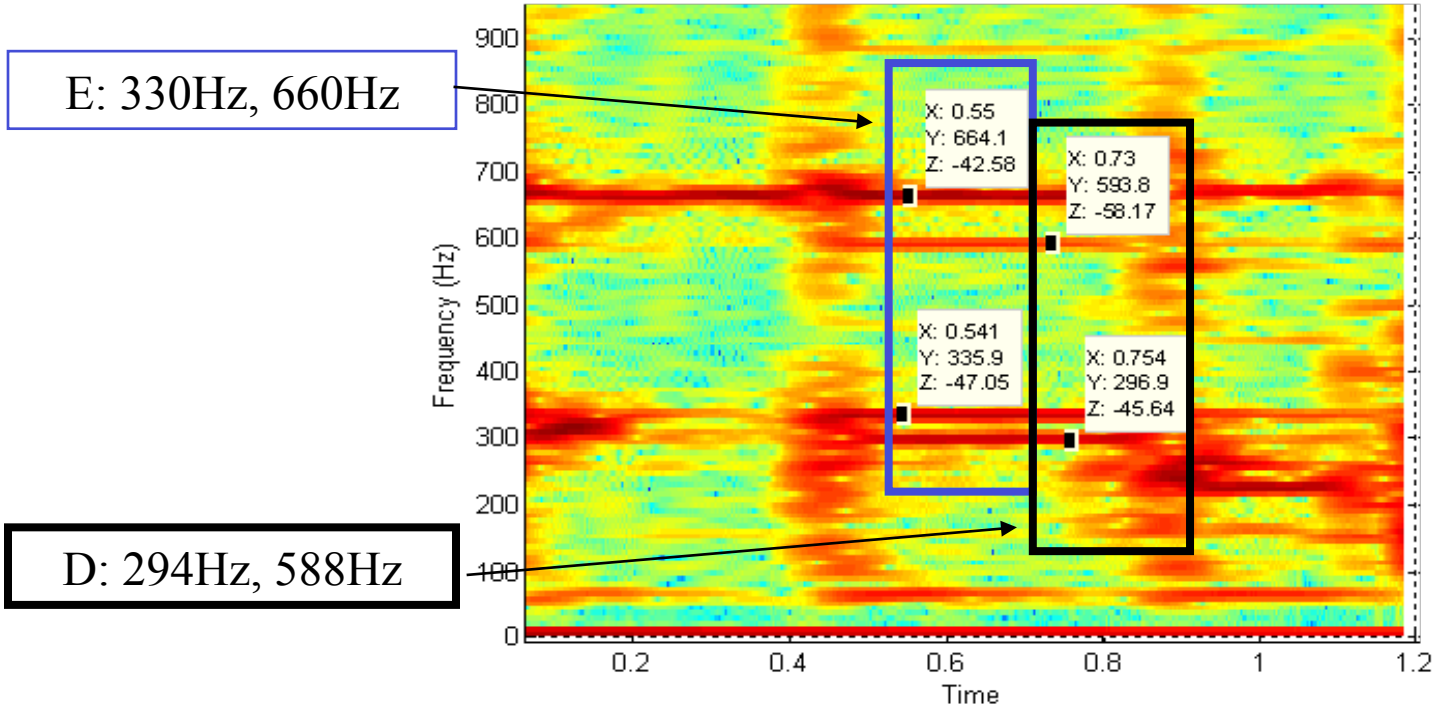
spectrogram(y(12001:22000), hamming(1024), 1000, 1024,'yaxis');



FT: Identification of musical notes [R. Cristi]

Closest notes

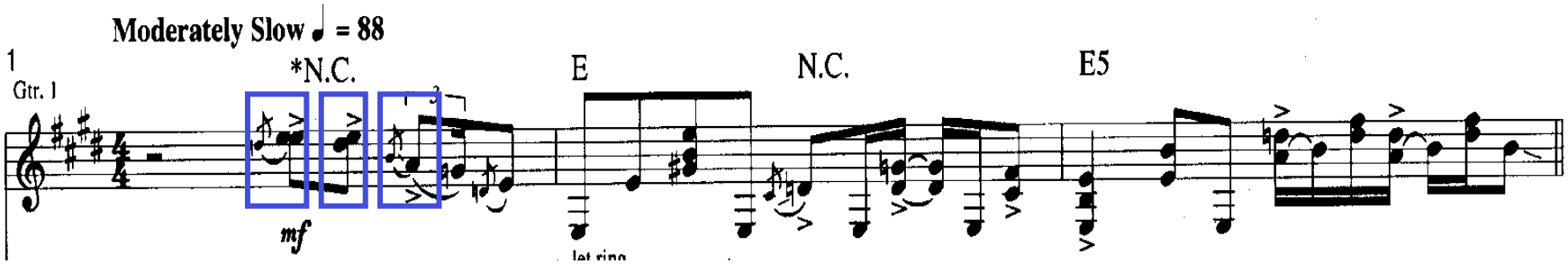
spectrogram(y(12001:22000), hamming(1024), 1000, 1024,'yaxis');

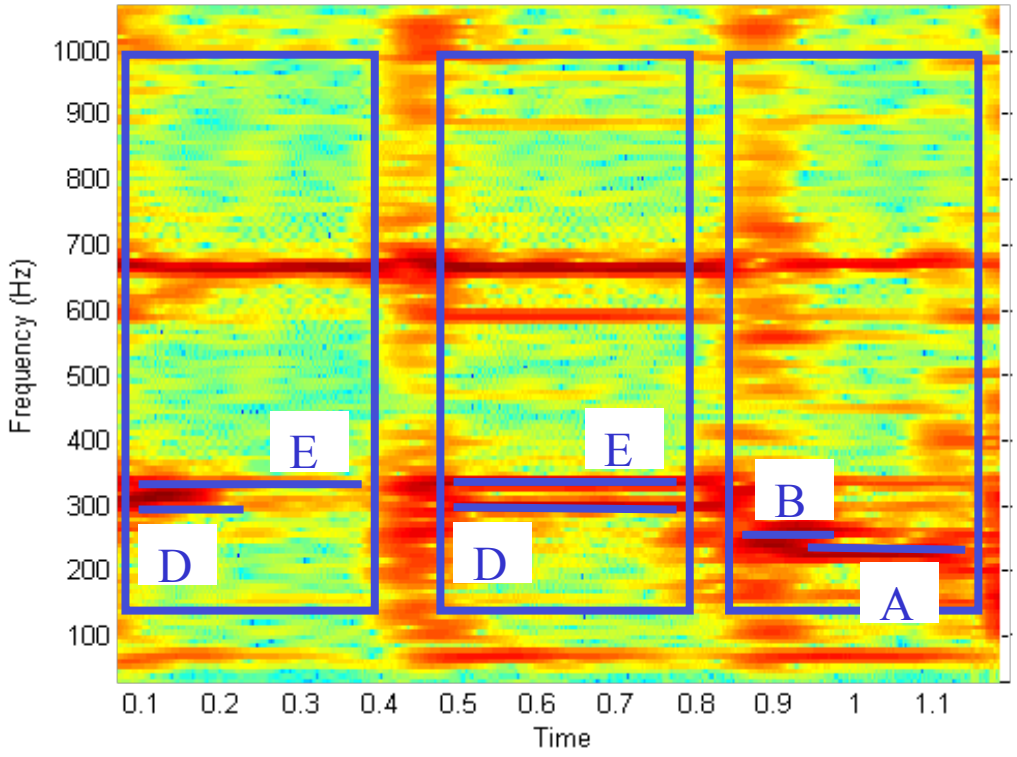


Notes	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
Freq. (Hz)	262	277	294	311	330	349	370	392	415	440	466	494

FT: identification of musical notes

[R. Cristi]





FT: Summary

Fourier transform represents a signal as a linear combination of sinus signals of various frequencies.

Some patterns (notes, chords) can be recognized easier in the Fourier domain than in the spatial or time domains.

Some processing tasks can be done with greater simplicity in the Fourier domain than in the spatial or time domains. For example: a convolution becomes a multiplication.

Typical procedure:

(1) Apply FT to transform a function (1D - signal, 2D – image, …) to the frequency domain.

(2) Carry the task(s) in the frequency domain.

(3) Apply IFT to return to the spatial or time domain (if needed).

Hough transform

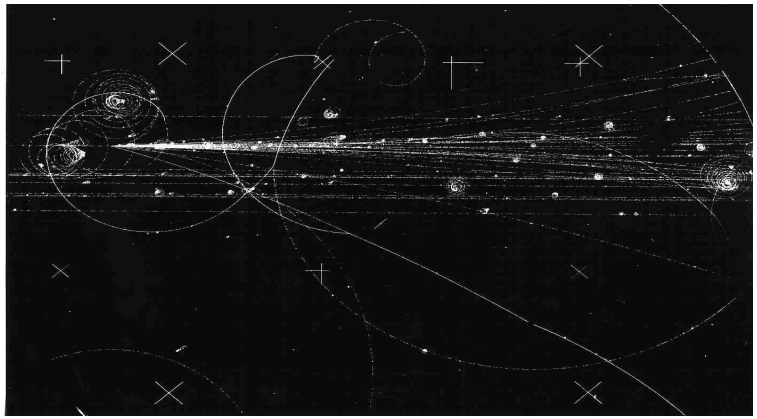
Hough transform for finding lines

HT: Introduction

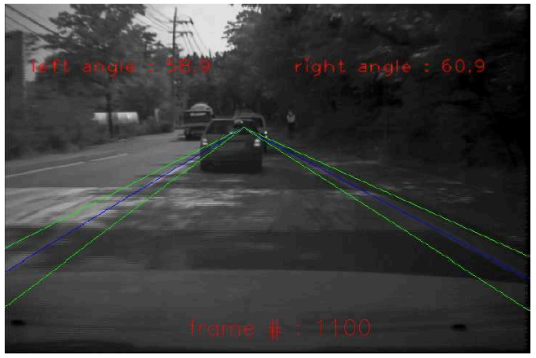
[Kristen Grauman]

Why find lines?

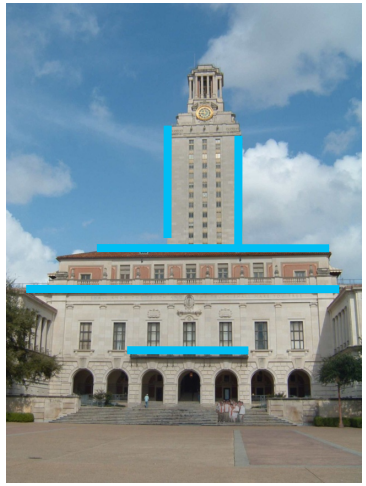
Many objects characterized by presence of straight lines.



Bubble chamber



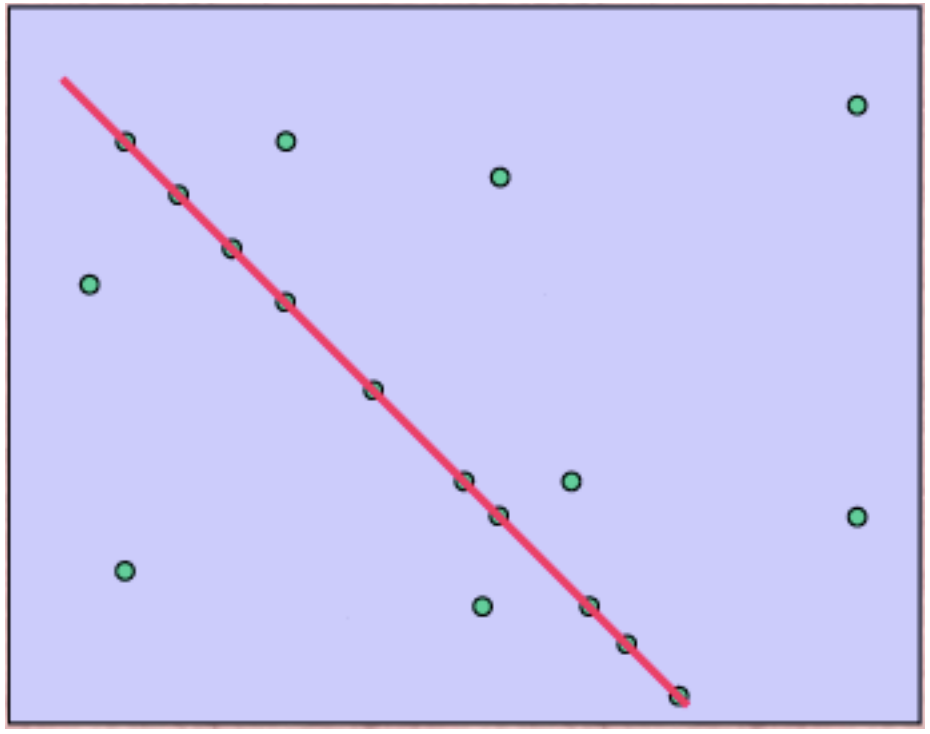
Remote landing lane localization



Wait, do we just not apply edge detection?

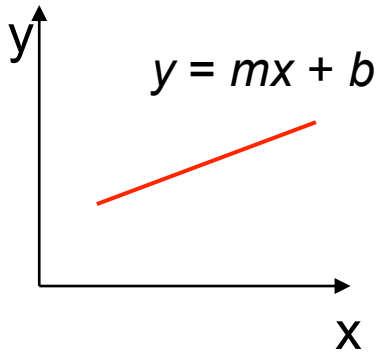
HT: The problem

Given a set of points in 2-D, find if a sub-set of these points, fall on a LINE.



HT: Parameter space representation

A line has two parameters (m,b)



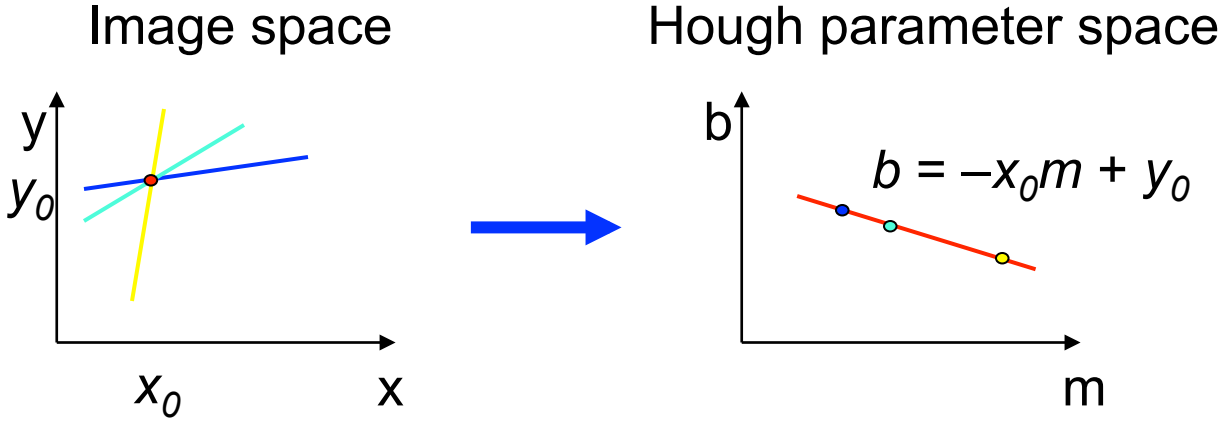
Line equation

$y = mx + b$

Rewrite this equation

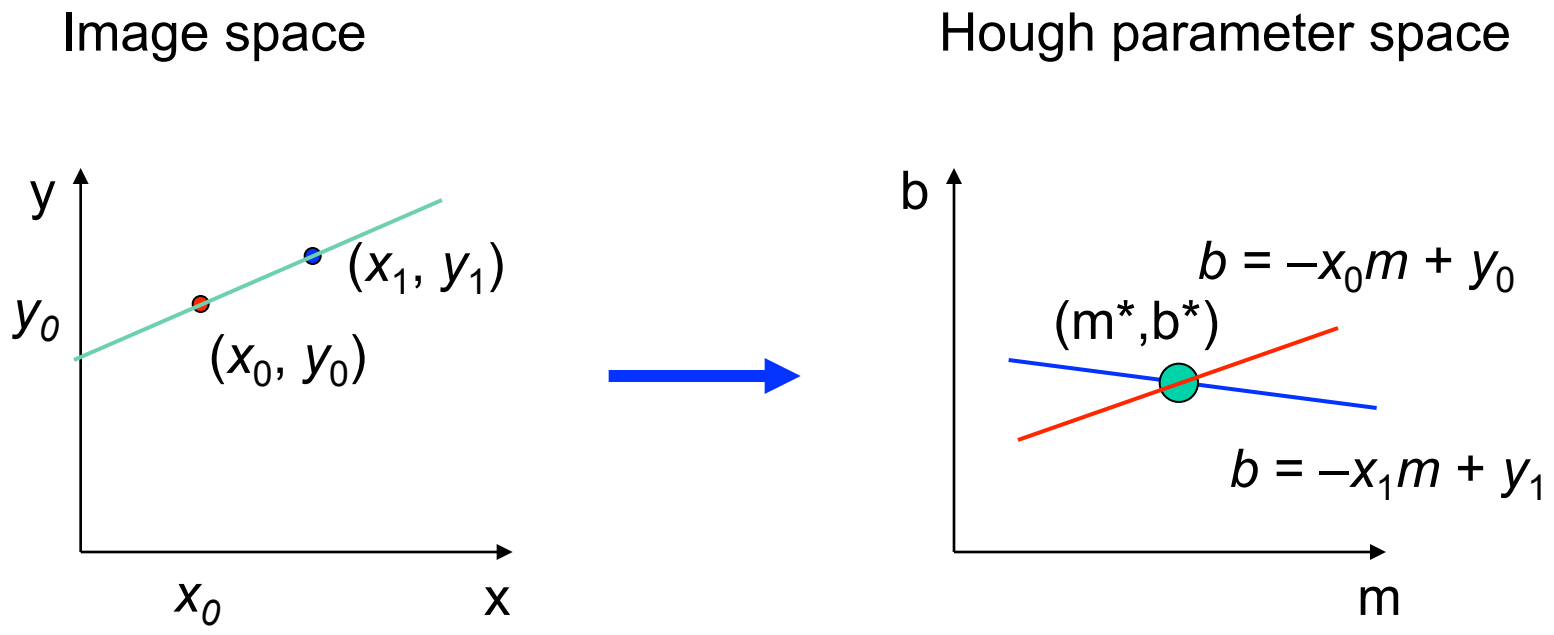
$b = -xm + y$

A point (x_0, y_0) in the image space is a line in Hough space $b = -x_0m + y_0$. This line is formed by the possible combinations of parameters (a,b) of lines that pass through the point (x,y)



HT: Parameter space representation

All points on a line in image space, yield lines in parameter space which intersect at a common point (m*,b*) of the parameters of the line in the space domain



HT: Hough transform algorithm

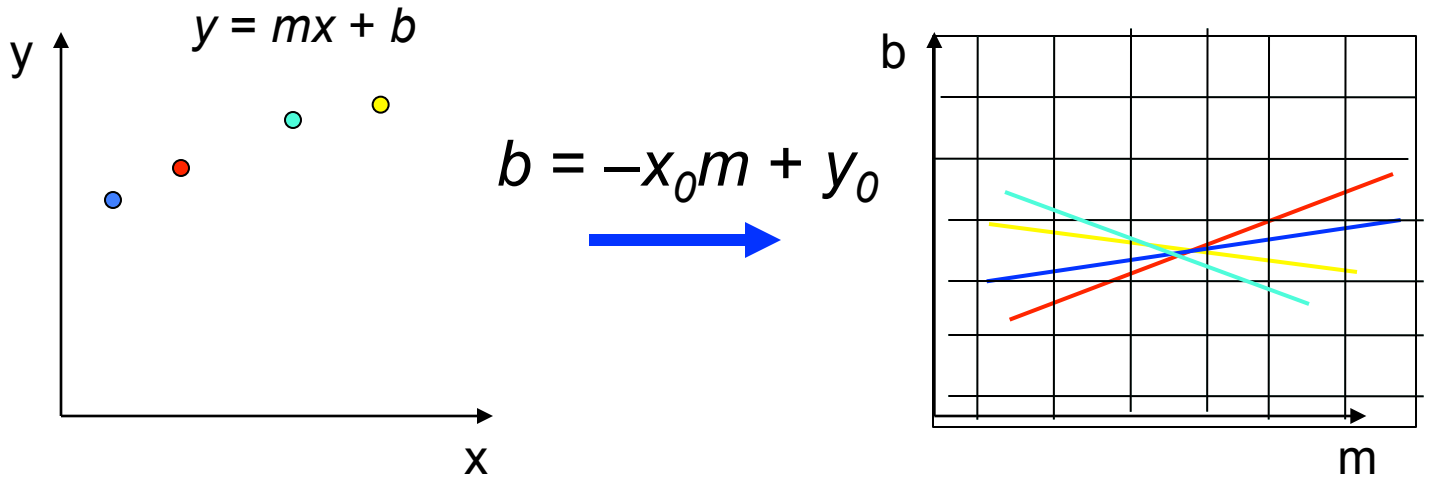


Image space

Hough parameter space

How can we find the most likely parameters (m,b) for a line in the image space?

- Let each edge point in image space *vote* for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

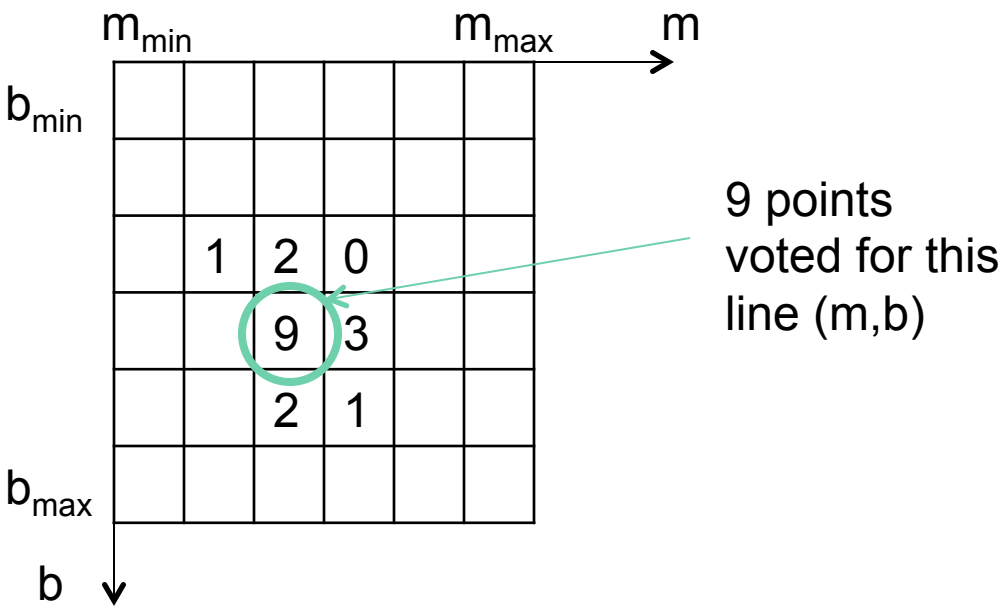
HT: Hough transform algorithm

- Quantize the parameter space H[m_{min},…m_{max},b_{min},…b_{max}]
- Initialize accumulator H(m,b) to all zeros
- For each feature point (x,y) in the image increment all cells that satisfy $b = -xm + y$

  For ($m = m_{min}$, $m \leq m_{min}$, m++)

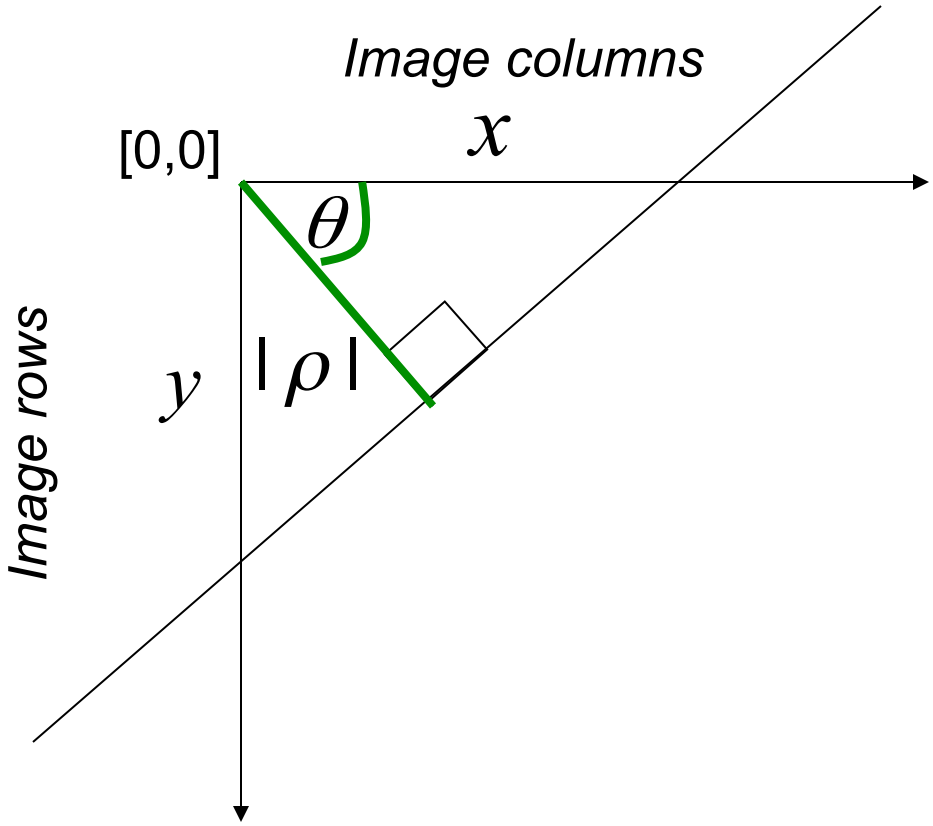
  $b = –xm + y$

  H(m,b) = H(m,b) + 1
- Local maxima in H(m,b) correspond to lines given by $y = mx + b$



HT: Polar representation of lines

- Avoids infinite slope (vertical lines)
- Constant resolution



$$\rho = x\cos\theta + y\sin\theta$$

$|\rho|$: smallest distance from line to origin

θ : angle the perpendicular makes with the x-axis (angle of the line and the y-axis)

Point in image space → sinusoid curve segment in Hough space

HT: Hough transform algorithm - Polar

- Quantize the parameter space H[$\rho_{min}, \ldots \rho_{max}$, $\theta_{min}, \ldots \theta_{max}$]
- Initialize accumulator H(θ, ρ) to all zeros
- For each feature point (x,y) in the image

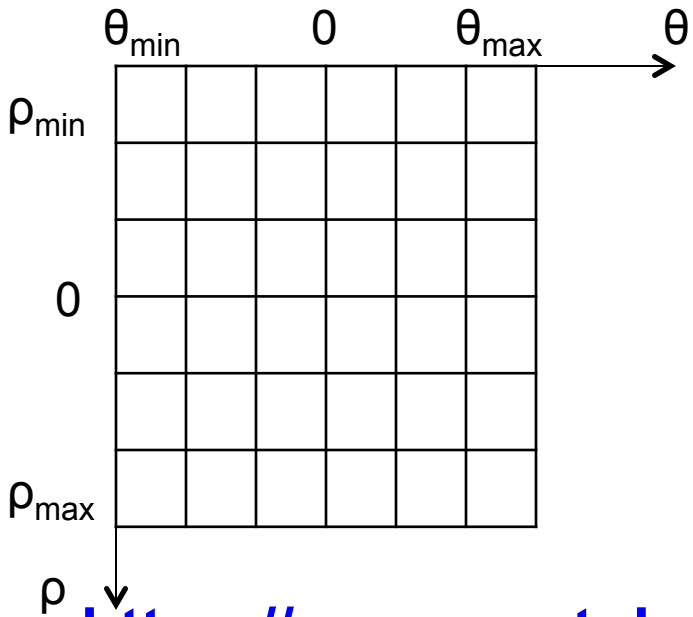
  - For ($\theta = \theta_{min}$, $\theta \leq \theta_{min}$, θ++)

$$\rho = x\cos\theta + y\sin\theta$$

H(θ, ρ) = H(θ, ρ) + 1

- Local maxima in H(θ, ρ) correspond to lines given by

$$\rho = x\cos\theta + y\sin\theta$$



Conventions:

ρ_{min} = - diagonal image

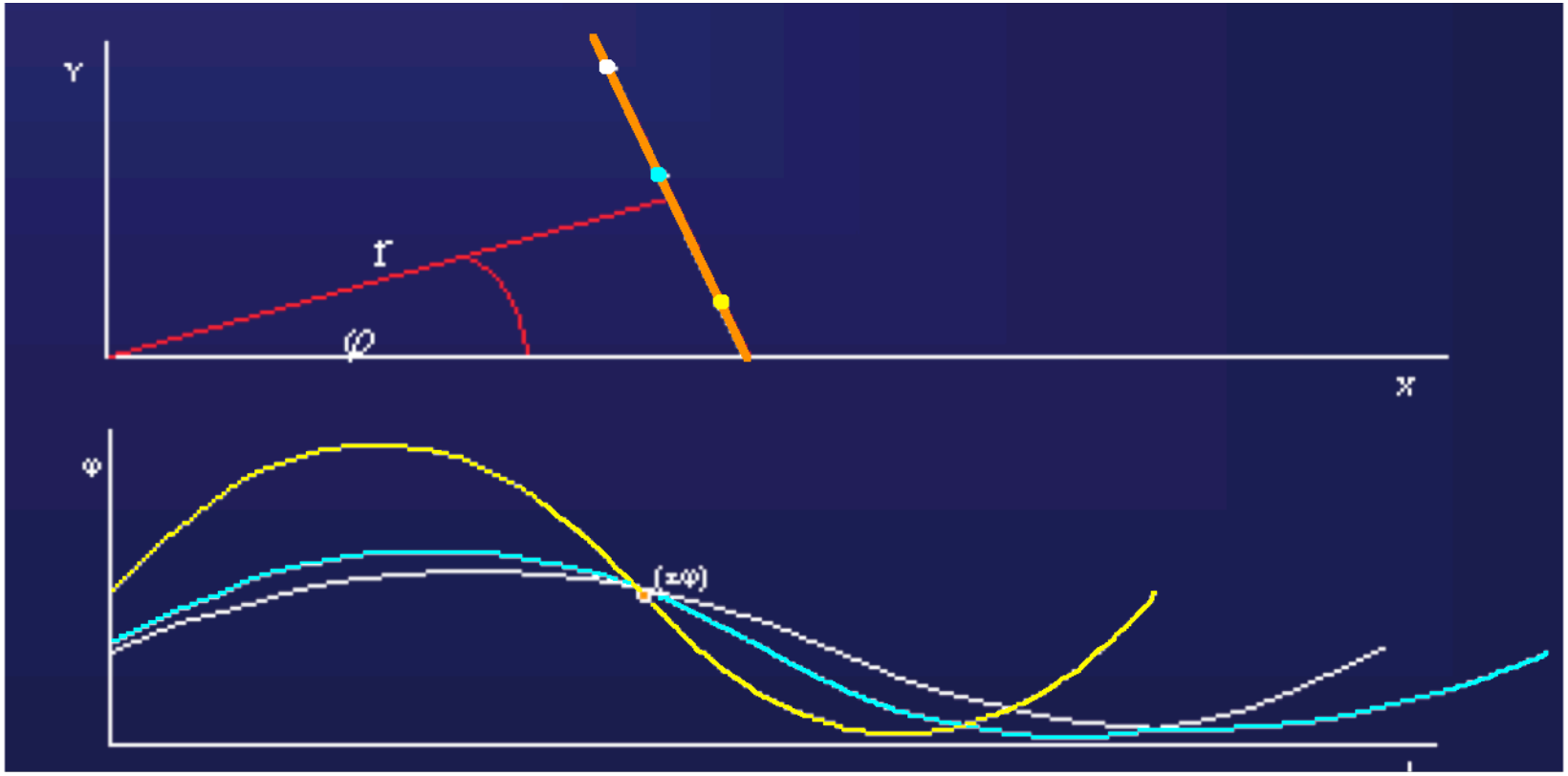
ρ_{max} = diagonal image

θ_{min} = -90º

θ_{max} = +90º

https://www.youtube.com/watch?v=o-n7NoLArcs

HT: Basic illustration

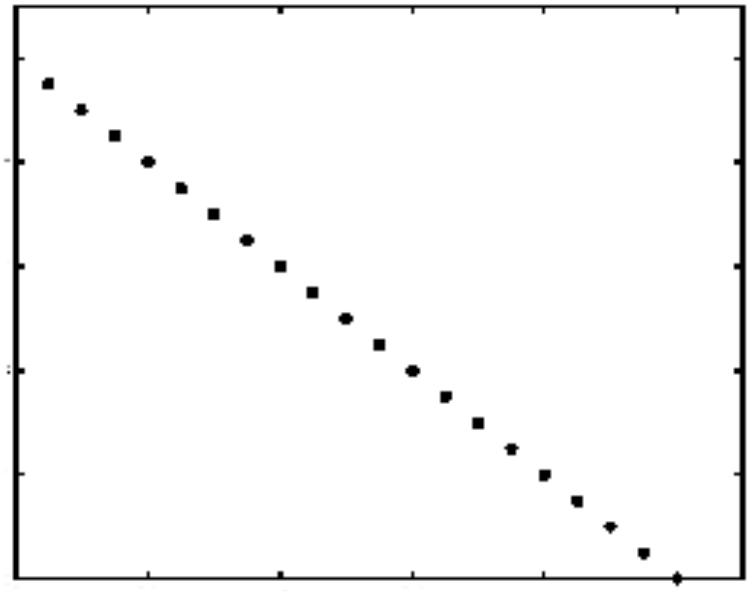


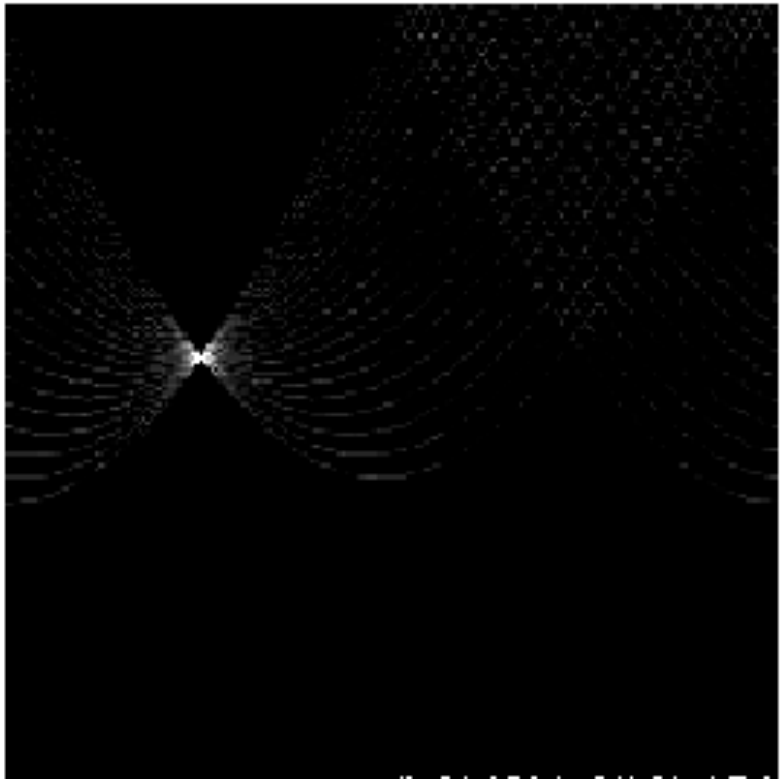
The points on a given line in the polar line parameter space are the votes casted by a given point in the image space

HT: Basic illustration

points

Votes in line parameter space





y

ρ

x

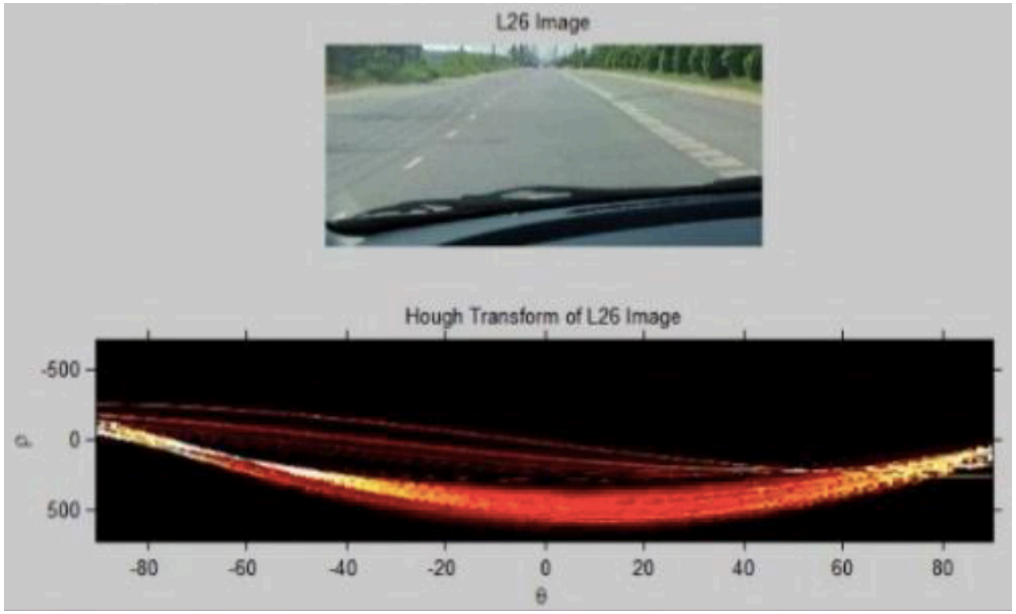
θ

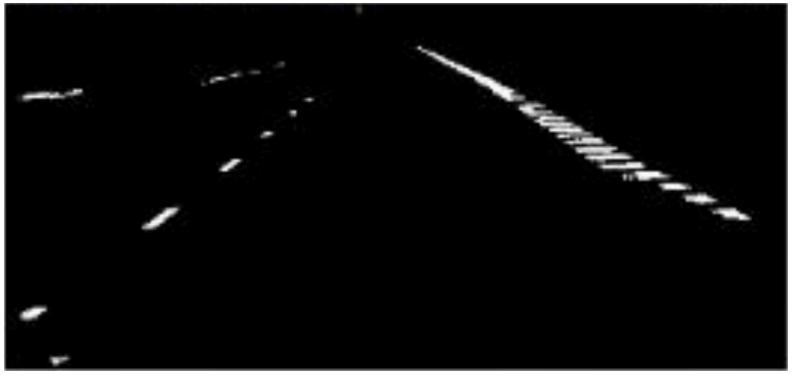
Bright value: high vote count

Black value: no votes

HT: Other shapes

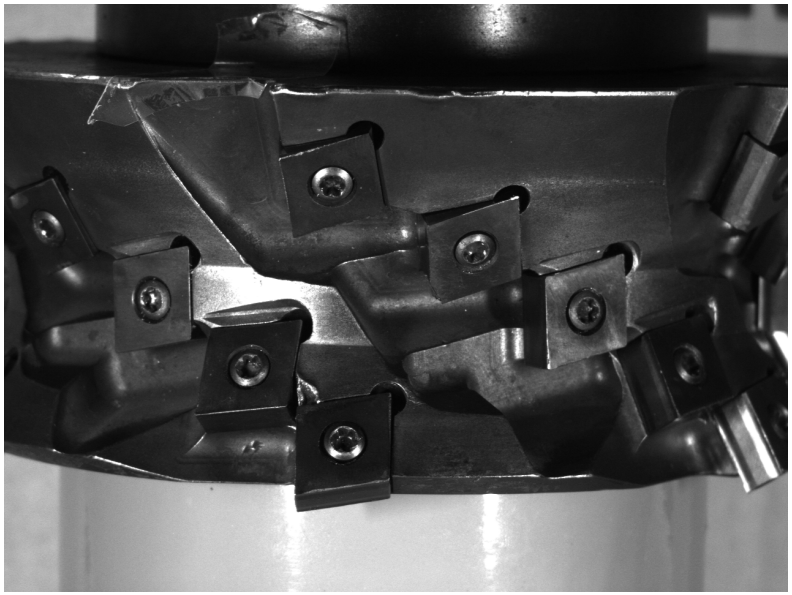
[Ganokratanaa et al., 2013]





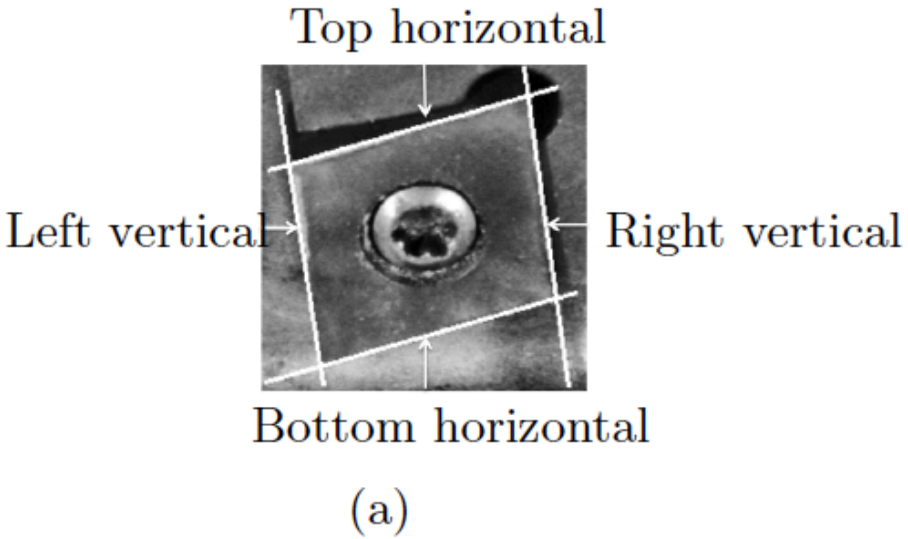
HT: Other shapes

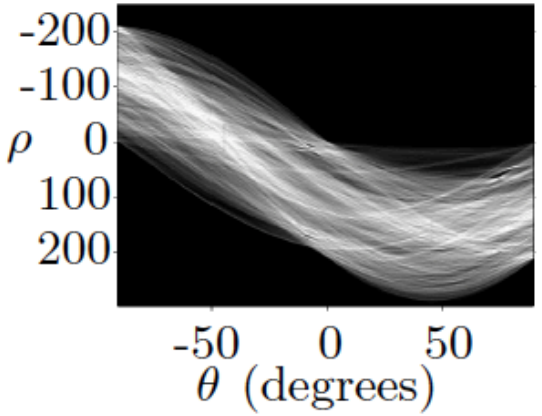
[Fernández-Robles et al., 2015]

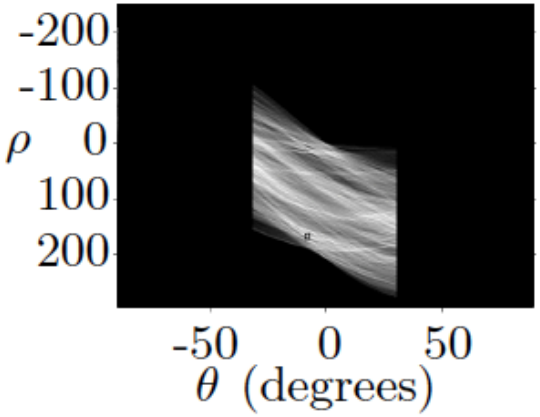


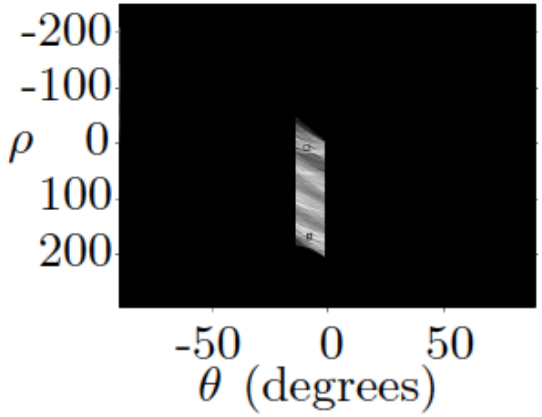
HT: Other shapes

[Fernández-Robles et al., 2015]



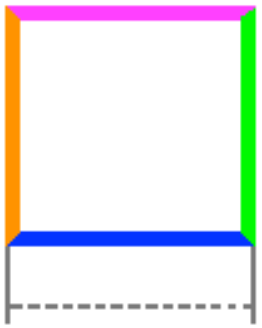




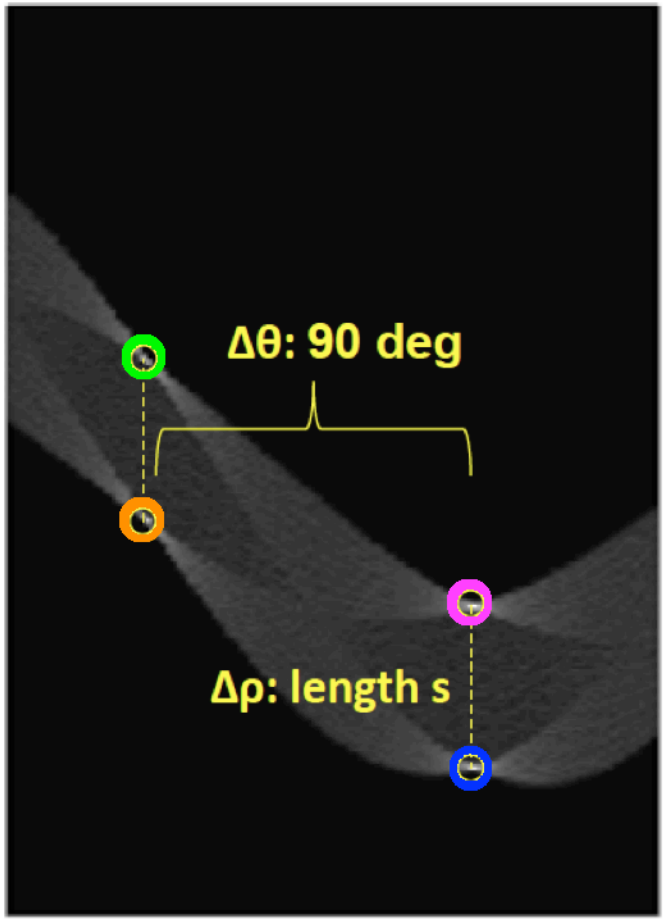


HT: Other shapes

Square

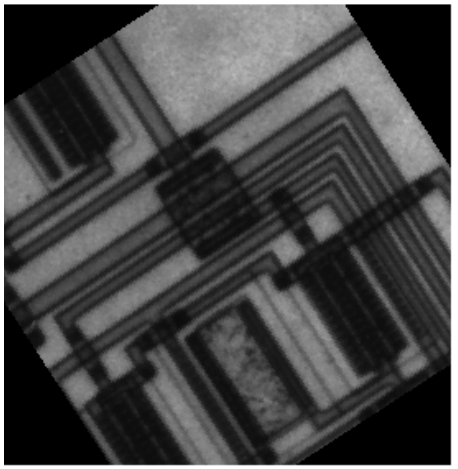


$\Delta\rho$

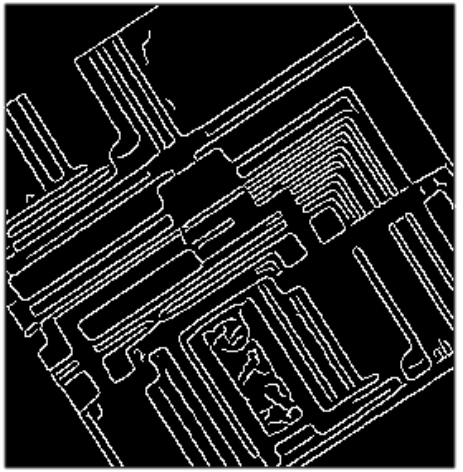


HT: Matlab

```
%READ AN IMAGE
  I = imread('circuit.tif');
  rotI = imrotate(I,33,'crop');
  imshow(rotI)
```



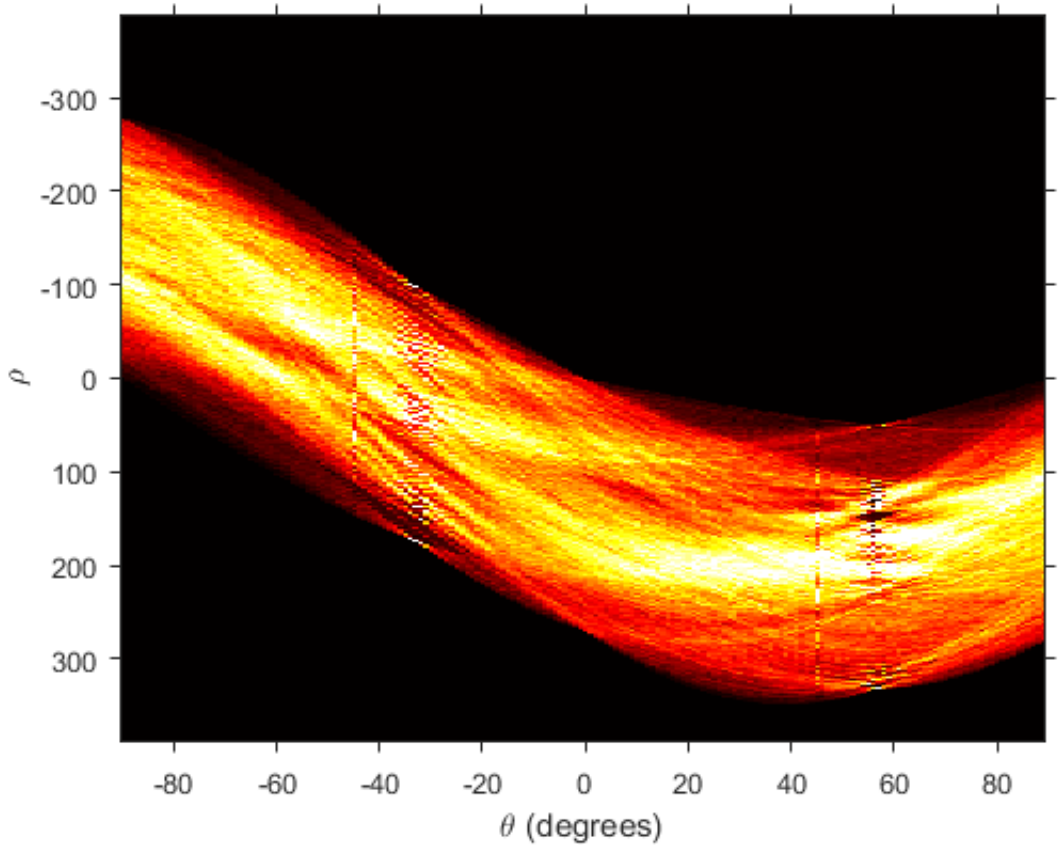
```
%FIND EDGES
  BW = edge(rotI,'canny');
  imshow(BW);
```



```
%COMPUTE THE HOUGH TRANSFORM
  [H,theta,rho] = hough(BW);
```

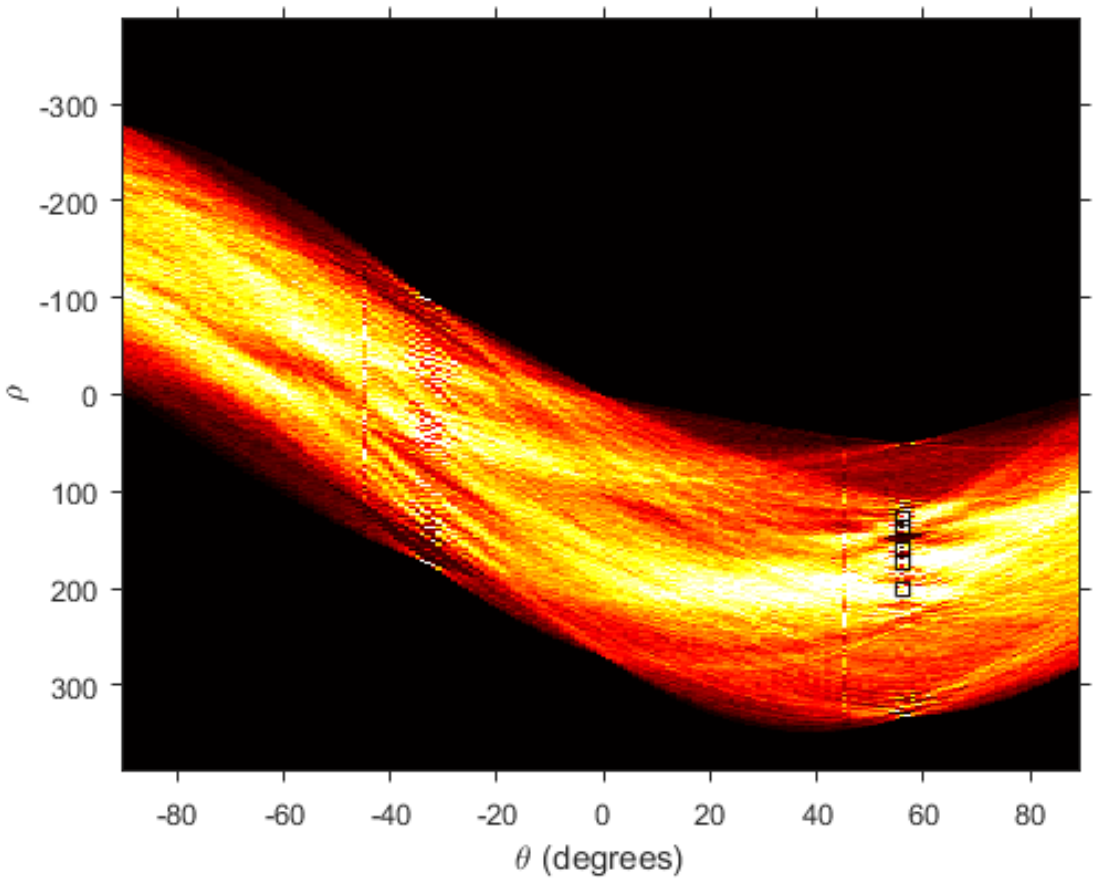
HT: Matlab

```
%DISPLAY THE RESULT OF THE HOUGH TRANSFORM
 figure
 imshow(imadjust(mat2gray(H)),[],...
   'XData',theta,...
   'YData',rho,...
   'InitialMagnification','fit');
 xlabel('\theta (degrees)')
 ylabel('\rho')
 axis on
 axis normal
 hold on
 colormap(hot)
```



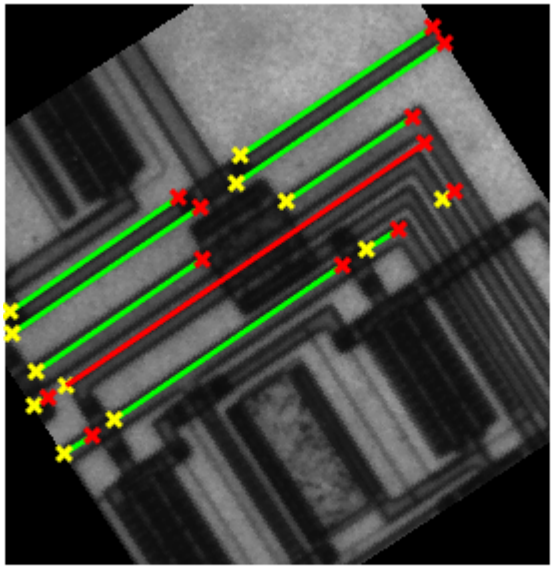
HT: Matlab

```
%FIND PEAKS
  P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));

%SUPERIMPOSE THE PEAKS IN THE HT
  x = theta(P(:,2));
  y = rho(P(:,1));
  plot(x,y,'s','color','black');
```



HT: Matlab

```
%FIND LINES
  lines = houghlines(BW,theta,rho,P,'FillGap',5,'MinLength',7);


%DISPLAY ORIGINAL IMAGE WITH LINES %SUPERIMPOSED
  figure, imshow(rotI), hold on
  max_len = 0;
  for k = 1:length(lines)
     xy = [lines(k).point1; lines(k).point2];
     plot(xy(:,1),xy(:,2),'LineWidth',…
       2,'Color','green');
  end
```



HT: Demo

http://www.activovision.com/octavi/doku.php?id=hough_transform

https://www.youtube.com/watch?v=4zHbI-fFIII

Hough transform for finding circles

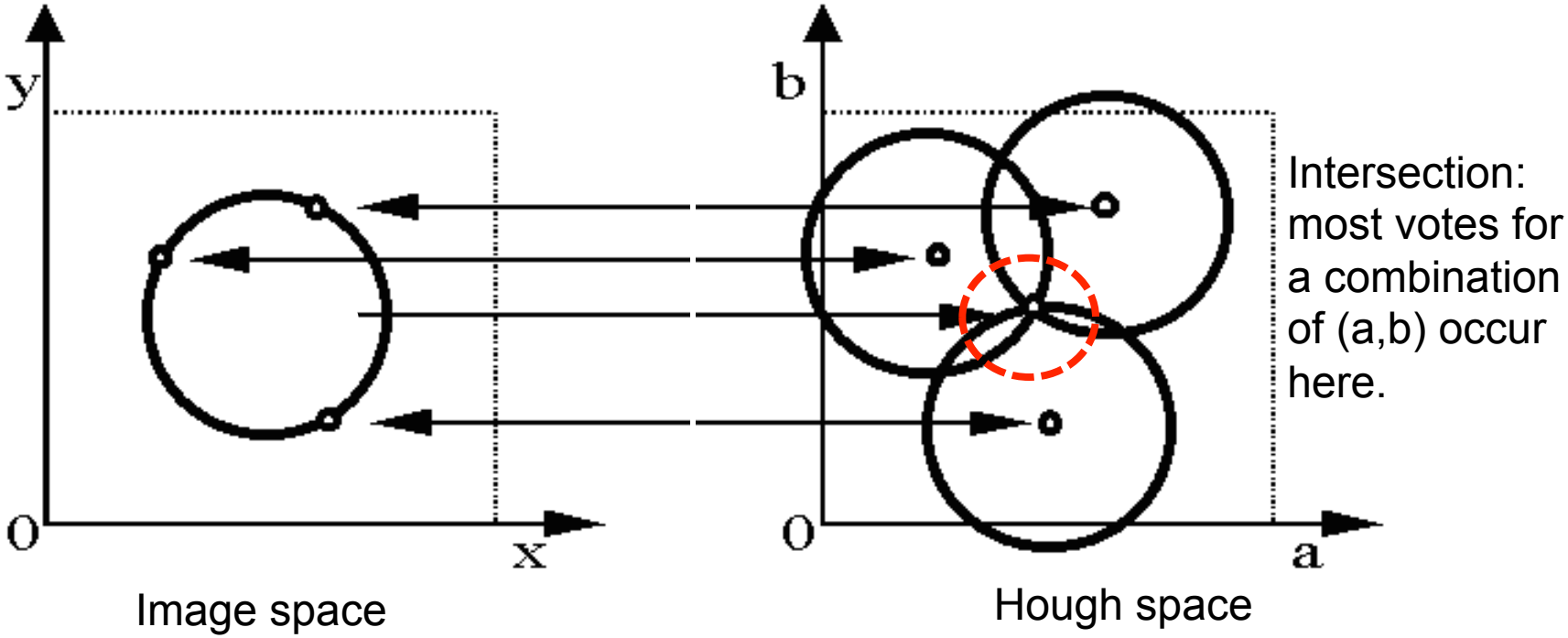
CHT: Hough transform for circles

- Circle: center (a,b) and radius r

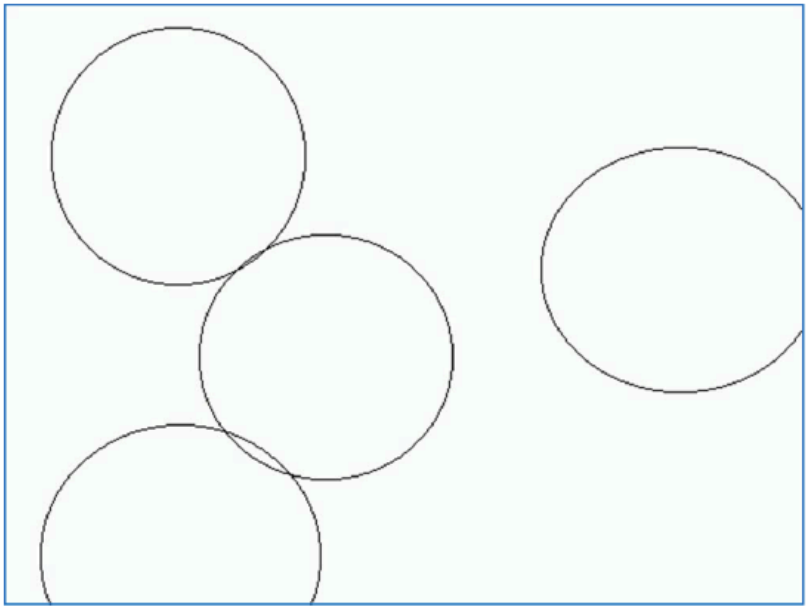
$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

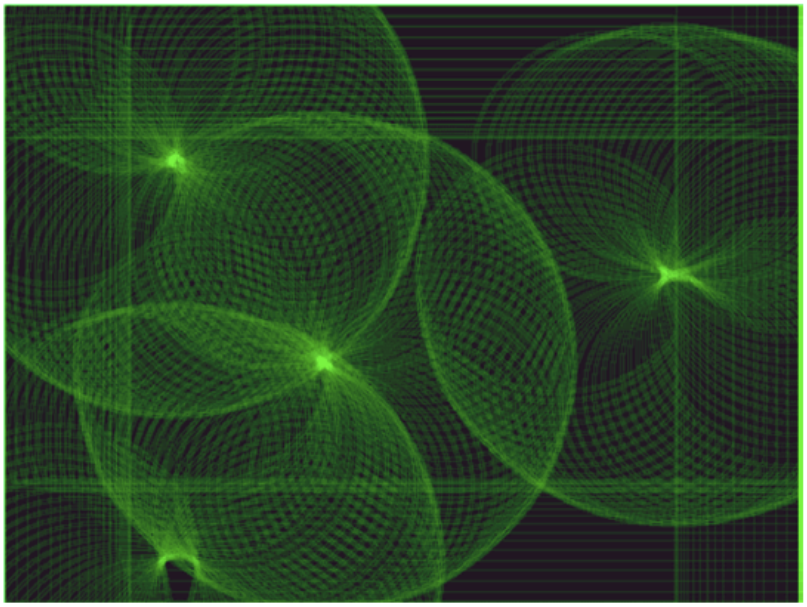
For a fixed radius r and unknown gradient direction.

Accumulator array H(a,b)



CHT: Hough transform for circles



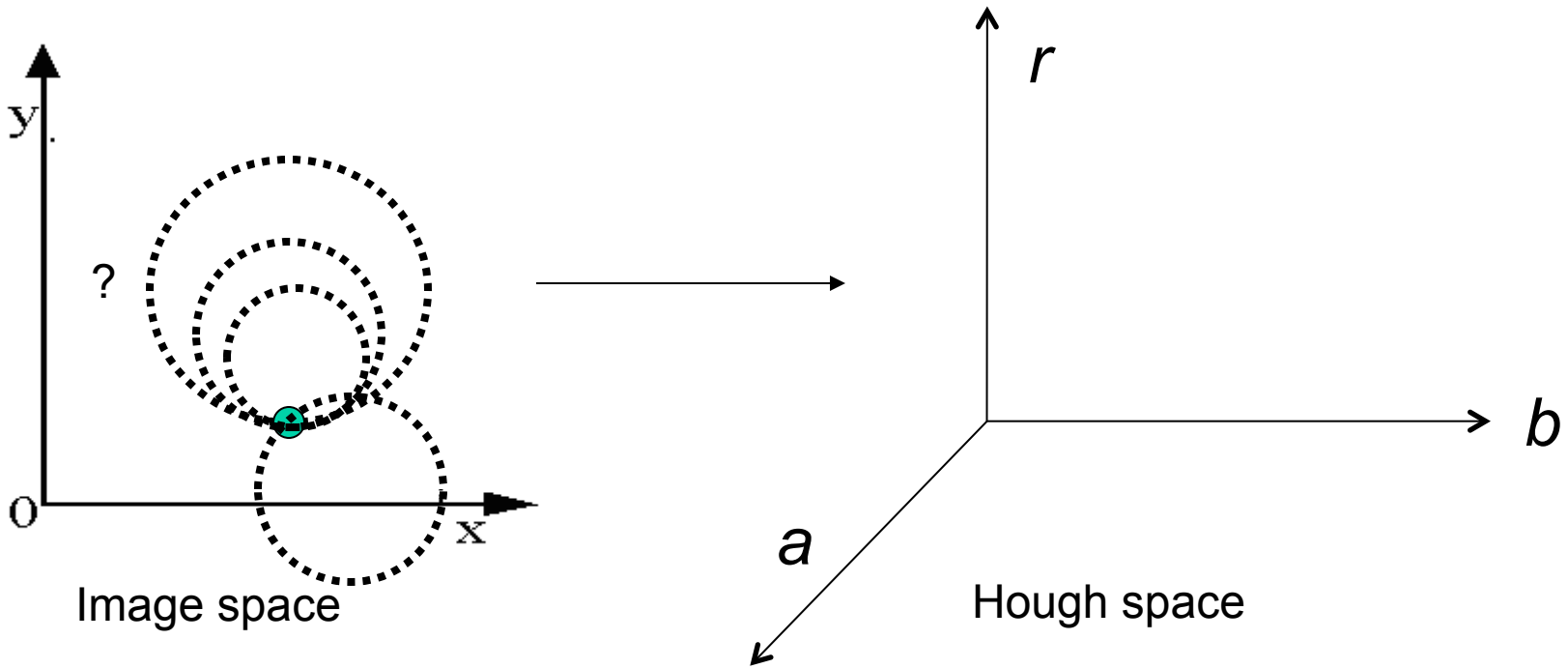


CHT: Hough transform for circles

Circle with a center (a,b) and radius r:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

For an unknown radius r and unknown gradient direction



CHT: Hough transform for circles

- Circle with a center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

For an unknown radius r and unknown gradient direction

3D accumulator array H(a,b,r)

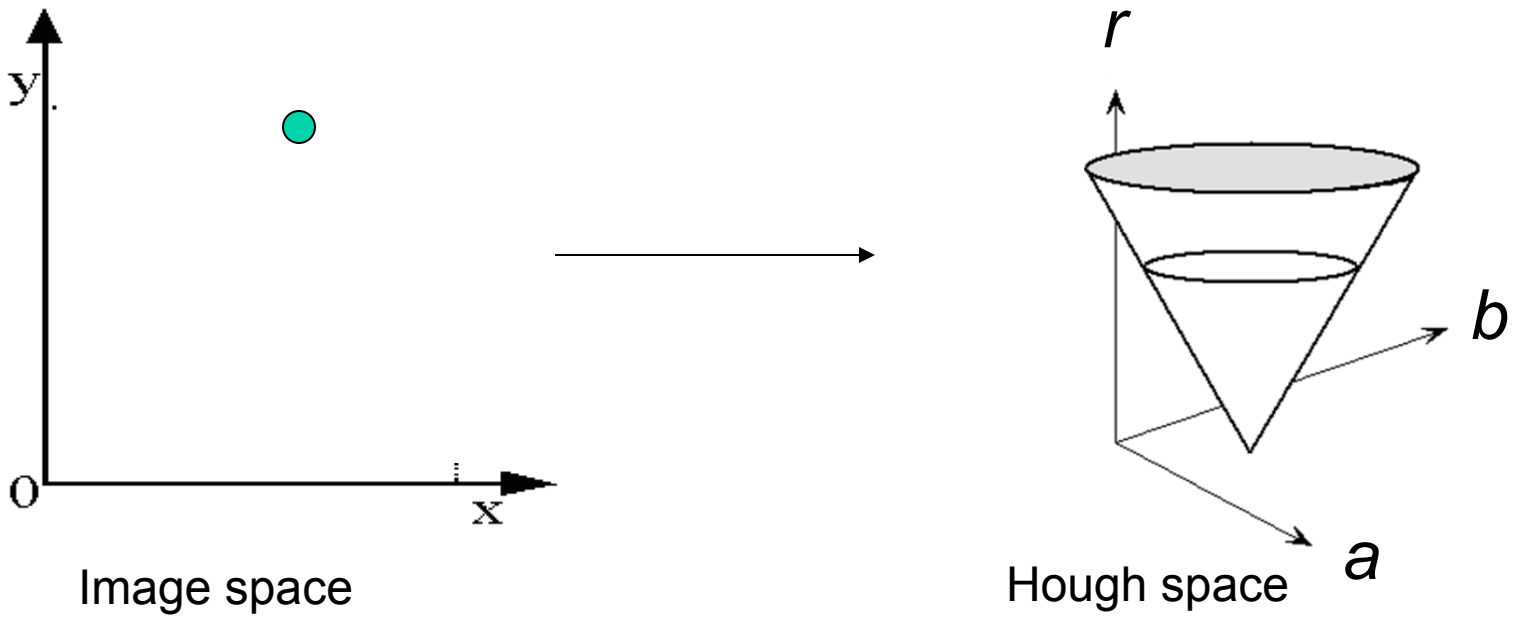


Image space

Hough space

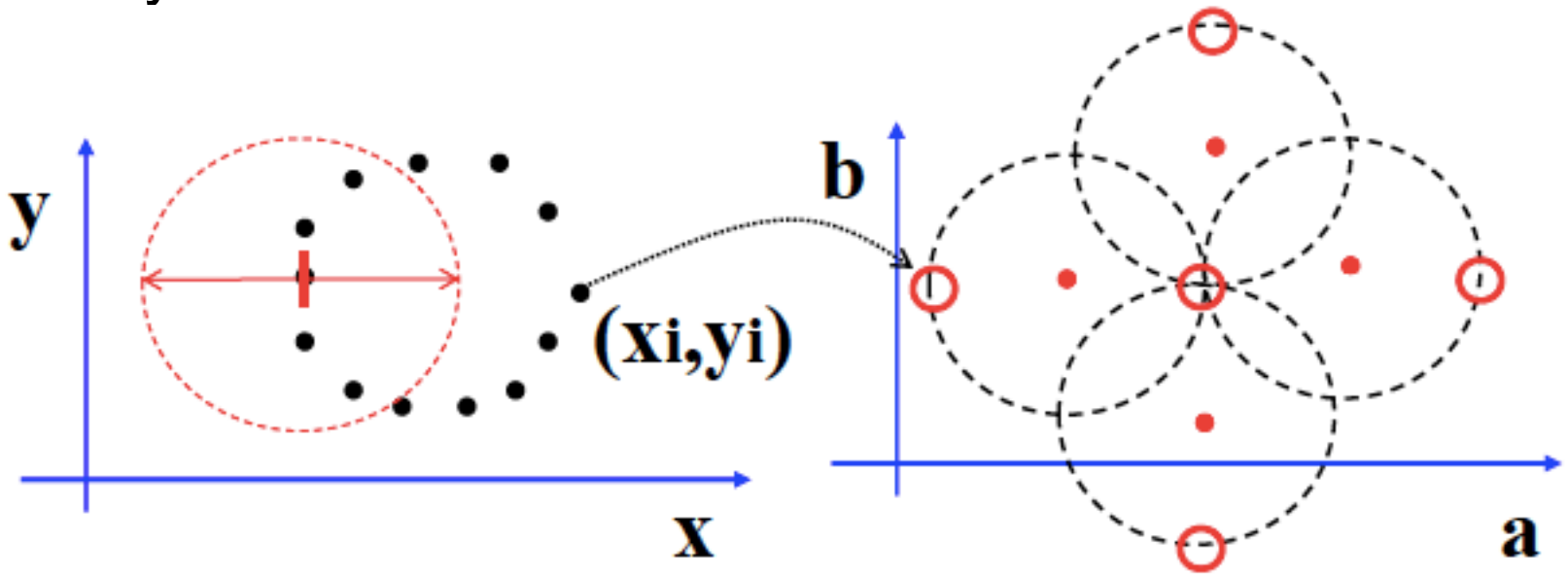
CHT: Hough transform for circles

- Circle with a center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

For radius r and edge known orientation

Search for center reduces from circle to two locations only.



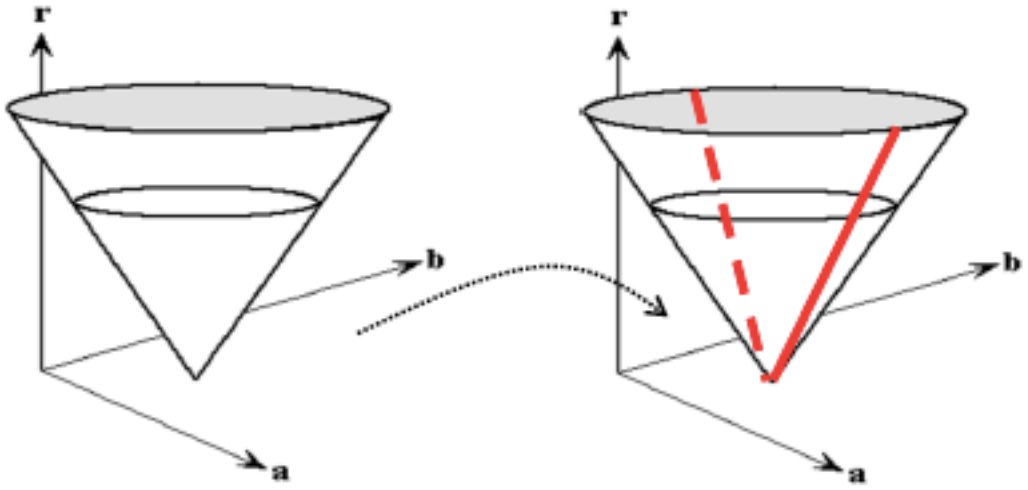
CHT: Hough transform for circles

- Circle with a center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

For an unknown radius r and **known** edge orientation

Search for center reduces from accumulation of cone to two lines only.



CHT: Hough transform for circles

For an known radius r, **known** gradient direction

Need to increment only one point in Accumulator.

For an unknown radius r, **known** gradient direction

Search for center reduces to one line only.

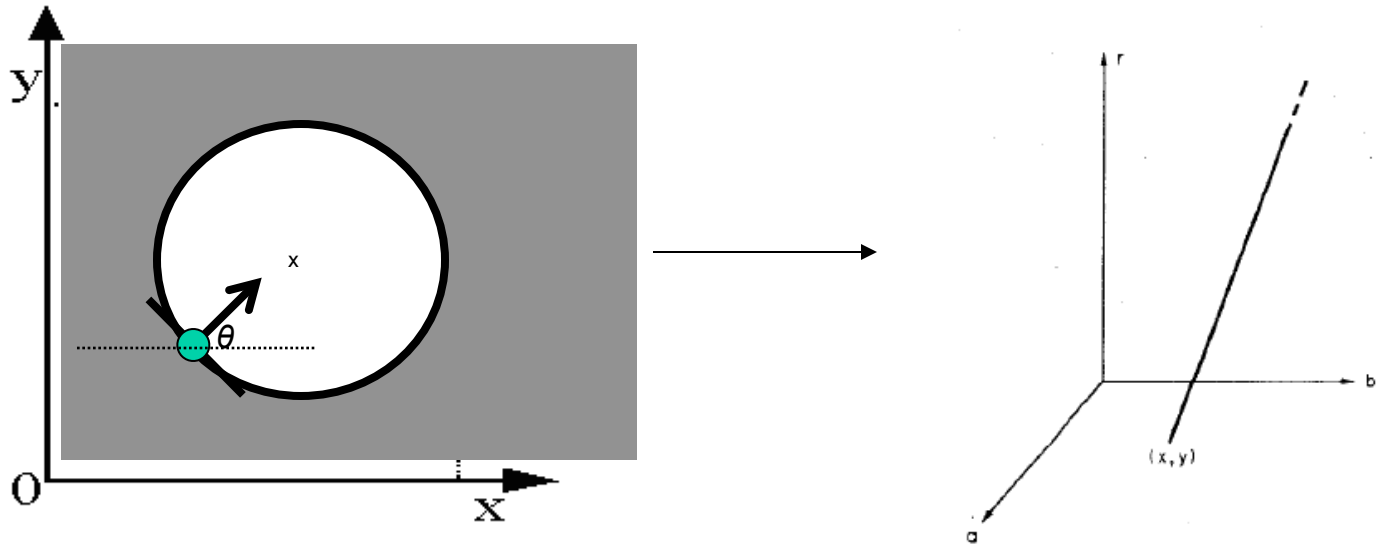


Image space

Hough space

CHT: Hough transform for circles

For every edge pixel (x,y) :

  For each possible radius value r:

    For each possible gradient direction θ:

      // or use estimated gradient at (x,y)

        $a = x - r\cos(\theta)$ // column

        $b = y + r\sin(\theta)$ // row

        $H[a,b,r] = H[a,b,r] + 1$

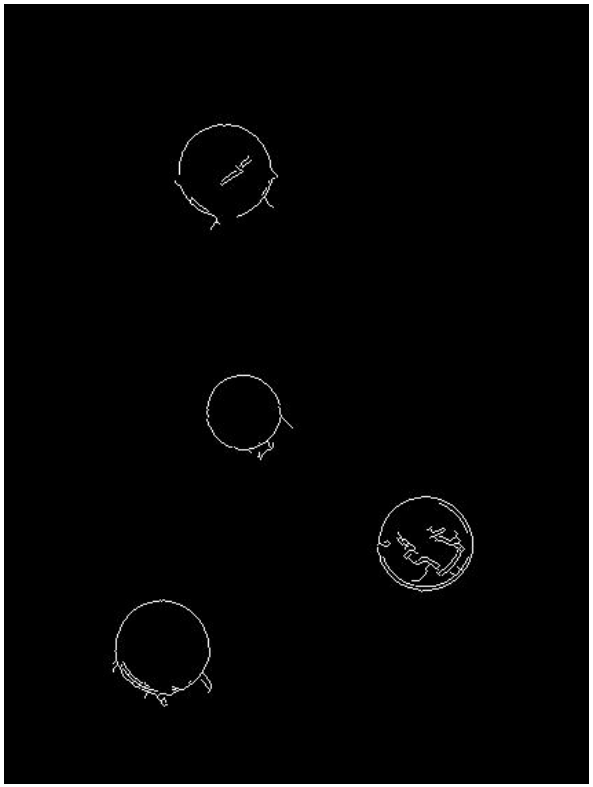
  end

end

CHT: Hough transform for circles

Original

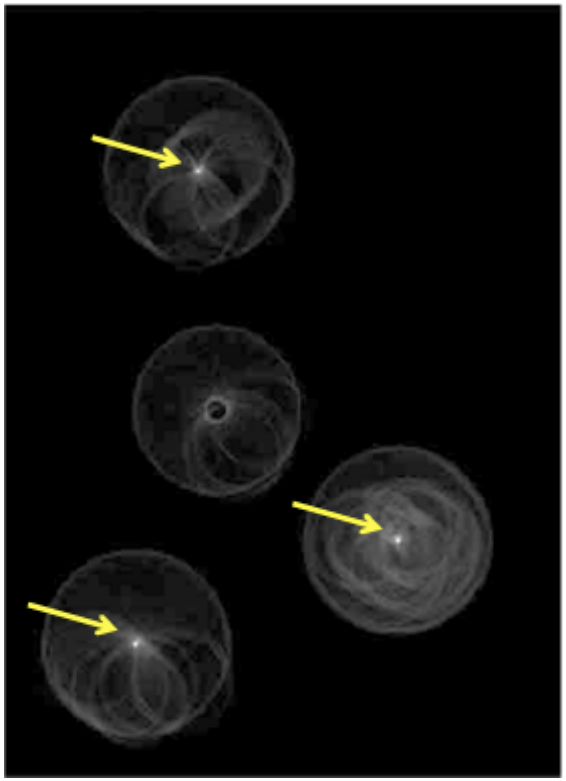
Edges



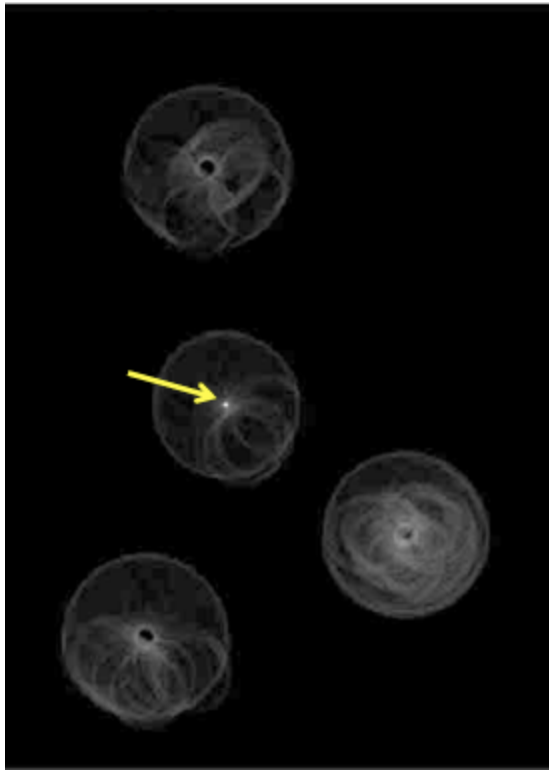
Coin finding sample images from: Vivek Kwatra

CHT: Hough transform for circles

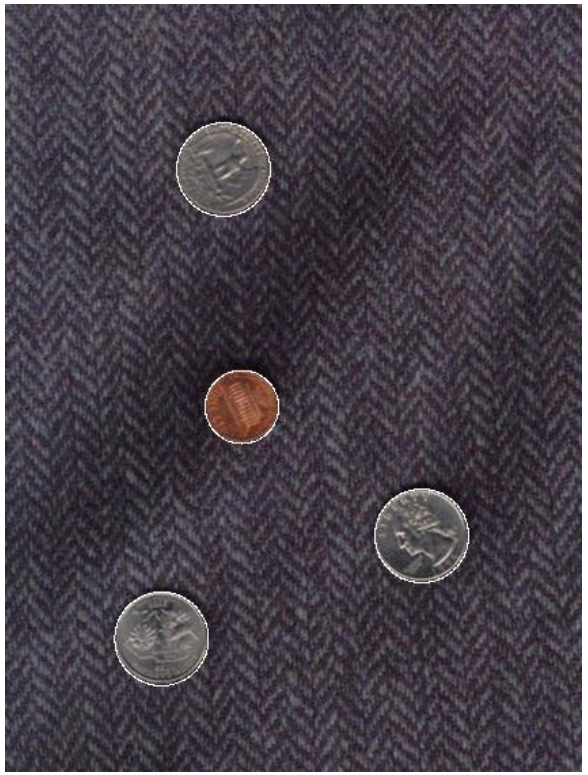
Votes: Quarter



Votes: Penn



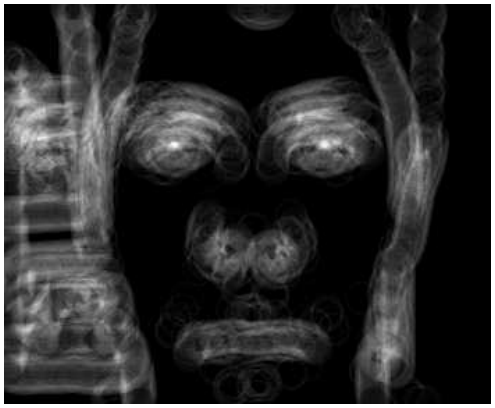
Combined detections



Coin finding sample images from: Vivek Kwatra

CHT: Hough transform for circles

Gradient+threshold



Hough space
(fixed radius)

Max detections

Hemerson Pistori and Eduardo Rocha Costa http://rsbweb.nih.gov/ij/plugins/hough-circles.html

CHT: Hough transform for circles

Figure 2. Original image

Figure 3. Distance image

Figure 4. Detected face region

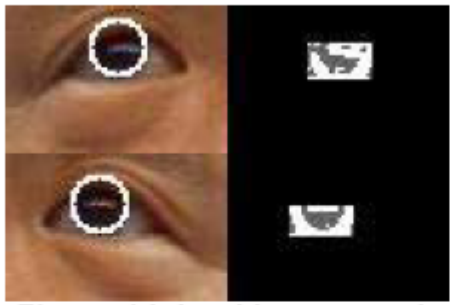


Figure 14. Looking upward

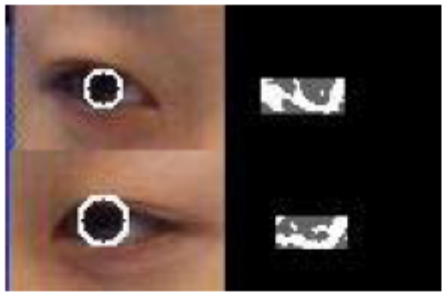


Figure 15. Looking sideways

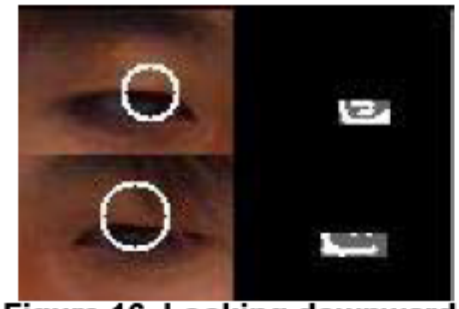


Figure 16. Looking downward

An Iris Detection Method Using the Hough Transform and Its Evaluation for Facial and Eye Movement, by Hideki Kashima, Hitoshi Hongo, Kunihito Kato, Kazuhiko Yamamoto, ACCV 2002.

CHT: Hough transform for circles

Iris recognition using Hough transform

https://www.youtube.com/watch?v=FdgYlwrupJ8

https://www.youtube.com/watch?v=tNI9nW-aGOg

Generalised Hough transform

GHT: Generalized Hough transform

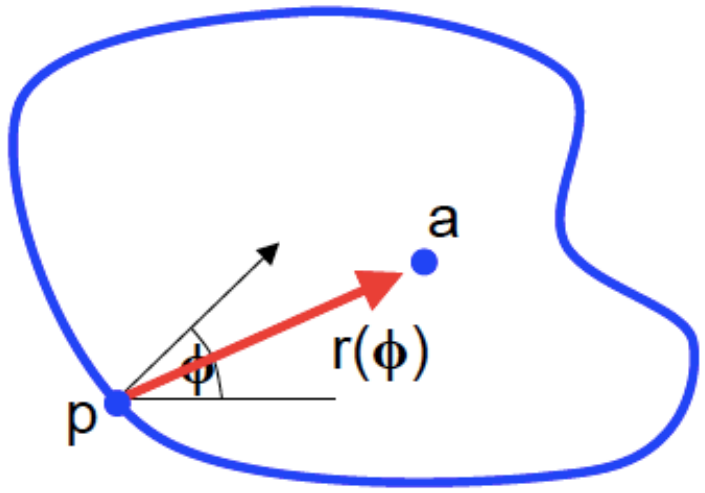
Two kinds:

- Using the boundary of an object of interest.
- Using some features of an object of interest.

GHT: Generalized Hough transform

Model Shape NOT described by equation but by sets of vectors from the boundary to the center, sorted by edge orientation.

We want to find a shape defined by its boundary points and a reference point



Compute centroid

For each edge point p, compute its distance to centroid r = a – p as a function of gradient orientation ϕ

Find edge orientation (gradient angle)

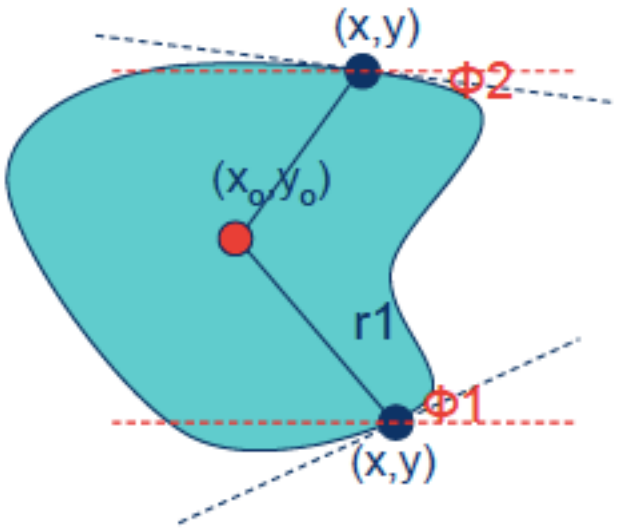
Construct a table of angles and r values

D. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, Pattern Recognition 13(2), 1981, pp. 111-122.

GHT: Generalized Hough transform

- Generating R-table

Φ1	r1, r2, r3 …
Φ2	r14, r21, r23 …
Φ3	r41, r42, r33 …
Φ4	r10, r12, r13 …



It requires training!

GHT: Generalized Hough transform

Detection:

- For each edge point p compute gradient orientation ϕ
- Retrieve all r indexed with ϕ
- For each $r(\phi)$, put a vote in the Hough space at $p + r(\phi)$
- Peak in this Hough space is reference point with most supporting edges

GHT: Generalized Hough transform

1. Quantize the parameter space $P[x_{cmin}, \ldots\ldots, x_{cmax}, y_{cmin}, \ldots\ldots, y_{cmax}]$.

2. For each edge point (x, y) do
   compute $\phi(x, y)$
   for each table entry for ϕ do

$$x_c = x + x' \tag{4.13}$$

$$y_c = y + y' \tag{4.14}$$

   $P[x_c, y_c] = P[x_c, y_c] + 1.$

3. Find the local maxima in the parameter space.

Figure 4.8: Generalized Hough transform algorithm.

GHT: Generalized Hough transform

Rotation and scale invariance

Rotation around Z-axis

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = x\sin\alpha + y\cos\alpha$$

Scaling

$$x' = sx$$
$$y' = sy$$

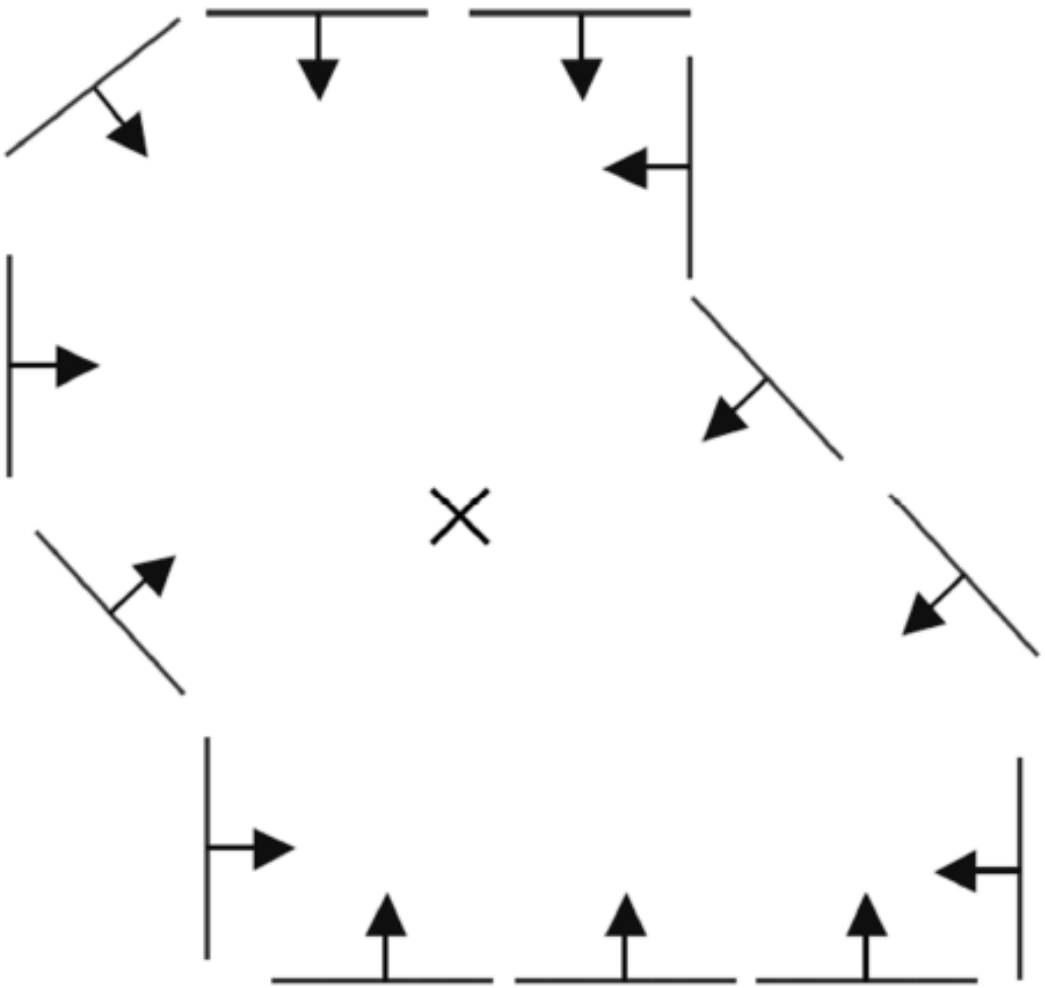
Rotation+scaling

$$x' = s(x\cos\alpha - y\sin\alpha)$$
$$y' = s(x\sin\alpha + y\cos\alpha)$$

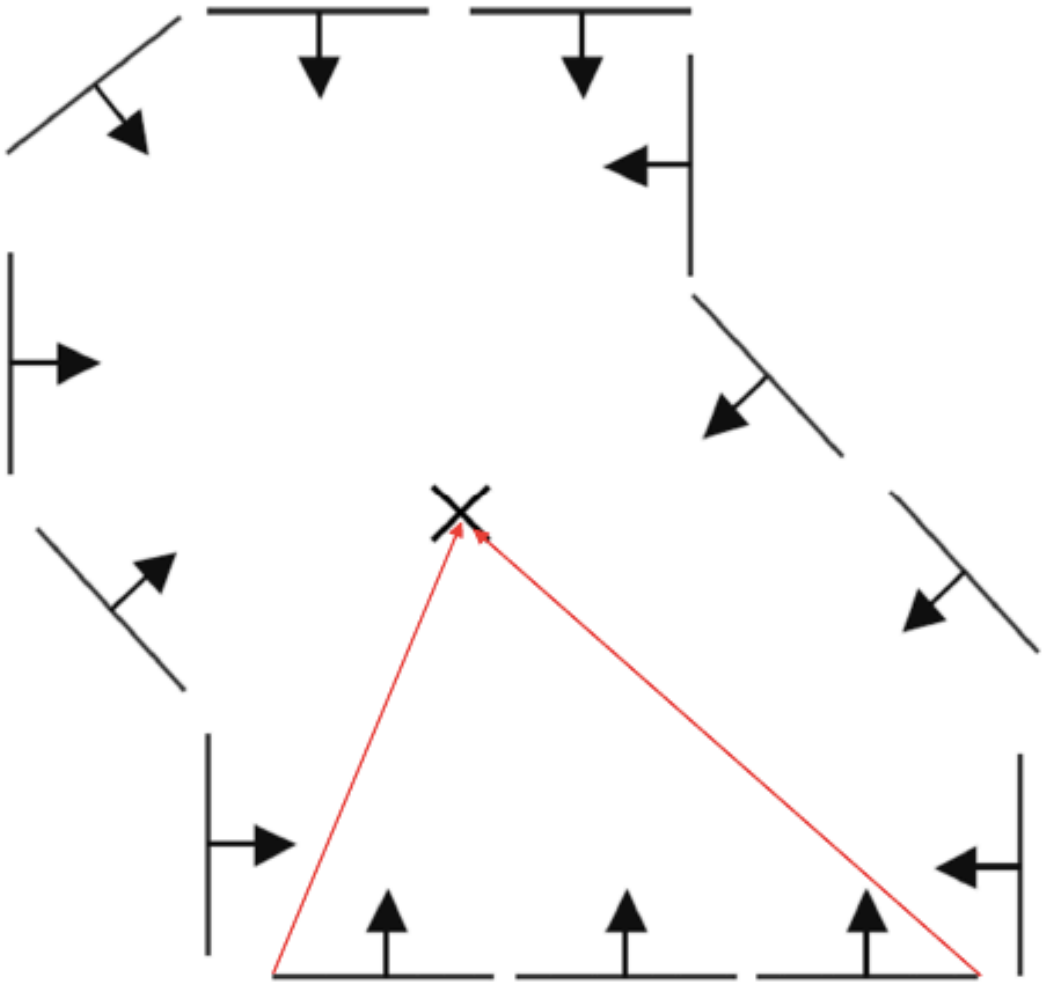
In the algorithm, replace by equations

$$x_c = x + s_x(x'\cos\theta + y'\sin\theta)$$
$$y_c = y + s_y(-x'\sin\theta + y'\cos\theta)$$

GHT: Generalized Hough transform

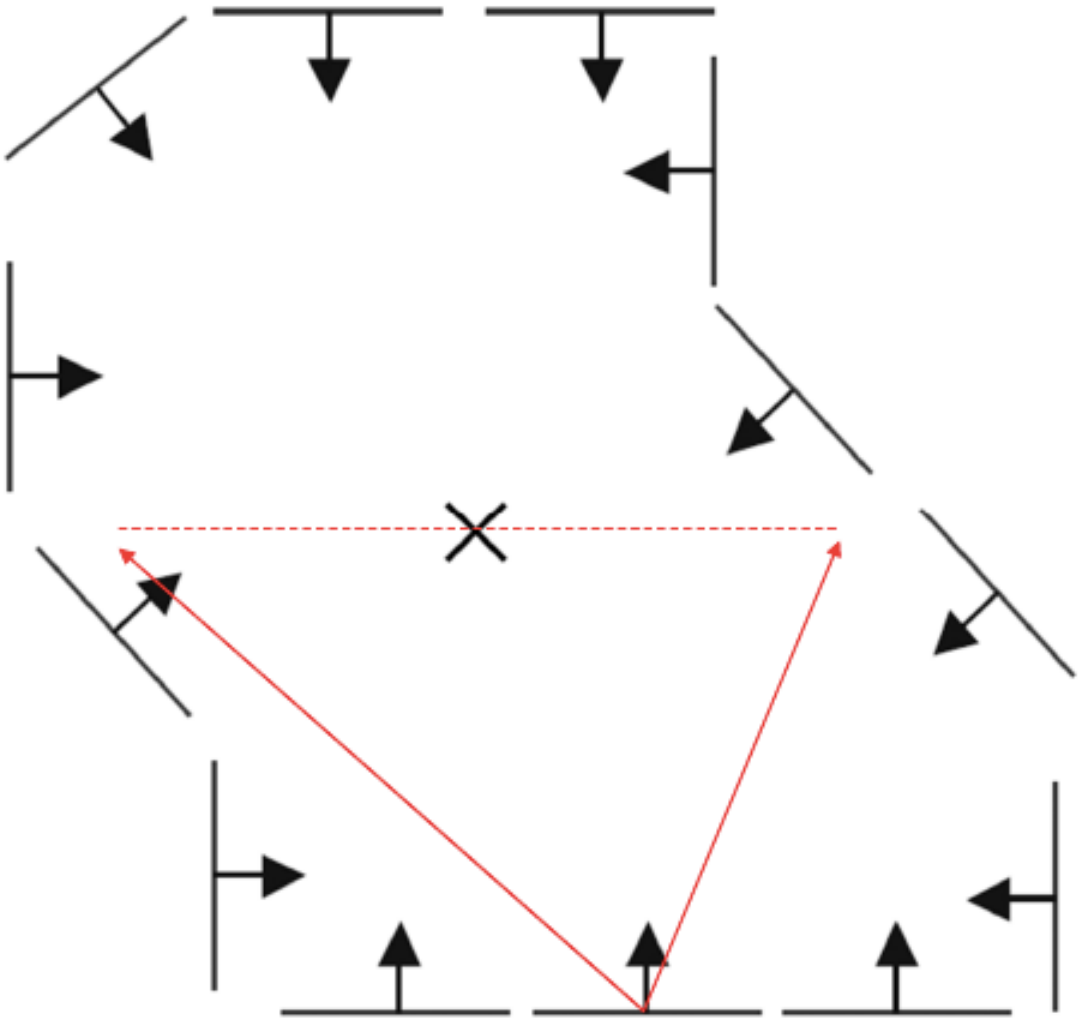


GHT: Generalized Hough transform

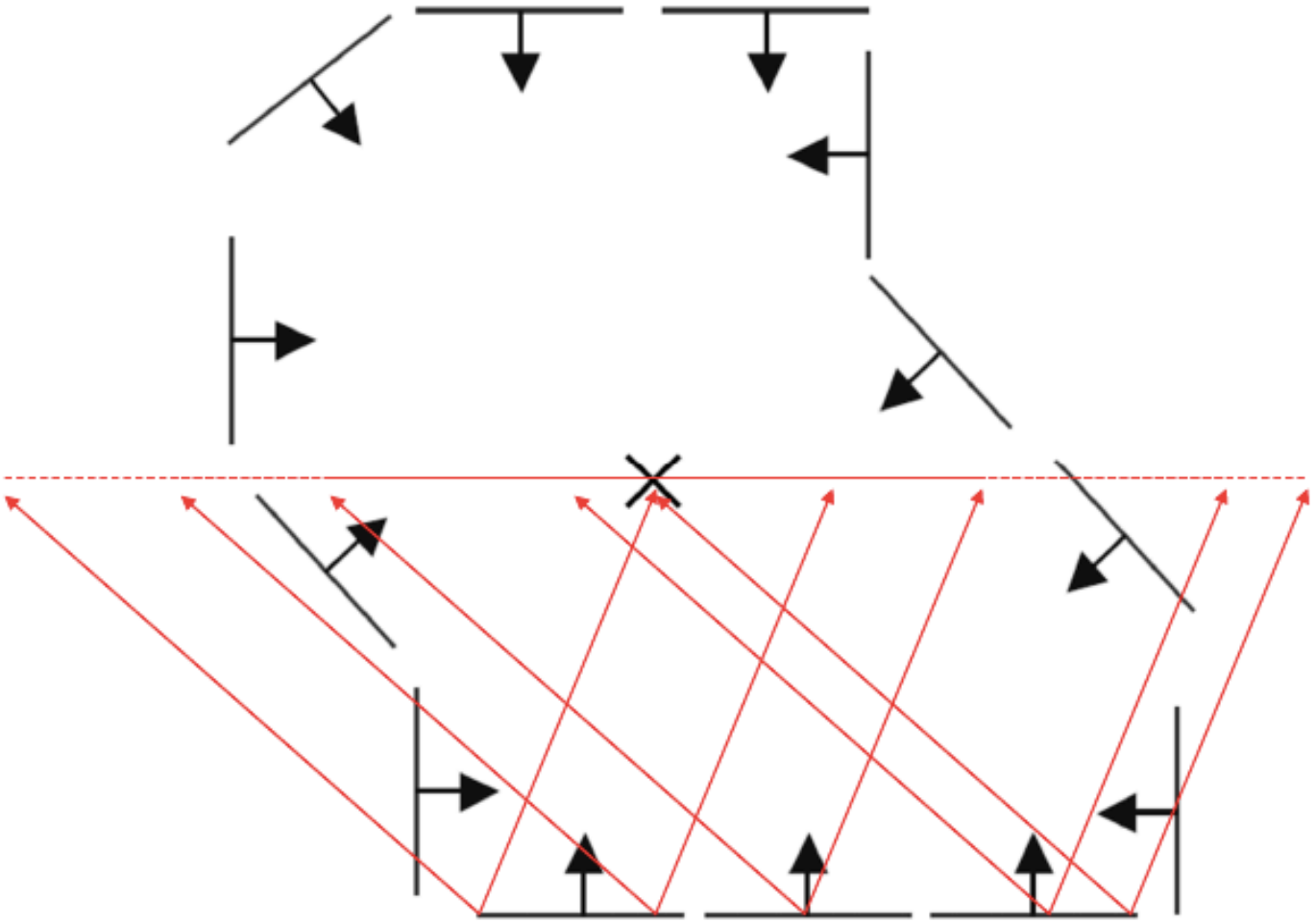


displacement vectors for model points

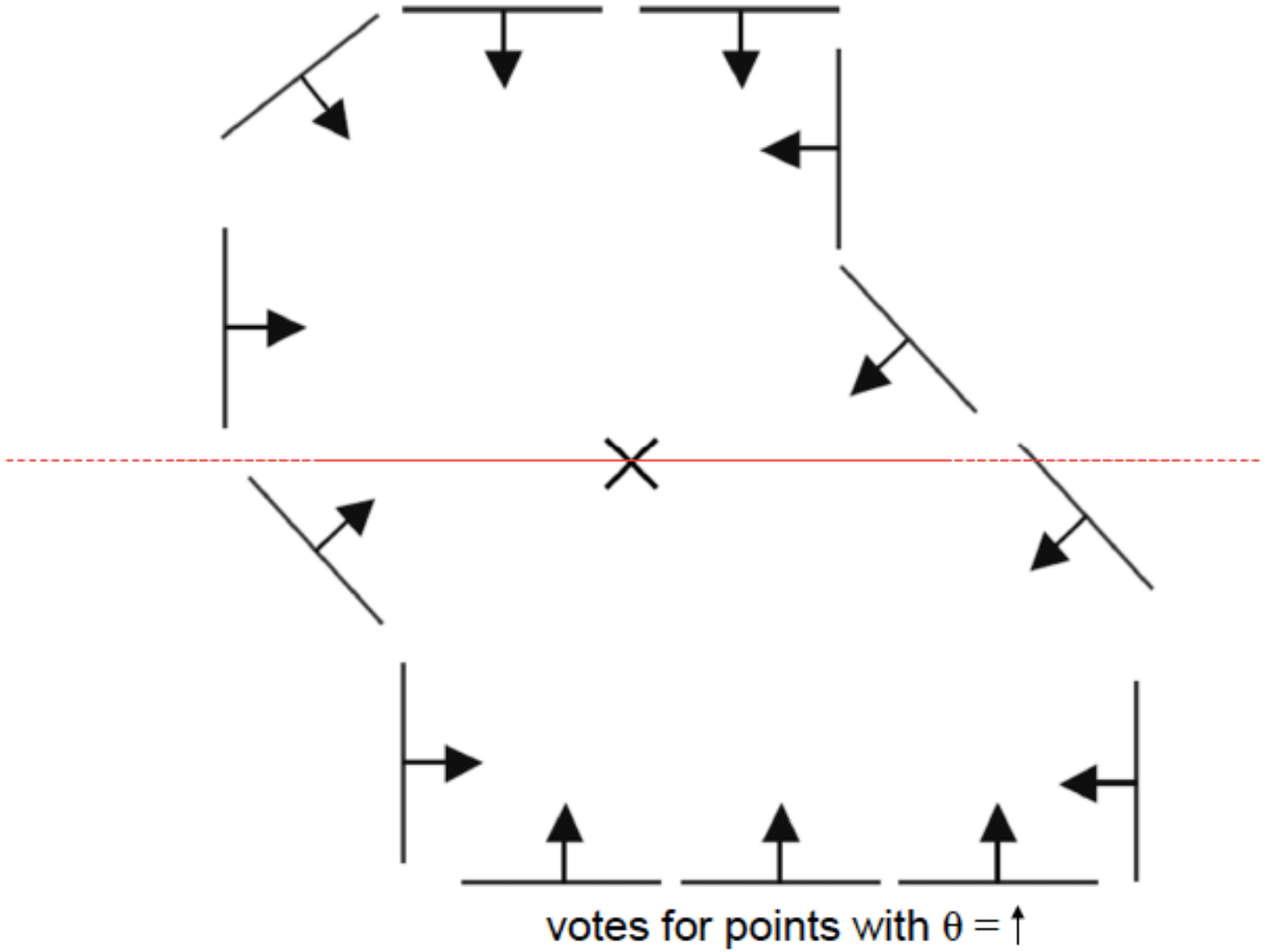
GHT: Generalized Hough transform



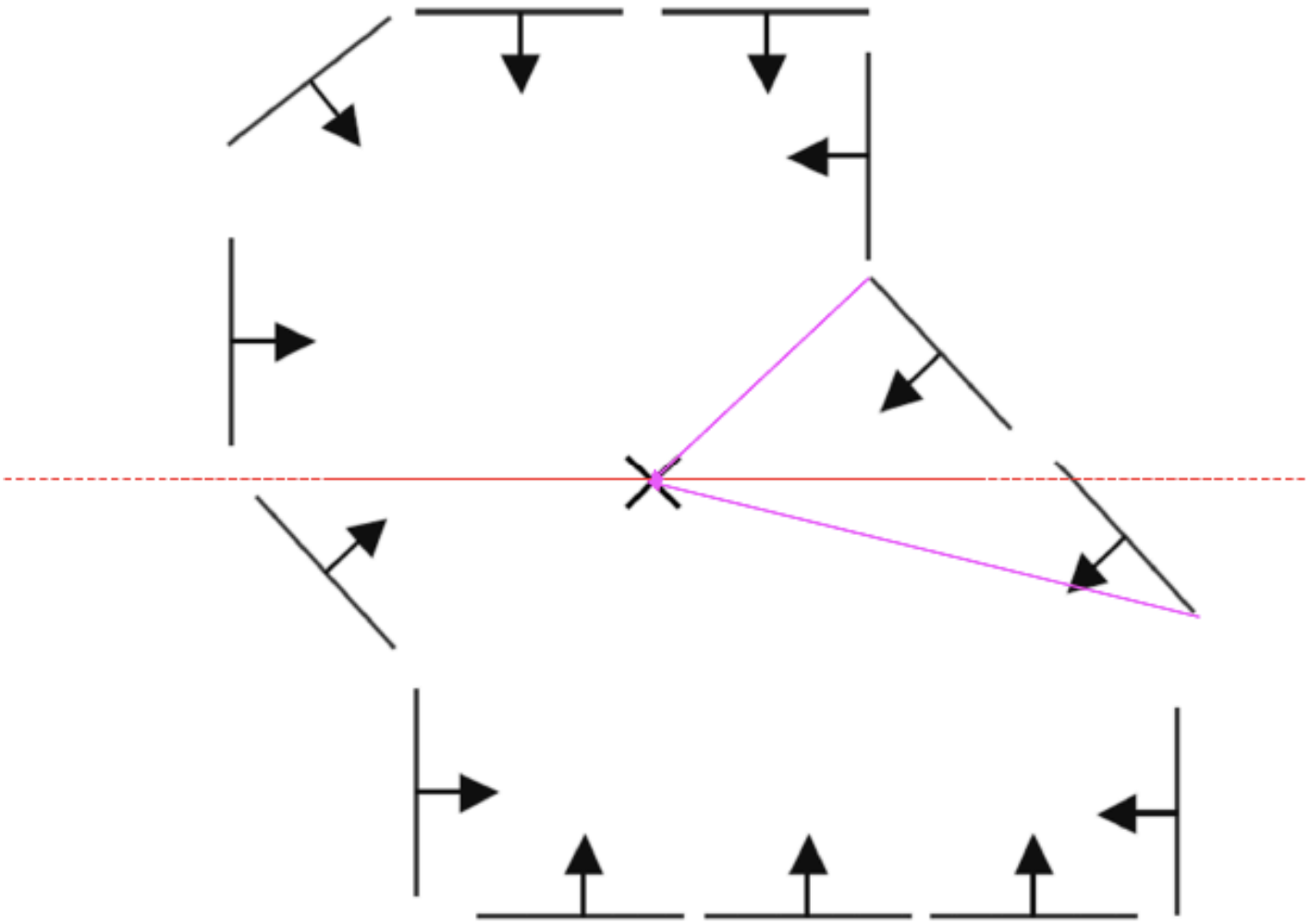
GHT: Generalized Hough transform



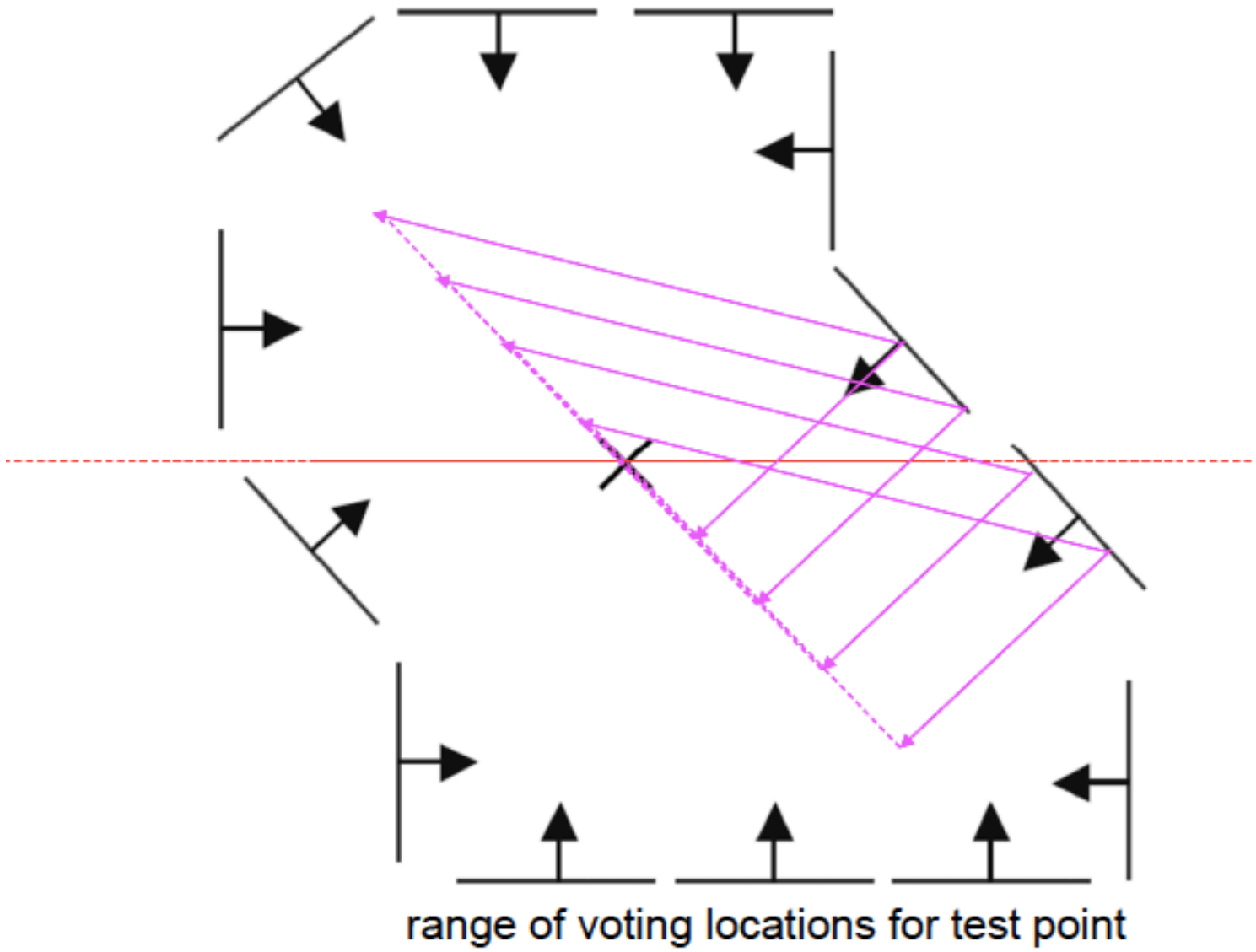
GHT: Generalized Hough transform



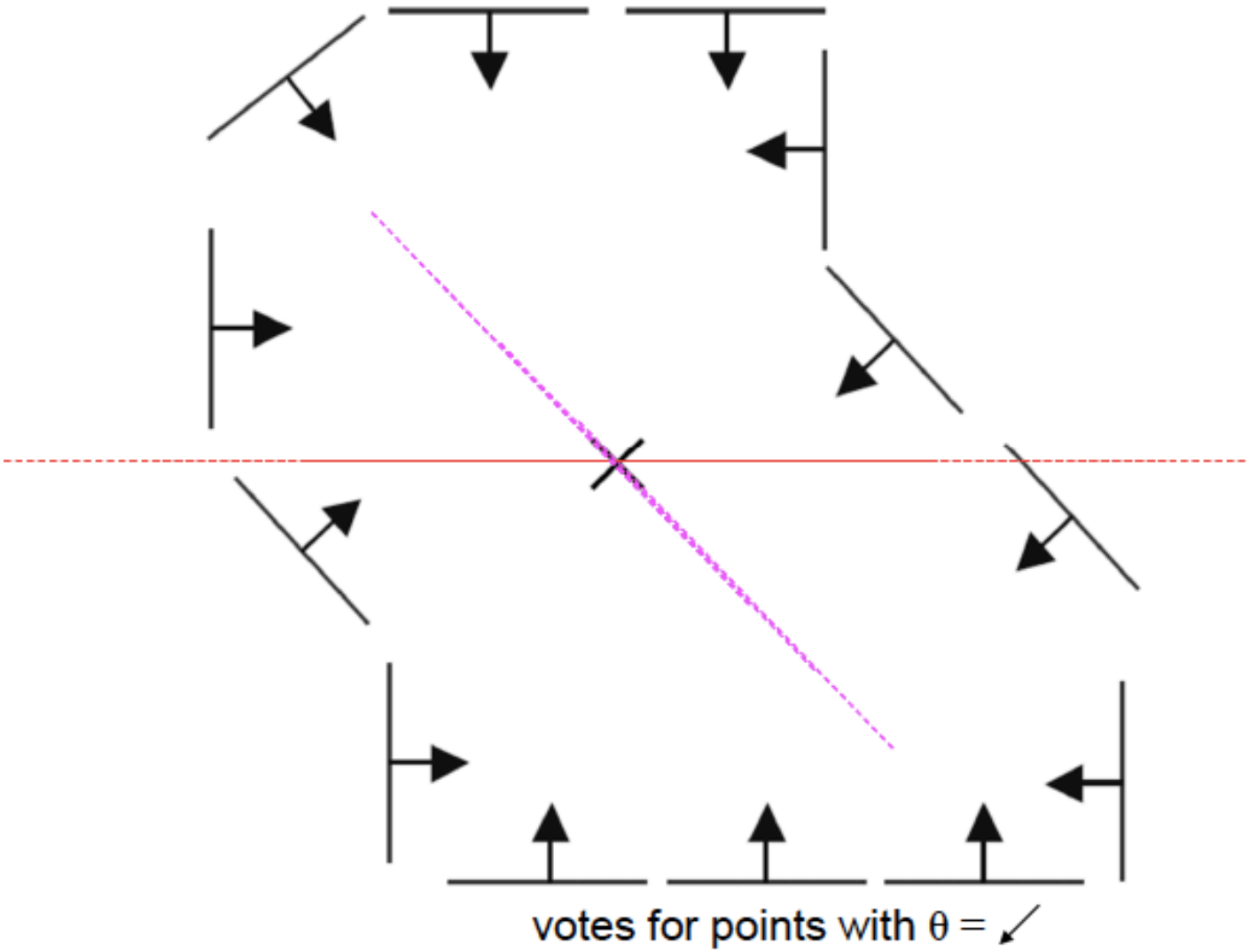
GHT: Generalized Hough transform



GHT: Generalized Hough transform

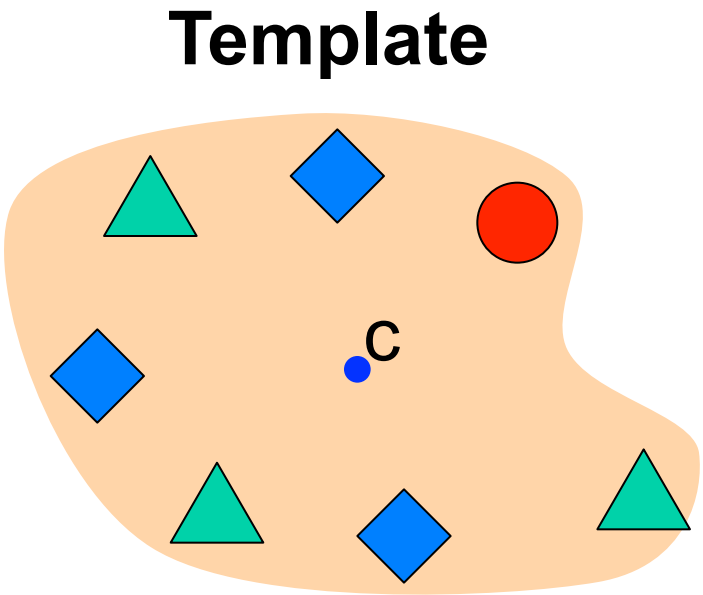


GHT: Generalized Hough transform



GHT: Generalized Hough transform

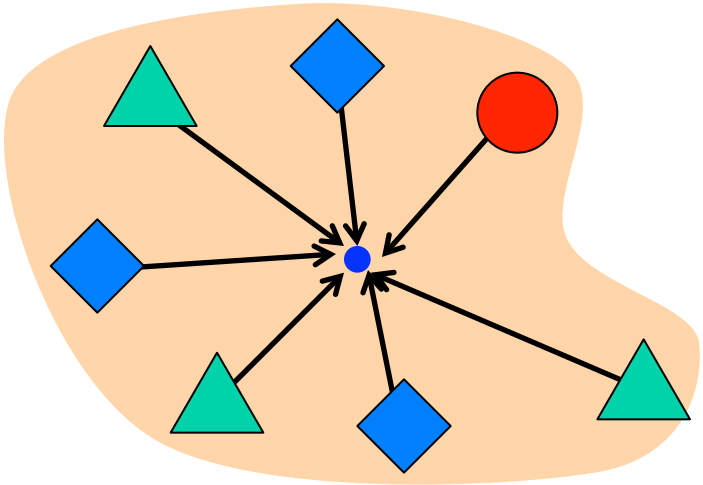
- We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration



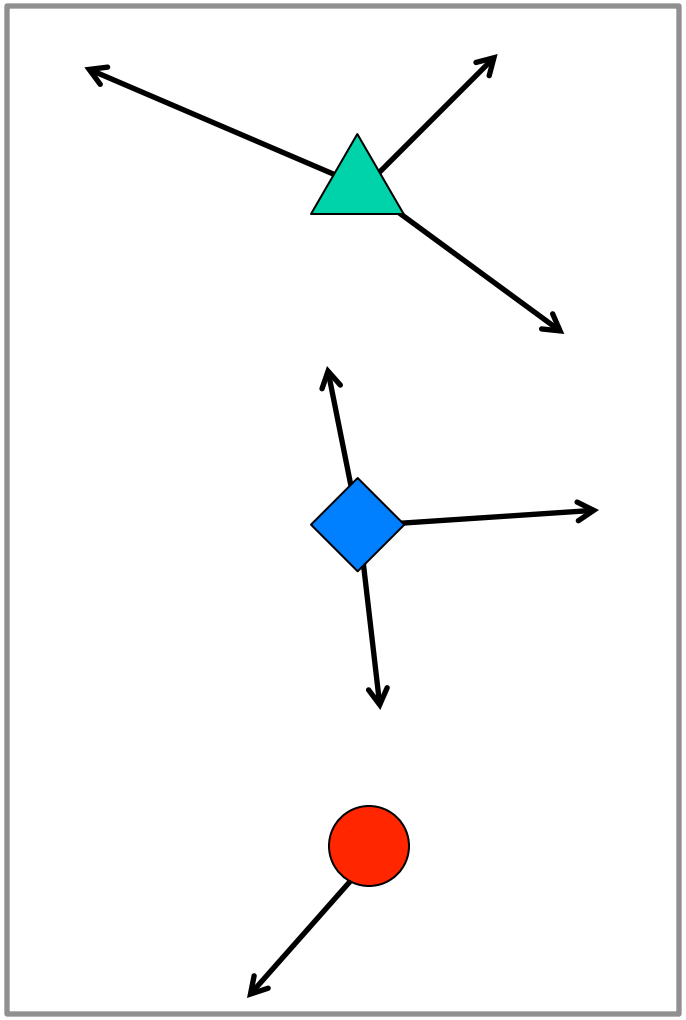
GHT: Generalized Hough transform

- Template representation: for each type of landmark point, store all possible displacement vectors towards the center

Template



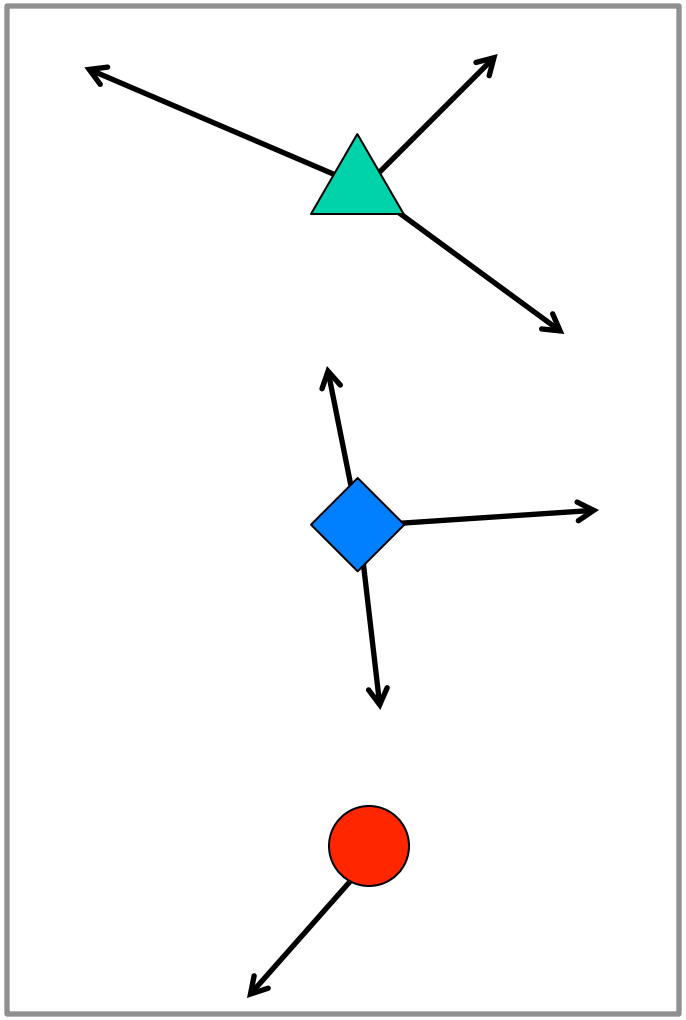
Model



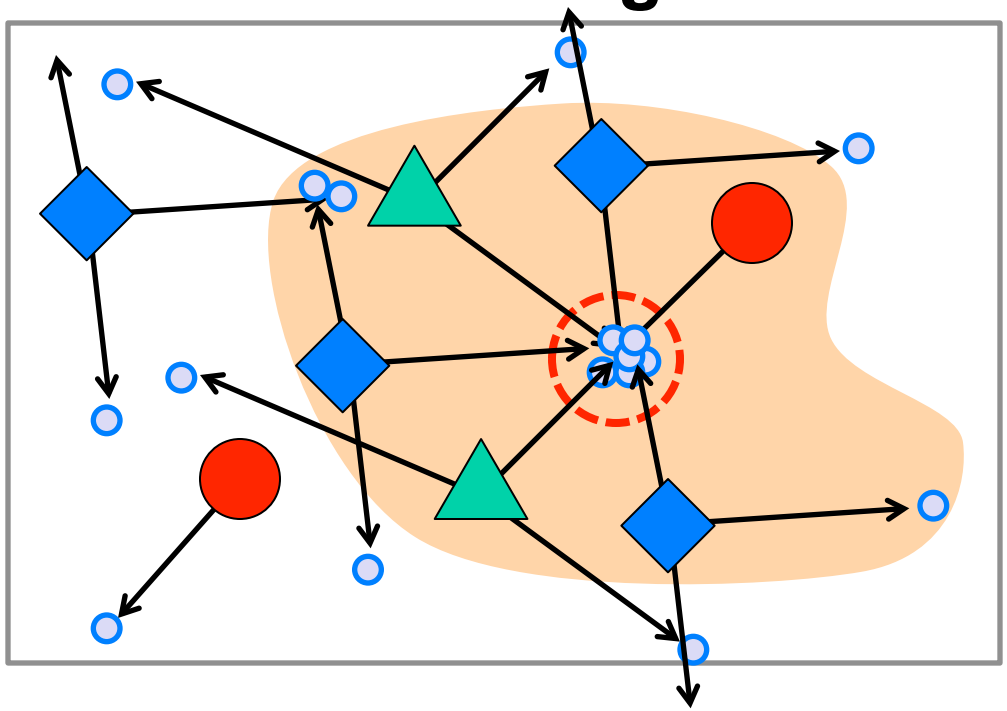
GHT: Generalized Hough transform

- Detecting the template:
  - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model

Model

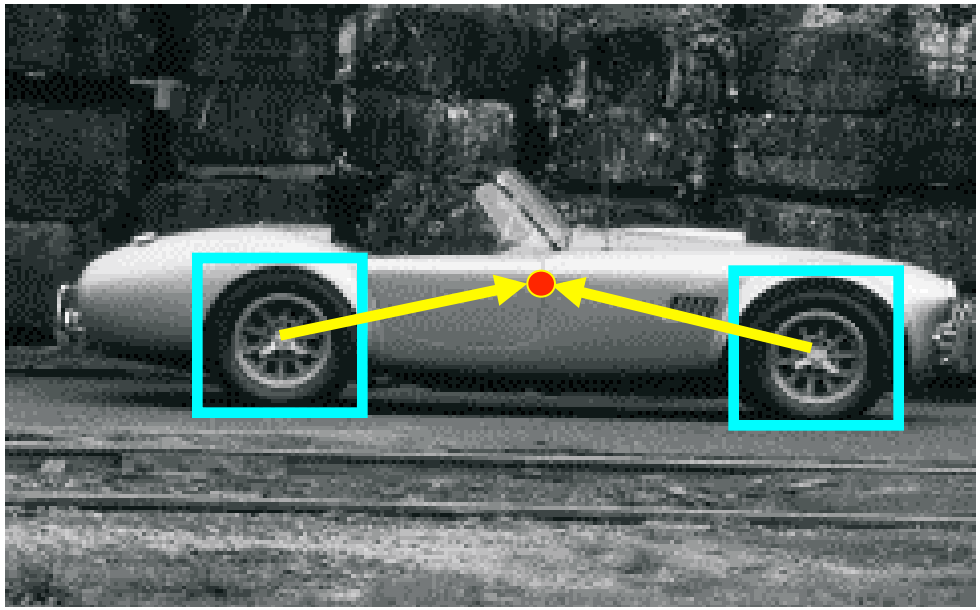


Test image



GHT: Application in recognition

- Index displacements by “visual codeword”



training image

visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

GHT: Application in recognition

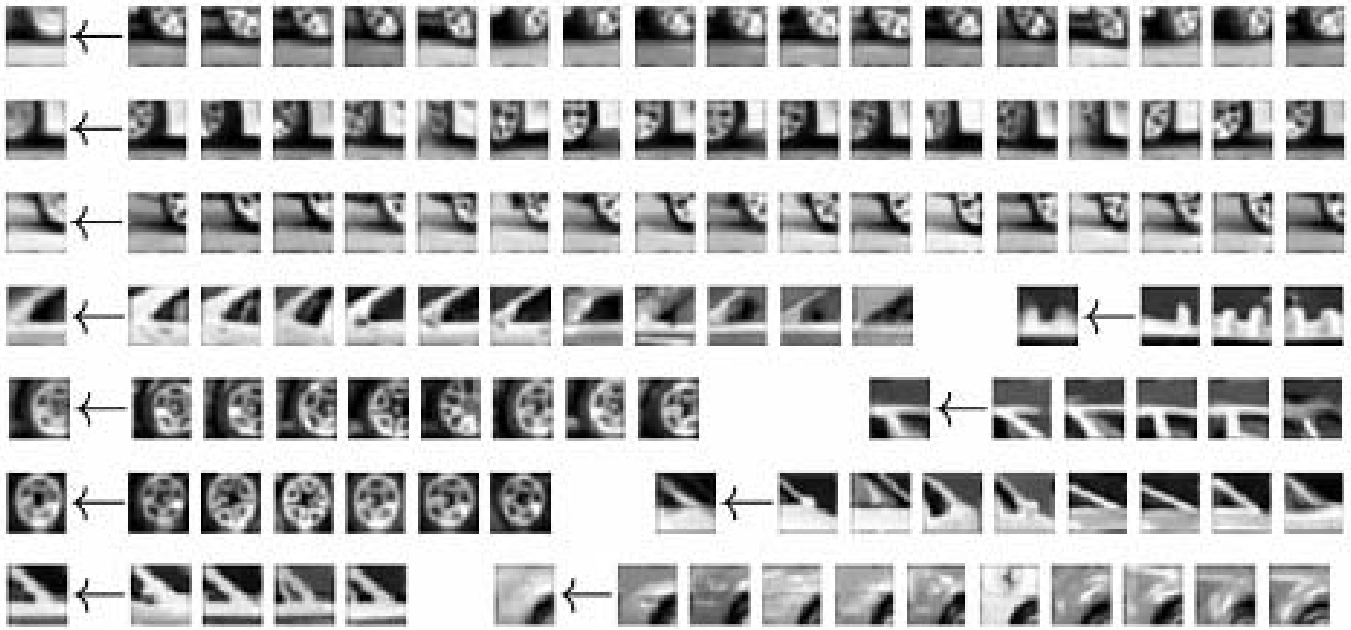
- Index displacements by “visual codeword”

test image

B. Leibe, A. Leonardis, and B. Schiele,
Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

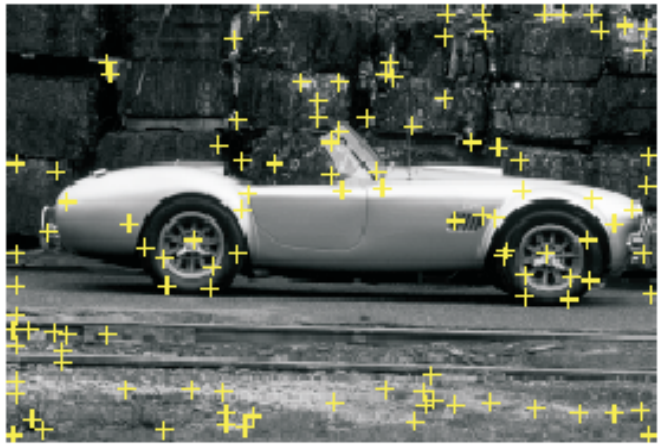
GHT: Implicit shape models: Training

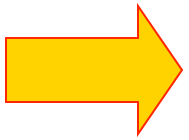
1. Build *codebook* of patches around extracted interest points using clustering

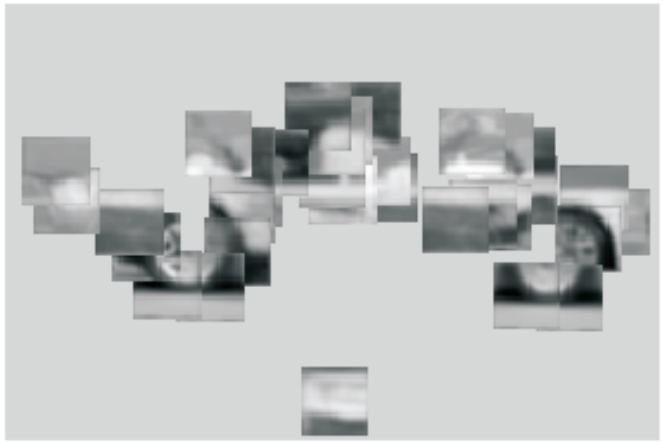


GHT: Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry

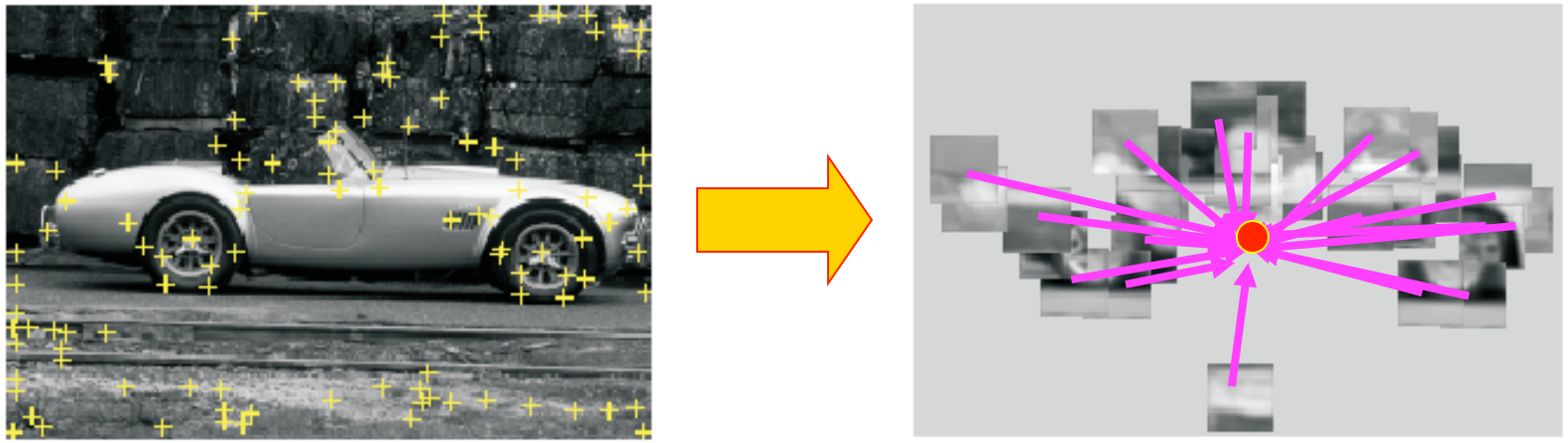






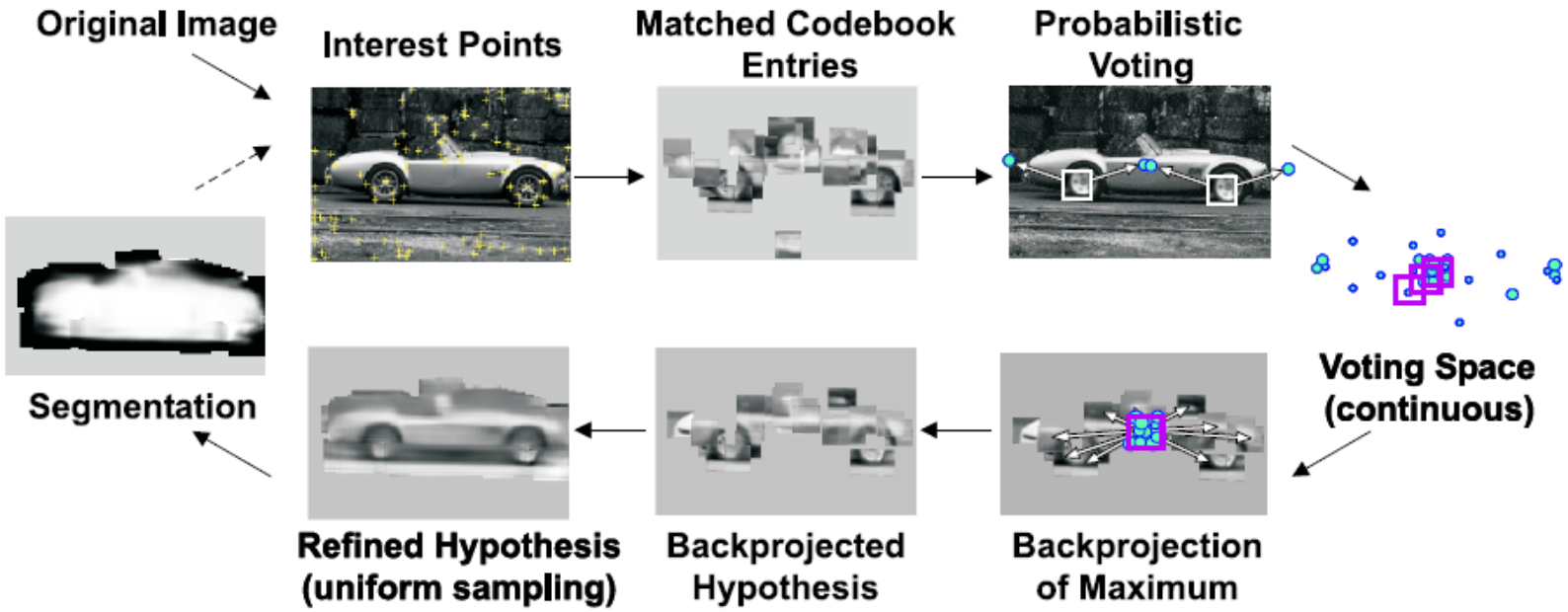
GHT: Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
3. For each codebook entry, store all positions it was found, relative to object center

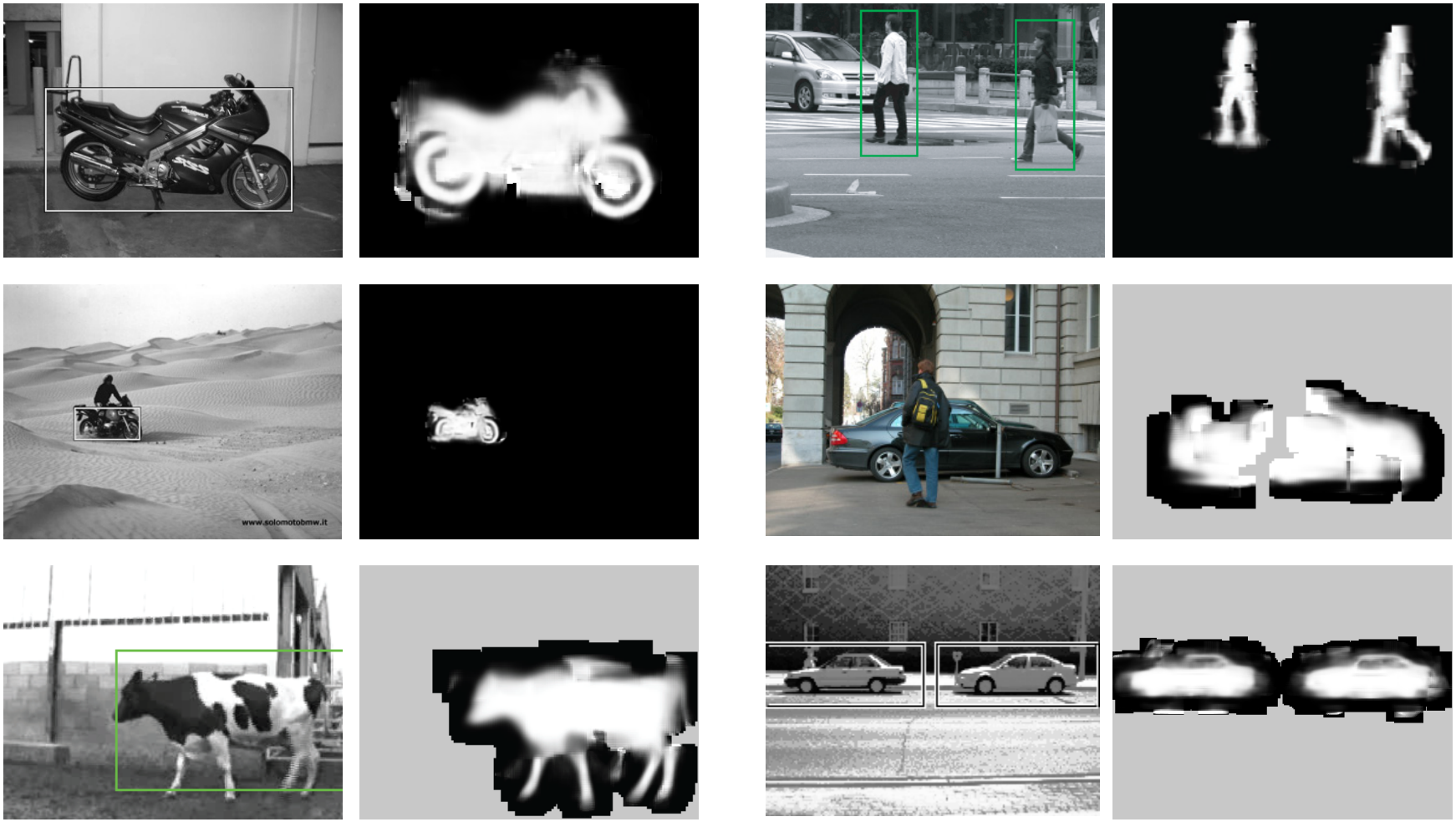


GHT: Implicit shape models: Testing

1. Given test image, extract patches, match to codebook entry
2. Cast votes for possible positions of object center
3. Search for maxima in voting space
4. Extract weighted segmentation mask based on stored masks for the codebook occurrences



GHT: Additional examples



B. Leibe, A. Leonardis, and B. Schiele,
Robust Object Detection with Interleaved Categorization and Segmentation, IJCV 77 (1-3), pp. 259-289, 2008.

GHT: Implicit shape models: Details

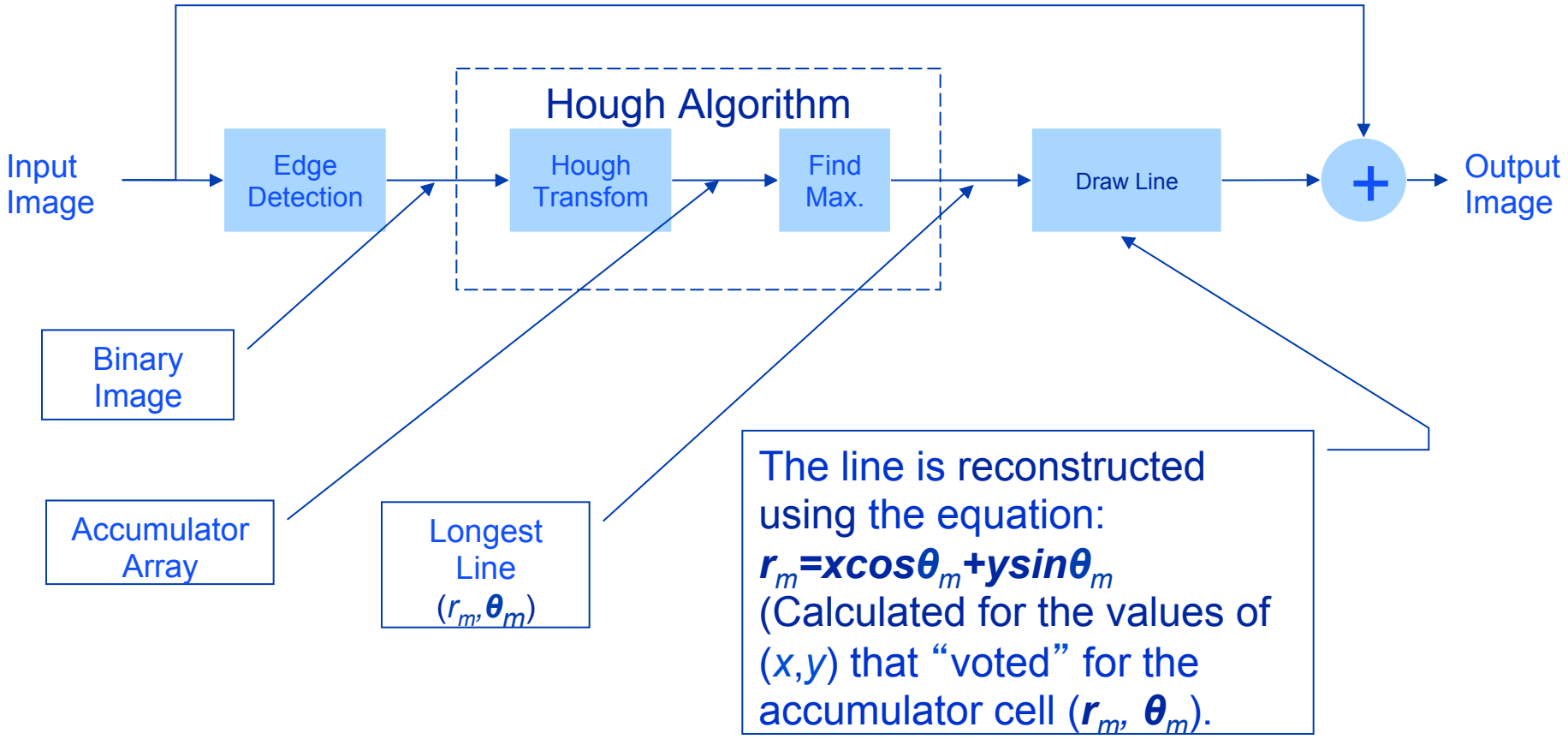
- Supervised training
  - Need reference location and segmentation mask for each training car
- Voting space is continuous, not discrete
  - Clustering algorithm needed to find maxima
- How about dealing with scale changes?
  - Option 1: search a range of scales, as in Hough transform for circles
  - Option 2: use interest points with characteristic scale
- Verification stage is very important
  - Once we have a location hypothesis, we can overlay a more detailed template over the image and compare pixel-by-pixel, transfer segmentation masks, etc.

Hough transform: Discussion

- Pros
  - Can deal with non-locality and occlusion
  - Can detect multiple instances of a model
  - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Cons
  - Complexity of search time increases exponentially with the number of model parameters
  - Non-target shapes can produce spurious peaks in parameter space
  - It’s hard to pick a good grid size

Hough transform: Review

- Hough transform for lines
- Hough transform for circles
- Generalized Hough transform



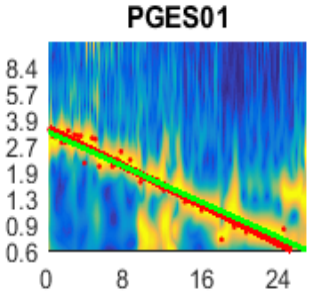
Radon transform

RT: Introduction

Theory by Johann Karl August Radon, 1917

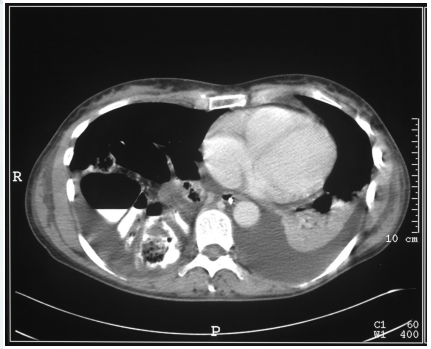
Radon transform

E. G.: Determine the slope of a line -> Epilepsy



Inverse Radon transform

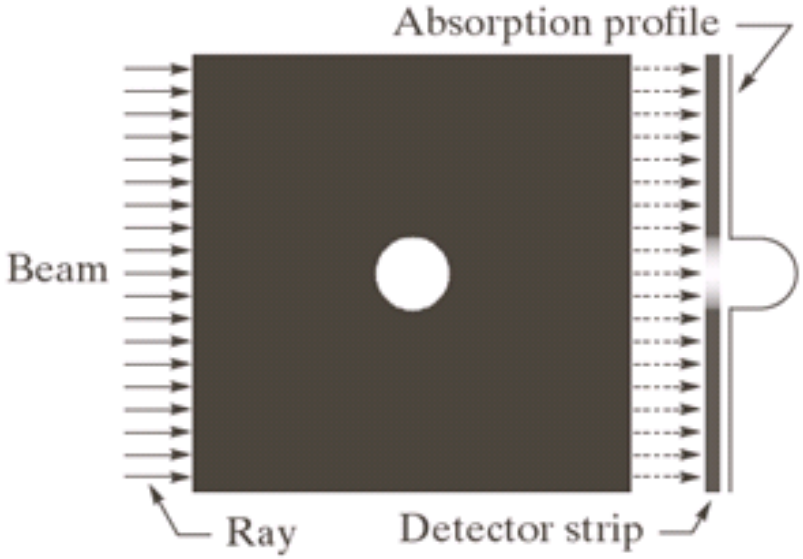
Important for CT (Computed-Tomography, 1964)



RT: Basic example in CT

Radon transform

- A beam of X-rays is emitted and part of it is absorbed by the object.
- The energy of absorption is detected by a set of detectors.
- The collected information is the absorption signal.

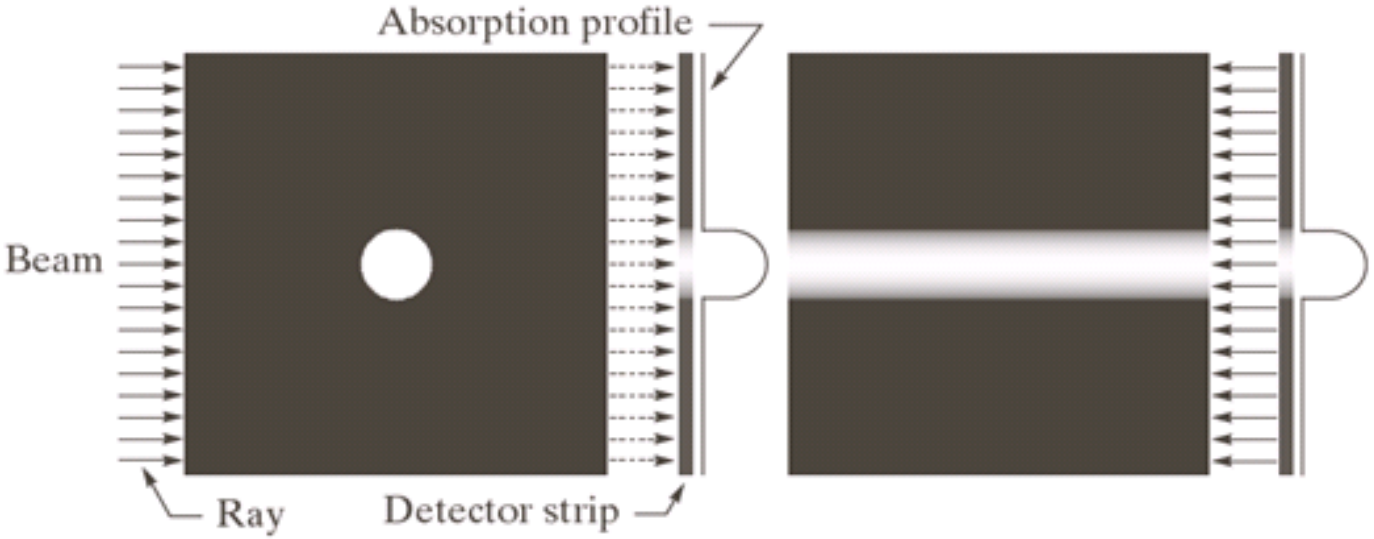


RT: Basic example in CT

Inverse Radon transform

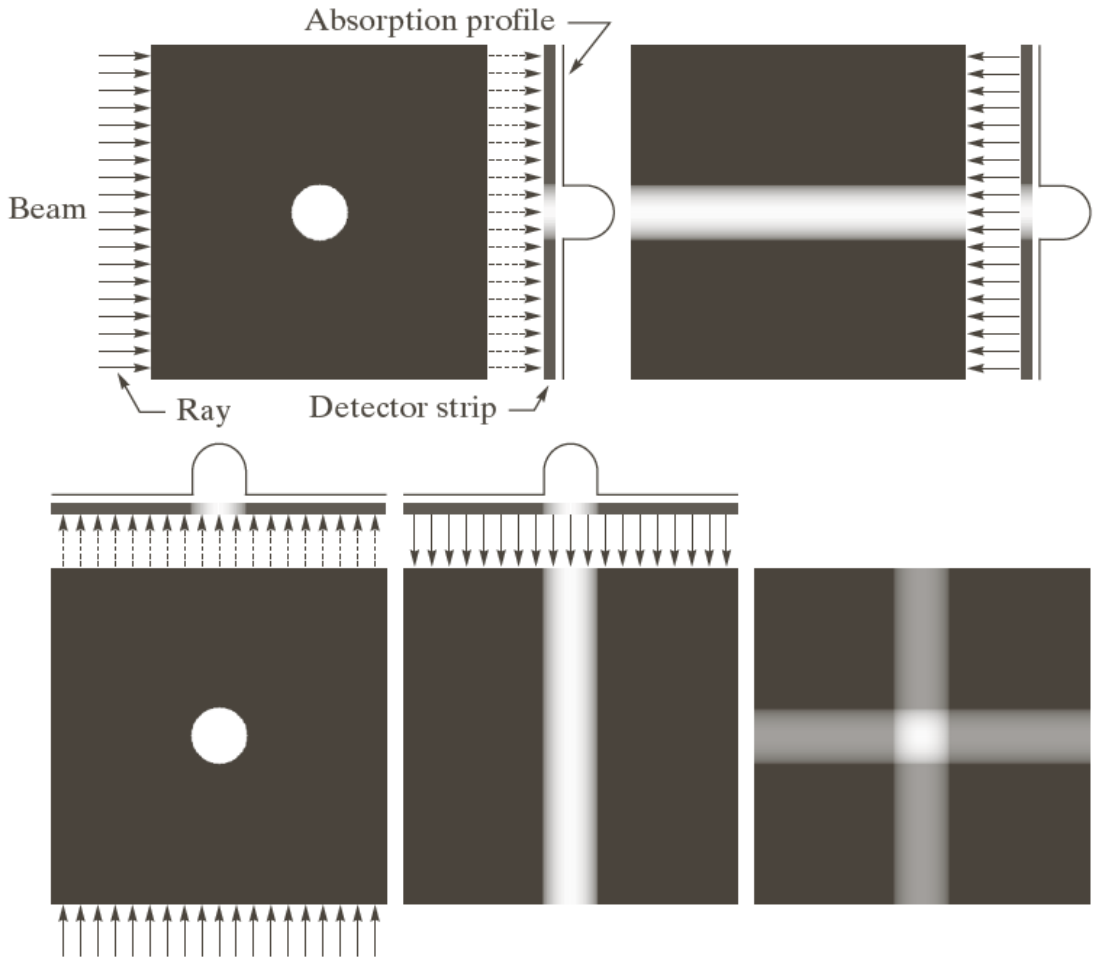
- A naïve way to recover the object is to back-project the 1D signal across the direction the beam came.

Naïve reconstruction –
A kind of voting



RT: Basic example in CT

- We now rotate the position of the source-detector pair and obtain another 1D signal.
- We repeat the procedure and add the signals from the previous back-projections.

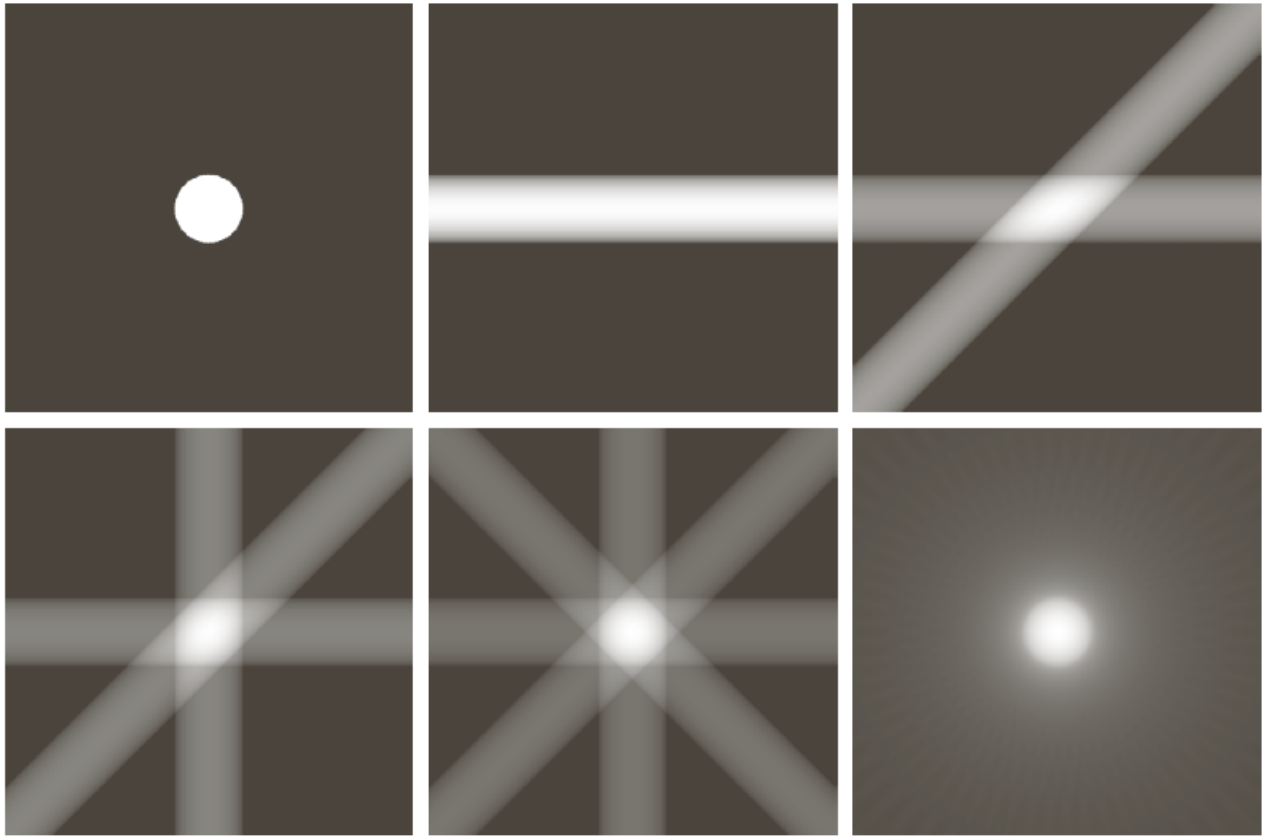


RT: Basic example in CT

By taking more projections:

- the form of the object becomes clearer as brighter regions will dominate the result

- back-projections with few interactions with the object will fade into the background.

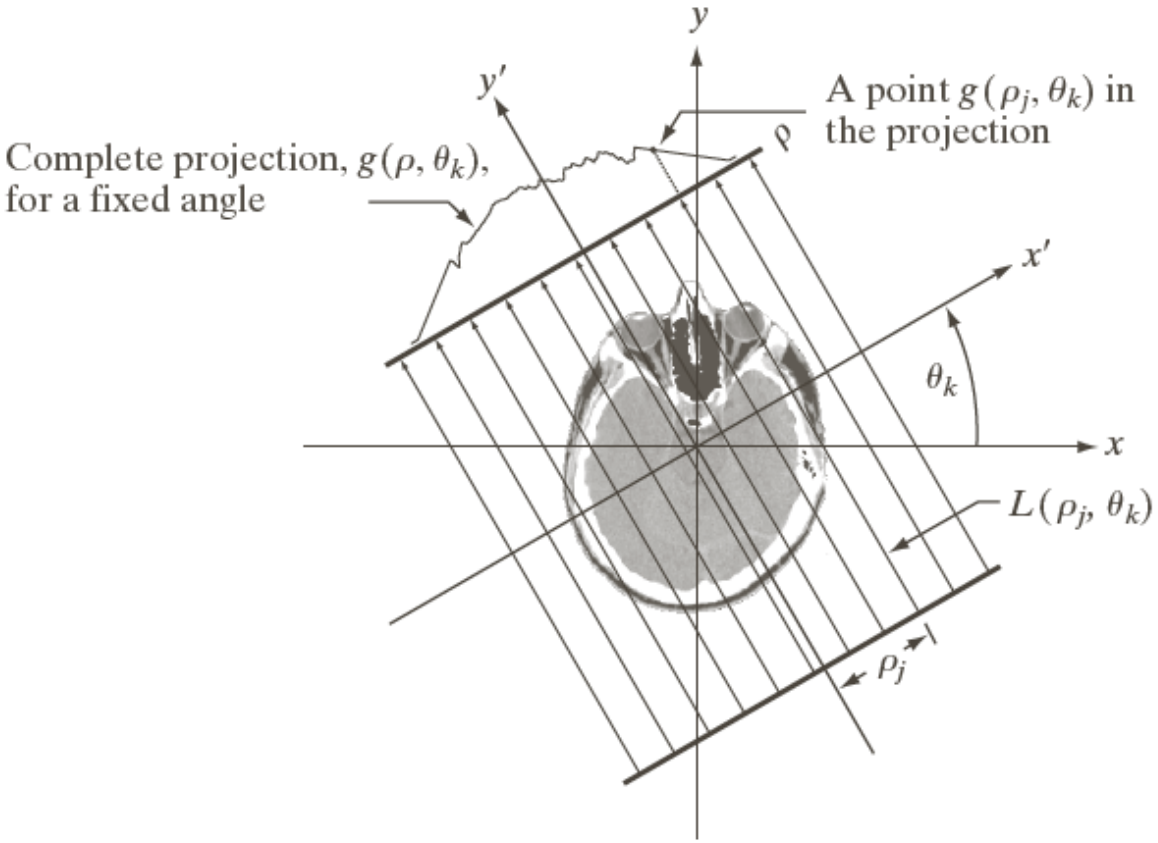


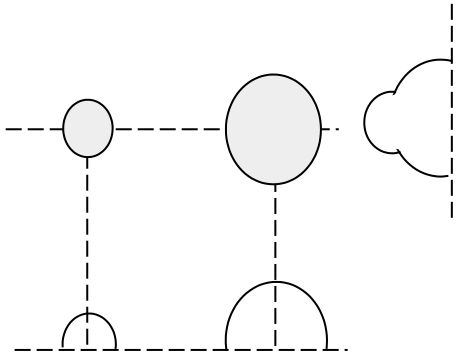
32 back-projections

RT: Radon transform

An arbitrary point (ρ_j, θ_k) in the projection signal is given by the **ray-sum** along the line

$$x\cos\theta_k + y\sin\theta_k = \rho_j.$$





RT: Radon transform

The ray-sum is a line integral:

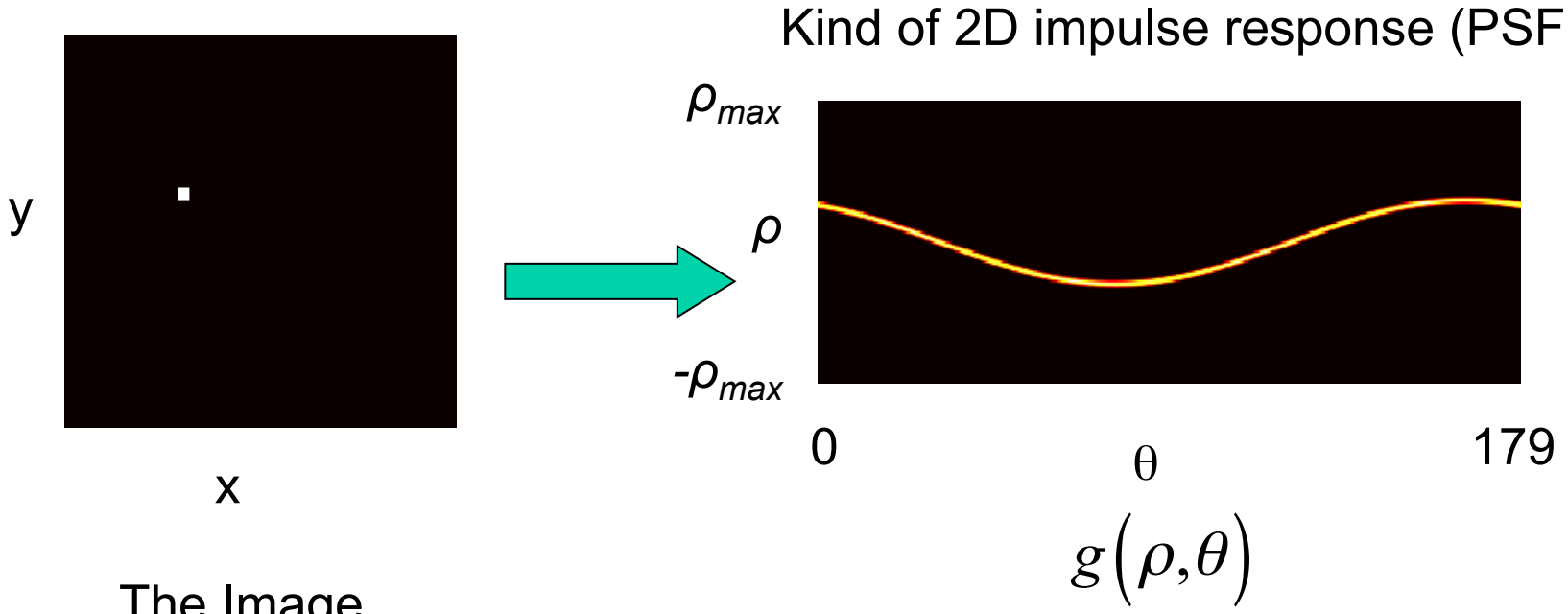
$$g(\rho_j, \theta_k) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y)\delta(x\cos\theta_k + y\sin\theta_k - \rho_j)dxdy$$

For all values of ρ and θ we obtain the **Radon transform**:

$$g(\rho, \theta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy$$

RT: Radon transform

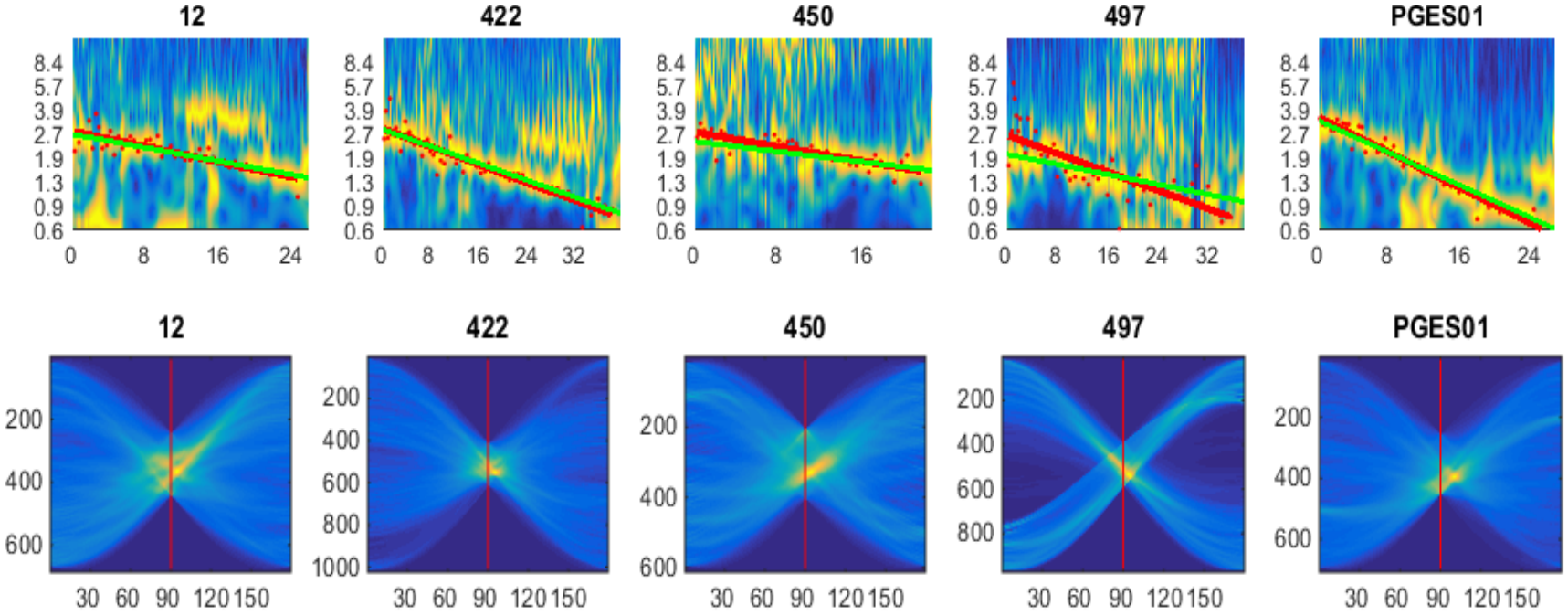
The representation of the Radon transform $g(\rho,\theta)$ as an image with ρ and θ as coordinates is called a ***sinogram***.



RT: Radon transform

Detection of Inter Clonus Interval evolution from Video Sequences: Epilepsy

Optic Flow ➔ Gabor spectrum ➔ Radon Transform

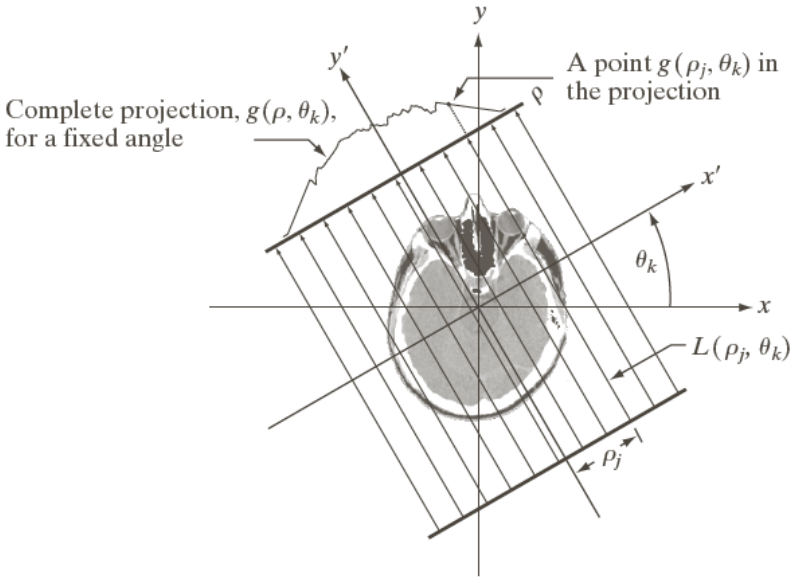


Courtesy Dr. S.N. Kalitzin, Netherlands Epilepsy Institute. From S.N. Kalitzin et al., Medicon 2016

RT: Inverse Radon transform

The approach is to back-project each projection and sum all the back-projections to generate a slice.

For a fixed rotation angle θ_k, and a fixed distance ρ_j, **back-projecting** the value of the projection $g(\rho_j,\theta_k)$ is equivalent to copying the value $g(\rho_j,\theta_k)$ to the image pixels belonging to the line $x\cos\theta_k+y\sin\theta_k=\rho_j$.



RT: Inverse Radon transform

Repeating the process for all values of ρ_j, having a fixed angle θ_k results in the following expression for the image values:

$$f_{\theta_k}(x,y) = g(\rho,\theta_k) = g(x\cos\theta_k + y\sin\theta_k, \theta_k)$$

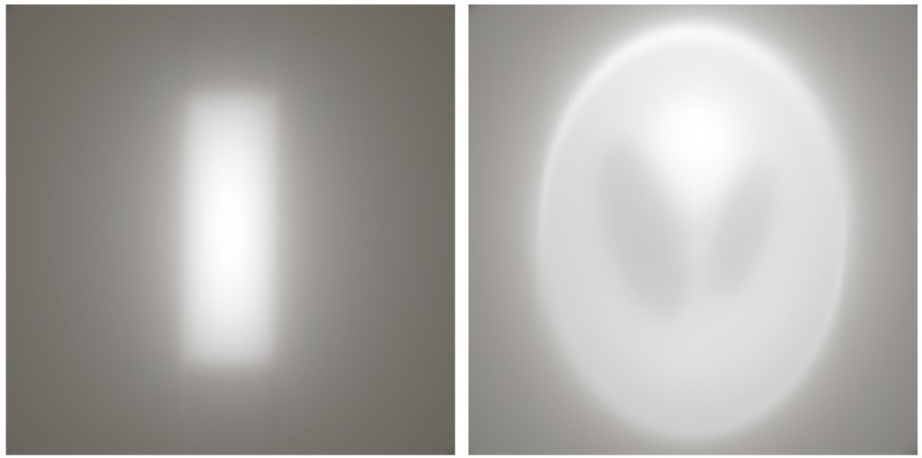
This equation holds for every angle θ:

$$f_{\theta}(x,y) = g(\rho,\theta) = g(x\cos\theta + y\sin\theta, \theta)$$

RT: Inverse Radon transform

The final image is formed by integrating over all the back-projected images:

$$f(x,y) = \int_0^{\pi} f_\theta(x,y)\, d\theta$$



Back-projection provides blurred images. We will reformulate the process to eliminate blurring.

RT: IRT - Fourier-slice theorem

The **Fourier-slice theorem** relates the 1D FT of a projection with the 2D FT of the region of the image from which the projection was obtained.

Let the 1D F.T. of a projection with respect to ρ (at a given angle) be:

$$G(\omega,\theta) = \int_{-\infty}^{+\infty} g(\rho,\theta)\, e^{-j2\pi\omega\rho}\, d\rho$$

Substituting the projection $g(\rho,\theta)$ by the ray-sum:

$$G(\omega,\theta) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\,\delta(x\cos\theta + y\sin\theta - \rho)\,dxdy\, e^{-j2\pi\omega\rho}\, d\rho$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\left[\int_{-\infty}^{+\infty} \delta(x\cos\theta + y\sin\theta - \rho) e^{-j2\pi\omega\rho}\, d\rho\right] dxdy$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y) e^{-j2\pi\omega(x\cos\theta + y\sin\theta)}\, dxdy$$

RT: IRT - Fourier-slice theorem

$$G(\omega,\theta)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)e^{-j2\pi\omega(x\cos\theta+y\sin\theta)}\,dxdy$$

Let now $u=\omega\cos\theta$ and $v=\omega\sin\theta$:

$$G(\omega,\theta)=\left[\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dxdy\right]_{u=\omega\cos\theta,v=\omega\sin\theta}$$

which is the 2D F.T. of the image $f(x,y)$ evaluated at the indicated frequencies u,v:

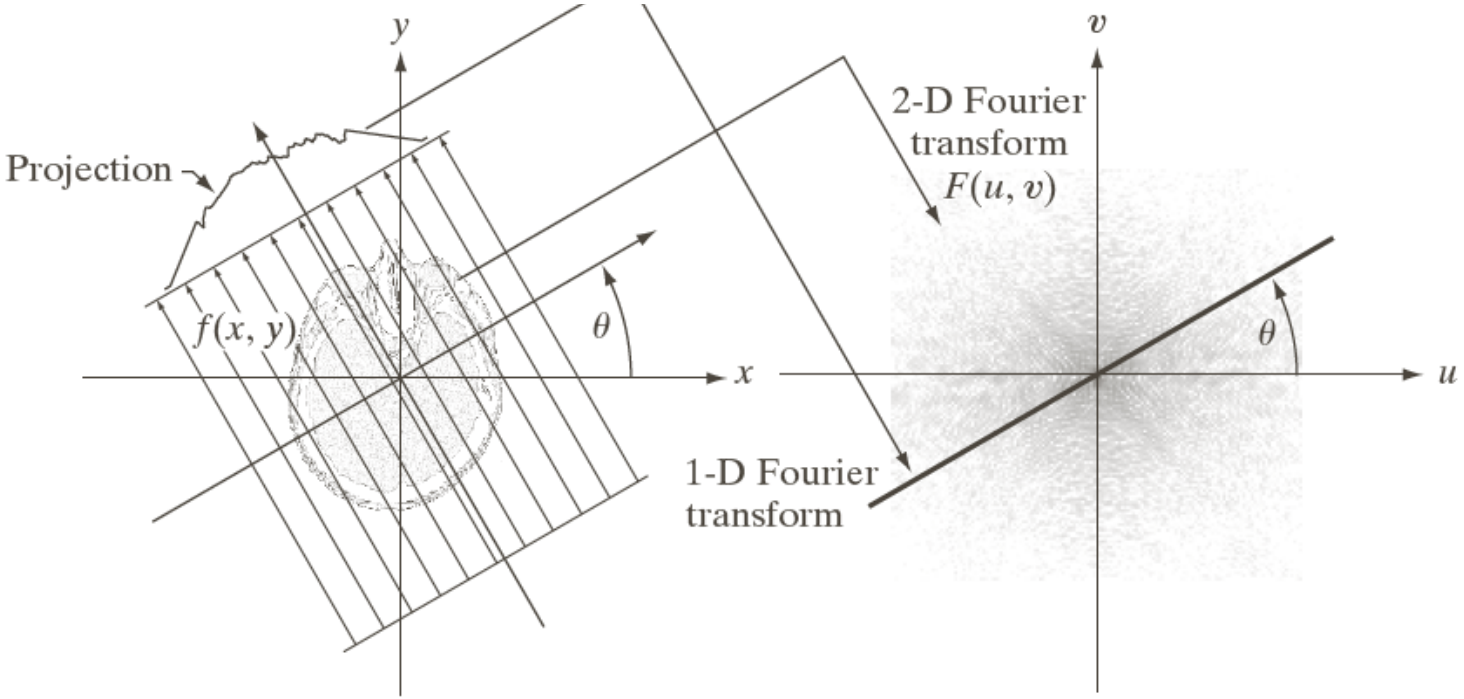
$$G(\omega,\theta)=\left[F(u,v)\right]_{u=\omega\cos\theta,v=\omega\sin\theta}=F(\omega\cos\theta,\omega\sin\theta)$$

RT: IRT - Fourier-slice theorem

The resulting equation $G(\omega,\theta) = F(\omega\cos\theta, \omega\sin\theta)$

is known as the Fourier-slice theorem.

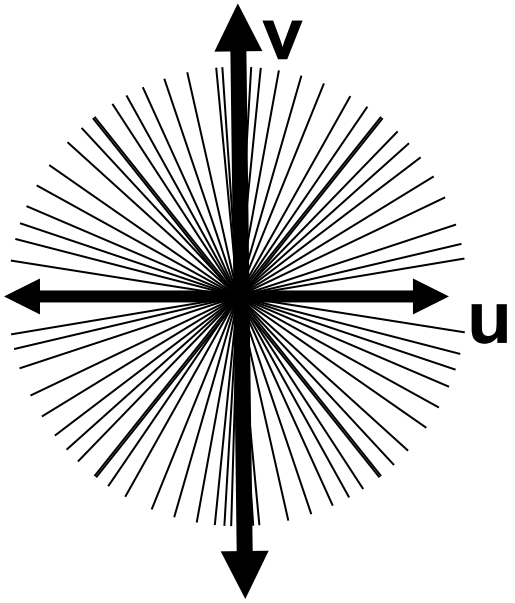
It states that the 1D F.T. of a projection (at a given angle θ) is a slice of the 2D F.T. of the image.



RT: IRT - Fourier-slice theorem

We could obtain *f(x,y)* by evaluating the F.T. of every projection and inverting them.

- It requires lots of Fourier Transforms
- We can't begin processing until we have all slices
- There's a more efficient way to organize things
- It requires "irregular" interpolation, worse at high frequencies



RT: IRT - Filtered back-projections

The 2D inverse Fourier transform of $F(u,v)$ is

$$f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

Letting $u=\omega\cos\theta$ and $v=\omega\sin\theta$ then the differential

$$du\,dv = \omega\,d\omega\,d\theta$$

and

$$f(x,y) = \int_{0}^{2\pi}\int_{0}^{+\infty} F(\omega\cos\theta,\omega\sin\theta)e^{j2\pi\omega(x\cos\theta+y\sin\theta)}\omega\,d\omega\,d\theta$$

RT: IRT - Filtered back-projections

Using the Fourier-slice theorem:

$$f(x,y)=\int_0^{2\pi}\int_0^{+\infty} G(\omega,\theta)e^{j2\pi\omega(x\cos\theta+y\sin\theta)}\omega\, d\omega\, d\theta$$

Make use of two facts: $G(\omega,\theta+\pi)=G(-\omega,\theta)$

$$\rho = x\cos\theta + y\sin\theta$$

$$f(x,y)=\int_0^{\pi}\int_{-\infty}^{+\infty} |\omega|\, G(\omega,\theta)e^{j2\pi\omega(x\cos\theta+y\sin\theta)}\, d\omega\, d\theta$$

The term $x\cos\theta+y\sin\theta=\rho$ and is independent of ω:

$$f(x,y)=\int_0^{\pi}\left[\int_{-\infty}^{+\infty} |\omega|\, G(\omega,\theta)e^{j2\pi\omega\rho}\, d\omega\right]_{\rho=x\cos\theta+y\sin\theta} d\theta$$

RT: IRT - Filtered back-projections

$$f(x,y) = \int_0^{\pi} \left[\int_{-\infty}^{+\infty} |\omega| G(\omega,\theta) e^{j2\pi\omega\rho} d\omega \right]_{\rho = x\cos\theta + y\sin\theta} d\theta$$

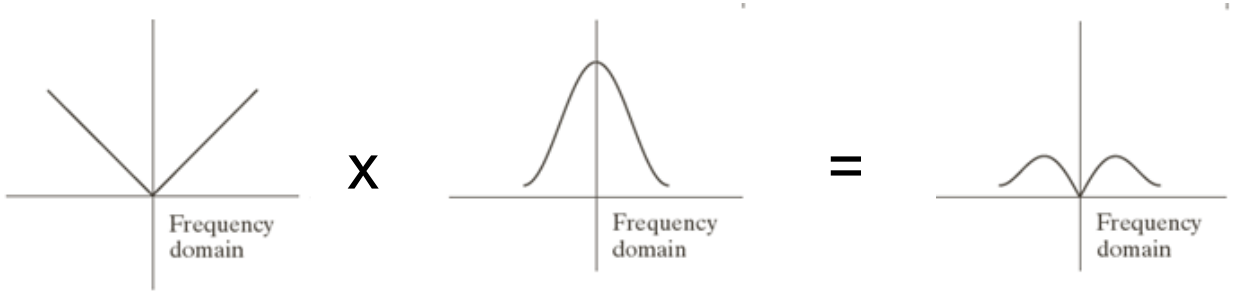
For a given angle θ, the inner expression is the 1-D Fourier transform of the projection multiplied by a ramp filter $|\omega|$.

This is equivalent in filtering the projection with a high-pass filter with Fourier Transform $|\omega|$ before back-projection.

RT: IRT - Filtered back-projections

Problem: the filter $H(\omega)=|\omega|$ is not integrable in the inverse Fourier transform as it extends to infinity in both directions. It should be truncated in the frequency domain.

The simplest approach is to multiply it by a box filter in the frequency domain. Ringing will be noticeable.



Windows with smoother transitions are used.

RT: IRT - Filtered back-projections

The *complete* back-projection is obtained as follows:

1. Compute the 1-D Fourier transform of each projection.
2. Multiply each Fourier transform by the filter function $|\omega|$ (multiplied by a suitable window, e.g. Hamming).
3. Obtain the inverse 1-D Fourier transform of each resulting filtered transform.
4. Back-project and integrate all the 1-D inverse transforms from step 3.

RT: Matlab

ORIGINAL IMAGE

```
P = phantom(256);
imshow(P)
```

Original image

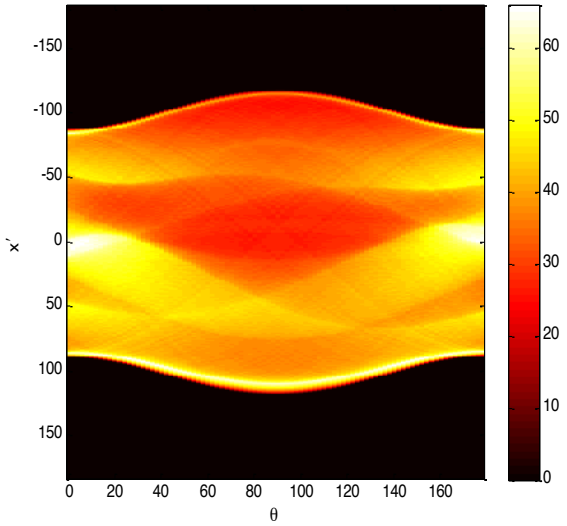
COMPUTE RADON TRANSFORM

```
theta = 0:2:178;  -> for 90 projections
[R,xp] = radon(P,theta);
```

DISPLAY SINOGRAM

```
figure, imagesc(theta,xp,R);
colormap(hot); colorbar
xlabel('\theta'); ylabel('x\prime');
```

Sinogram



RT: IRT - Matlab

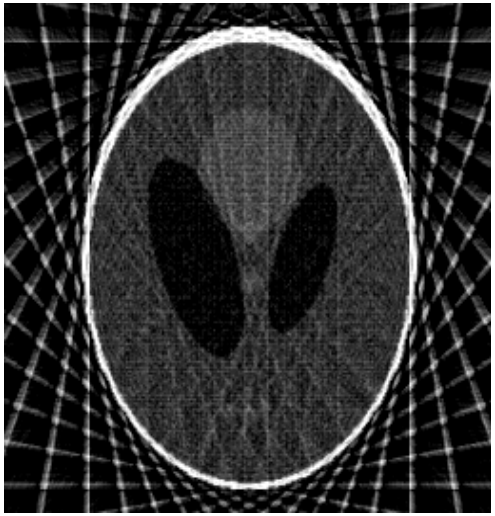
RECONSTRUCT

```
I = iradon(R,2);
```

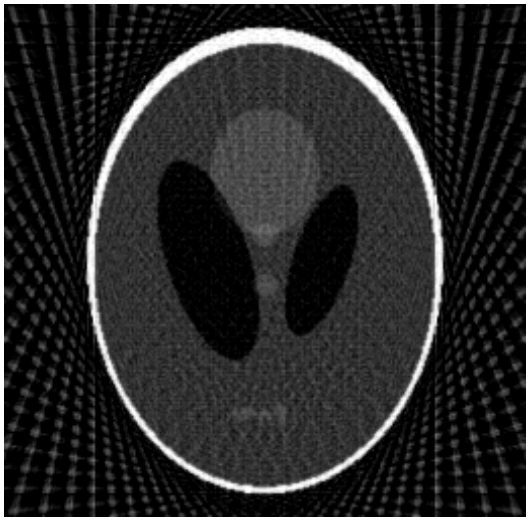
DISPLAY

```
figure, imshow(I)
```

18 projections



36 projections



RT: IRT - Matlab

Back-projection

Ramp FBP

Hamming FBP

RT: Overview of steps

Computed-Tomography