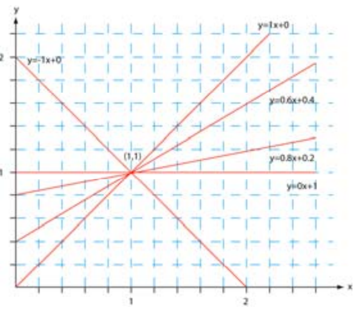
INF 4300 – Digital Image Analysis

HOUGH TRANSFORM



Fritz Albregtsen 11.09.2013

Plan for today

This lecture includes more details than G&W 10.2 !

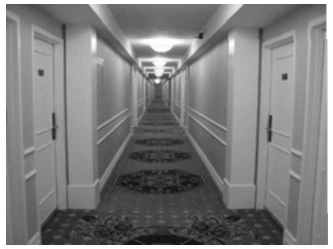
- Introduction to Hough transform
- Using gradient information to detect lines
- Representing a line
- The [a,b]-representation
- The [ρ,θ]-representation
- Algorithms for detection of lines
- Detecting circles and ellipses

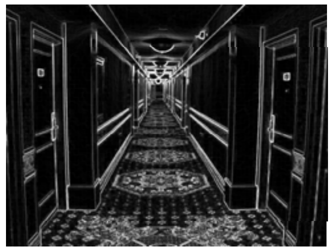
Introduction to HT

- The Hough transform (HT) can be used to detect
  - Lines
  - Circles
  - Other parametric curves
- Introduced by Paul Hough (1962)
- First used to find lines in images a decade later (Duda & Hart 1972).
- Our goal is to find the location of lines in images.
- This problem could be solved by e.g.
  - Morphology and a linear structuring element
    - Then we would need to handle rotation, zoom, distortions etc
  - Fitting a line (regression)
    - Then we need to know which edge pixels to use as input
- HT can give robust detection under noise and partial occlusion.

An image with linear structures

- Borders between the regions are straight lines.
- These lines separate regions with different grey levels.
- Edge detection is often used as preprocessing to Hough transform.





Remember edge types?

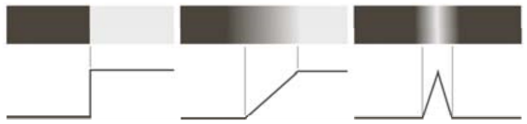

a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

Step edge Ramp edge Roof edge

Hough-transform – the input

- The input image must be a thresholded edge image.
- The magnitude results computed by e.g. the Sobel operator can be thresholded and used as input.

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

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

Repetition - Basic edge detection

- A thresholded edge image is the starting point for HT.
- What does a Sobel filter produce?
- An approximation to the image gradient which is a vector quantity given by:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Repetition – Edge magnitude

- The gradient is a measure of how the function f(x,y) changes as a function of changes in the arguments x and y.
- The gradient vector points in the direction of maximum change.
- The length of this vector indicates the size of the gradient:

$$|\nabla f| = \sqrt{g_x^2 + g_y^2}$$

G_x, G_y, and the gradient operator

- Horisontal edges:
  - Compute $g_x(x,y)=H_x*f(x,y)$
    - Convolve with the horisontal filter kernel H_x
- Vertical edges:
  - Compute $g_y(x,y)=H_y*f(x,y)$
    - Convolve with the vertical filter kernel H_y
- Compute the gradient parameters as:

$$G(x,y)=\sqrt{g_x^2(x,y)+g_y^2(x,y)}$$ Gradient magnitude

$$\theta(x,y)=\tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$$ Gradient-direction

- Note: from one input image, you produce four images:
  - $g_x(x,y)$, $g_y(x,y)$, $G(x,y)$, and $\theta(x,y)$.

Input to Hough – thresholded edge image

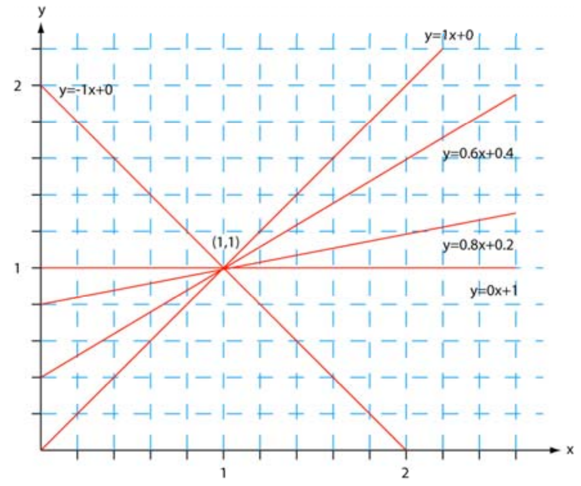
Prior to applying Hough transform:

- Compute edge magnitude from input image.
- As always with edge detection, simple lowpass filtering can be applied first.
- Threshold the gradient magnitude image.

Hough-transform

- Assume that we have performed some edge detection, and a thresholding of the gradient magnitude image.
- We have *n* pixels that may partially describe the boundaries of some objects.
- We wish to find sets of pixels that make up straight lines.
- Regard a point (x_i; y_i) and a straight line $y_i = ax_i + b$
  - Many lines may pass through the point (x_i,y_i).
  - Common to them is that they satisfy the equation above for varying values of the parameters (a,b).

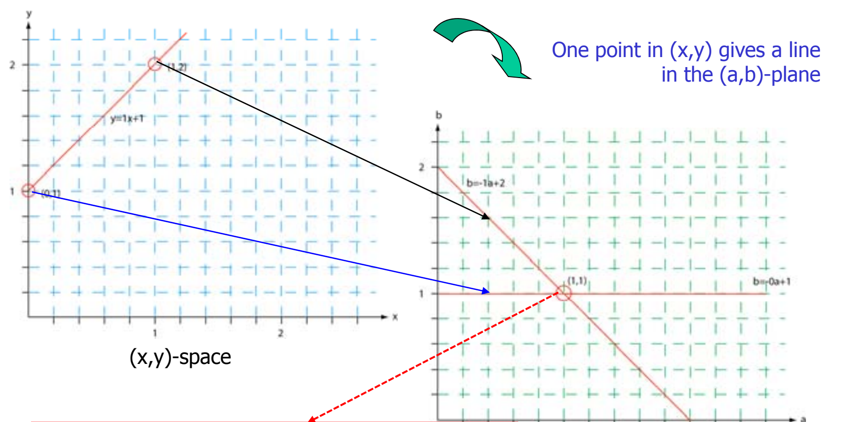
Hough transform – basic idea



Hough transform – basic idea

- Given a line in the *(x,y)* - plane:
  $y = ax + b$
- This equation can obviously be rewritten as follows:
  $$b = -xa + y$$
- We now consider *x* and *y* as parameters
  and *a* and *b* as variables.
- This is a line in (*a,b*) - plane parameterized by *x* and *y*.
  - So: a point (x_1,y_1) in *xy* - space gives a line in *(a,b)* space.
  - Another point (x_1,y_1) on the same line $y = ax + b$ in *xy* - space will give rise to another line in (*a,b*) - space.

Hough transform – basic idea



One point in (x,y) gives a line in the (a,b)-plane

Line intersection in (ab)-space gives line parameters in (x,y)-space

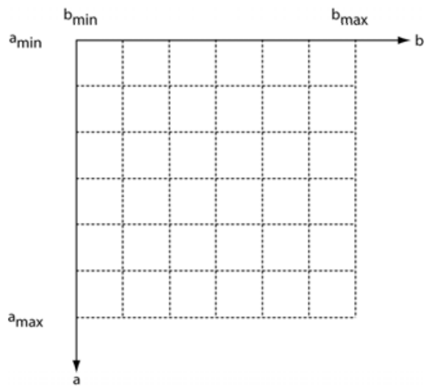
Hough transform – basic idea

- Two points (x,y) and (z,k) define a line in the (x,y) plane.
- These two points give two different lines in (a,b) space.
- In (a,b) space these lines will intersect in a point (a’,b’) where a’ is the slope and b’ the intercept of the line defined by (x,y) and (z,k) in (x,y) space.
- In fact, all points on the line defined by (x,y) and (z,k) in (x,y) space will parameterize lines that intersect in (a’,b’) in (a,b) space.
- Points that lie on a line in the (x,y) space will form a “cluster of crossings” in the (a,b) space.

Hough transform – algorithm

- Quantize the parameter space (a,b), i.e., divide it into cells.
- This quantized space is often referred to as <u>the accumulator cells</u>.
- In the figure in the next slide a_{min} is the minimal value of a, etc.
- Count the number of times a line intersects a given cell.
  - For each point (x,y) with value 1 in the binary image, find the values of (a,b) in the range $[[a_{min},a_{max}],[b_{min},b_{max}]]$ defining the line corresponding to this point.
  - Increase the value of the accumulator for these [a’,b’] point.
  - Then proceed with the next point in the image.
- Cells receiving more than a given number of “votes” are assumed to correspond to lines in (x,y) space.
  - Lines can be found as peaks in this accumulator space.

Hough transform - algorithm



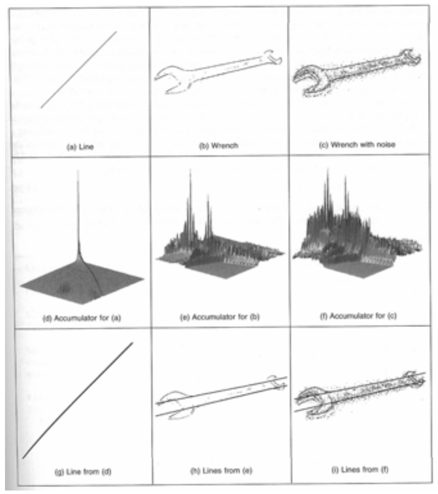
Hough accumulator cells

Example – images and accumulator space

Thresholded edge images

Visualizing the accumulator space
The height of the peak will be defined by the number of pixels in the line.

Thresholding the accumulator space and superimposing corresponding lines onto the edge image



Note how noise affects the accumulator. Even with noise, the largest peaks correspond to the major lines.

Polar representation of lines

- In practice, we do not use the equation

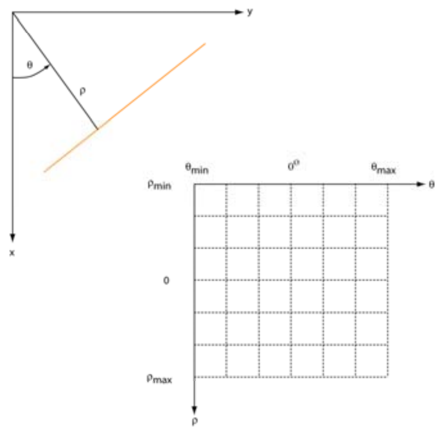
$$y = ax + b$$

in order to represent lines (why?)

- Rather, we use the polar representation of lines:

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

HT and polar representation of lines



Polar representation of lines

Hough transform and the polar representation

- The polar (also called normal) representation of straight lines is

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

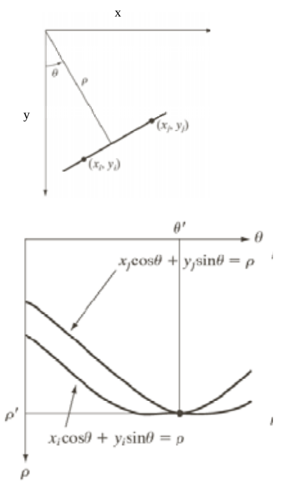
- Each point (x_i,y_i) in the xy-plane gives a sinusoid in the $\rho\theta$-plane.
- M colinear point lying on the line

$$x_i\cos\theta + y_i\sin\theta = \rho$$

  will give M curves that intersect at (ρ_i,θ_j) in the parameter plane.
- Local maxima in parameter domain
  correspond to significant lines in the image domain.

Two points on a line …

- Each curve in the figure represents the familiy of lines that may pass through a particular point (x_i,y_i) in the xy-plane.
- The intersection point (ρ',θ') corresponds to the line that passes through two points (x_i,y_i) and (x_j,y_j)
- A horizontal line will have θ=0 and ρ equal to the intercept with the y-axis.
- A vertical line will have θ=90 and ρ equal to the intercept with the x-axis.


x
y
θ
ρ
(x_j, y_j)
(x_i, y_i)
θ'
θ
$x_j\cos\theta + y_j\sin\theta = \rho$
ρ'
$x_i\cos\theta + y_i\sin\theta = \rho$
ρ

HT with (ρ,θ)-representation of lines

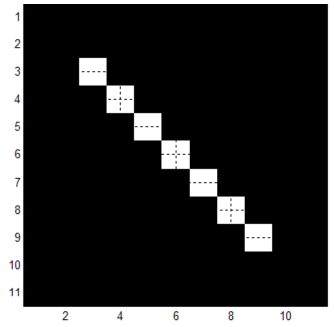
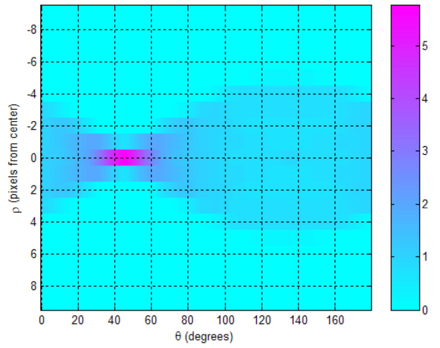
- Partition the $\rho\theta$-plane into accumulator cells $A[\rho,\theta]$, $\rho\in[\rho_{min}, \rho_{max}]$; $\theta\in[\theta_{min}, \theta_{max}]$
- The range of θ is $\pm 90°$
  - Horizontal lines have $\theta=0°$, $\rho\geq 0$
  - Vertical lines have $\theta=90°$, $\rho\geq 0$
- The range of ρ is $\pm$ sqrt(M^2+N^2)
  if the image is of size MxN
- The discretization of θ and ρ must give acceptable precision within and size of the parameter space.

Algorithm continued

- The cell (i,j) corresponds to the square associated with parameter values (θ_j, ρ_i).
- Initialize all A(i,j) cells with value 0.
- For each foreground point (x_k,y_k) in thresholded edge image
  - For all possible θ-values
    - Solve for ρ using $\rho=x_k \cos\theta_j + y_k \sin\theta_j$
    - Round ρ to the closest cell value, ρ_q
    - Increment A(i,q) if the θ_j results in ρ_q
- A(i,j)=P means that P points in the xy-space lie on the line
  $\rho_j = x \cos\theta_j + y\sin\theta_j$
- Find line candiates where A(i,j) is above a suitable threshold.

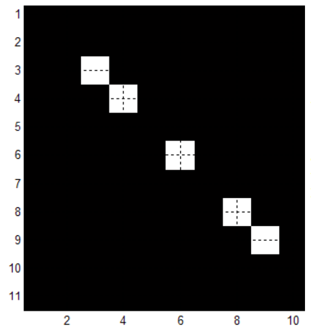
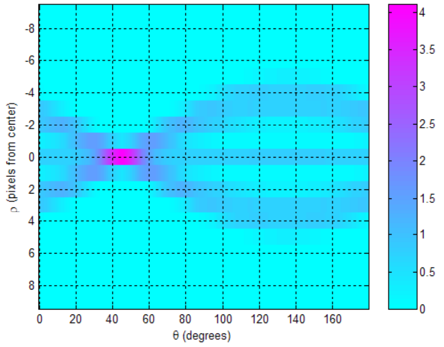
Hough transform – example 1

- Example 1: 11x11 image and its Hough transform:

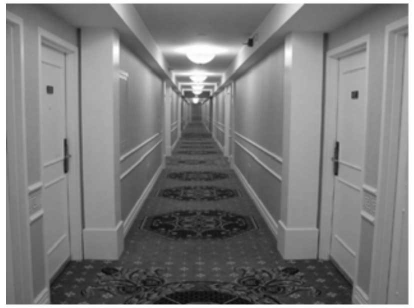
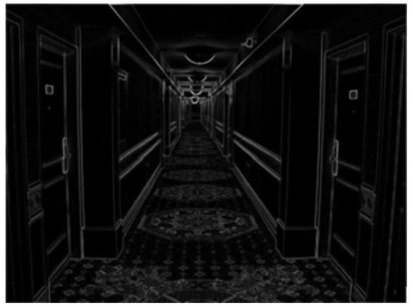
Hough transform – example 2

- Example 2: 11x11 image and its Hough transform:

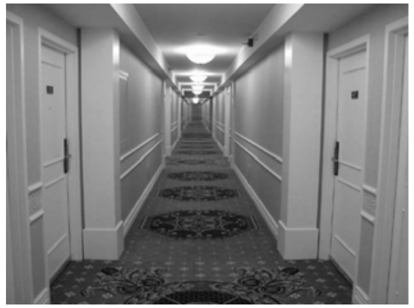
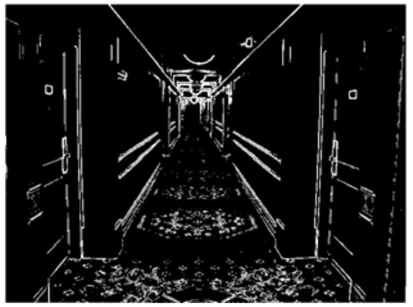
Hough transform – example 3

- Example 3: Natural scene & result of Sobel magnitude:

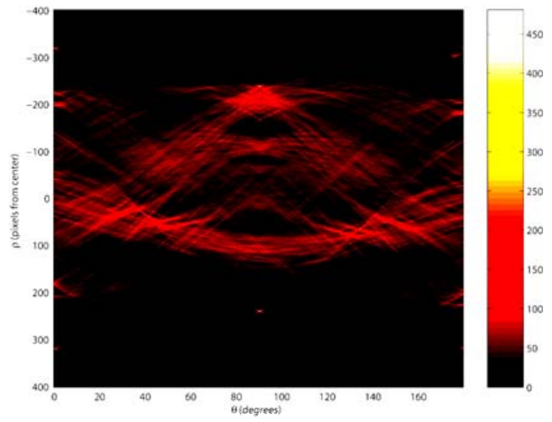
Hough transform – example 3

- Example 3: Natural scene and result of Sobel gradient magnitude followed by thresholding:

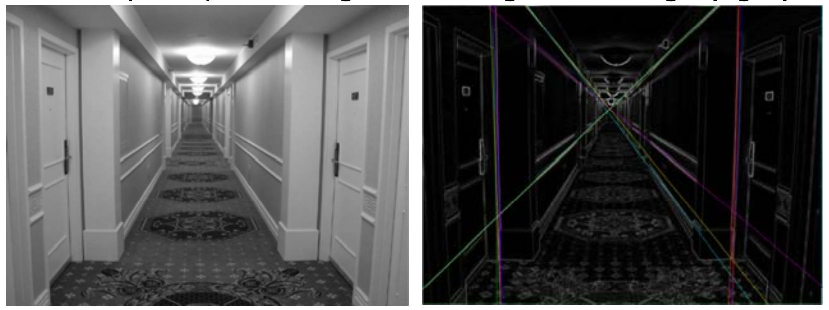
Hough transform – example 3

- Example 3: Accumulator matrix:



Hough transform – example 3

- Example 3: Original image (left) and 20 most prominent lines superimposed on gradient magnitude image (right):



HT - advantages

- Advantages:
  - Conceptually simple.
  - Easy implementation.
  - Handles missing and occluded data very gracefully.
  - Can be adapted to many types of forms, not just lines.

HT - disadvantages

- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object.
  - Can be “fooled” by “apparent lines”.
  - Length and position of a line segment cannot be determined.
  - Co-linear line segments cannot be separated.

HT using the full gradient information

- Given a gradient magnitude image g(x,y) containing a line.
- Simple algorithm:

```
for all $g(x_i,y_i)>T$ do
    for all $\theta$ do
        $\rho = x_i \cos\theta + y_i \sin\theta$
        find indexes (m,n) corresponding to $(\rho, \theta)$ and increment A(m,n);
```

- Better algorithm if we have both
  - The gradient magnitude g(x,y)
  - And the gradient components g_x and g_y
    - So we can compute gradient direction

$$\phi_g(x,y) = \arctan\left(\frac{g_y}{g_x}\right)$$

- The new algorithm:

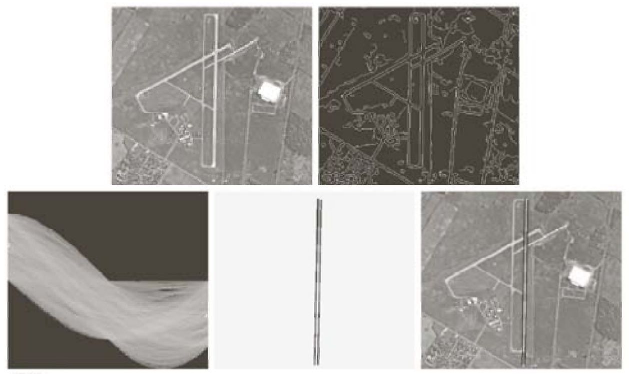
```
for all $g(x_i,y_i)>T$ do
    $\rho = x_i \cos(\phi_g(x,y)) + y_i \sin(\phi_g(x,y))$
    find indexes (m, n) corresponding to $(\rho, \phi_g(x,y))$, increment A(m,n);
```

HT and line & edge linking

Q: How can we look for lines with a certain orientation?

1. Obtain a thresholded edge image
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells
   for high pixel concentrations (= length of line).
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.
   - Continuity here normally means distance between disconnected pixels.
   - A gap in the line can be bridged if the length of the gap is less than a certain threshold.

Using edge linking



a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

Hough transform for circles

- A circle in the xy-plane is given by

$$(x-x_c)^2 + (y-y_c)^2 = r^2$$

- So we have a 3D parameter space.

- **Simple 3D accumulation procedure:**

```
set all $A[x_c,y_c,r]=0$;
for every (x,y) where $g(x,y)>T$
    for all $x_c$ and $y_c$
        $r=\mathrm{sqrt}((x-x_c)^2+(y-y_c)^2)$;
        $A[x_c,y_c,r] = A[x_c,y_c,r]+1$;
```

- **Better procedure … ?**

Hough transform for ellipses

- A general ellipse in the xy-plane has 5 parameters:
  - Position of centre (x_c, y_c), semi-axes (a,b), and orientation (θ).
- Thus, we have a 5D parameter space.
- For large images and full parameter resolution,
  straight forward HT may easily overwhelm your computer!
- **Reducing accumulator dimensionality:**
  - Pick pixel pairs with opposite gradient directions
  - Midpoint of pair ”votes” for centre of ellipse!
  - Reduces HT-accumulator from 5D to 3D.
  - Min and max distance of pixel pairs => 2b & 2a
- **Reducing accumulator extent & resolution**
- **Using other tricks ...**

Hough transform – exercise 1

- Familiarize with Matlab function for line detection:
  - Functions hough(), houghpeaks(), and houghlines()
- Next exercise:
  - Test Hough transform for equal size circles on the coins image.

Hough transform – exercise 2

Next exercise: The randomized Hough transform.

- Simple idea (line case):
  - From the edge image, pick two points.
  - Find the ρ and θ corresponding to this set of points.
  - Increment the indicated (ρ,θ) cell.
  - Once a cell reaches a certain (low) count, assume that an edge is present in the image.
  - Verify this.
  - If truly present:
    - Store line parameters
    - Erase this line from the edge image
  - Continue until no more points or until the number of iterations between two detections is to high.
  - Orders of magnitude faster than the ordinary transform.

Learning goals - HT

- Understand the basic Hough Transform, and its limitations
- Understand that the normal representation is more general
- Be able to implement line detecting HT with accumulator matrix
  - Understand how this may be improved by gradient direction information.
- Be able to implement simple HT for circles of given size,
  in order to find position of circular objects in image.
  - Understand how this may be improved.
- Understand simple HT to detect ellipses
  - Understand how this may be improved.
- Understand the basic random Hough transform.
- Do the exercises!