Camera Calibration
Edge Detection

Lecture 9

Outline of Alternate Proof: Plane + Perspective Model

- Consider Z=0 Plane in the world
- Then,

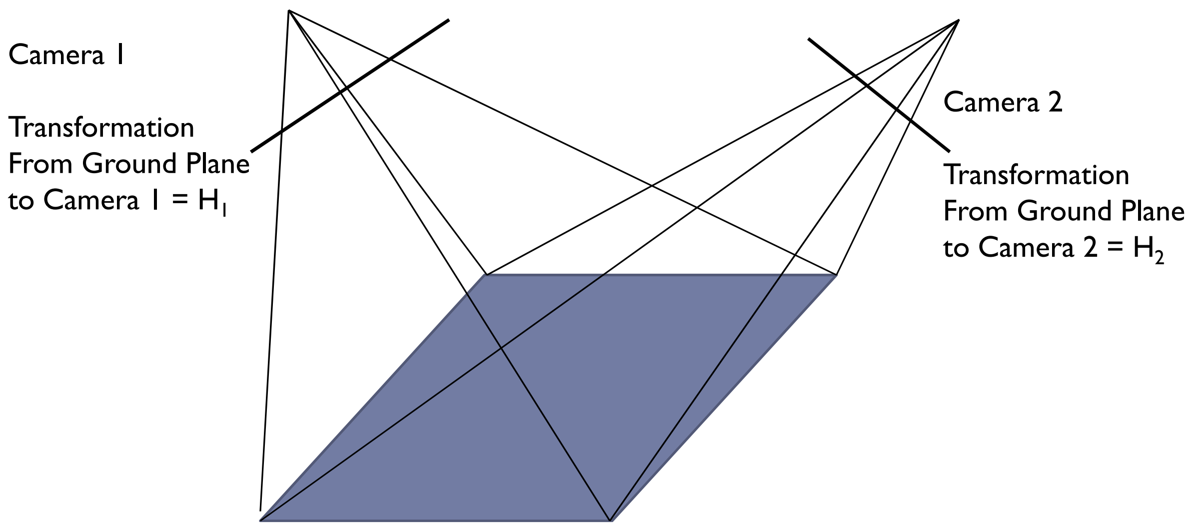
$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z=0 \\ 1 \end{bmatrix} = \begin{bmatrix} . & . & . \\ . & . & . \\ . & . & . \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

3x4 perpective transform matrix
Note: 3^{rd} column does not matter because of Z=0

- Since in homogeneous coordinates, scale factor does not matter, the 3x3 matrix is a projective transform between the world plane and the camera image plane.

Continued…

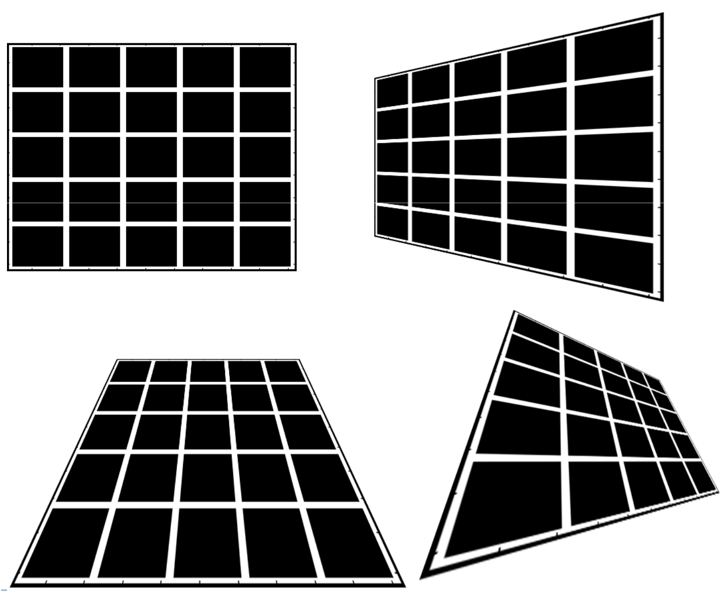
Outline of Alternate Proof



Transformation From Camera 2 to Camera 1 will be $H_1H_2^{-1}$ (Why?)

It will be a projective transform if projective transformation operation forms a group (Prove)

Examples of Projective Transformations



Camera Calibration

- To relate 3D world points to 2D camera points, we need to know a lot of things about the camera
  - Camera Location X,Y,Z
  - Camera orientation α, β
  - Gimbal vector $(r_1, r_2, r_3)^T$
  - Focal length f
  - Size of CCD array
  - Center of projection
- Intrinsic parameters: internal to the camera.
  - Do not change when camera is moved
- Extrinsic parameters: External to the camera.
  - Change when camera is moved

Camera Calibration

- In general, the camera model looks like:

$$C_h = AW_h;$$

$$\begin{bmatrix} C_{h_1} \\ C_{h_2} \\ C_{h_3} \\ C_{h_4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Calibration is the process of finding the parameters $[a_{11} \ldots a_{44}]$
- If W_h and C_h are known, then we can solve for the unknown parameters

Camera Calibration

$$x = \frac{C_{h_1}}{C_{h_4}},$$
$$y = \frac{C_{h_2}}{C_{h_4}}.$$

$$\begin{aligned}
C_{h_1} &= a_{11}X + a_{12}Y + a_{13}Z + a_{14} = C_{h_4}x \\
C_{h_2} &= a_{21}X + a_{22}Y + a_{23}Z + a_{24} = C_{h_4}y \\
C_{h_4} &= a_{41}X + a_{42}Y + a_{43}Z + a_{44}
\end{aligned}$$

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - a_{41}Xx - xa_{42}Y - xa_{43}Z - xa_{44} = 0$$
$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - a_{41}Xy - ya_{42}Y - ya_{43}Z - ya_{44} = 0.$$

Camera Calibration

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - a_{41}Xx - xa_{42}Y - xa_{43}Z - xa_{44} = 0$$
$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - a_{41}Xy - ya_{42}Y - ya_{43}Z - ya_{44} = 0.$$

- This equation has 12 unknowns
- Each correspondence yields two equations
- If 6 correspondences are known, we can solve for the unknowns

Camera Calibration

- Separating out the knowns and the unknowns

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ \vdots & & & & & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \vdots & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

or

$$CP = \mathbf{0},$$

Camera Calibration

- This system CP = 0 is a homogeneous system.
- C is rank deficient: rank(C) = 11
- Has multiple solutions (other than the trivial solution)… Can be solved uniquely only up to a scale factor
- Solution?

Solving for P

- The null space of C represents the P which are the solutions to the system CP = 0
- How to find null space?
  1. null(C) in MATLAB
  2. Take SVD of C, as svd(C) = USV^T. The column of V corresponding to the eigen value of zero represents the solution

  (in practice, you will have to take the smallest eigen value)

End of Module 1

- Introduction, overview of the course, policies
- General introduction of Computer Vision and the course modules
- Human eye
- Digital images
- Capturing color images
- Image histogram
- Interlacing
- 2D Transformations
- 2D Displacement Models
  - Affine, Projective
  - Recovering best transformation
- 3D Transformations
- Perspective, orthographic transforms
- Camera Model
- Planarity assumption
- Camera Calibration

Edge Detection

Applying Masks to Images

- Convolution Operation
- Mask
  - Set of pixel positions and weights
  - Origin of mask

1	1	1
1	1	1
1	1	1

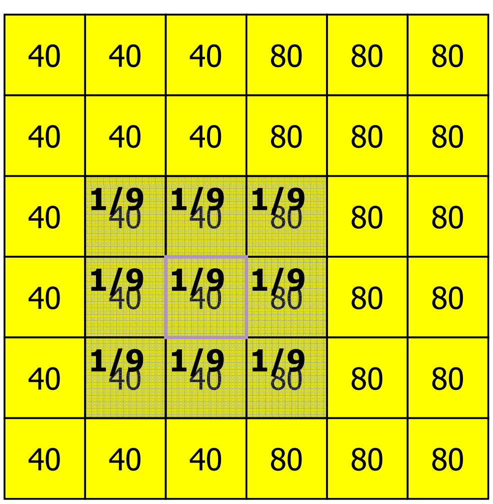
1	2	1
2	4	2
1	2	1

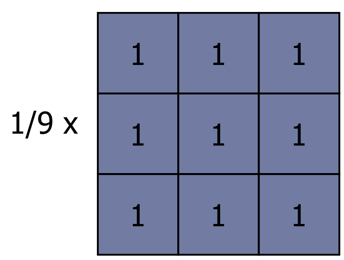
1
1
1
1
1

Applying Masks to Images

- $I_1 \otimes mask = I_2$
- Convention: I_2 is the same size as I_1
- Mask Application:
  - For each pixel
  - Place mask origin on top of pixel
  - Multiply each weight with pixel under it
  - Sum the result and put in location of the pixel

Applying Masks to Images



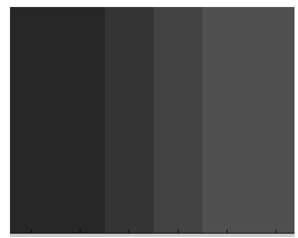


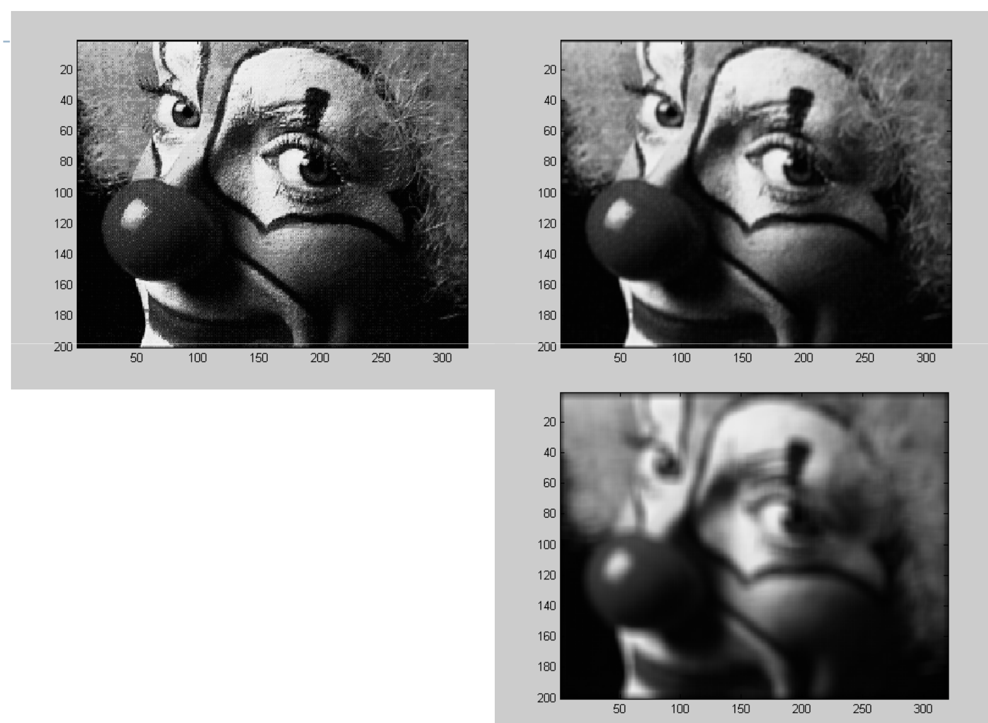
6*(1/9*40)+3*(1/9*80) = 53

Applying Masks to Images

40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80
40	40	53	67	80	80

- Overall effect of this mask?
- Smoothing filter



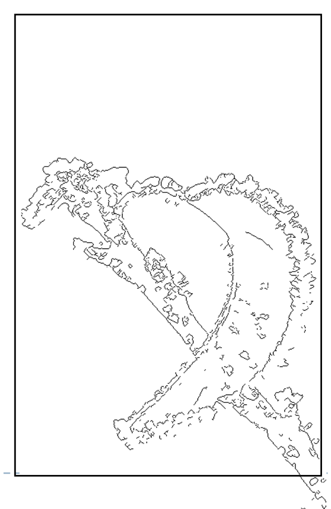


What about corner pixels

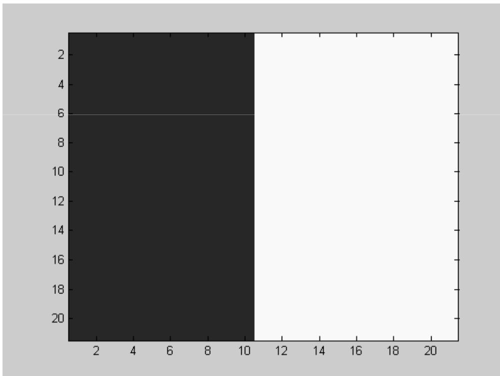
- Expand image with virtual pixels
- Options
  - Fill with a particular value, e.g. zeros
  - Fill with nearest pixel value
- Or just ignore them

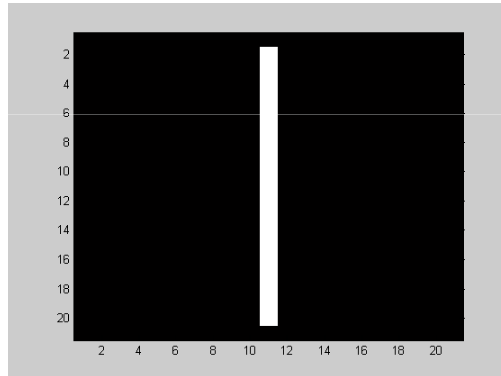
Edge Detection in Images

- Finding the contour of objects in a scene



Edge Detection



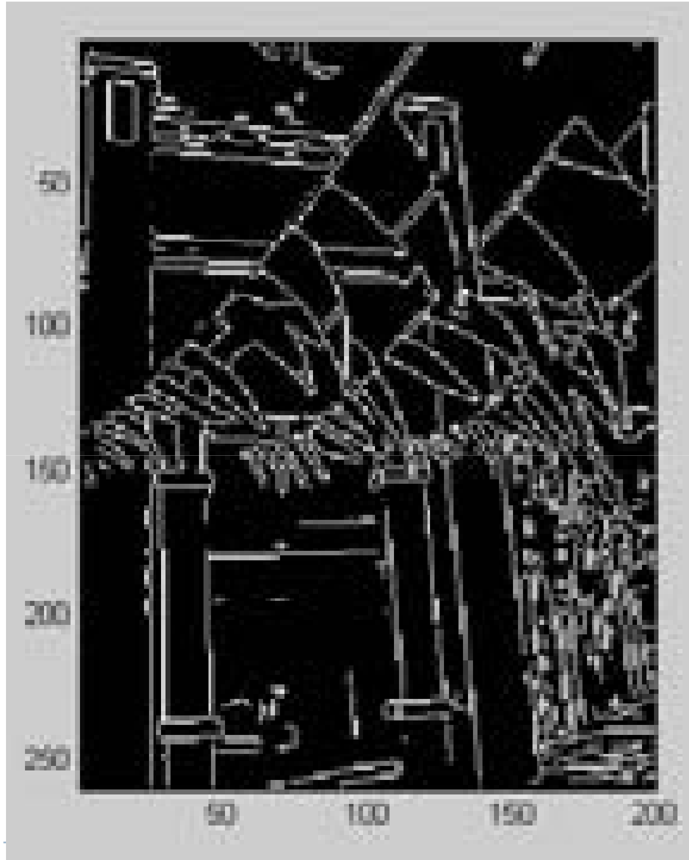


Choice of Mask?

- Should give 0 output on smooth regions
- Should give high output on non-smooth regions

-1	0	1
-1	0	1
-1	0	1

Prewitt mask



Why Edge Detection?

- Historical Background
- Discarding Useless Data?
- Simplification of Matching Problem
- Illumination Invariant Algorithms

Stages in Edge Detection

- Filtering
  - Removal of noise
- Differentiation
  - Assigns high values to regions of intensity change
- Detection
  - Indicates regions where intensity changes are significant

Filtering

1/9 x

1	1	1
1	1	1
1	1	1

- Mean Filter

$$
\begin{aligned}
h(x,y) &= f(x-1,y-1)\frac{1}{9} + f(x-1,y)\frac{1}{9} + f(x-1,y+1)\frac{1}{9} + f(x,y-1)\frac{1}{9} \\
&+ f(x,y)\frac{1}{9} + f(x,y+1)\frac{1}{9} + f(x+1,y-1)\frac{1}{9} + f(x+1,y)\frac{1}{9} + f(x+1,y+1)\frac{1}{9}
\end{aligned}
$$

$$
\begin{aligned}
h(x,y) &= f(x-1,y-1)g(-1,-1) + f(x-1,y)g(-1,0) + f(x-1,y+1)g(-1,1) \\
&+ f(x,y-1)g(0,-1) + f(x,y)g(0,0) + f(x,y+1)g(0,1) \\
&+ f(x+1,y-1)g(1,-1) + f(x+1,y)g(1,0) + f(x+1,y+1)g(1,1)
\end{aligned}
$$

$$
h(x,y) = \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} f(x+i,y+j)g(i,j)
$$

$$
h(x,y) = f(x,y) * g(x,y).
$$

Filtering

- Gaussian Filter
- Pixel weight is inversely proportional to distance from origin

$$g(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Value of σ has to be specified

Gaussian Filtering

- Example

$$g(x,y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- x = -1, y = -1, σ = 1

$$g(-1,-1) = e^{-\frac{(-1)^2+(-1)^2}{2}} \approx 0.367$$

0.367		

Gaussian Filtering

- Implementation problem
  - Float multiplications are slow
- Solution
  - Multiply mask with 255, round to nearest integer
  - Scale answer by sum of all weights

Gaussian Filtering

94	155	94
155	255	155
94	155	94

$\approx$

1	2	1
2	4	2
1	2	1

Round(g/70)

5	21	35	21	5
21	94	155	94	21
35	155	255	155	35
21	94	155	94	21
5	21	35	21	5

Gaussian Filtering

0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	3	11	26	50	73	82	73	50	26	11	3	0
1	6	20	50	93	136	154	136	93	50	20	6	1
2	9	30	73	136	198	225	198	136	73	30	9	2
2	11	34	82	154	225	255	225	154	82	34	11	2
2	9	30	73	136	198	225	198	136	73	30	9	2
1	6	20	50	93	136	154	136	93	50	20	6	1
0	3	11	26	50	73	82	73	50	26	11	3	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0

Figure 2.3: Gaussian mask with $\sigma = 2$.

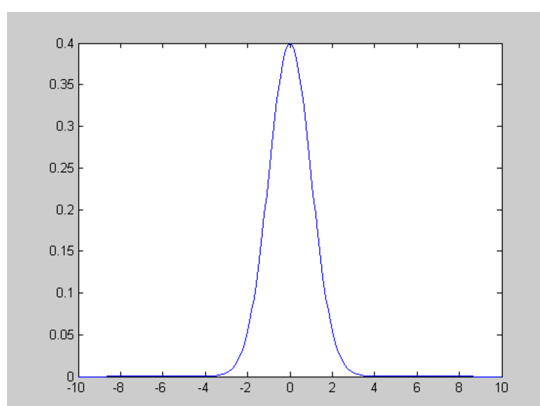
Continuous Gaussian Function

- 1-D Gaussian

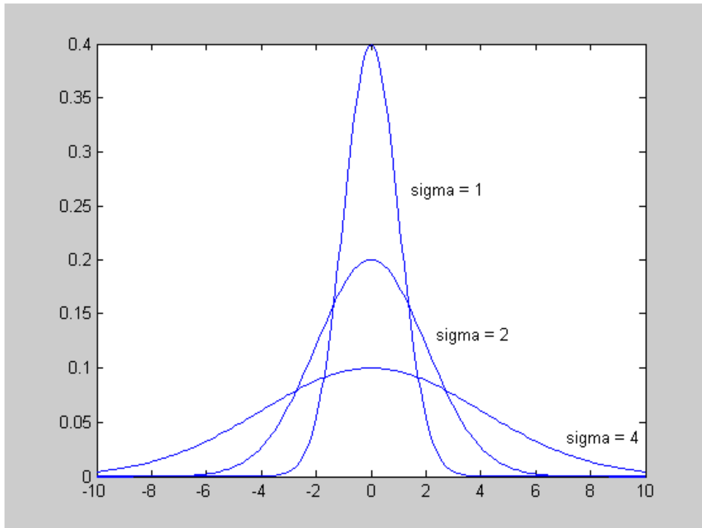
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

- Non zero from -∞ to ∞
- Width is controlled by σ
- Symmetric
- Area under curve = 1

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} = 1$$

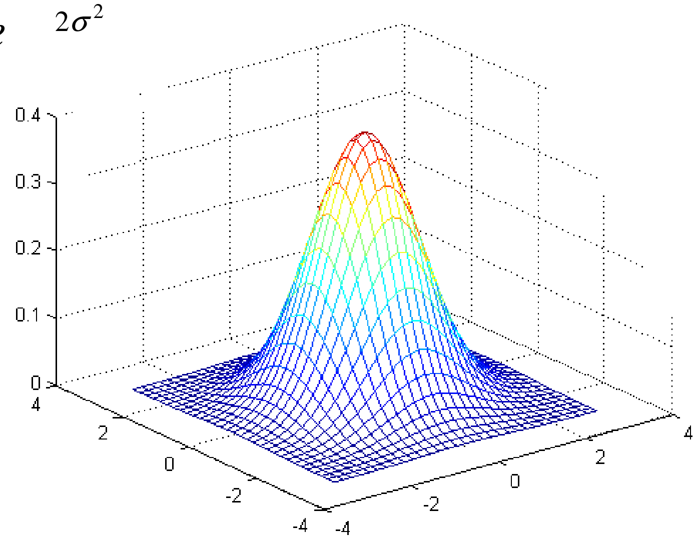


Continuous Gaussian Function

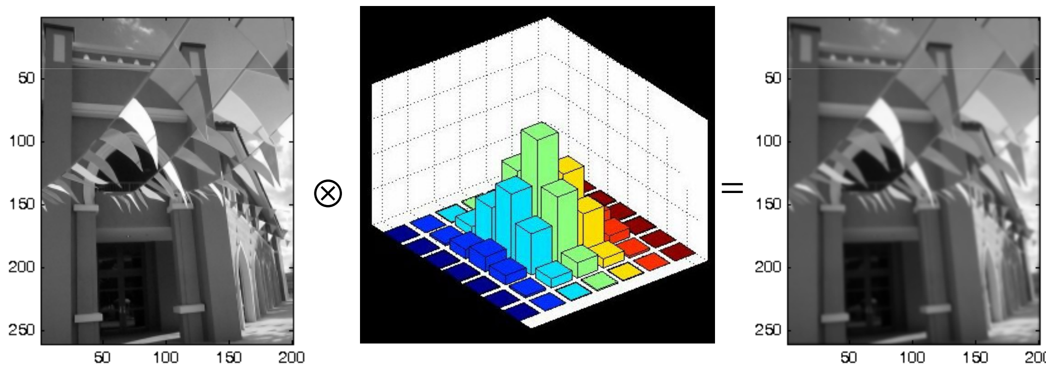


2-D Gaussian Function

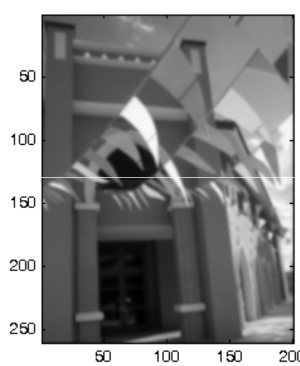
$$g(x, y) = \frac{1}{2\pi\sigma} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



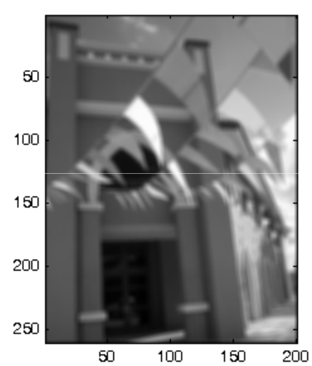
Gaussian Filter



Gaussian Vs Average



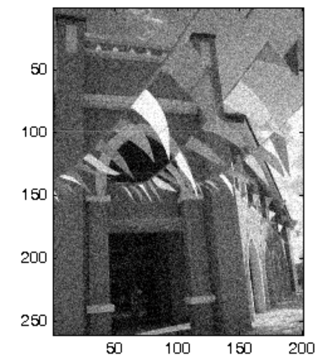
Gaussian Smoothing



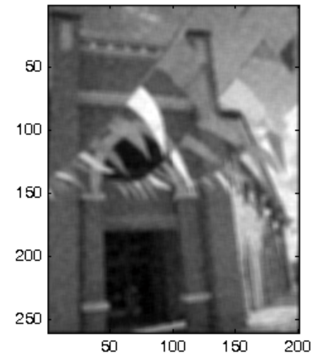
Smoothing by Averaging

Is this a realistic comparison? What issues need to be looked at?

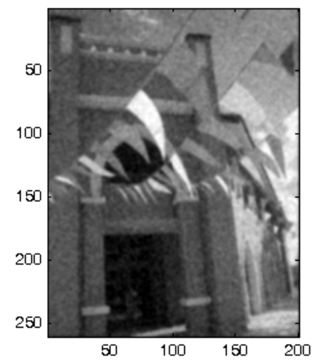
Noise Filtering



Gaussian Noise

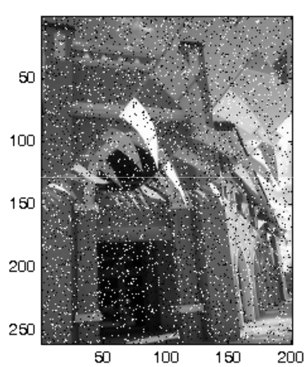


After Averaging

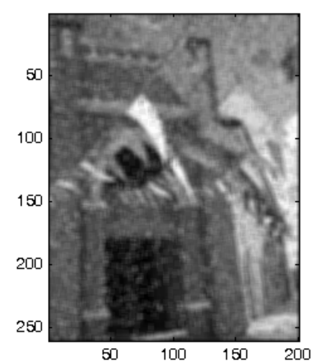


After Gaussian Smoothing

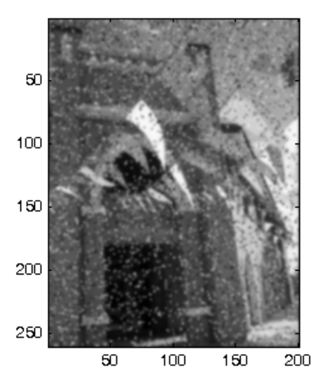
Noise Filtering



Salt & Pepper Noise



After Averaging



After Gaussian Smoothing

Stage 2: Differentiation

- Continuous form

$$f' = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

- Discrete form (put Δx = 1)

$$f' = \frac{df}{dx} = f(x) - f(x-1)$$

Discrete Derivatives

$f(x)$	40	40	40	40	40	80	80	80	80	80
$f'(x)$	0	0	0	0	0	40	0	0	0	0

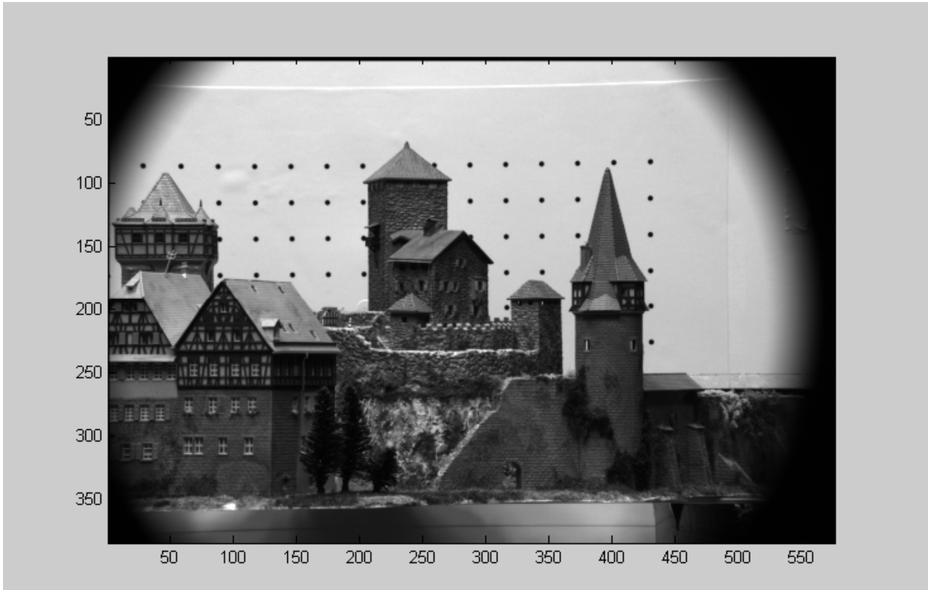
If $g(x)$

-1	1

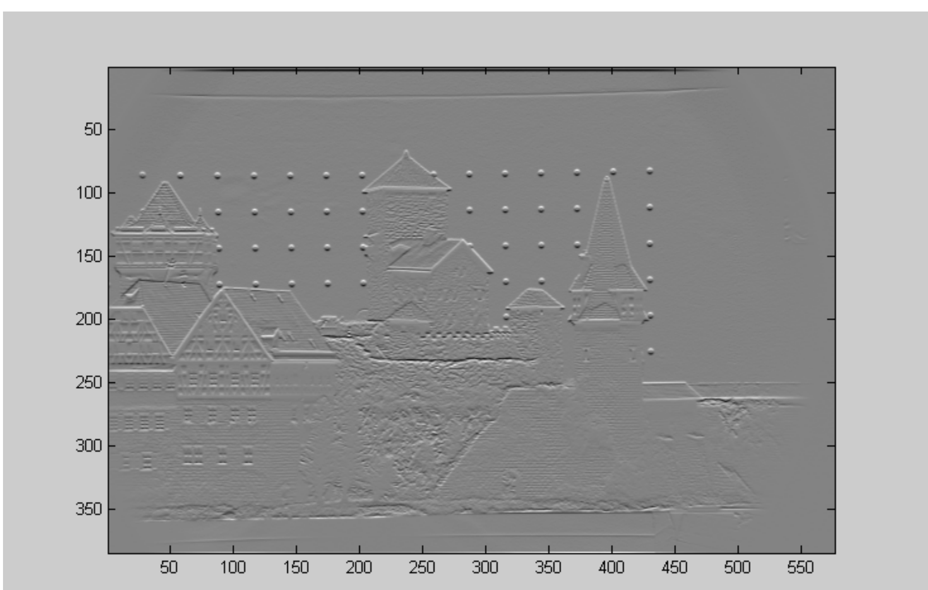
origin

then $f'(x) = f(x) \otimes g(x)$

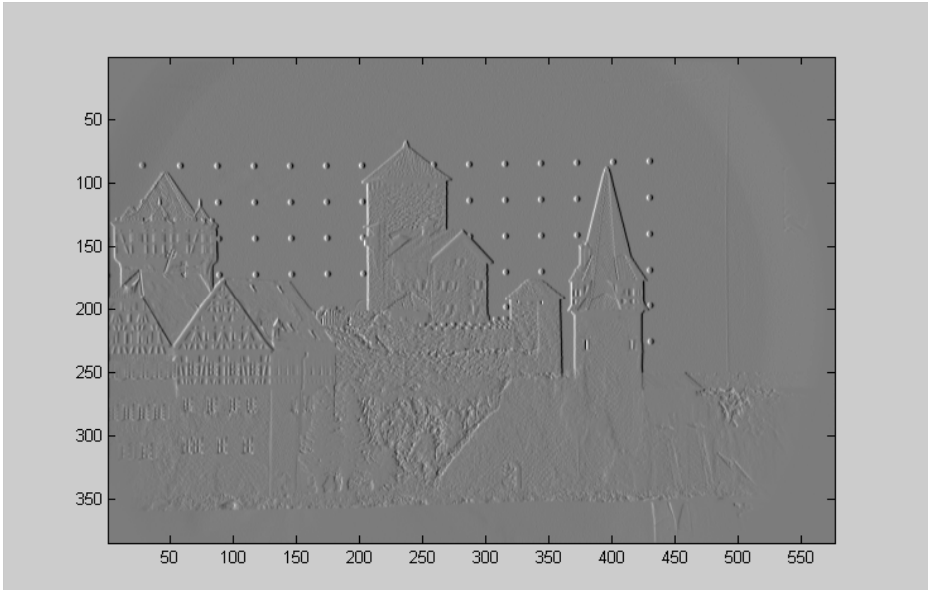
$f''(x)$	0	0	0	0	0	40	-40	0	0	0



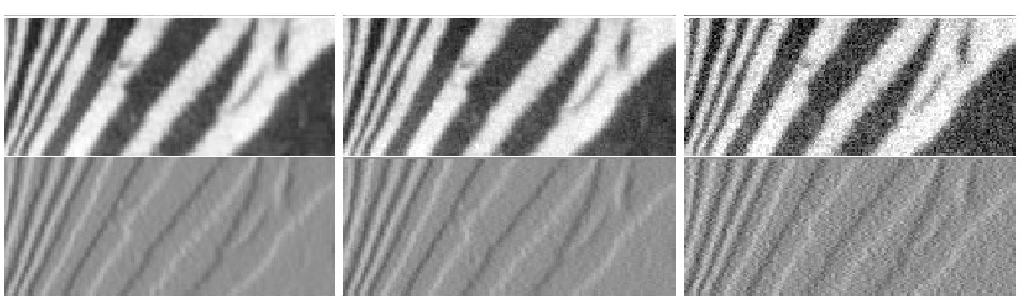
50
100
150
200
250
300
350
50 100 150 200 250 300 350 400 450 500 550



50
100
150
200
250
300
350
50 100 150 200 250 300 350 400 450 500 550



Effect of Noise (without filtering)



Recap: Filtering Masks

- Mean Filter
- Gaussian Filter

1/9 x

1	1	1
1	1	1
1	1	1

$$g(x,y) = ce^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

5	21	35	21	5
21	94	155	94	21
35	155	255	155	35
21	94	155	94	21
5	21	35	21	5

Recap: Derivative Masks

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

-1	1

-1
1

Properties of Masks

- Filtering Masks
  - All values are +ve
  - Sum to 1
  - Output on smooth region is unchanged
  - Blurs areas of high contrast
  - Larger mask -> more smoothing
- Derivative Masks
  - opposite signs
  - Sum to zero
  - Output on smooth region is zero
  - Gives high output in areas of high contrast
  - Larger mask -> more edges detected

Derivative Masks

- Prewit Operator

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

- Sobel Operator

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

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

- Robert's Operator

1	0
0	-1

0	1
-1	0

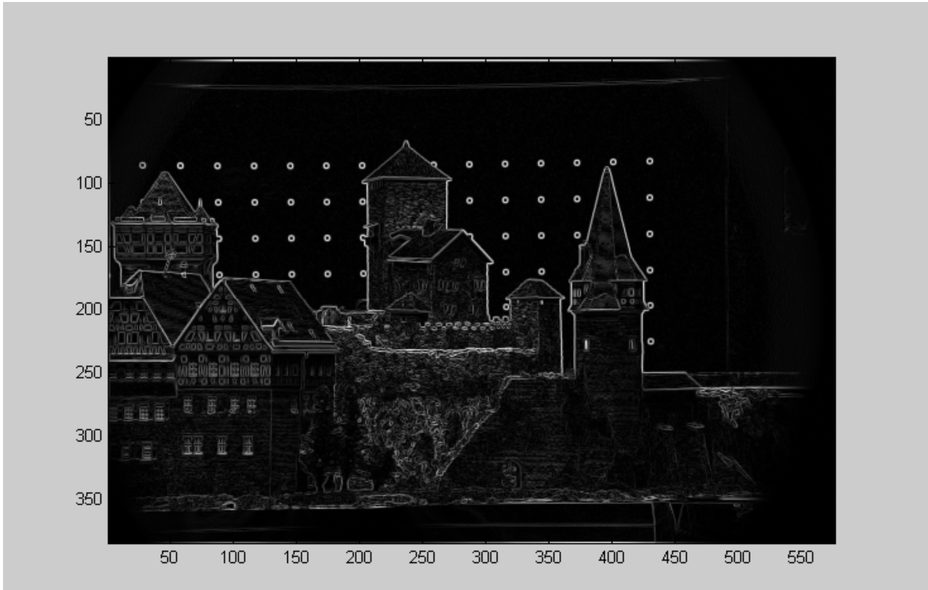
Application of 2-D Masks

- If f_x is derivative in x-direction, f_y is derivative in y-direction
- Gradient Magnitude

$$M = \sqrt{f_x^2 + f_y^2}$$

- Gradient Direction

$$\theta = \arctan \frac{f_y}{f_x}$$



Detection Stage - Threshold

- Gradient Magnitude

$$M(x,y) = \sqrt{f_x^2(x,y) + f_y^2(x,y)}$$

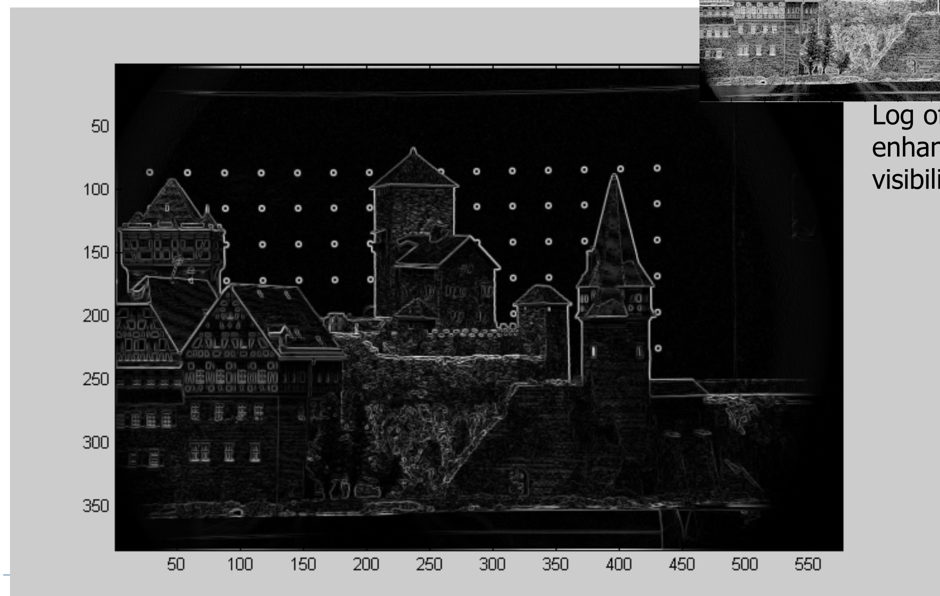
- Gradient Magnitude normalized b/w 0-100

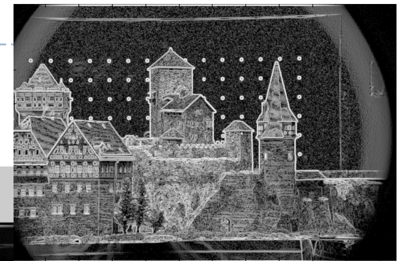
$$N(x,y) = \frac{M(x,y)}{max_{i=1,\ldots,n,j=1,\ldots,n} M(i,j)} \times 100.$$

- Application of a threshold

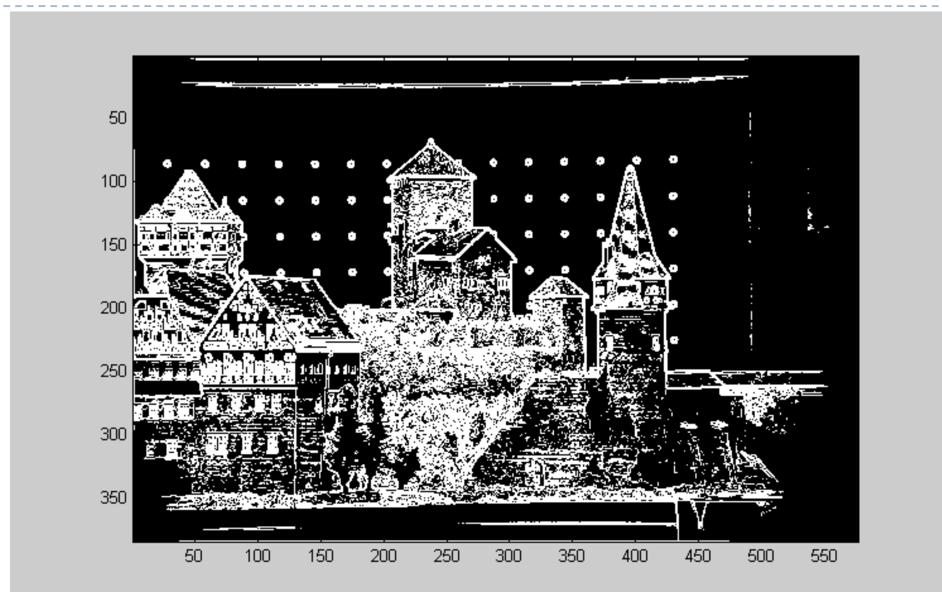
$$E(x,y) = \begin{cases} 1 & \text{if } N(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

Gradient Magnitude

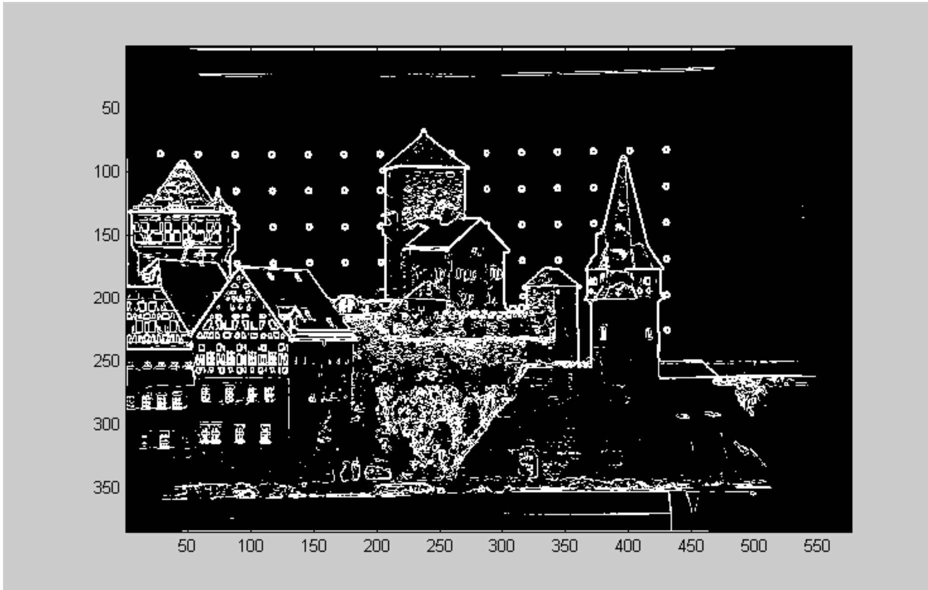




Log of M to enhance visibility

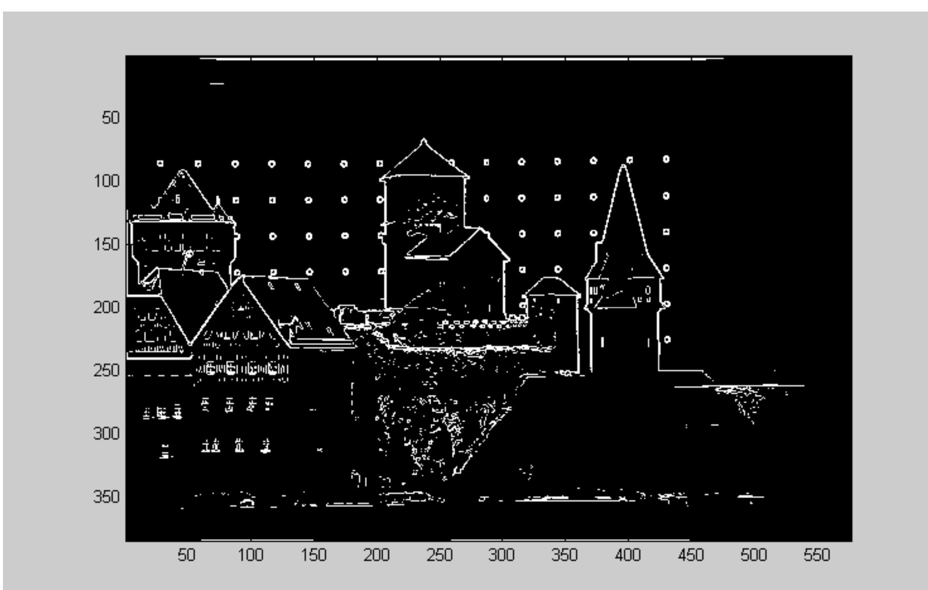


T = 10



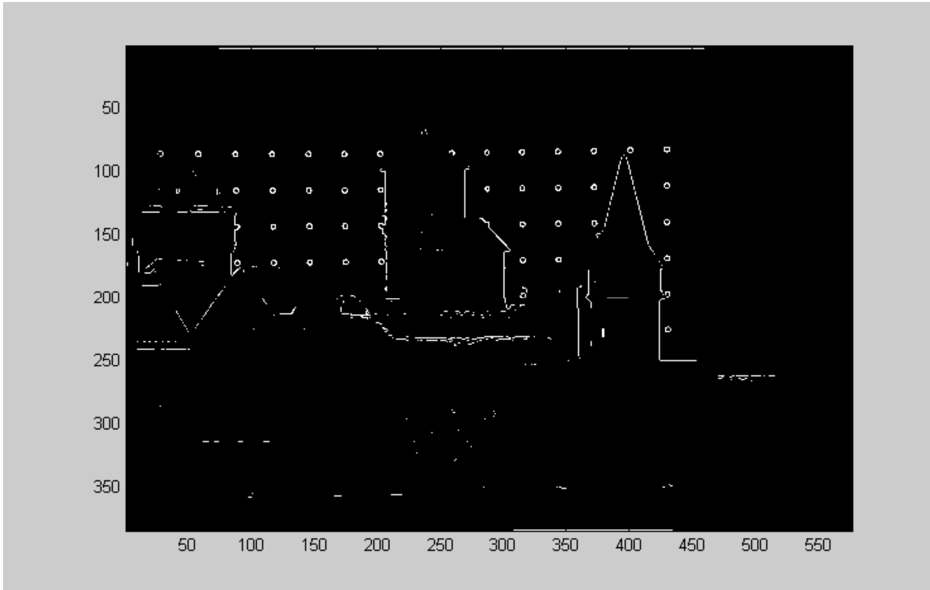
50
100
150
200
250
300
350
50 100 150 200 250 300 350 400 450 500 550

T = 20



50
100
150
200
250
300
350
50 100 150 200 250 300 350 400 450 500 550

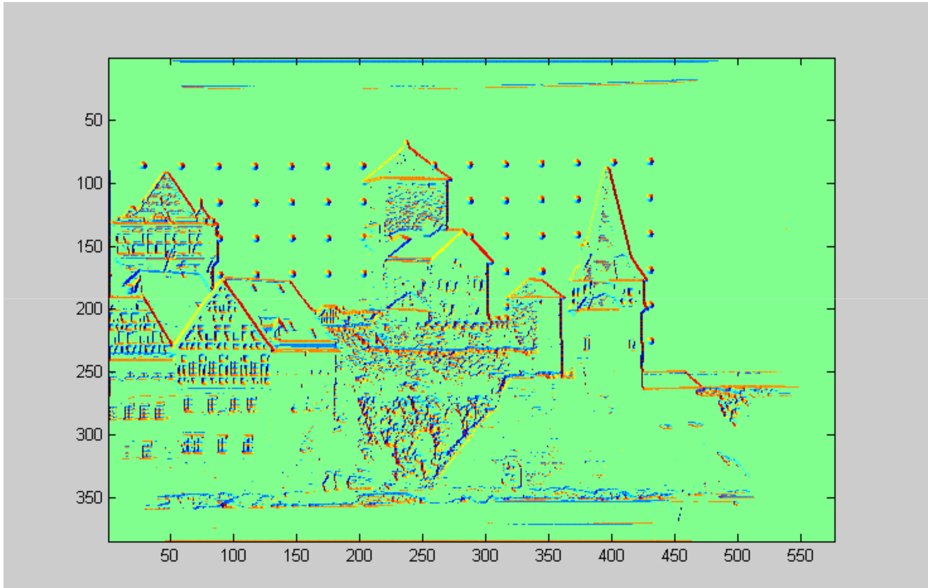
T = 40



T = 80

What about Gradient Direction?

- Gradient Direction is always perpendicular to edge
- Direction of most change of gray levels
- Thick edges can be eliminated using gradient direction
- Weak edges also captured in this manner

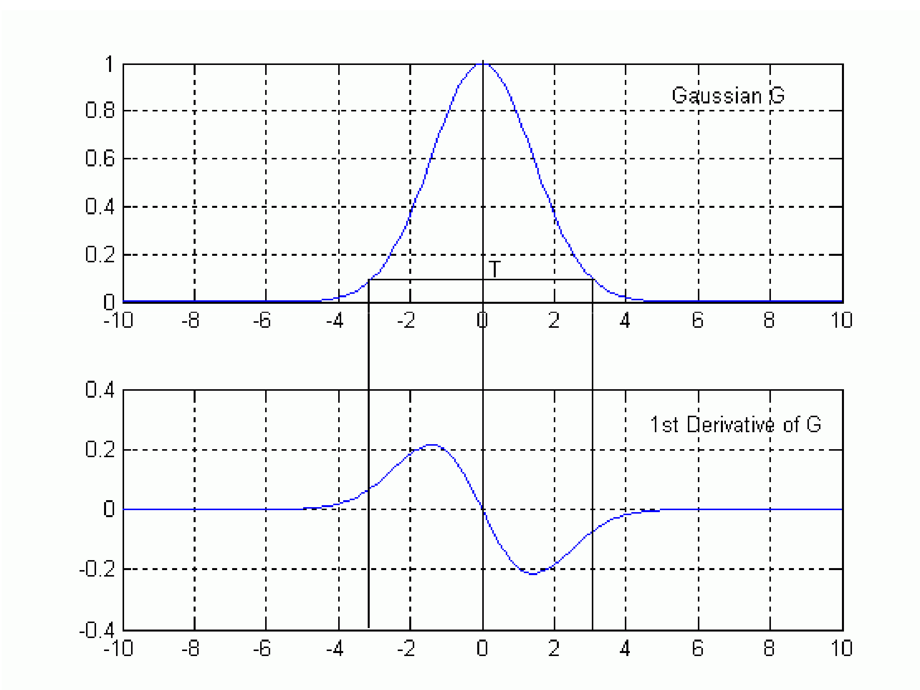


Canny’s Edge Detector

- Filtering + Derivative:
  - Uses first derivative Gaussian masks
- Detection:
  - Uses Non-Maxima Suppression
  - Uses Hysteresis Thresholding

First Derivative of Gaussian

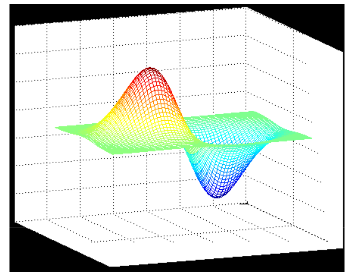
- Expression?
- Effect?
- Filtering + Derivative



Canny Edge Operator

$$\Delta S = \Delta(G_\sigma * I) = \Delta G_\sigma * I$$

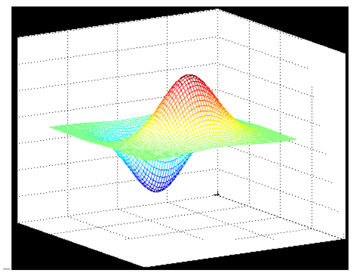
$$\Delta G_\sigma = \begin{bmatrix} \dfrac{\partial G_\sigma}{\partial x} & \dfrac{\partial G_\sigma}{\partial y} \end{bmatrix}^T$$



$$\Delta S = \begin{bmatrix} \dfrac{\partial G_\sigma}{\partial x} * I & \dfrac{\partial G_\sigma}{\partial y} * I \end{bmatrix}^T$$

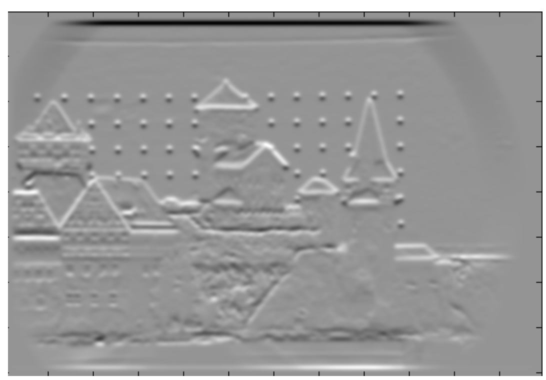
$$f_x(x, y) = f(x, y) * \left(\frac{-x}{\sigma^2}\right) e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

$$f_y(x, y) = f(x, y) * \left(\frac{-y}{\sigma^2}\right) e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

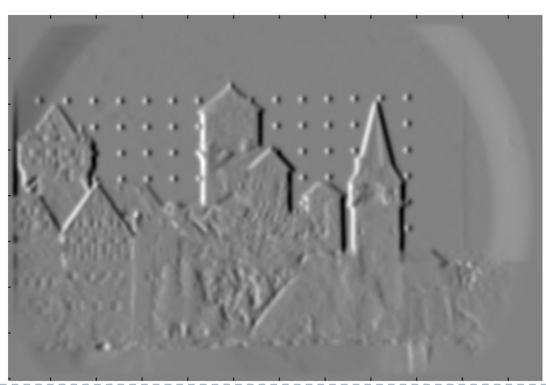


I

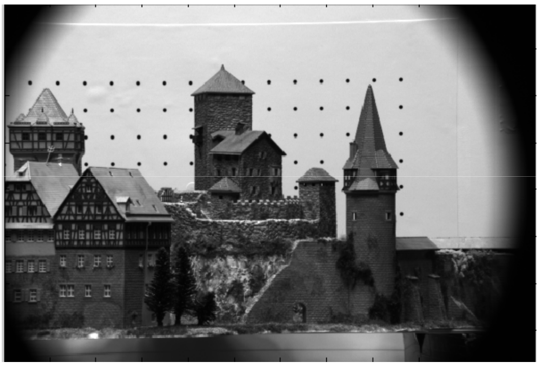
f_x

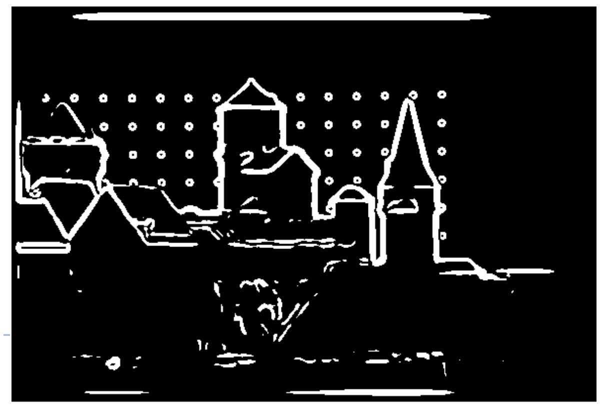


f_y



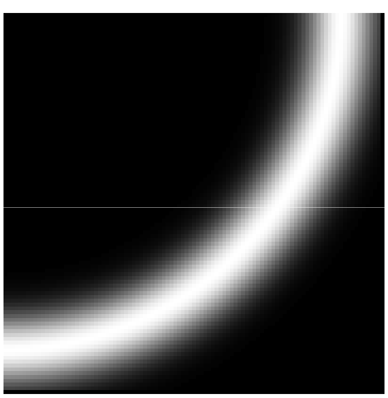
$$M = \sqrt{f_x^2 + f_y^2}$$

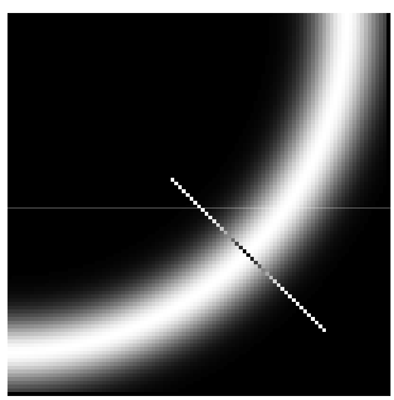




$$M \geq Threshold = 10$$

Non-Maximum Suppression





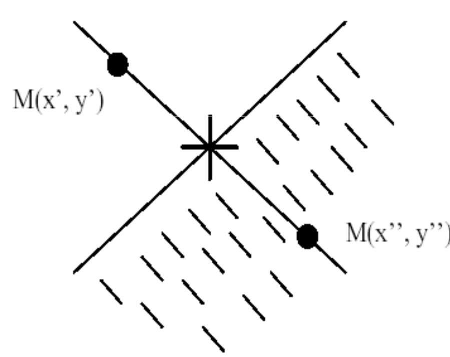
Remove all points along the gradient direction that are not maximum points

Non-Maxima Suppression

$$M(x,y) = \begin{cases} M(x,y) & \text{if } M(x,y) > M(x\prime, y\prime) \text{ and} \\ & \text{if } M(x,y) > M(x\prime\prime, y\prime\prime) \\ 0 & \text{otherwise} \end{cases}$$

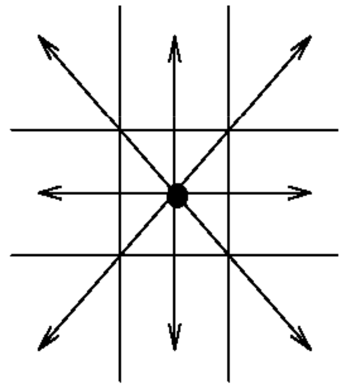
Where M(x′,y′) and M(x″,y″) are gradient magnitudes on both sides of edge at (x,y) in the gradient magnitude direction

Non-Maxima Supression

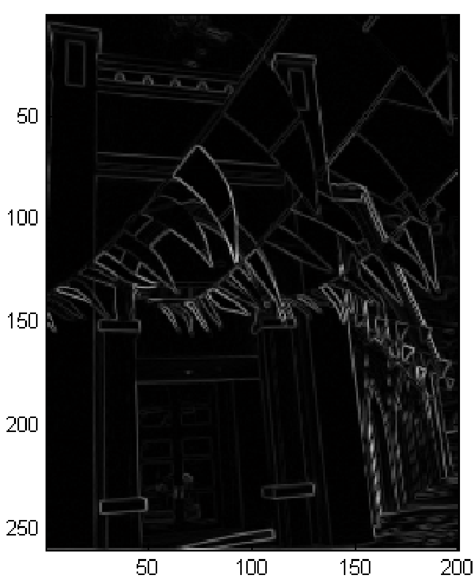


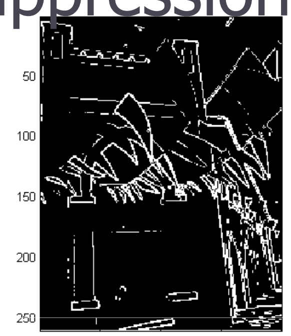
Quantization of Gradient Direction

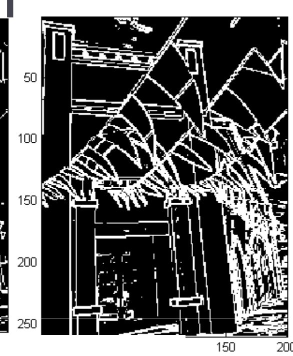
$$\theta = \arctan \frac{f_y}{f_x}$$

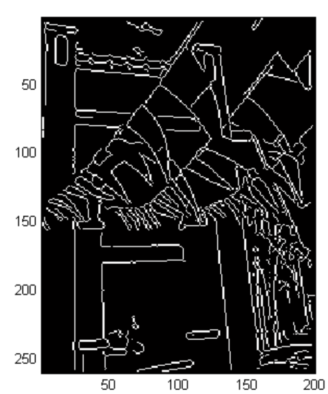


Non-Maximum Suppression





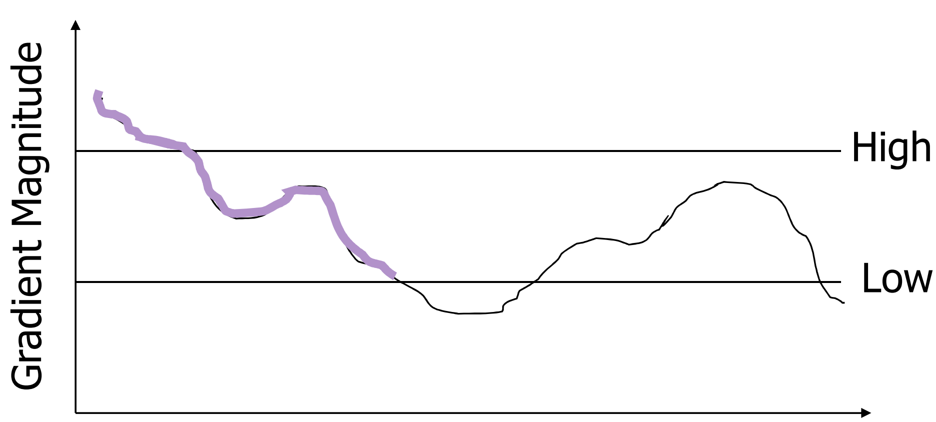




Hysteresis Thresholding

- Two thresholds, T_H and T_L
- Apply non-maxima suppression to M (gradient magnitude)
- Scan image from left to right, top to bottom
- If M(x,y) is above T_H mark it as edge
- Recursively look at neighbors; if gradient magnitude is above T_L mark it as edge

Hysteresis Thresholding



Algorithm Summary

- Compute gradient of image *f*(*x*,*y*) by convolving with first derivative of Gaussian in x and y directions

$$f_x(x,y) = f(x,y) * (\frac{-x}{\sigma^2})\, e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

$$f_y(x,y) = f(x,y) * (\frac{-y}{\sigma^2})\, e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

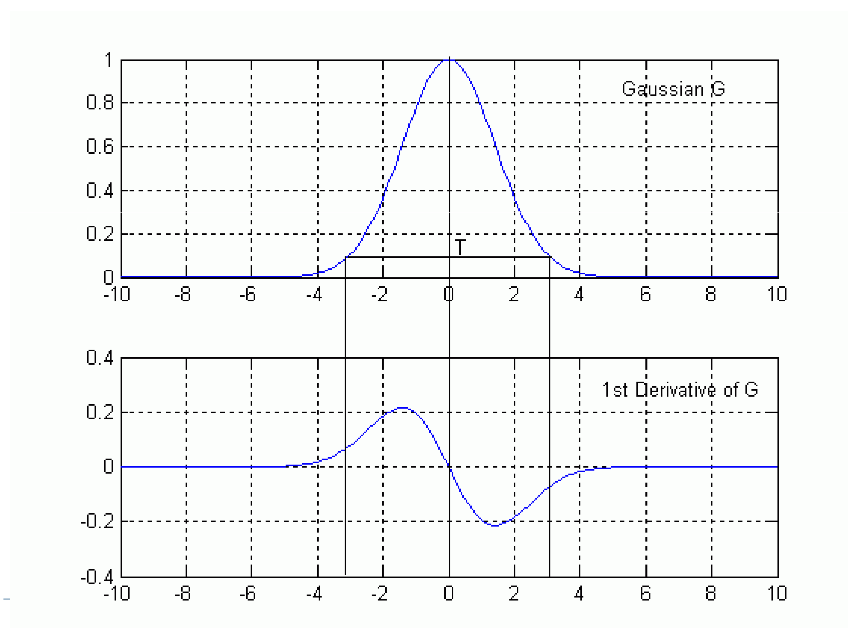
Algorithm Summary

- Compute gradient magnitude and direction at each pixel
- Perform non-maxima suppression
  - Find gradient direction at pixel
  - Quantize it in 8 directions {0, 45, 90, 135, … 315}
  - Compare current value of M with two neighbors in appropriate direction
  - If maximum, keep it, otherwise make it zero

Algorithm Summary

- Perform Hysteresis thresholding
  - Scan image from left to right, top to bottom
  - If $M(x,y)$ is above T_H mark it as edge
  - Recursively look at neighbors; if gradient magnitude is above T_L mark it as edge

Choice of Sigma



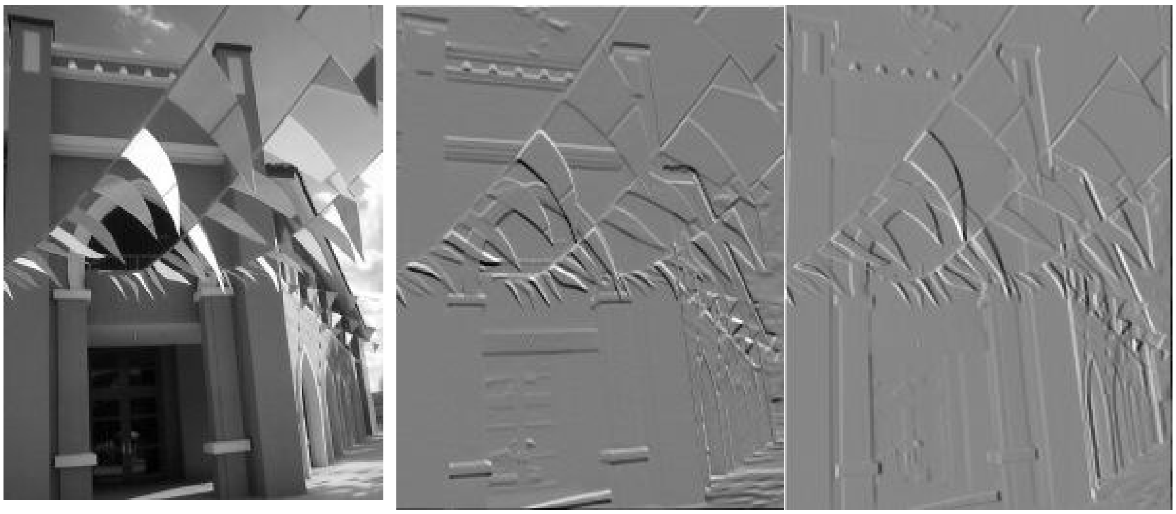
Choice of Sigma

- sHalf is computed by finding the point on the curve where the Gaussian value drops below T
- exp(-x^2/(2*sigma^2) = T
- sHalf = round(sqrt(-log(T) * 2 * sigma^2))
- mask size is then 2*sHalf + 1

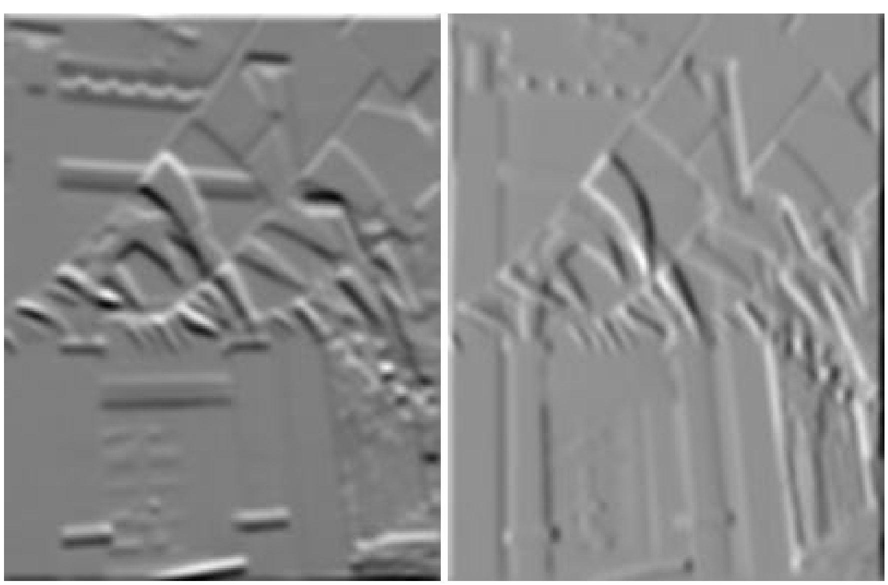
Choice of Signma

Sigma	Size of Mask
0.5	3x3
1	5x5
2	9x9
3	13x13
4	19x19

Conv. Results



sigma = 0.5



sigma = 2

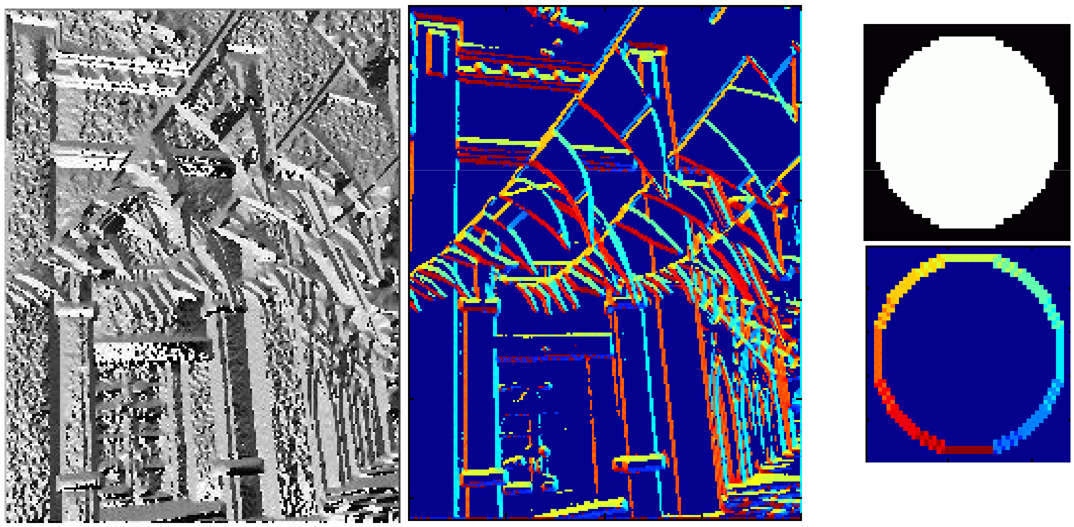
Gradient Magnitude Results

sigma = 0.5   sigma = 2

M when σ = 2

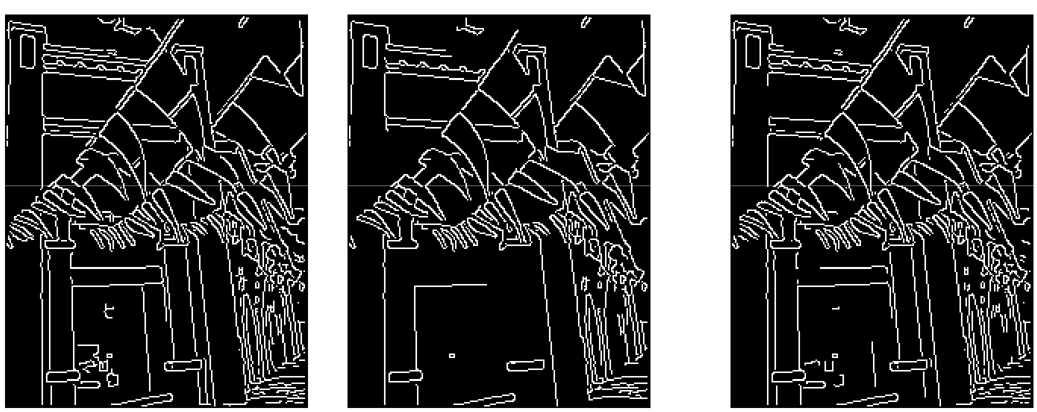
Non-maxima suppressed image

Gradient Direction Results



Raw

Quantized



Th = 50
Tl = 10

Th = 100
Tl = 10
(incr. in Th only)

Th = 50
Tl = 40
(incr. in Tl only)

Recursion Movie