Canny Edge Detector

1) Smooth image with a Gaussian
   - optimizes the trade-off between noise filtering and edge localization
2) Compute the Gradient magnitude using approximations of partial derivatives
   - 2x2 filters
3) Thin edges by applying non-maxima suppression to the gradient magnitude
4) Detect edges by double thresholding

Gradient

- At each point convolve with

$$G_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

- magnitude and orientation of the Gradient are computed as

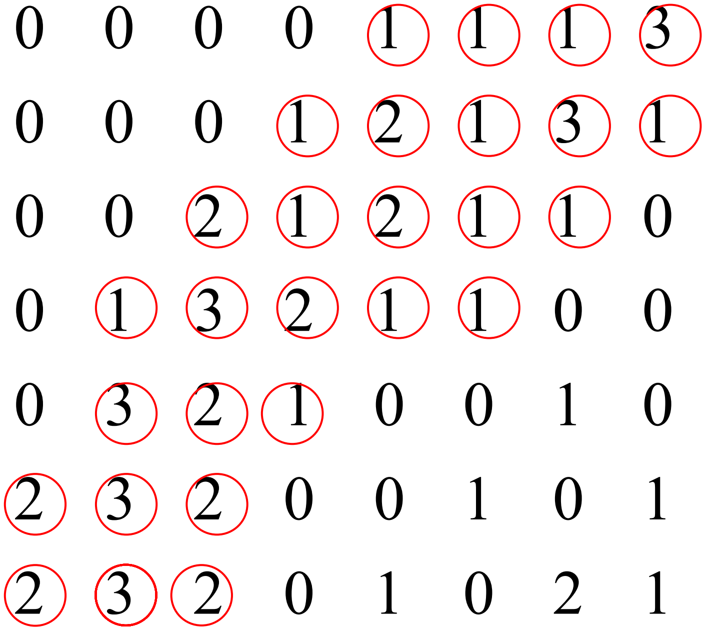
$$M[i,j] = \sqrt{P[i,j]^2 + Q[i,j]^2}$$

$$\theta[i,j] = \tan^{-1}(Q[i,j], P[i,j])$$

- Avoid floating point arithmetic for fast computation

Non-Maxima Suppression

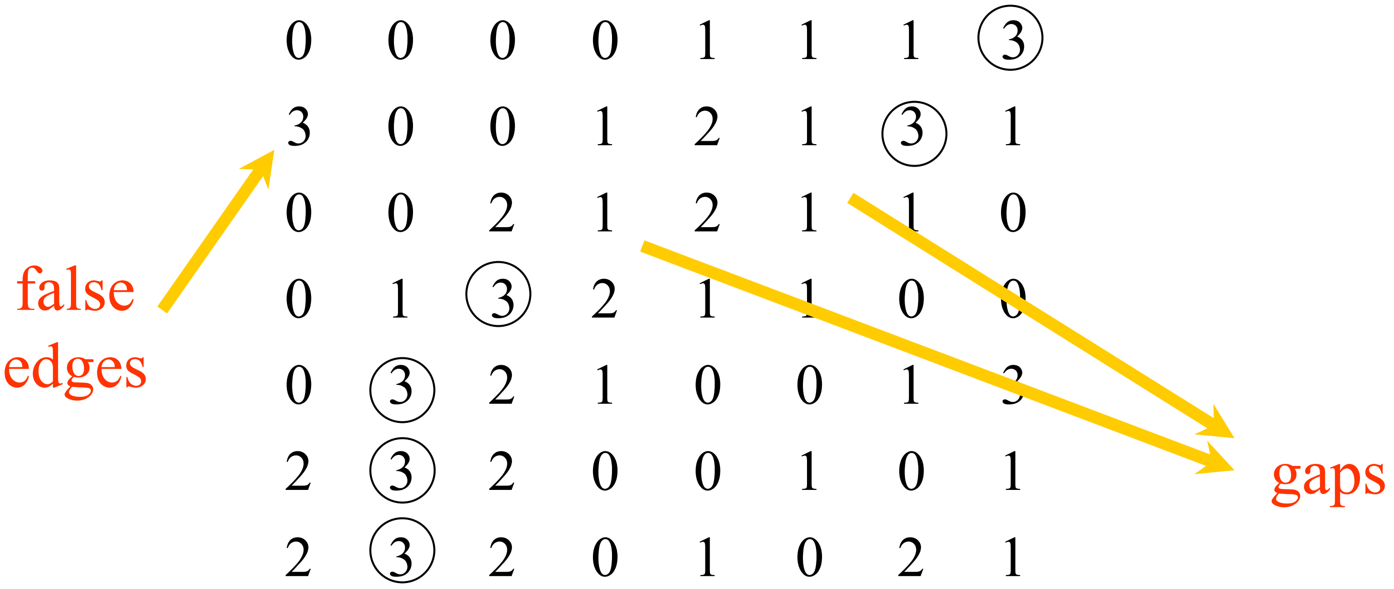
- Thin edges by keeping large values of Gradient
  - not always at the location of an edge
  - there are many thick edges



0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

Non-Maxima Suppression (2)

- Thin the broad ridges in M[i,j] into ridges that are only one pixel wide
- Find local maxima in M[i,j] by suppressing all values along the line of the Gradient that are not peak values of the ridge



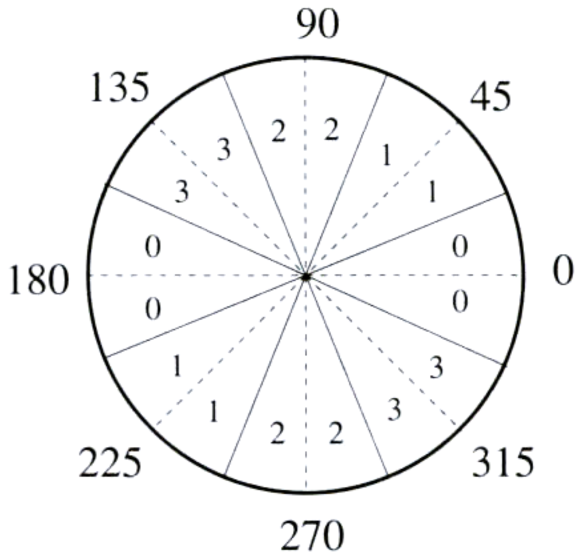
0	0	0	0	1	1	1	③
3	0	0	1	2	1	③	1
0	0	2	1	2	1	1	0
0	1	③	2	1	1	0	0
0	③	2	1	0	0	1	3
2	③	2	0	0	1	0	1
2	③	2	0	1	0	2	1

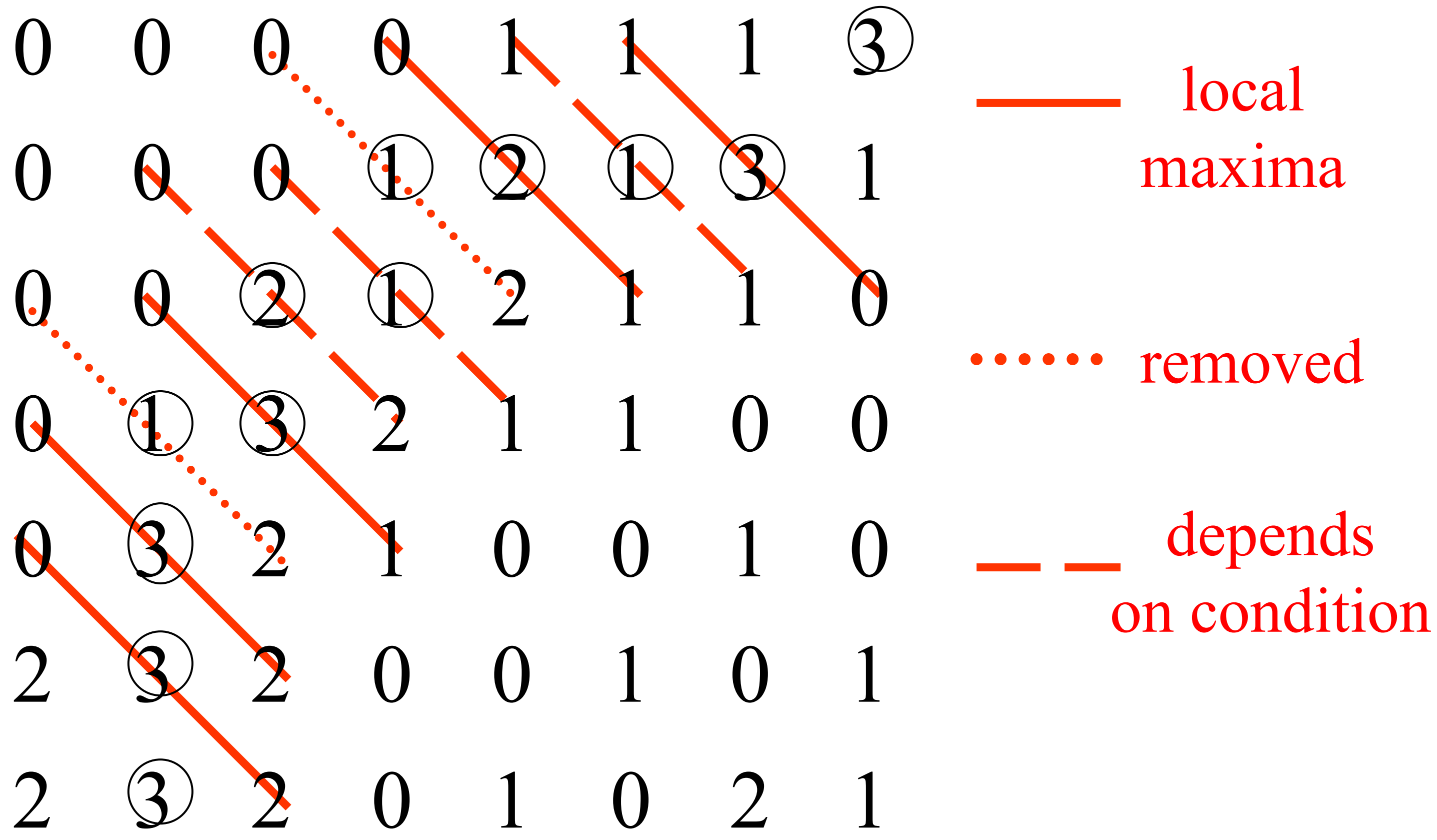
false edges

gaps

Gradient Orientation

- Reduce angle of Gradient θ[i,j] to one of the 4 sectors
- Check the 3x3 region of each M[i,j]
- If the value at the center is not greater than the 2 values along the gradient, then M[i,j] is set to 0



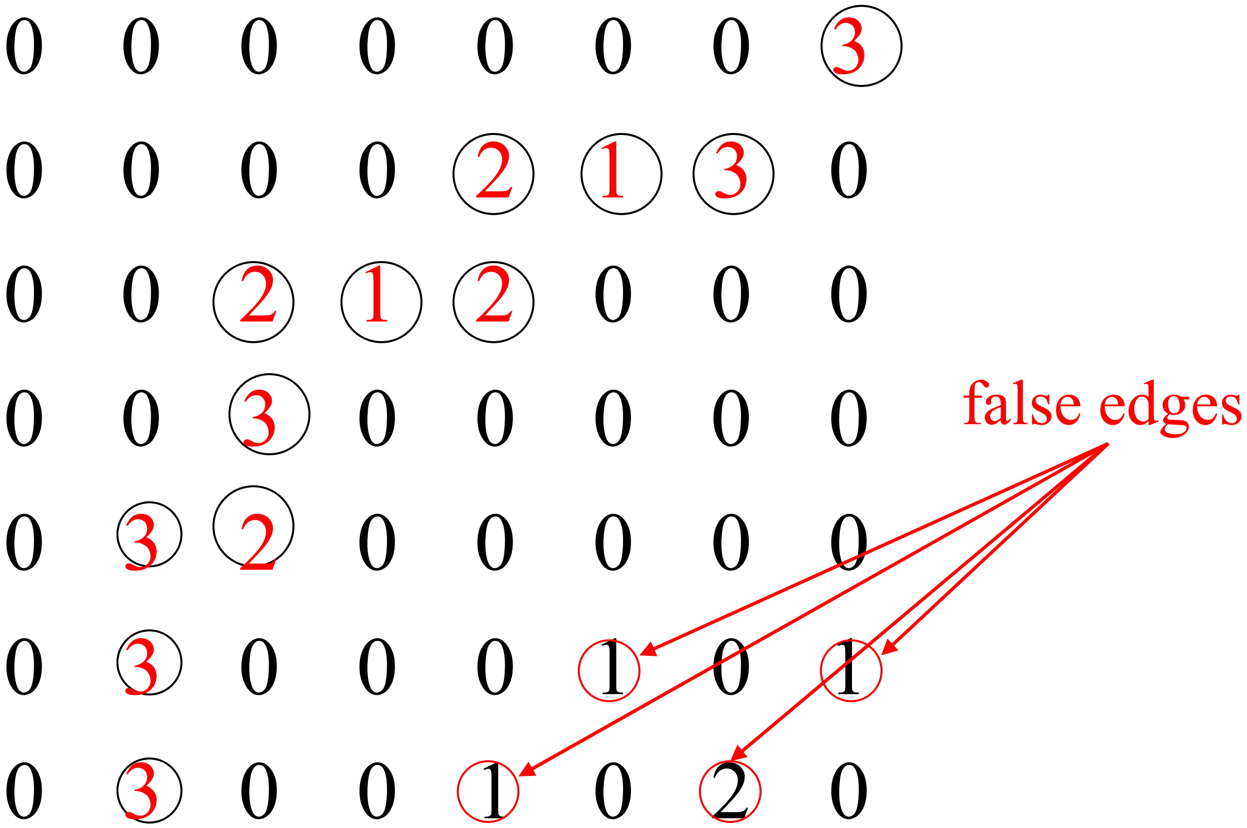


0	0	0	0	1	1	1	③
0	0	0	①	②	①	③	1
0	0	②	①	2	1	1	0
0	①	③	2	1	1	0	0
0	③	2	1	0	0	1	0
2	③	2	0	0	1	0	1
2	③	2	0	1	0	2	1

—— local maxima

•••••• removed

— — depends on condition



0	0	0	0	0	0	0	3
0	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

false edges

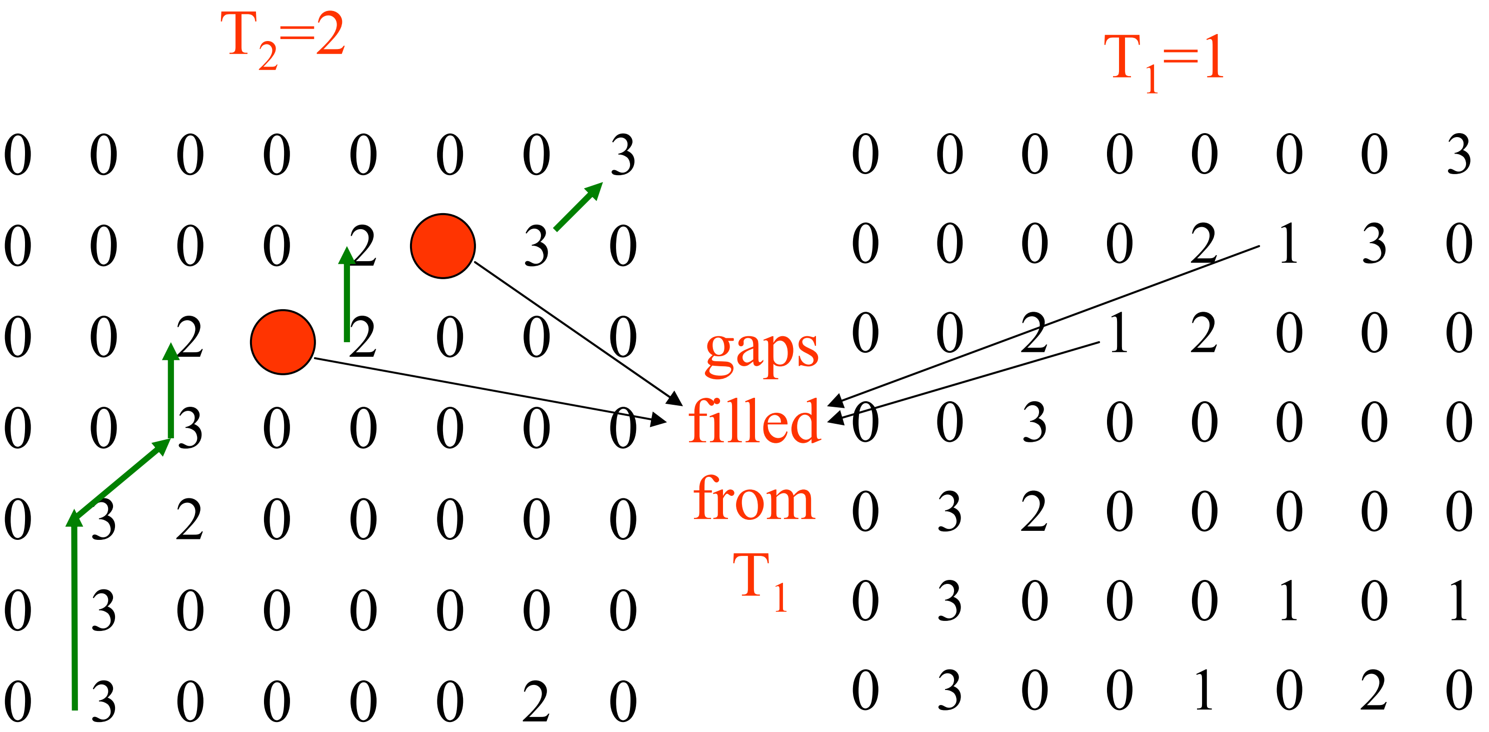
- The suppressed magnitude image will contain many false edges caused by noise or fine texture

Thresholding

- Reduce number of false edges by applying a threshold T
  - all values below T are changed to 0
  - selecting a good values for T is difficult
  - some false edges will remain if T is too low
  - some edges will disappear if T is too high
  - some edges will disappear due to softening of the edge contrast by shadows

Double Thresholding

- Apply two thresholds in the suppressed image
  - $T_2 = 2T_2$
  - two images in the output
  - the image from T_2 contains fewer edges but has gaps in the contours
  - the image from T_1 has many false edges
  - combine the results from T_1 and T_2
  - link the edges of T_2 into contours until we reach a gap
  - link the edge from T_2 with edge pixels from a T_1 contour until a T_2 edge is found again



$T_2=2$

0	0	0	0	0	0	0	3
0	0	0	0	2	●	3	0
0	0	2	●	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	0	0	0
0	3	0	0	0	0	2	0

gaps filled from T_1

$T_1=1$

0	0	0	0	0	0	0	3
0	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

- A T_2 contour has pixels along the green arrows
- Linking: search in a 3x3 of each pixel and connect the pixel at the center with the one having greater value
- Search in the direction of the edge (direction of Gradient)

Edge Linking

- Fill gaps in the Canny edge
  - e.g., after thresholding follow edge

    4 $T_2=8$

    9 $T_1=4$ → 4 is lost!!

    5

    6

  - scan bottom-up and combine the edges
  - scan left-to-right and right-to-left
  - scan across the diagonals

Line Detection

- Model of a line: two edges with opposite polarity in distance less than the size of the smoothing filter
  - edge detection filters respond to step edges
  - they do not provide meaningful response to lines
- Apply nonmaxima suppression on the smoothed output
  - a line is the derivative of a step → the derivative step of the Canny algorithm is not necessary