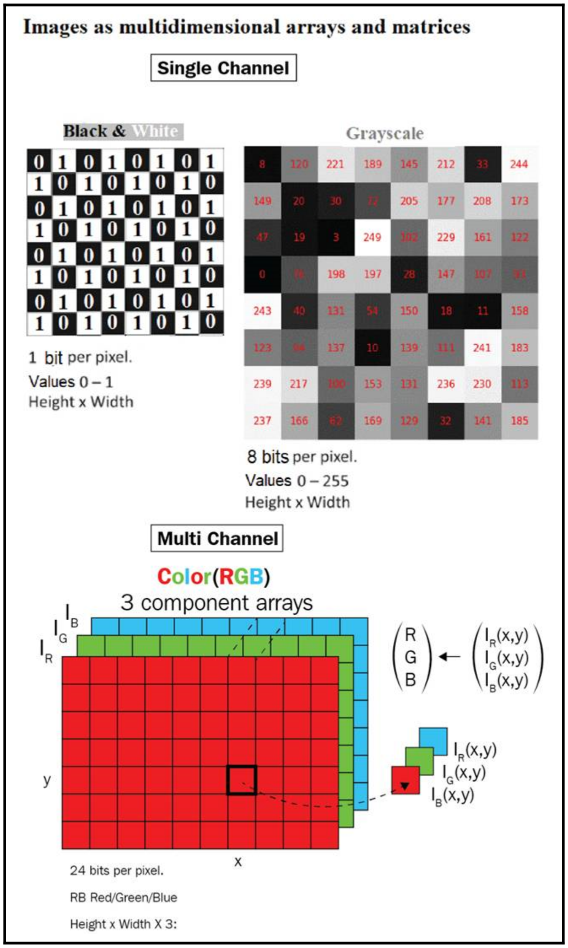
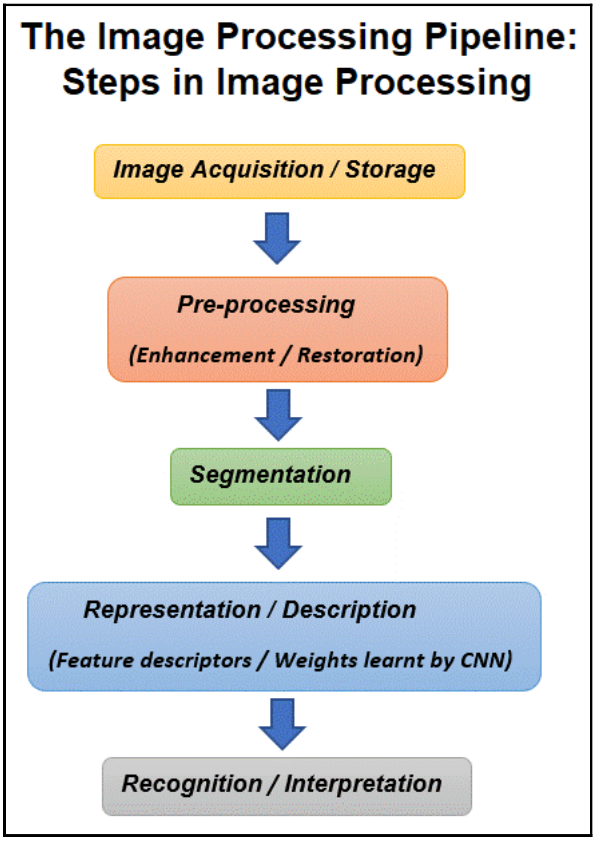
Graphics

Chapter 1: Getting Started with Image Processing




Binary
Gray Scale
RGB
Hands-on Image Processing with Python


The Image Processing Pipeline: Steps in Image Processing
Image Acquisition / Storage
Pre-processing
(Enhancement / Restoration)
Segmentation
Representation / Description
(Feature descriptors / Weights learnt by CNN)
Recognition / Interpretation

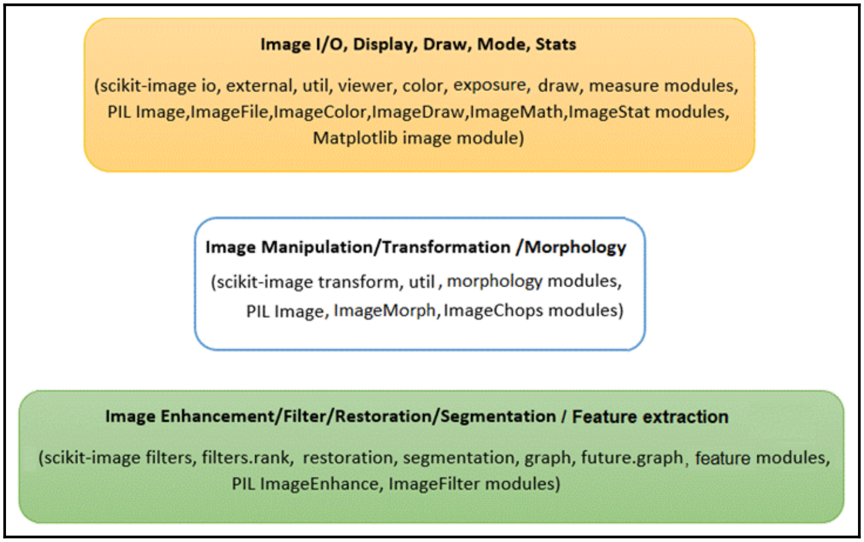
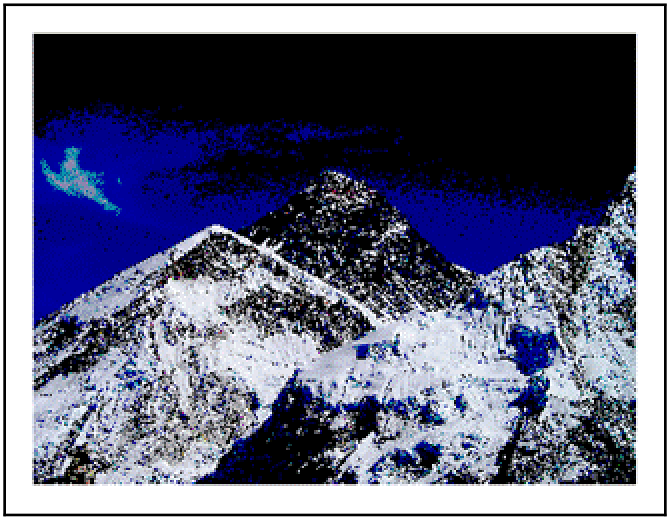
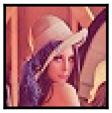
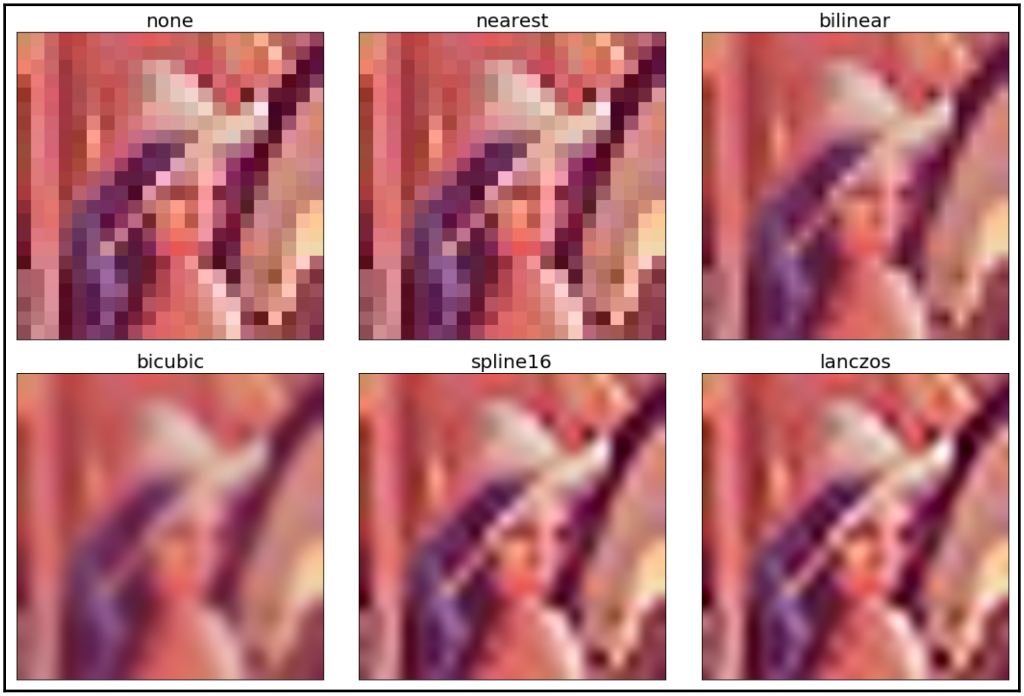
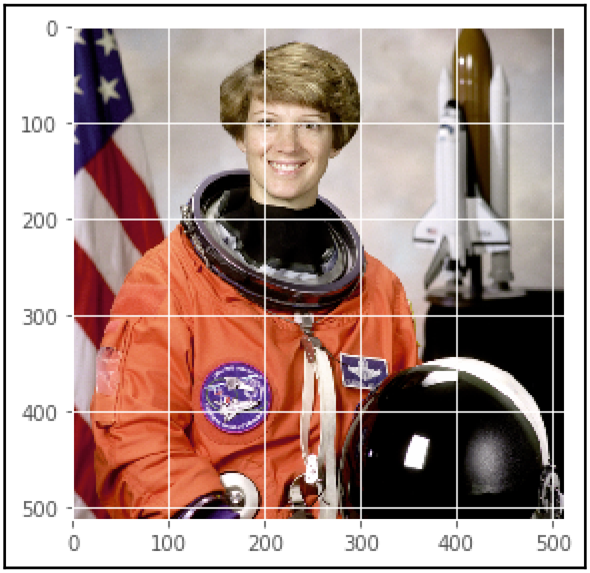


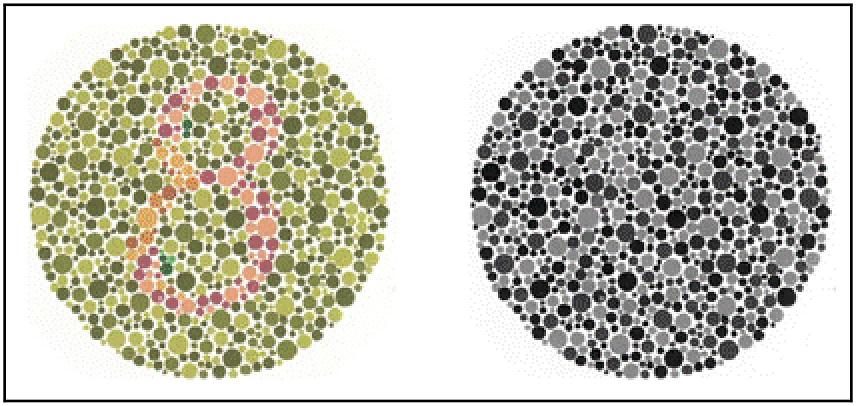
Image I/O, Display, Draw, Mode, Stats
(scikit-image io, external, util, viewer, color, exposure, draw, measure modules,
PIL Image,ImageFile,ImageColor,ImageDraw,ImageMath,ImageStat modules,
Matplotlib image module)
Image Manipulation/Transformation /Morphology
(scikit-image transform, util , morphology modules,
PIL Image, ImageMorph, ImageChops modules)
Image Enhancement/Filter/Restoration/Segmentation / Feature extraction
(scikit-image filters, filters.rank, restoration, segmentation, graph, future.graph, feature modules,
PIL ImageEnhance, ImageFilter modules)






none
nearest
bilinear
bicubic
spline16
lanczos


0
100
200
300
400
500
0
100
200
300
400
500

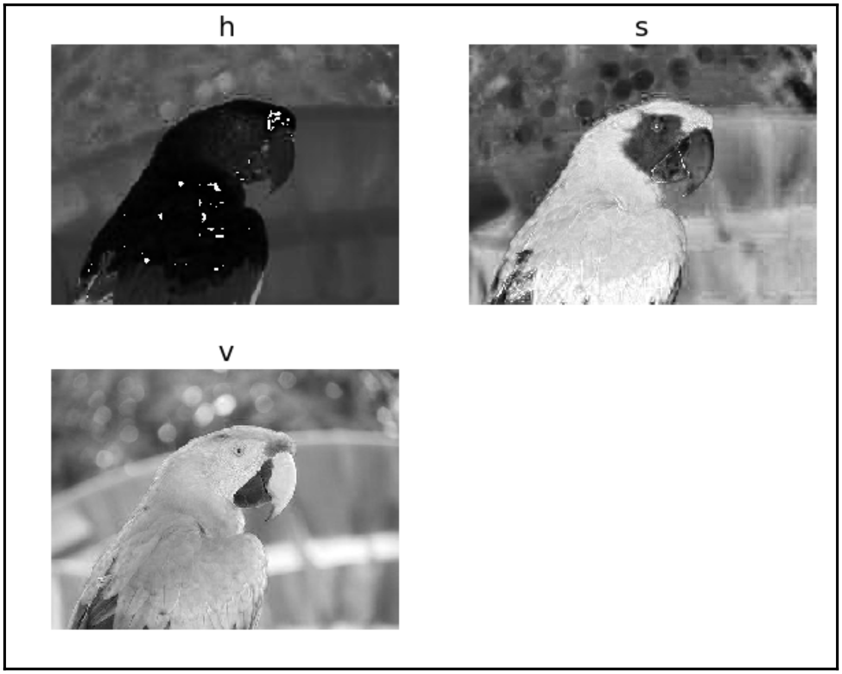


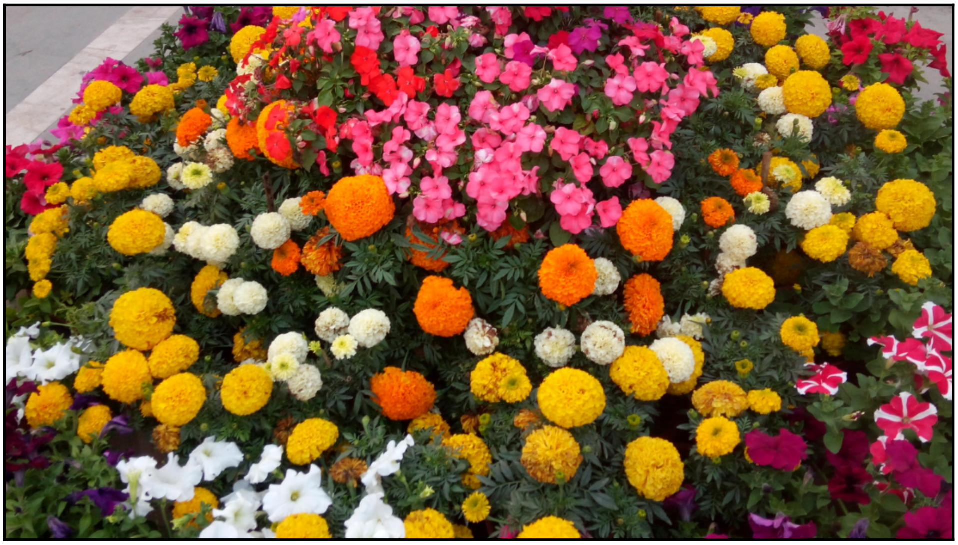
RGB to YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

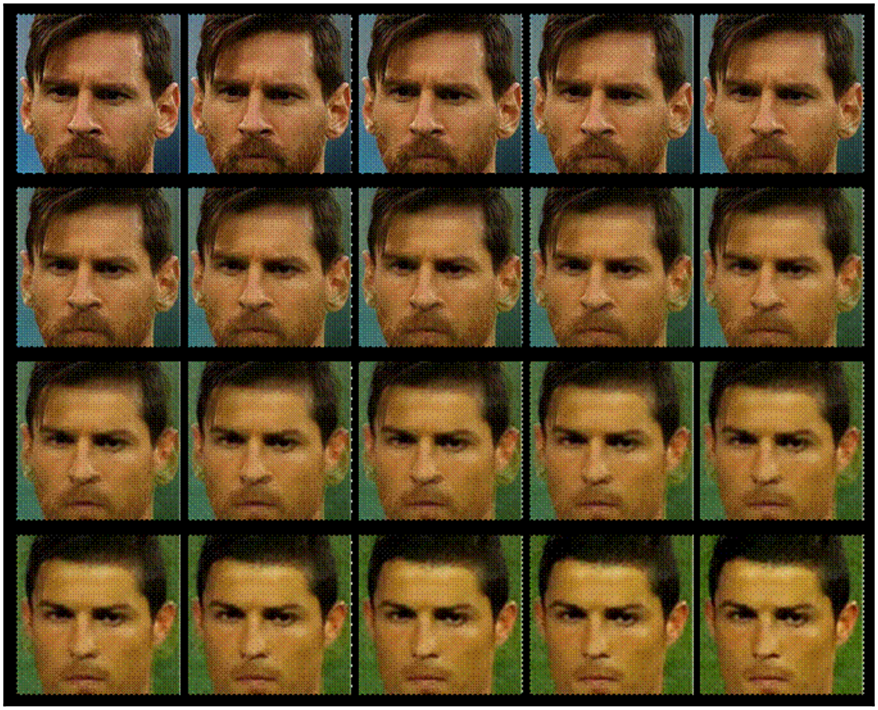
YIQ to RGB

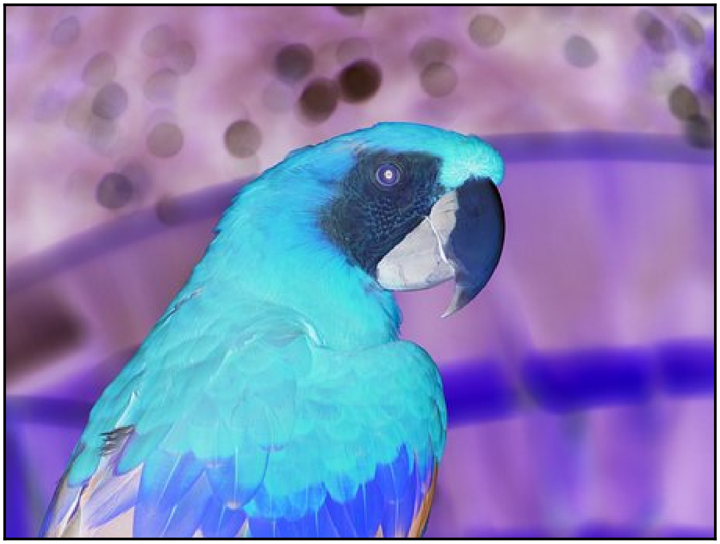
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

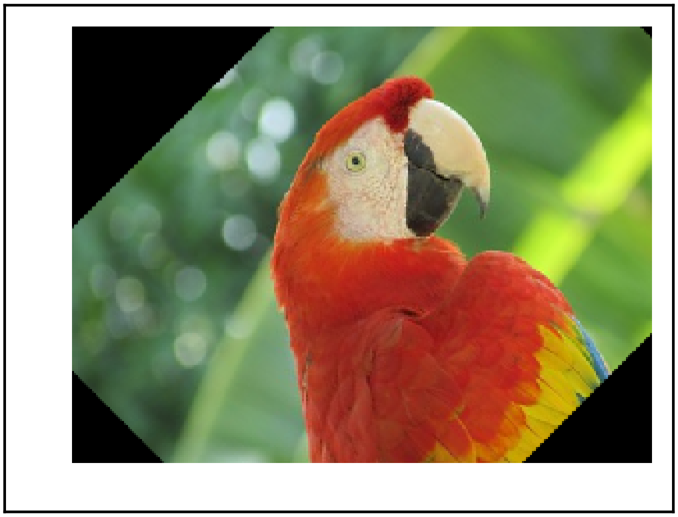

h
s
v

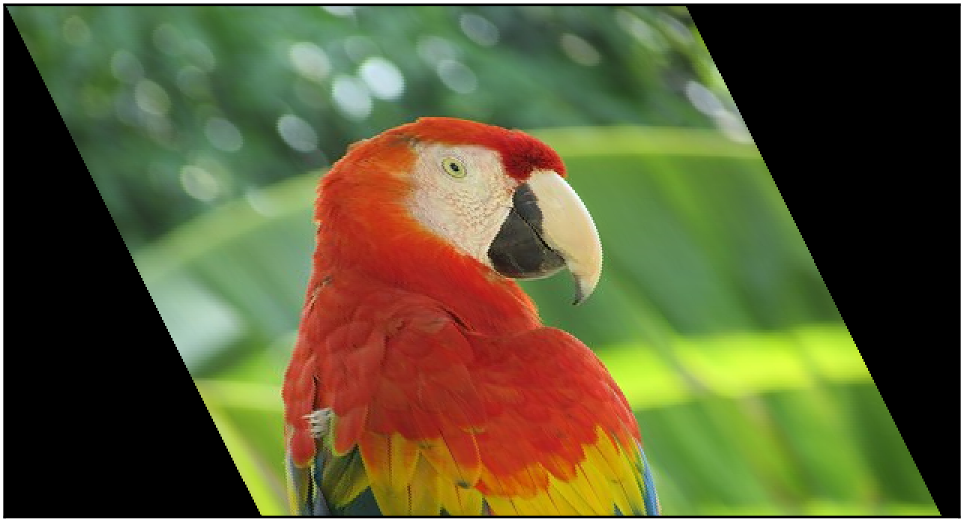


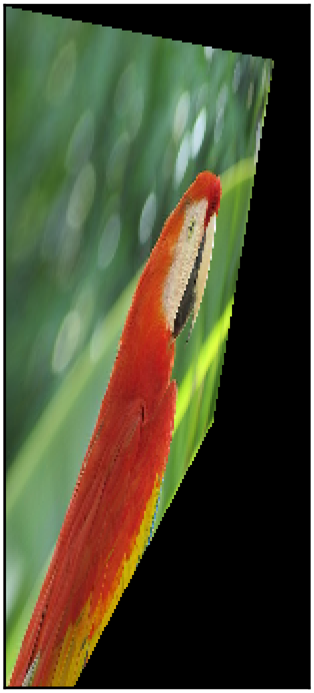

y
x

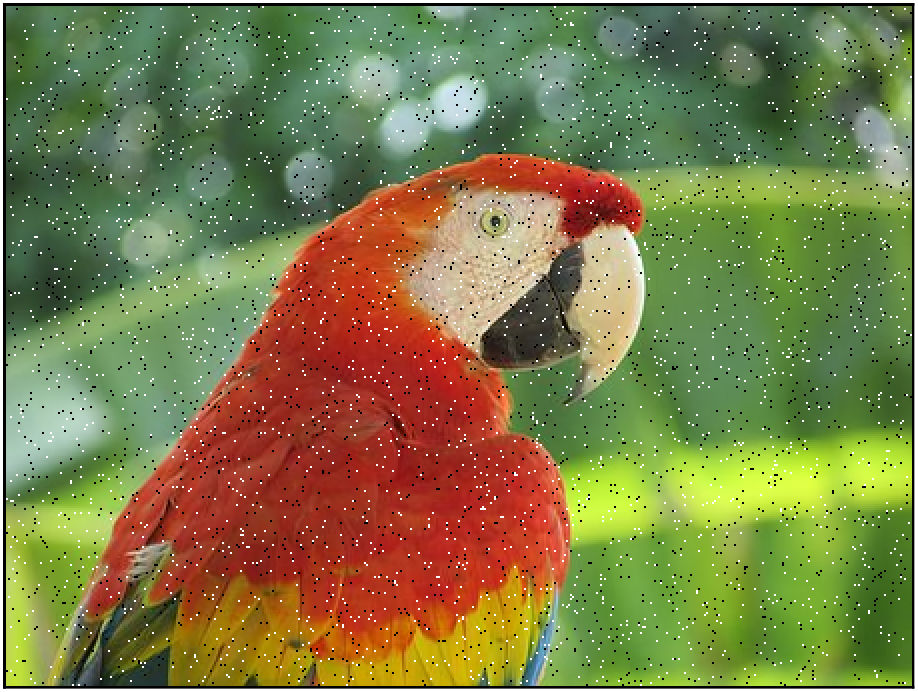


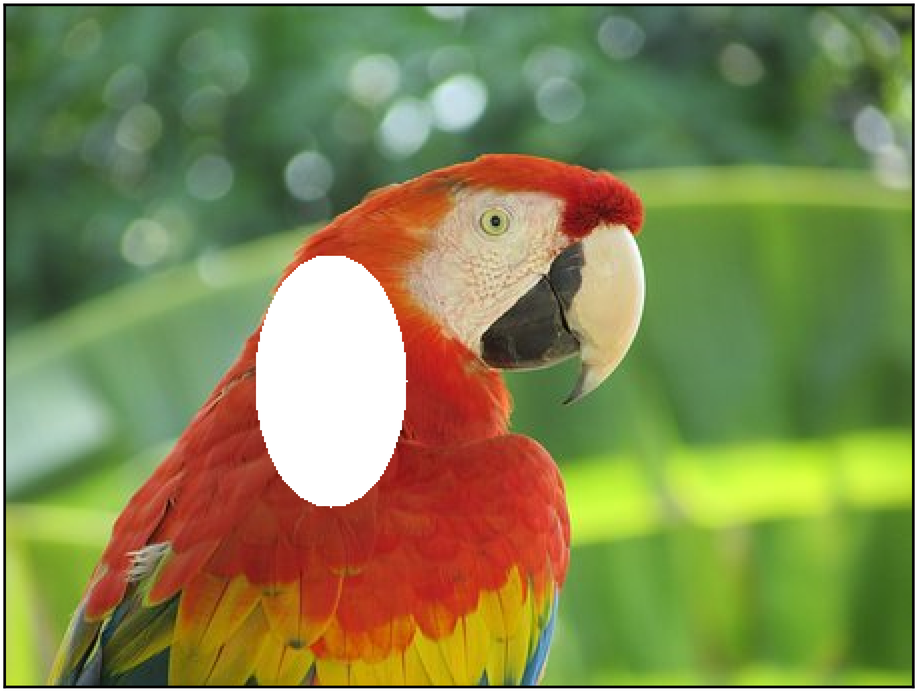




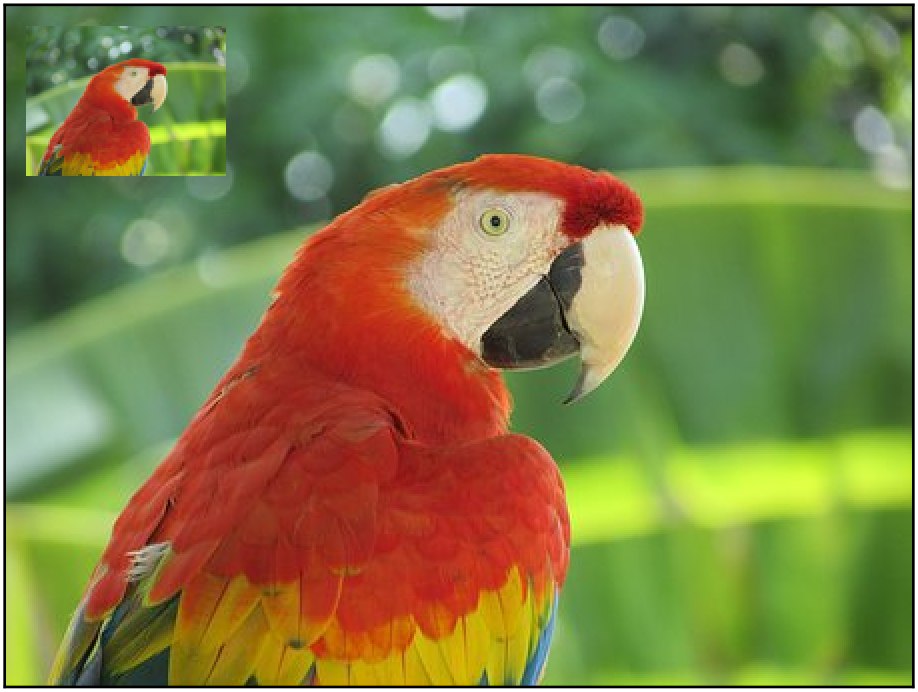


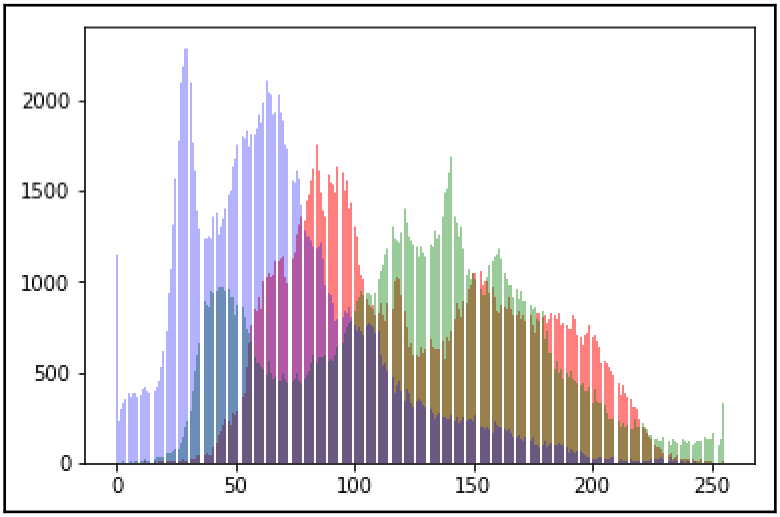


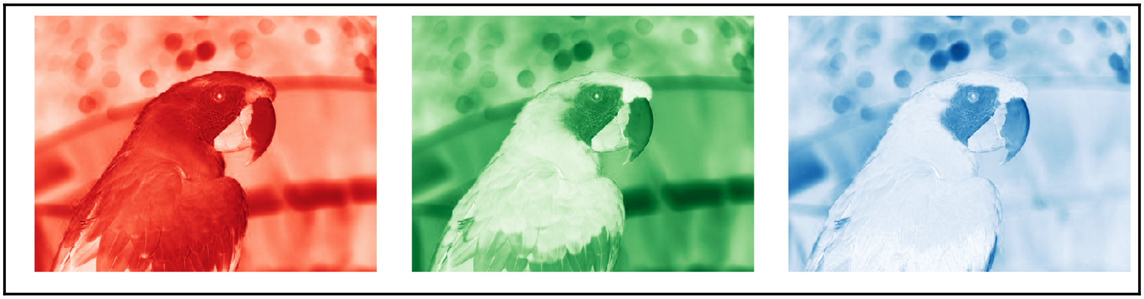





Welcome to image processing with python

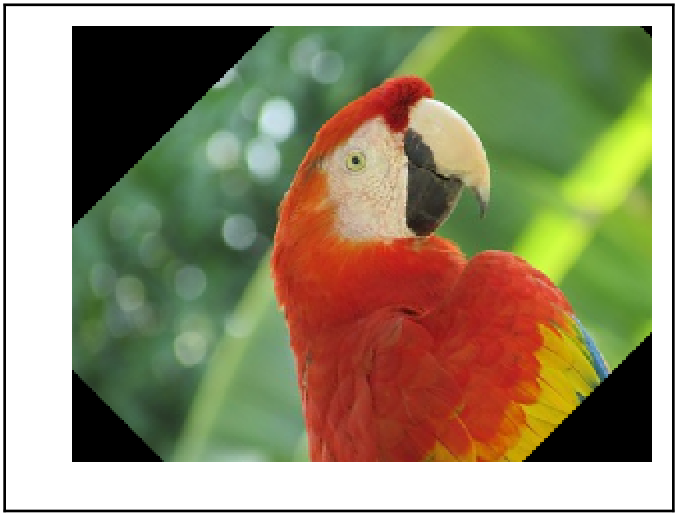


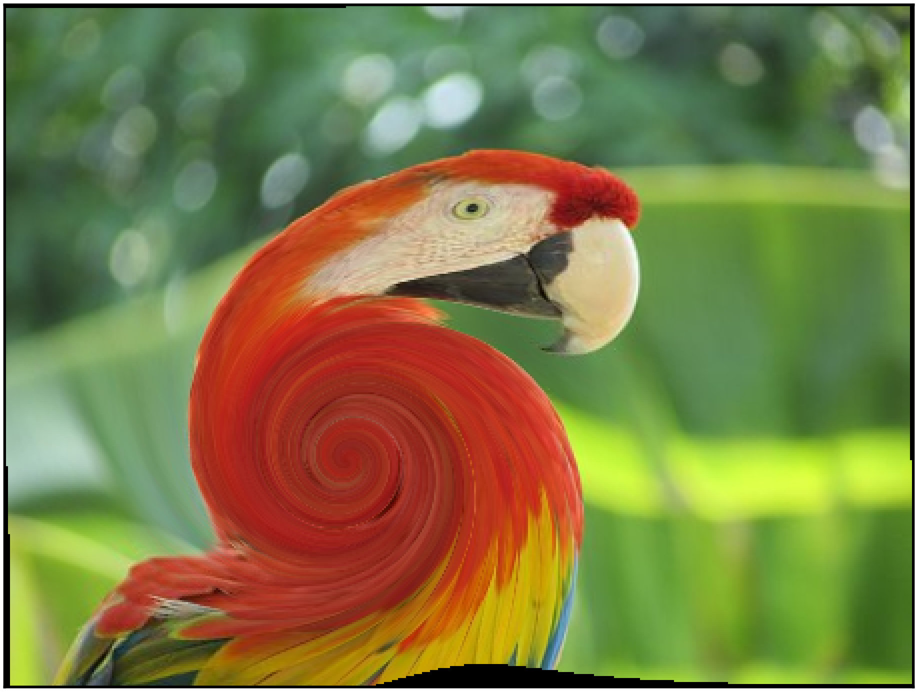

2000
1500
1000
500
0
0
50
100
150
200
250

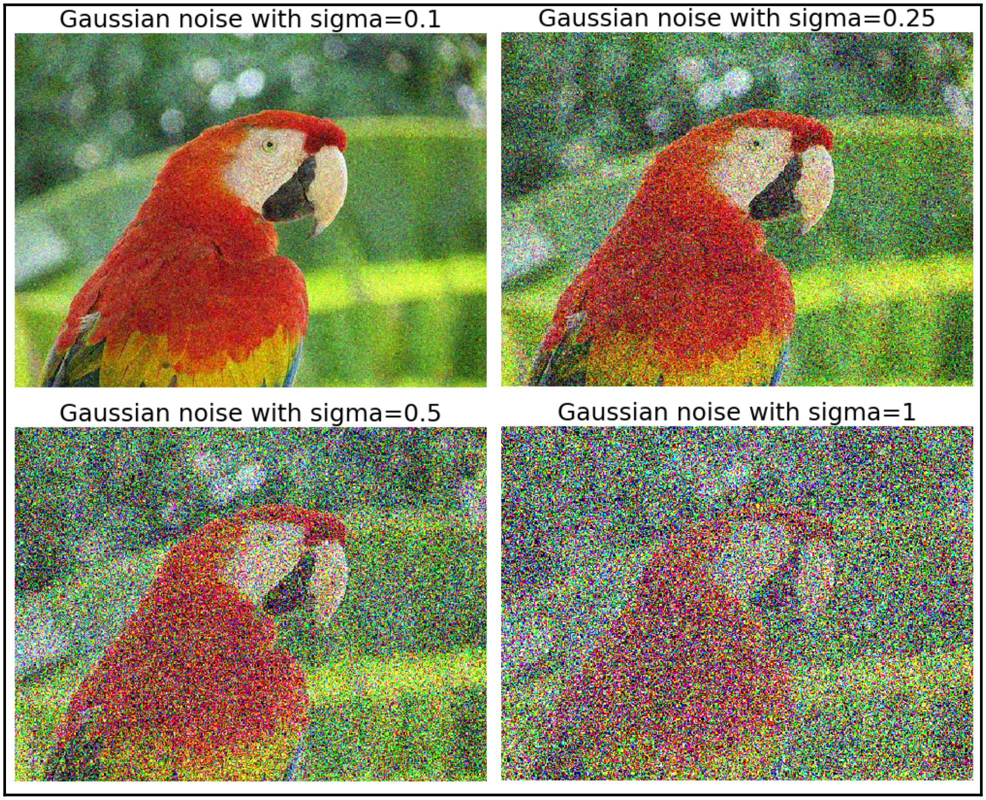


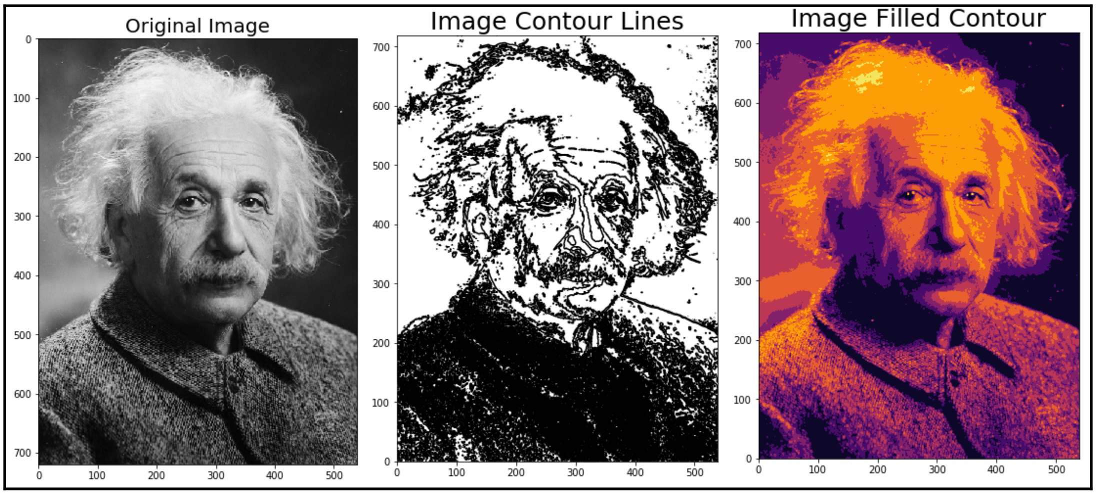

POR 1 0 ESP
FIFA TV
GAZPROM

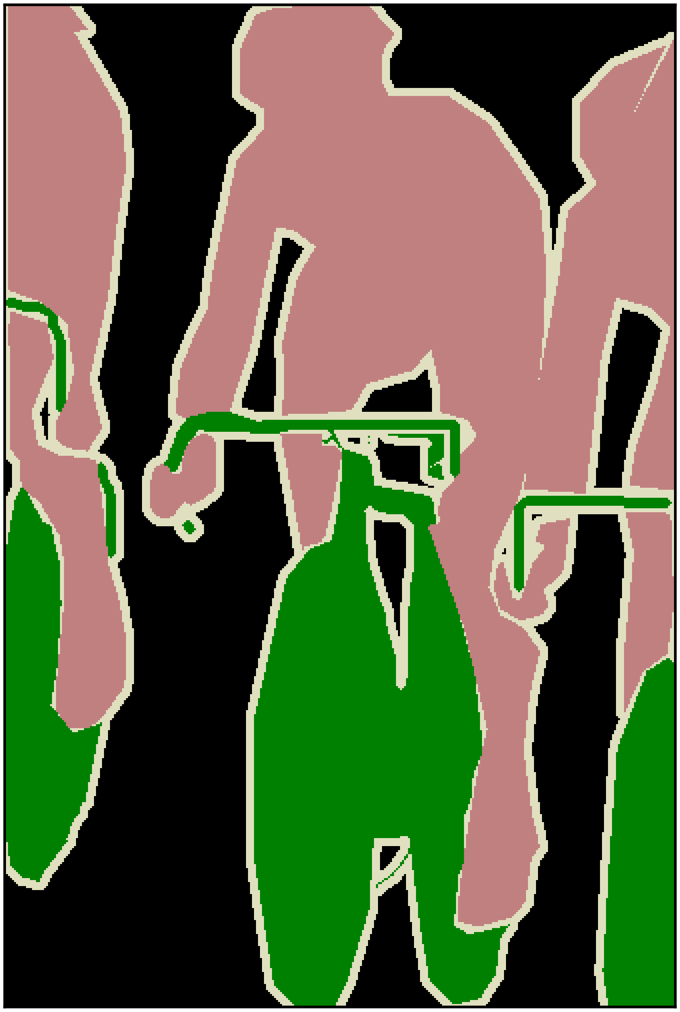

POR 1 0 ESP
FIFA TV
GAZPROM






Gaussian noise with sigma=0.1
Gaussian noise with sigma=0.25
Gaussian noise with sigma=0.5
Gaussian noise with sigma=1

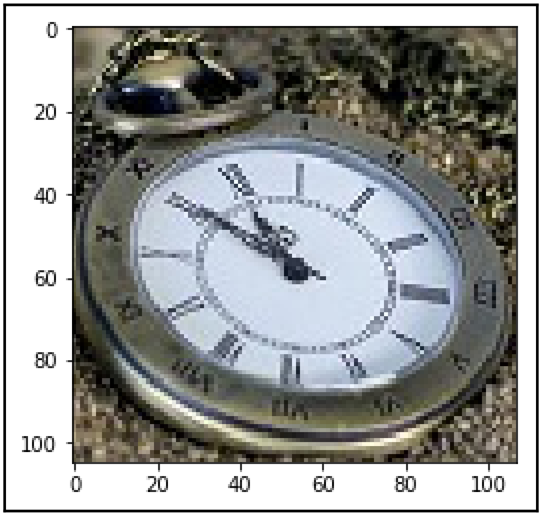

Original Image
Image Contour Lines
Image Filled Contour




shutterstock
www.shutterstock.com • 146430692


www.shutterstock.com • 146430692

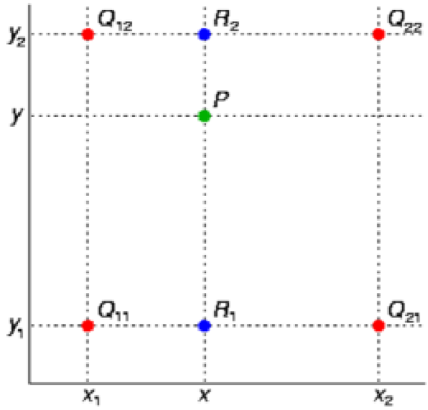
Chapter 2: Sampling, Fourier Transform, and Convolution




0
100
200
300
400
500
0
100
200
300
400
500

Bilinear interpolation

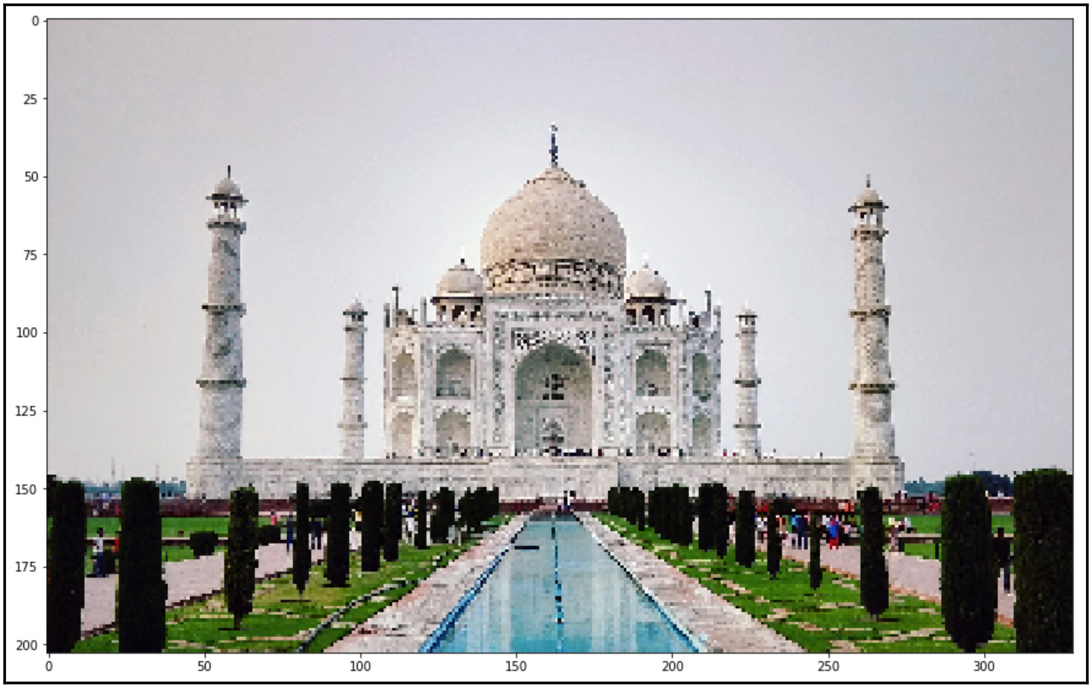
$$f(x,y) \approx \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1).$$



http://en.wikipedia.org/wiki/Bilinear_interpolation


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
25
50
75
100
125
150
175
200
0
50
100
150
200
250
300


0
25
50
75
100
125
150
175
200
0
50
100
150
200
250
300

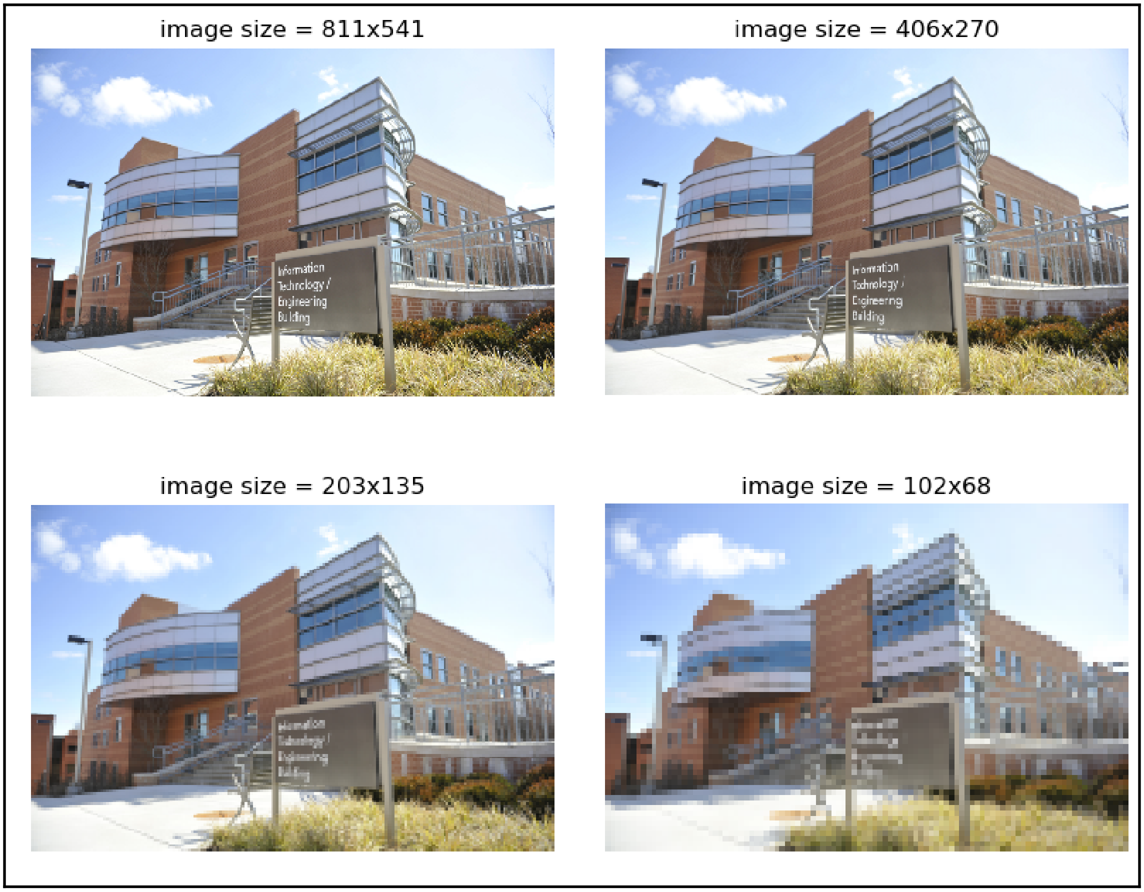


image size = 811x541
image size = 406x270
image size = 203x135
image size = 102x68
Information
Technology /
Engineering
Building

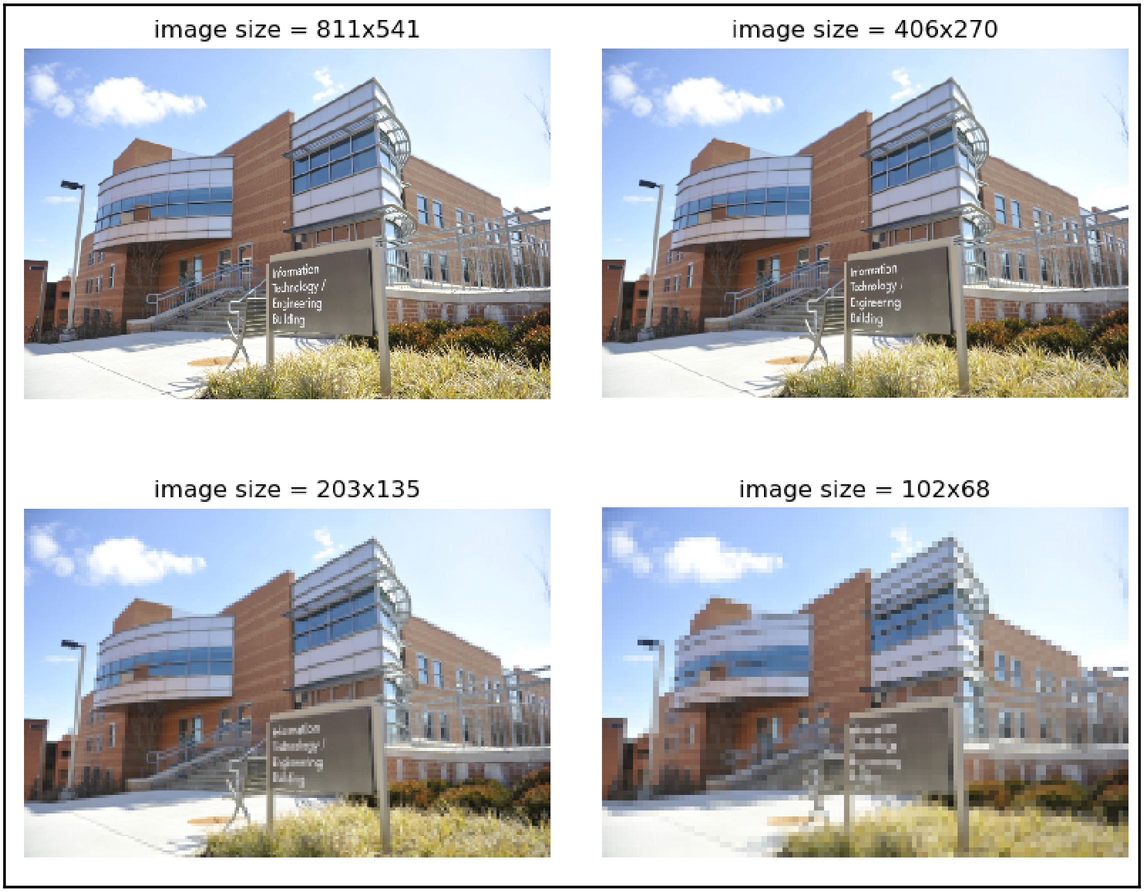


image size = 811x541
image size = 406x270
image size = 203x135
image size = 102x68
Information
Technology /
Engineering
Building


Image with max # colors = 256 SNR = 1.697
Image with max # colors = 128 SNR = 1.684
Image with max # colors = 64 SNR = 1.726
Image with max # colors = 32 SNR = 1.689

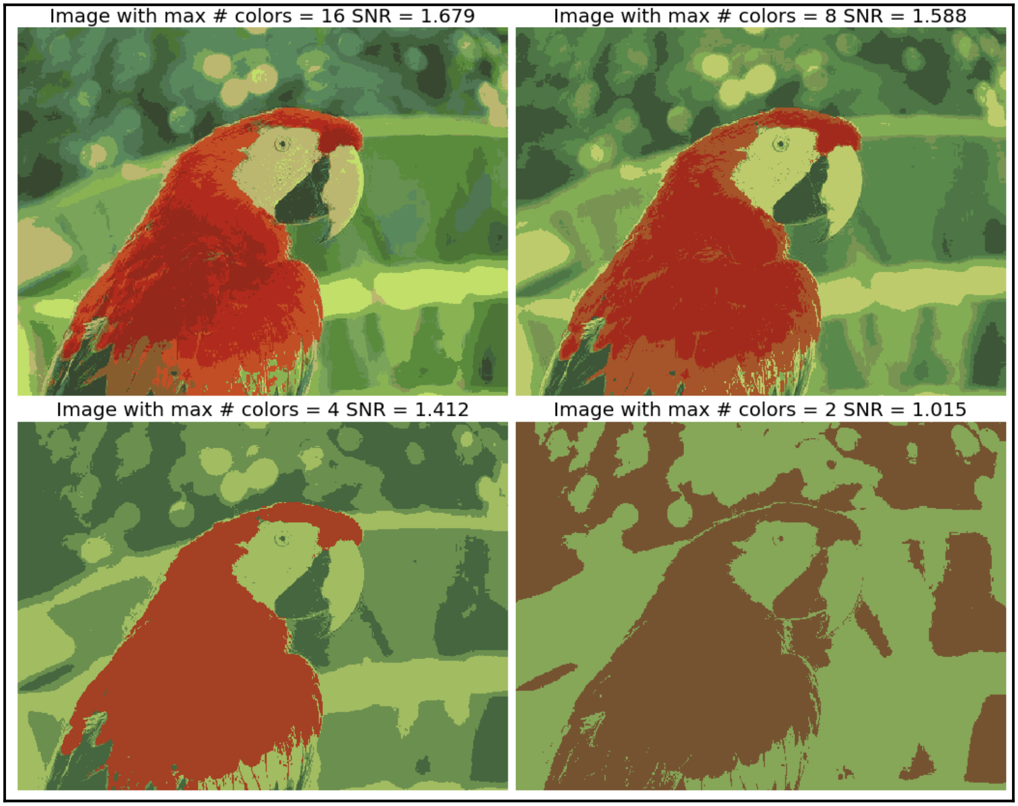
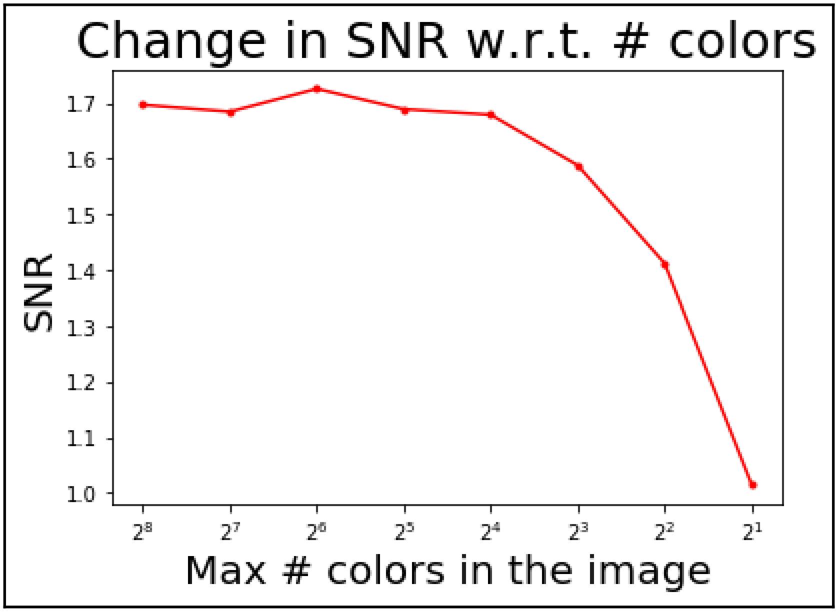


Image with max # colors = 16 SNR = 1.679
Image with max # colors = 8 SNR = 1.588
Image with max # colors = 4 SNR = 1.412
Image with max # colors = 2 SNR = 1.015


Change in SNR w.r.t. # colors
SNR
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
2^8
2^7
2^6
2^5
2^4
2^3
2^2
2^1
Max # colors in the image

2D Discrete Fourier Transform (DFT)

$$F[u, v] = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f[x, y] e^{-2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

frequencies(frequeny domain) *image (time domain)*

$u = 0, 1, .., M\text{-}1$

$v = 0, 1, .., N\text{-}1$

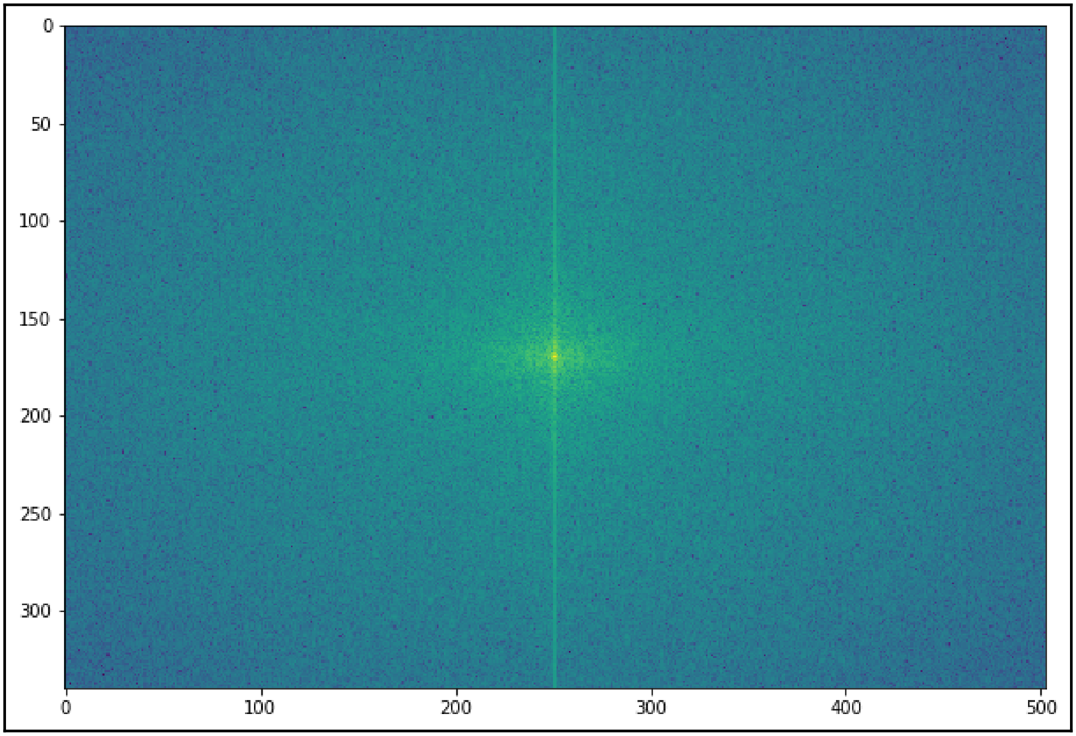
2D Inverse Discrete Fourier Transform (IDFT)

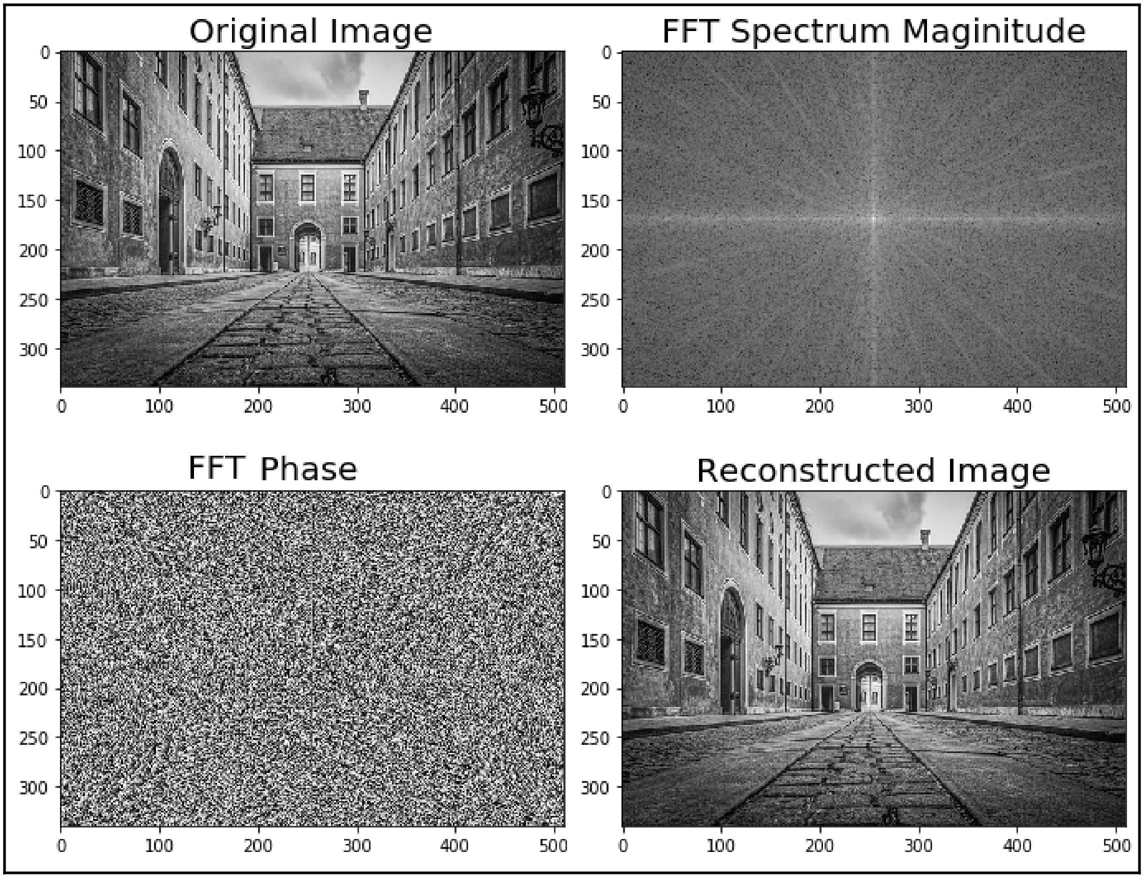
$$f[x, y] = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F[u, v] e^{2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

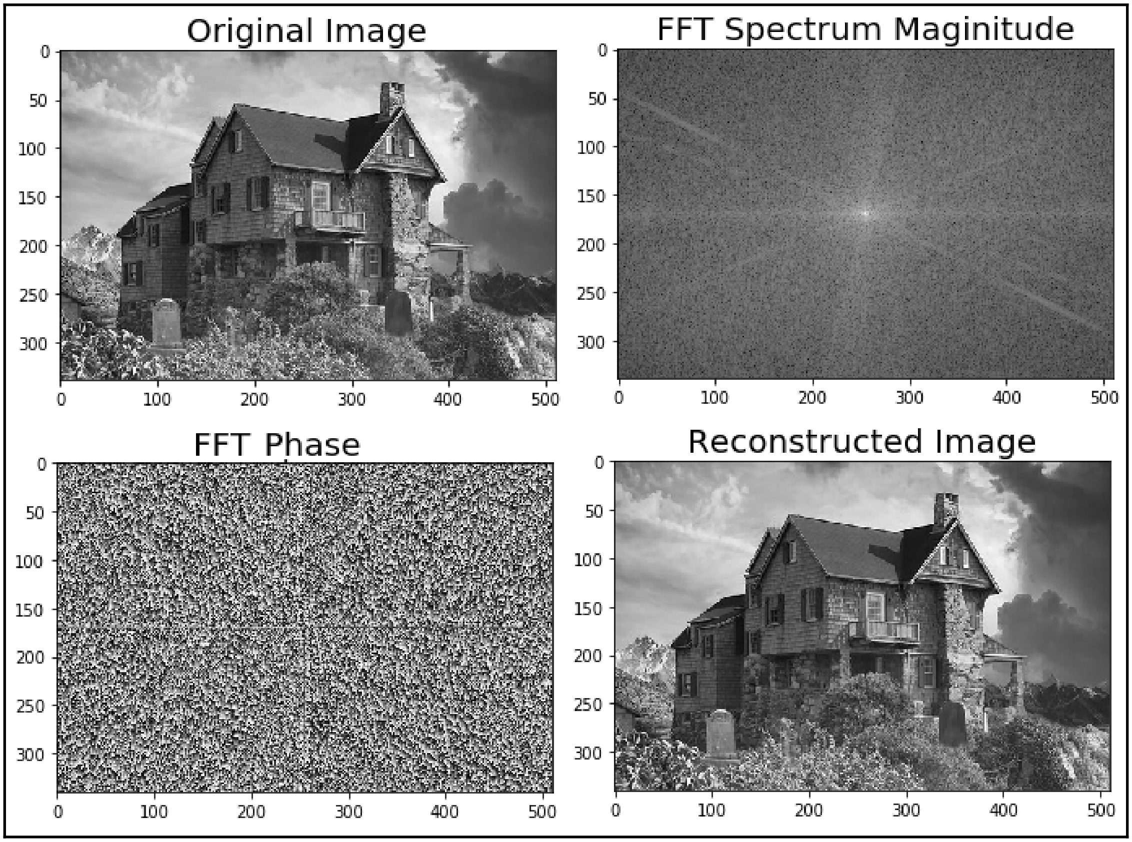
image (time domain) *frequencies(frequeny domain)*

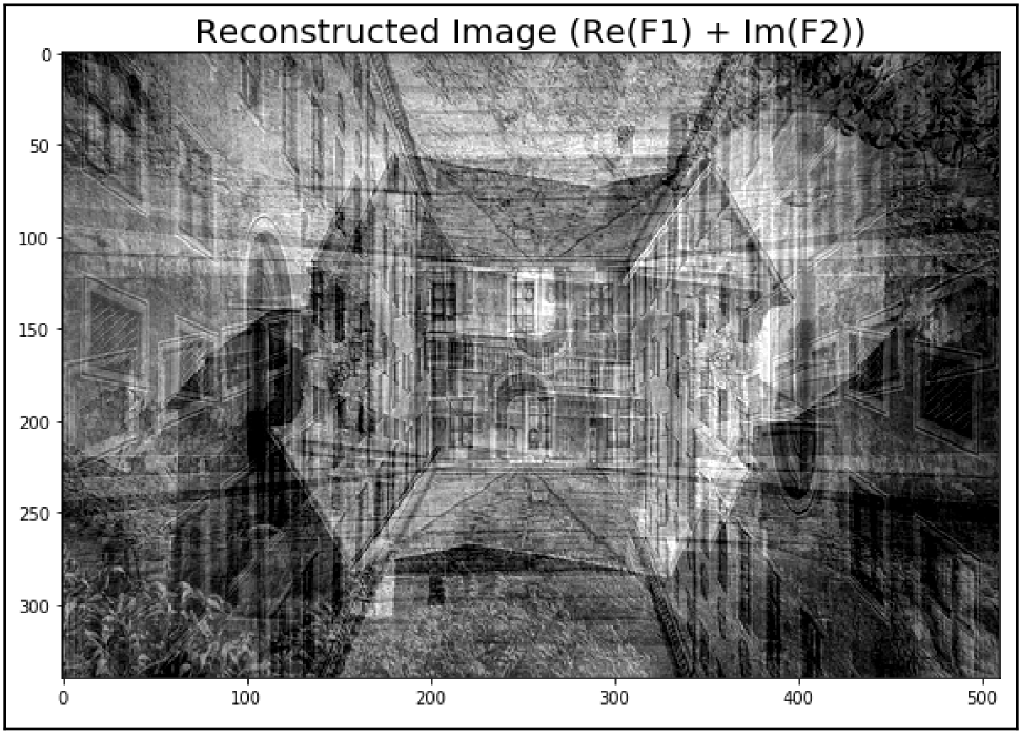
$x = 0, 1, .., M\text{-}1$

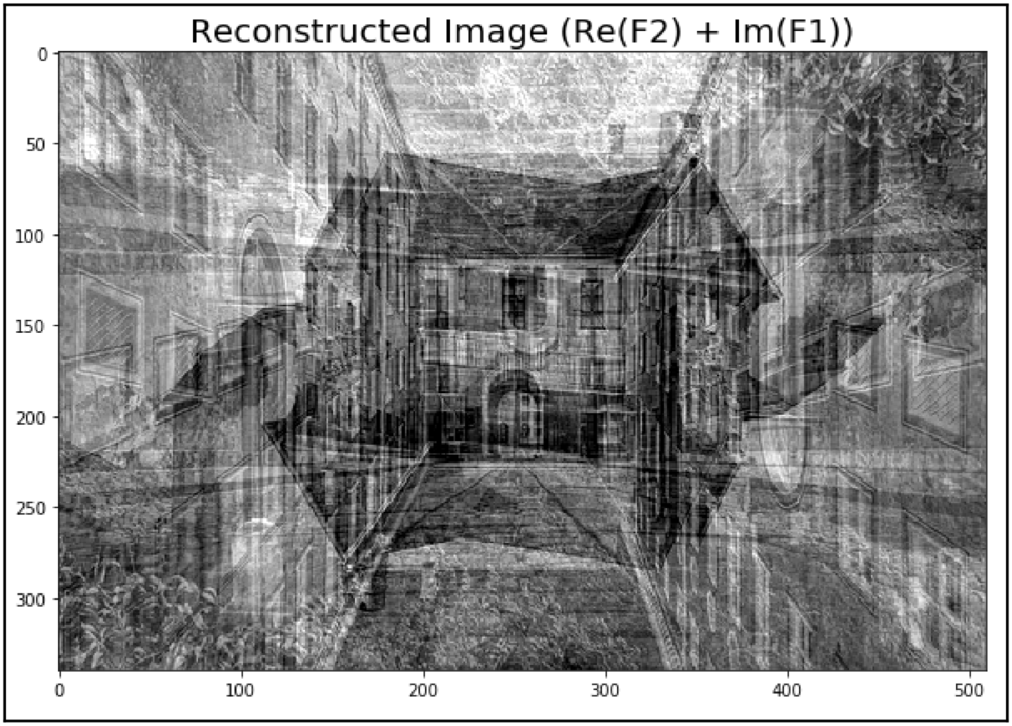
$y = 0, 1, .., N\text{-}1$

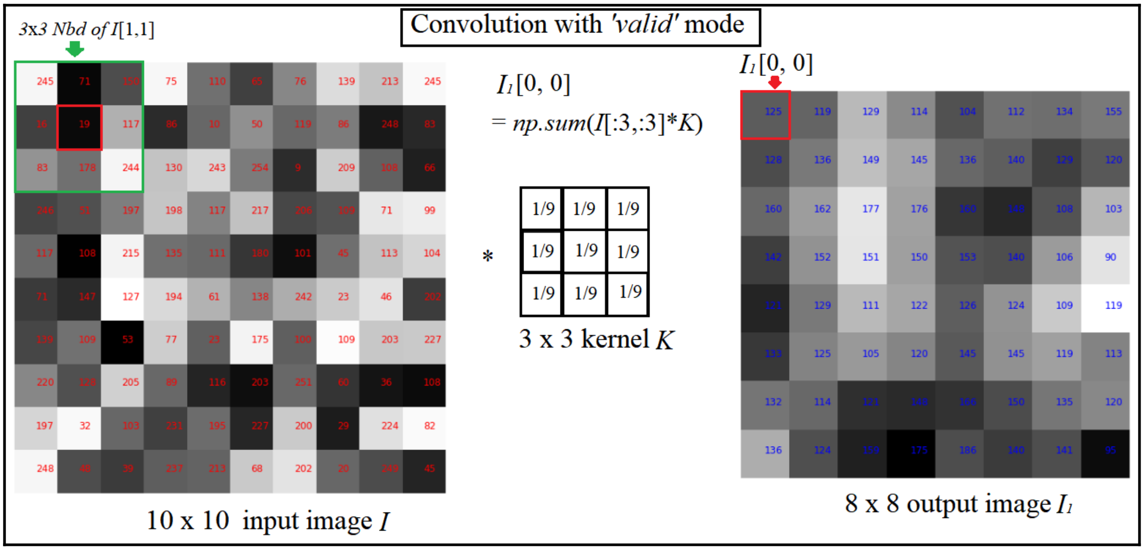

0
50
100
150
200
250
300
0
100
200
300
400
500


Original Image
FFT Spectrum Maginitude
FFT Phase
Reconstructed Image
0
50
100
150
200
250
300
0
100
200
300
400
500


Original Image
FFT Spectrum Maginitude
FFT Phase
Reconstructed Image
0
50
100
150
200
250
300
0
100
200
300
400
500

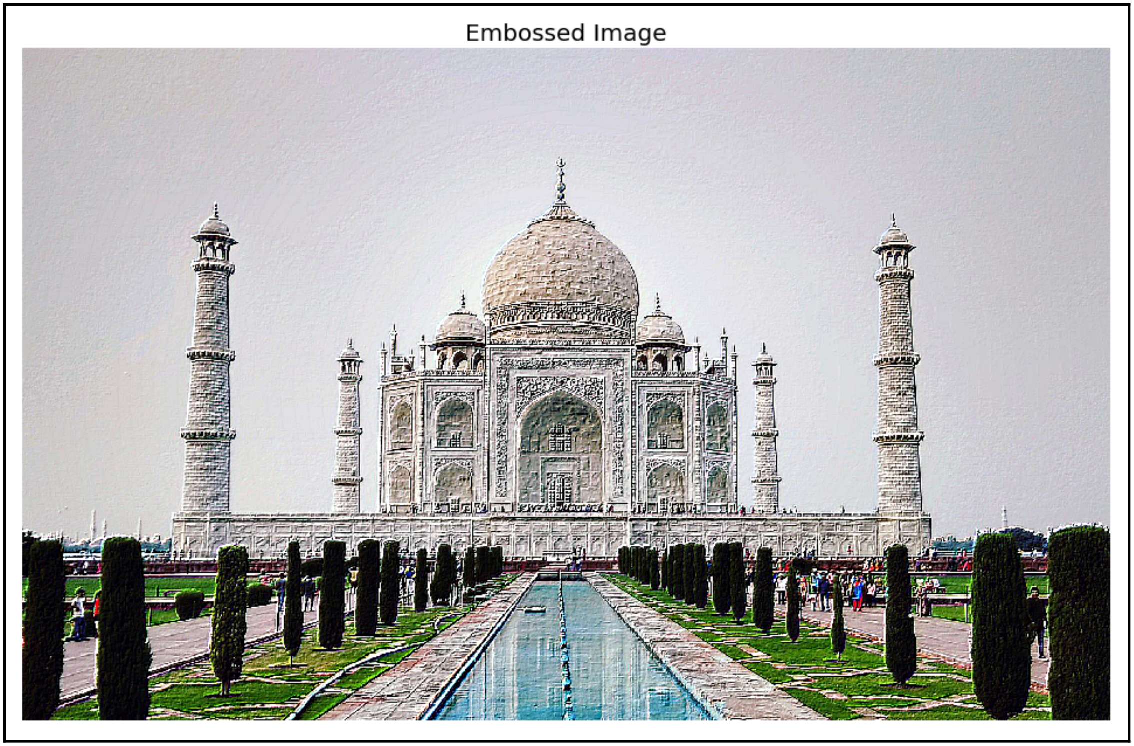

Reconstructed Image (Re(F1) + Im(F2))
0
50
100
150
200
250
300
0
100
200
300
400
500

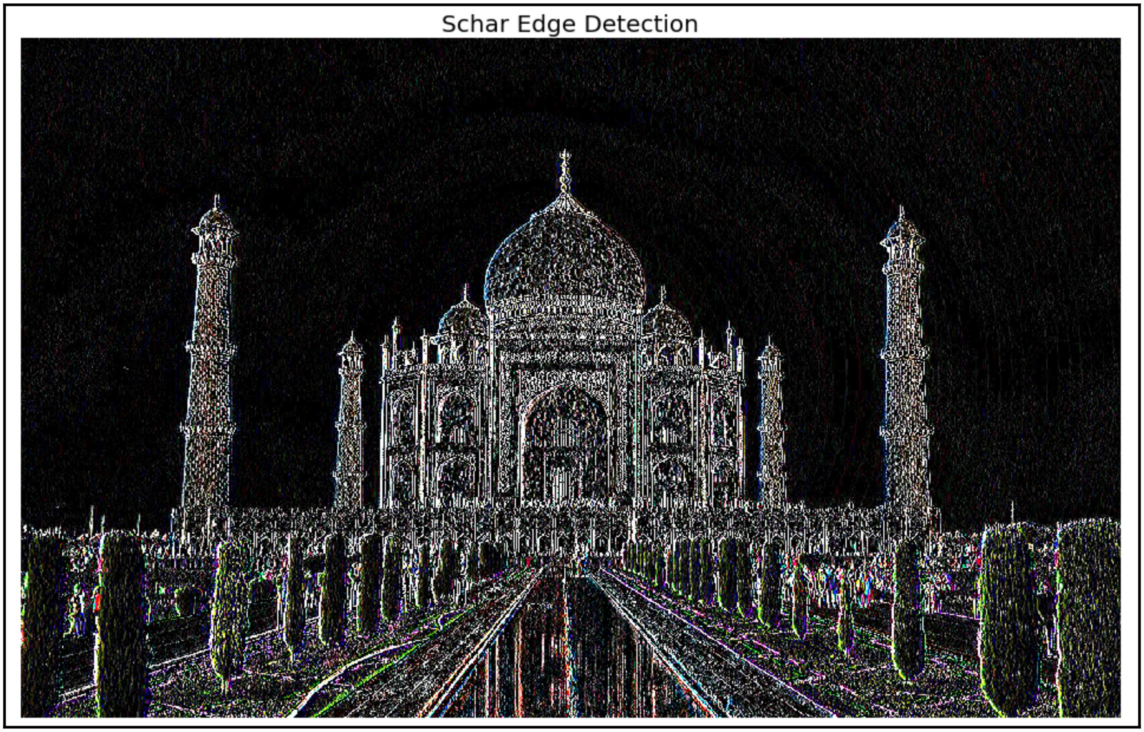

Reconstructed Image (Re(F2) + Im(F1))
0
50
100
150
200
250
300
0
100
200
300
400
500


Convolution with 'valid' mode
3x3 Nbd of I[1,1]
I1[0, 0]
= np.sum(I[:3,:3]*K)
I1[0, 0]
*
1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
3 x 3 kernel K
10 x 10 input image I
8 x 8 output image I1

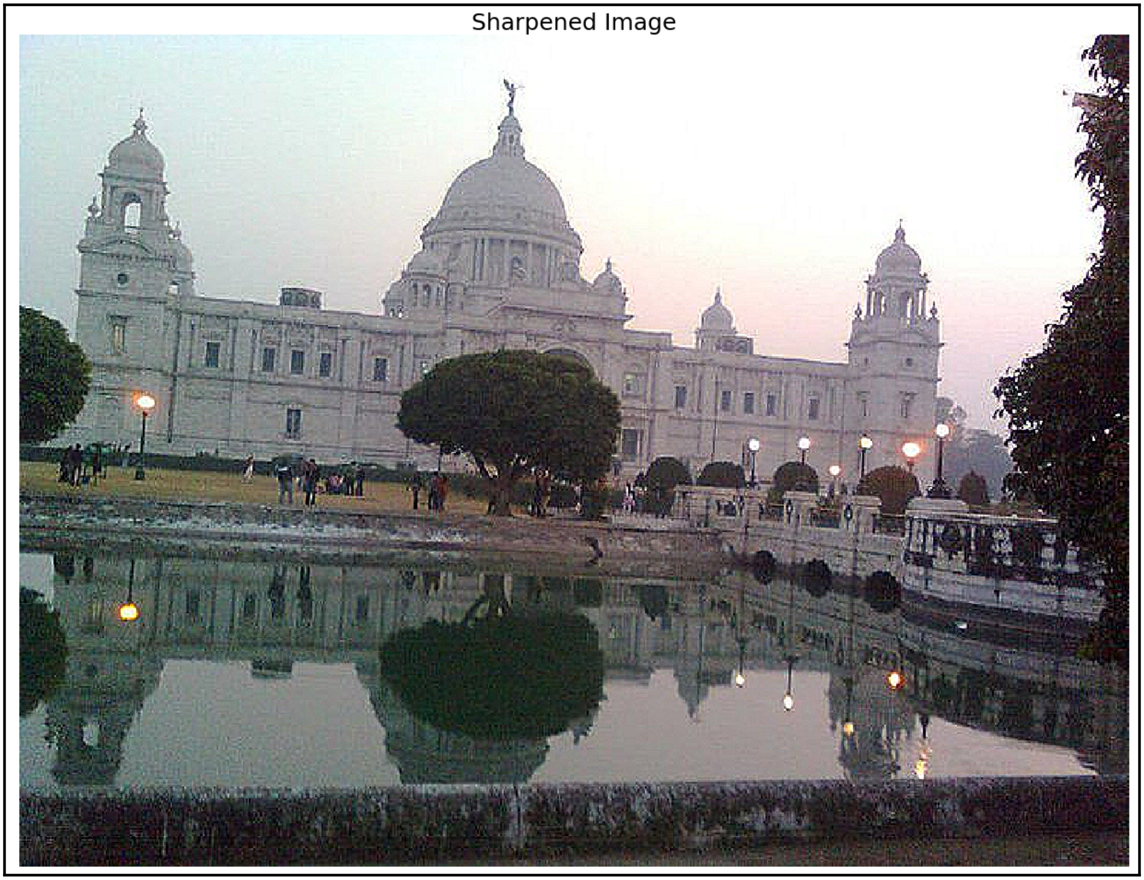

Original Image
Box Blur
Laplace Edge Detection

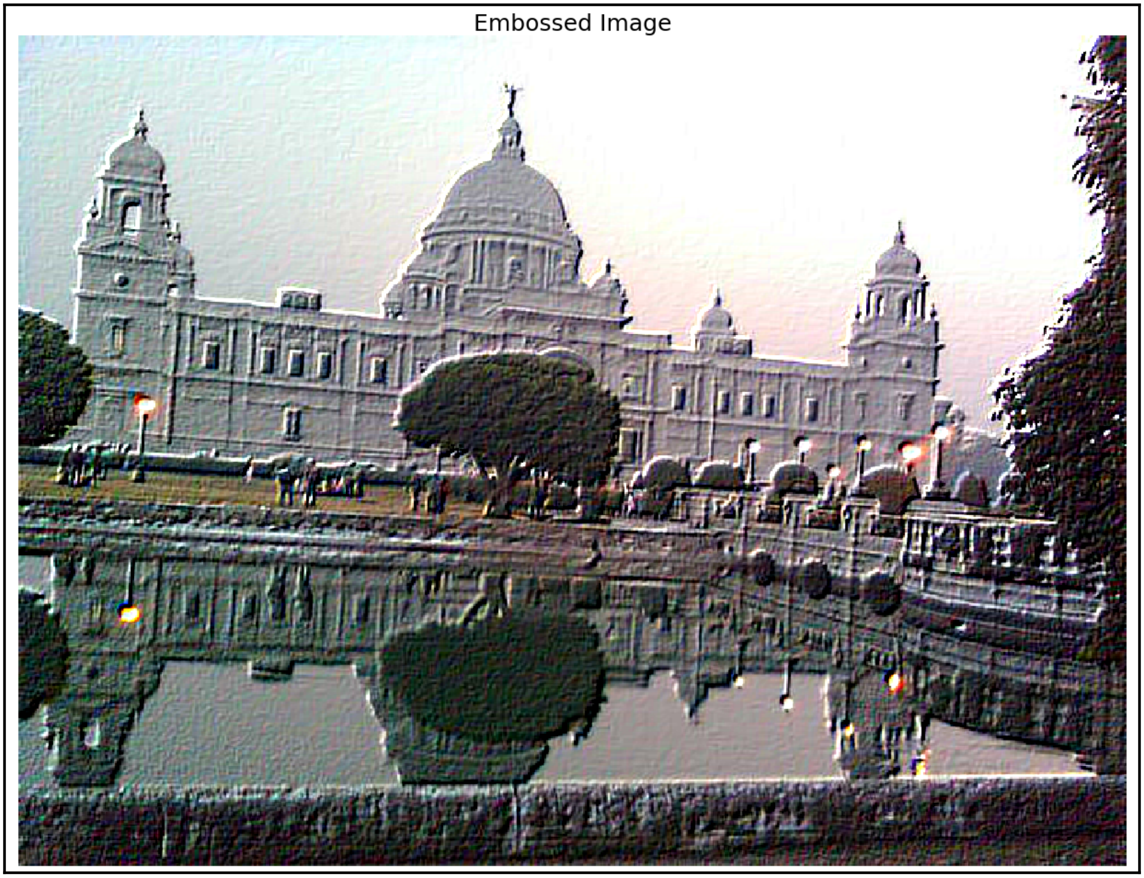

Original Image


Embossed Image


Schar Edge Detection

Original Image


Sharpened Image



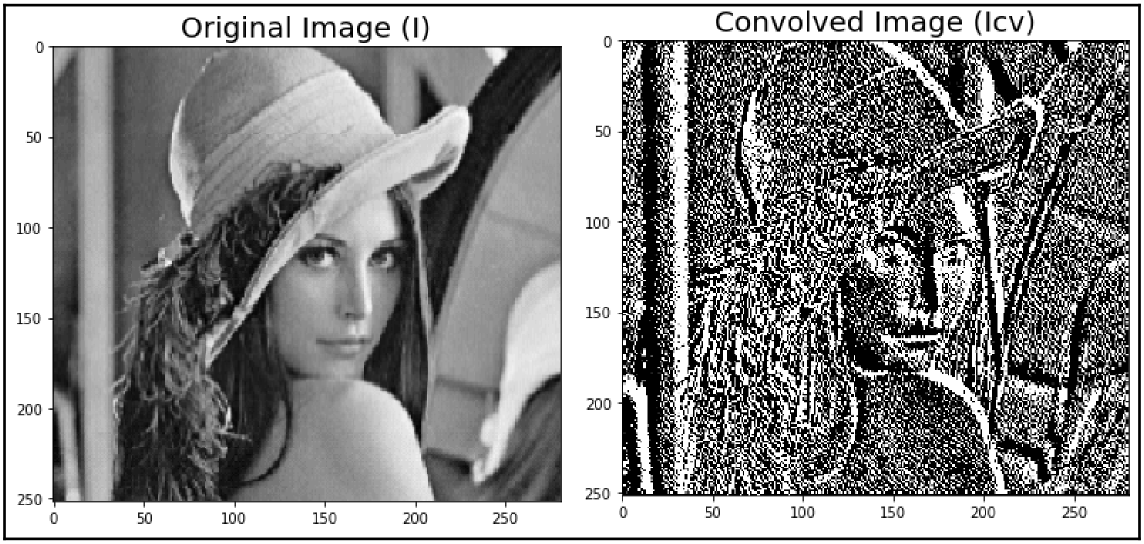
Correlation

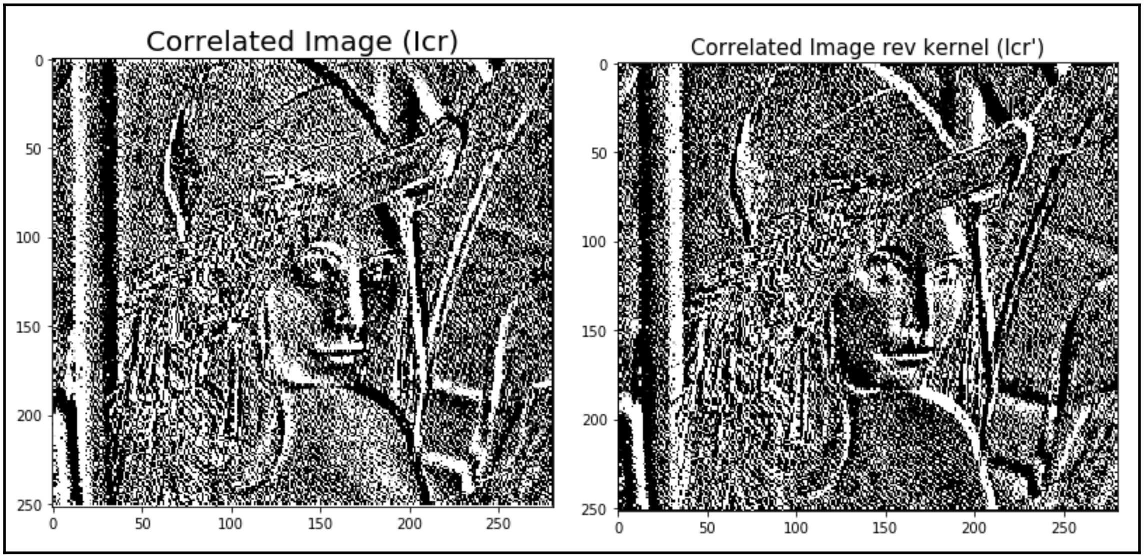
$$g(x, y) = f \star K = \sum_{u=-h}^{h} \sum_{v=-h}^{h} f(x+u, y+v)\mathbf{K}(u, v)$$

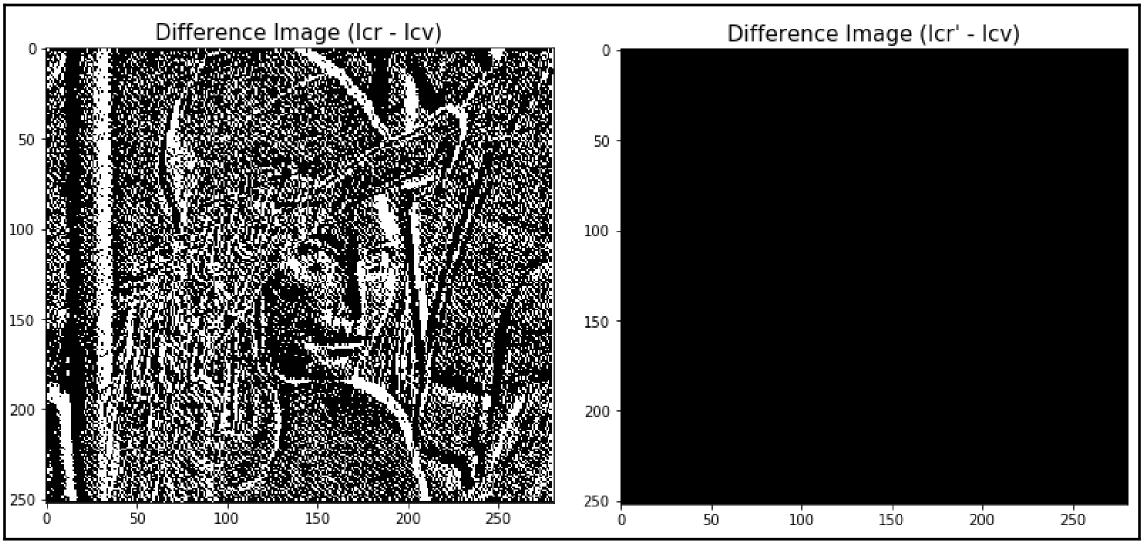
h x h kernel K

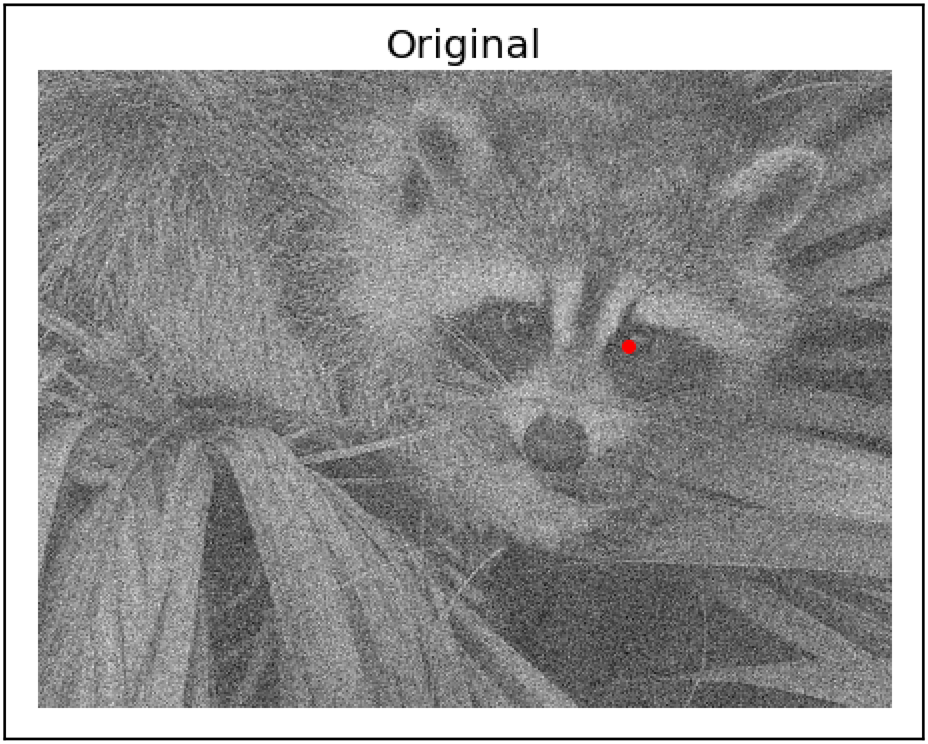
Convolution

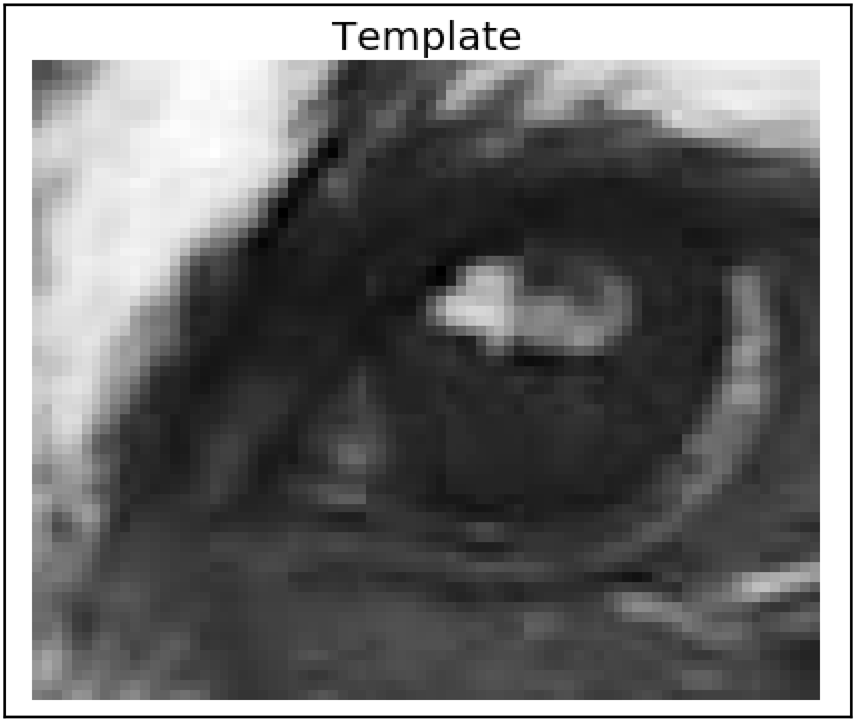
$$g(x, y) = f * K = \sum_{u=-h}^{h} \sum_{v=-h}^{h} f(x-u, y-v)\mathbf{K}(u, v)$$

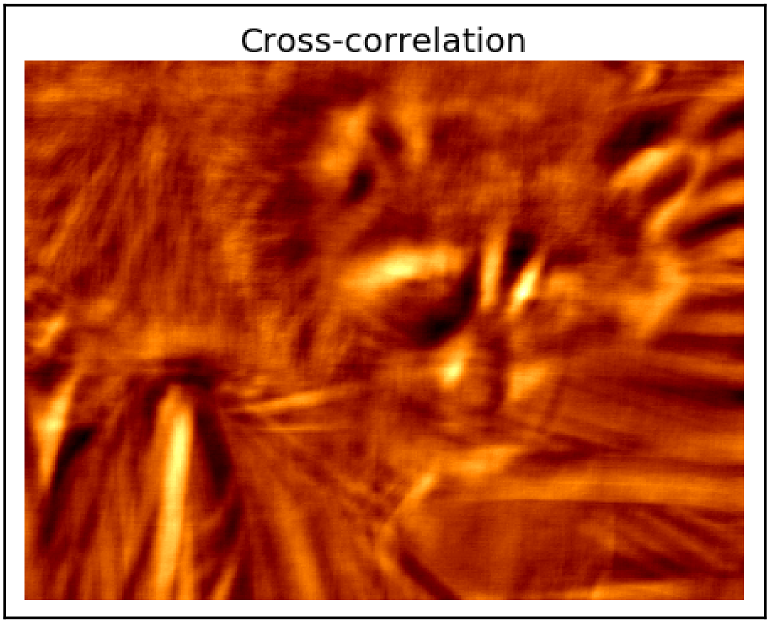



Correlated Image (Icr)
Correlated Image rev kernel (Icr')
0
50
100
150
200
250


Difference Image (Icr - Icv)
Difference Image (Icr' - Icv)
0
50
100
150
200
250


Original


Template

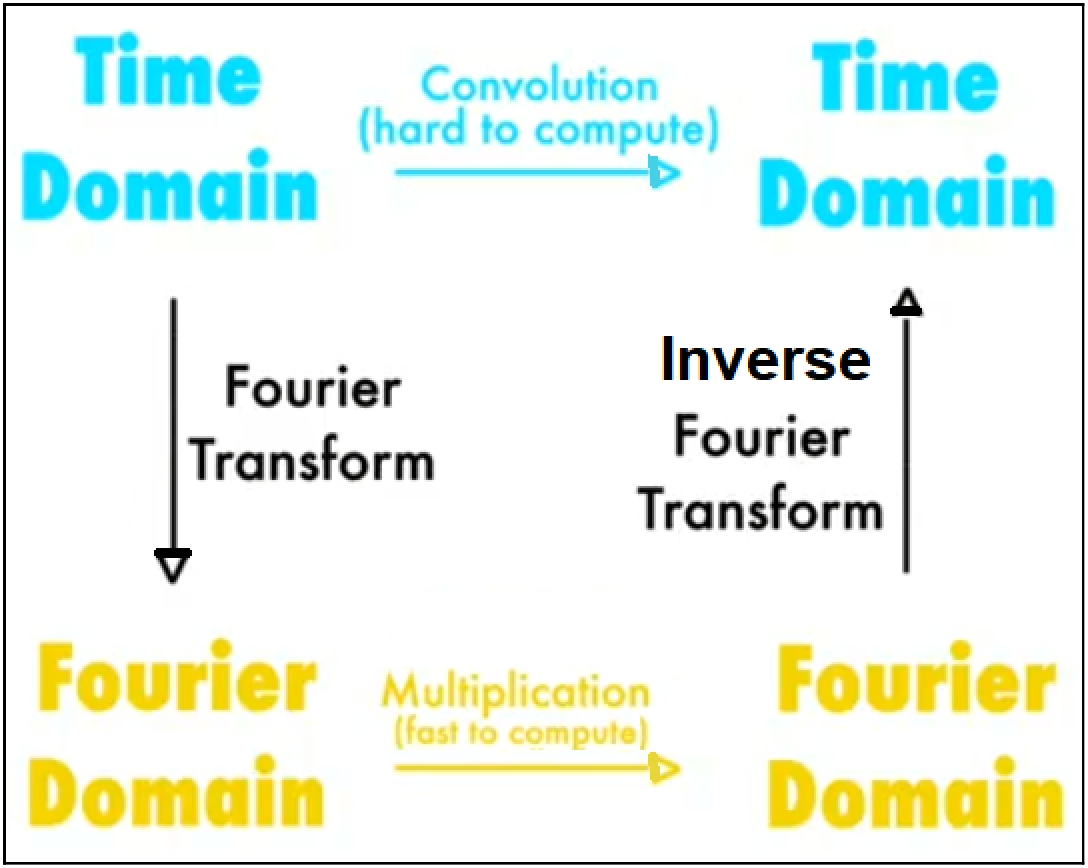

Cross-correlation

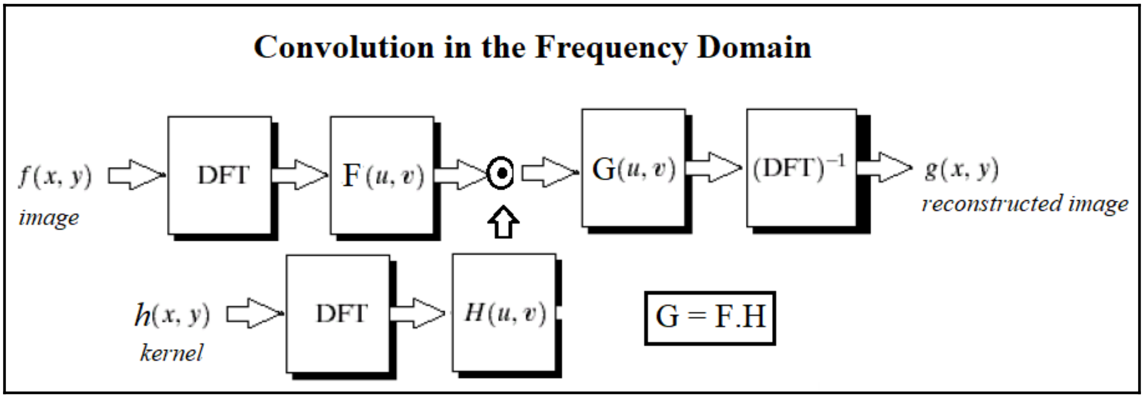
Chapter 3: Convolution and Frequency Domain Filtering

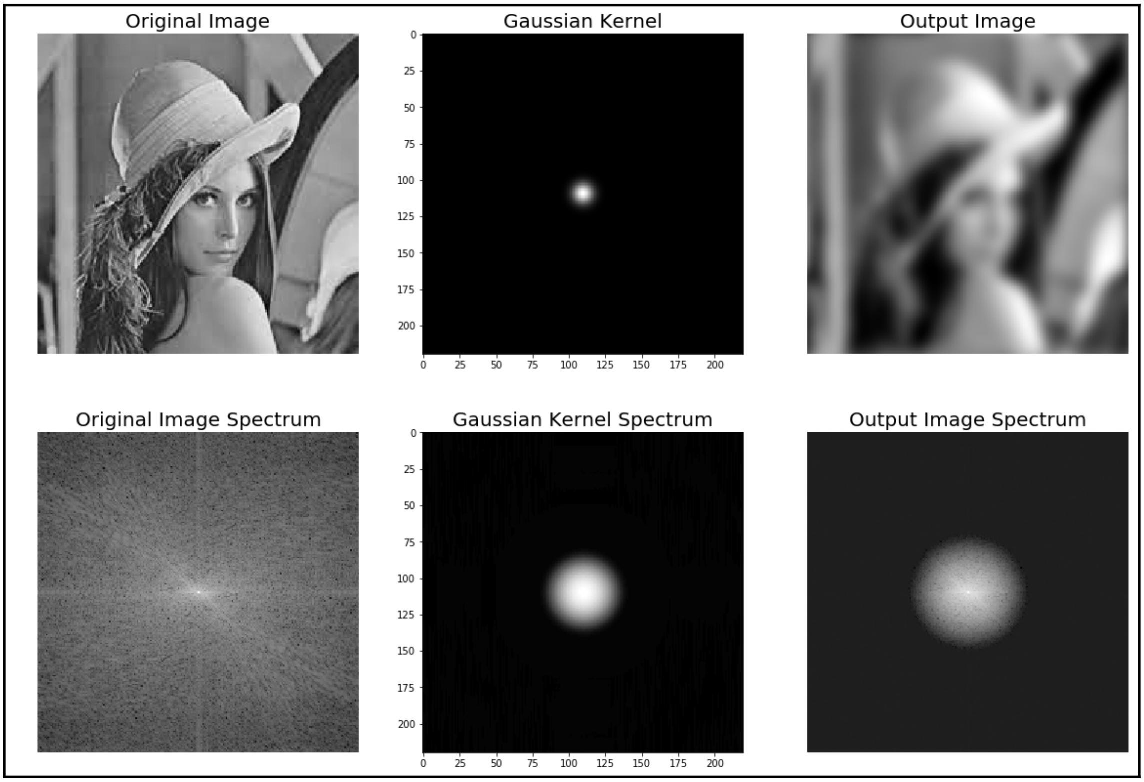
Convolution theorem

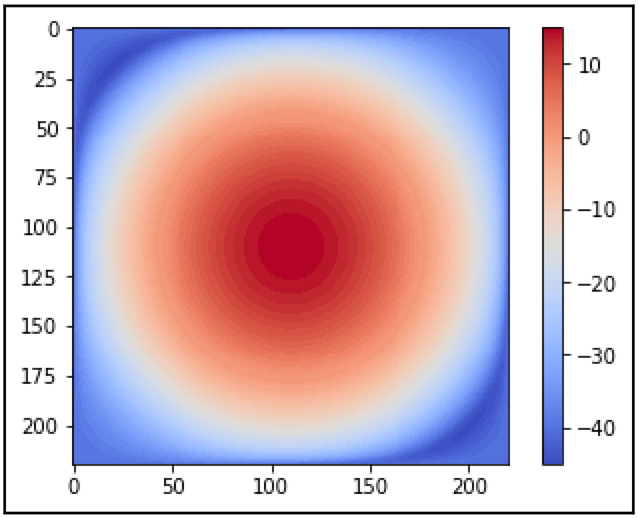
$$f(x,y) * h(x,y) \quad \Leftrightarrow \quad F(u,v)H(u,v)$$

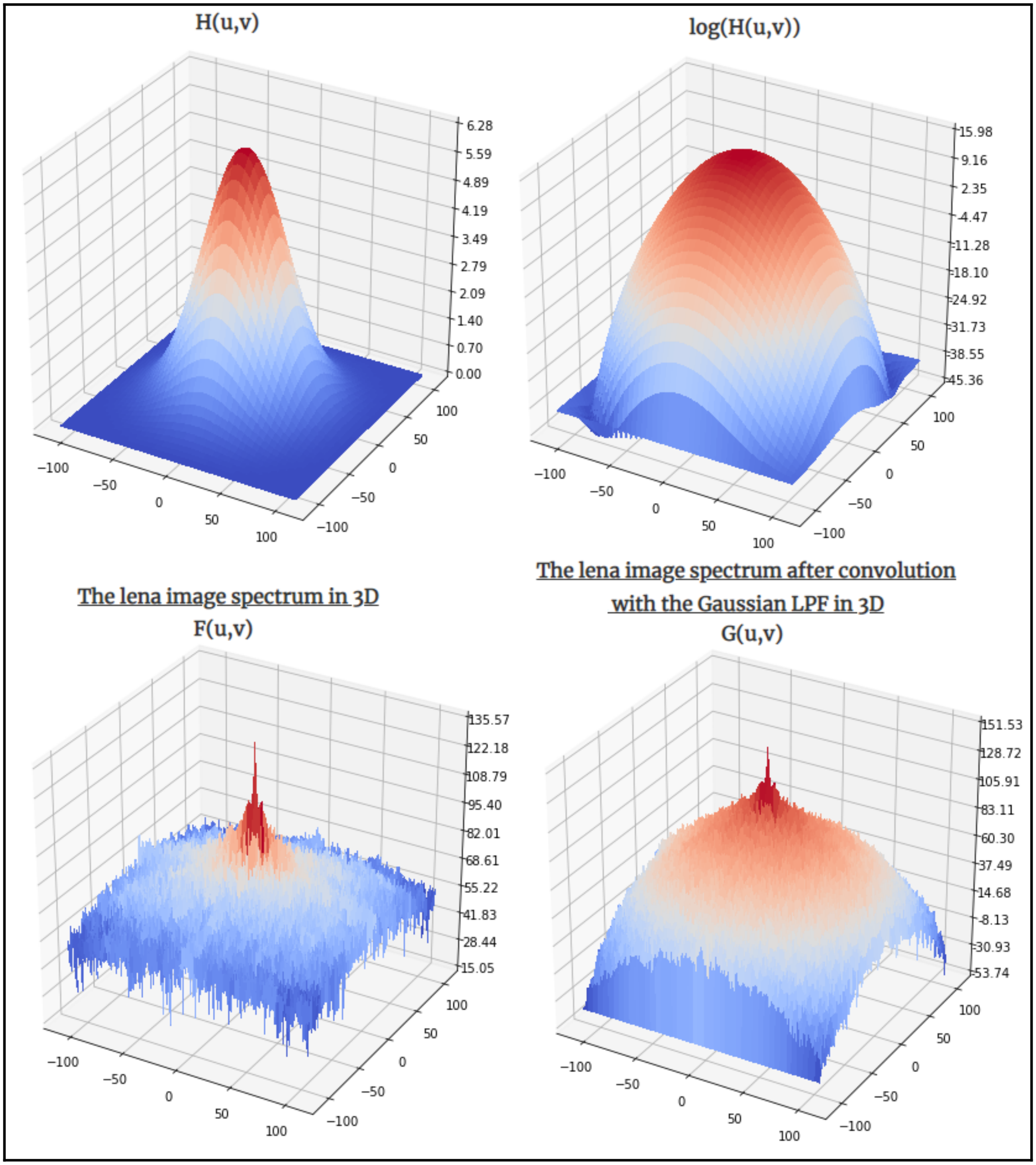
Space convolution = frequency multiplication

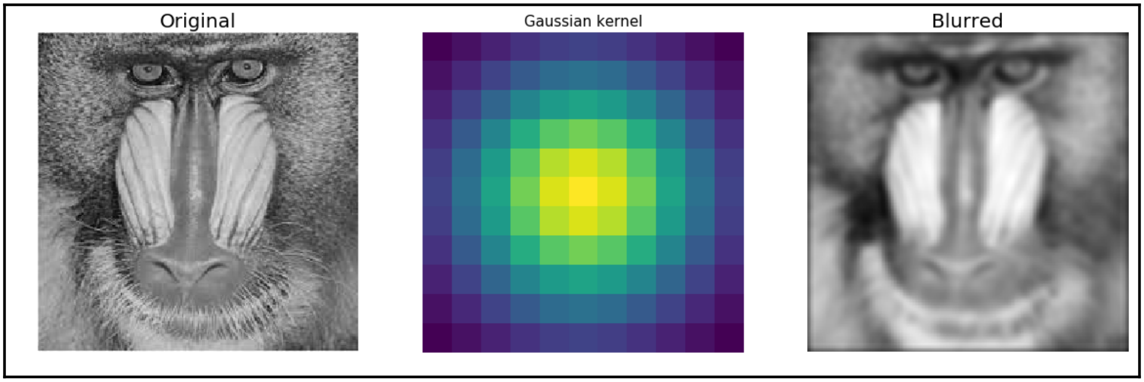

Time
Domain
Convolution
(hard to compute)
Time
Domain
Fourier
Transform
Inverse
Fourier
Transform
Fourier
Domain
Multiplication
(fast to compute)
Fourier
Domain

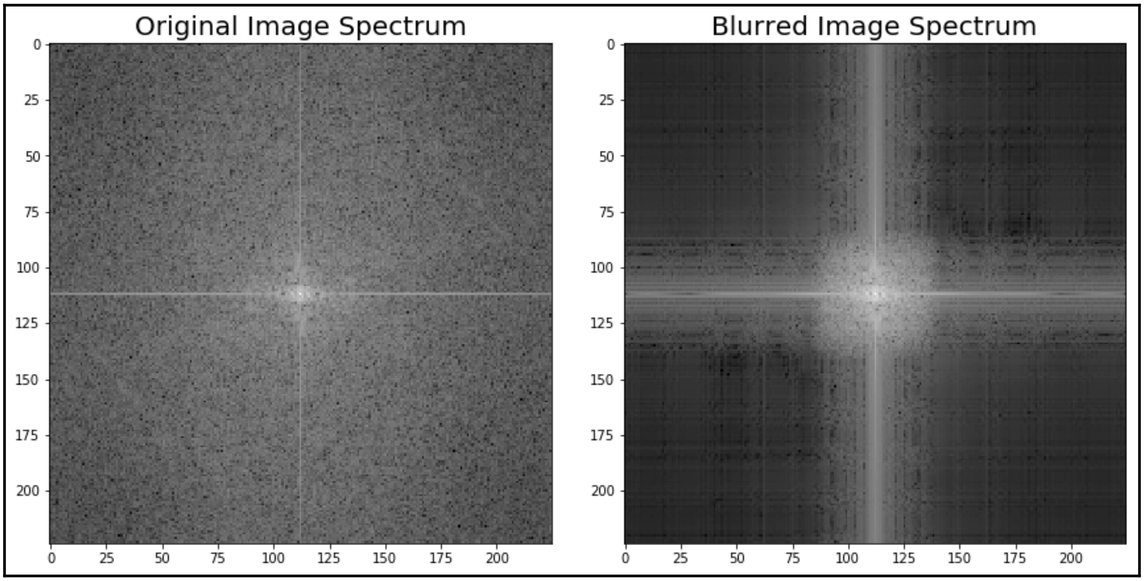

Convolution in the Frequency Domain
f(x, y)
image
DFT
F(u, v)
G(u, v)
(DFT)^-1
g(x, y)
reconstructed image
h(x, y)
kernel
DFT
H(u, v)
G = F.H

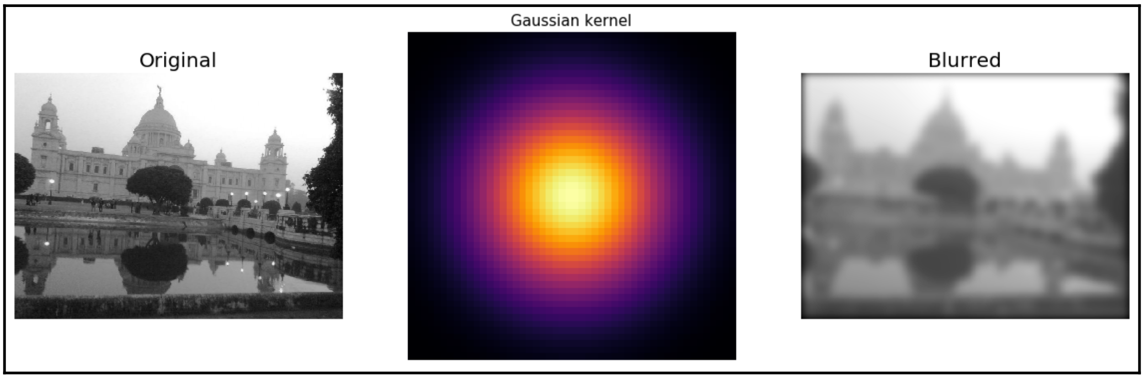

Original Image
Gaussian Kernel
Output Image
Original Image Spectrum
Gaussian Kernel Spectrum
Output Image Spectrum

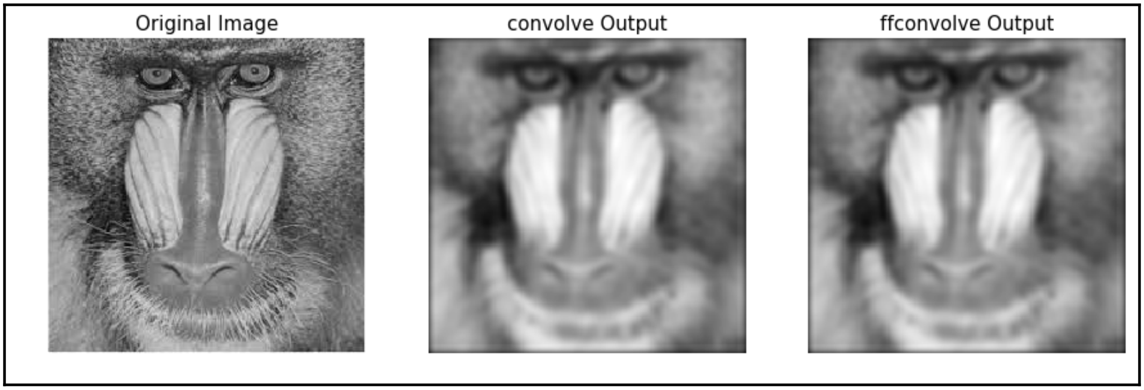


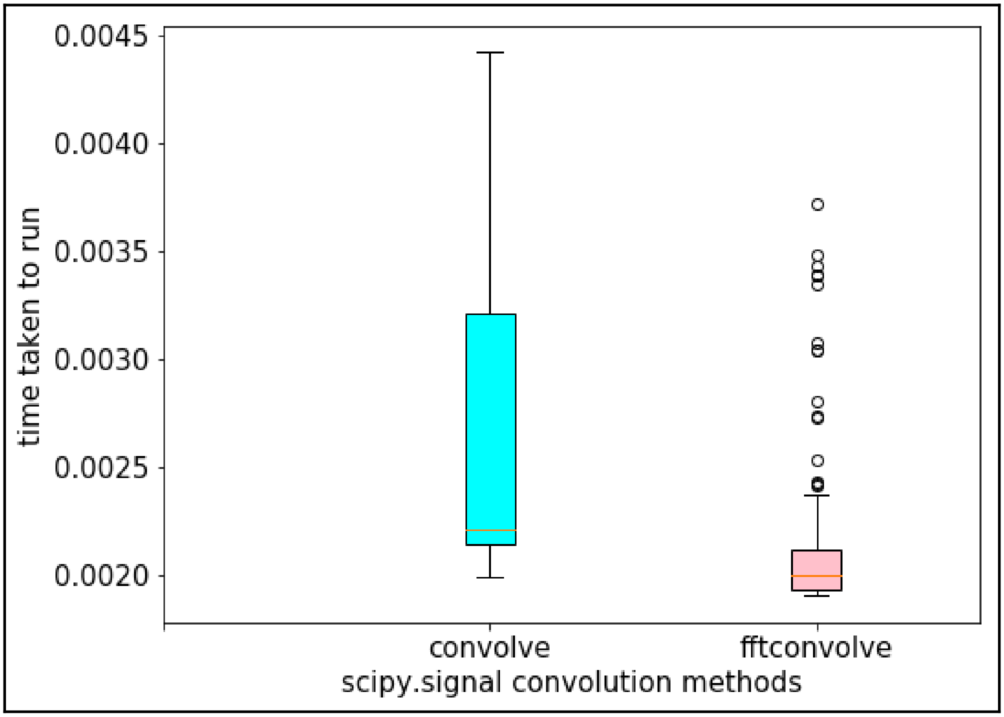

H(u,v)
log(H(u,v))
The lena image spectrum in 3D
F(u,v)
The lena image spectrum after convolution with the Gaussian LPF in 3D
G(u,v)

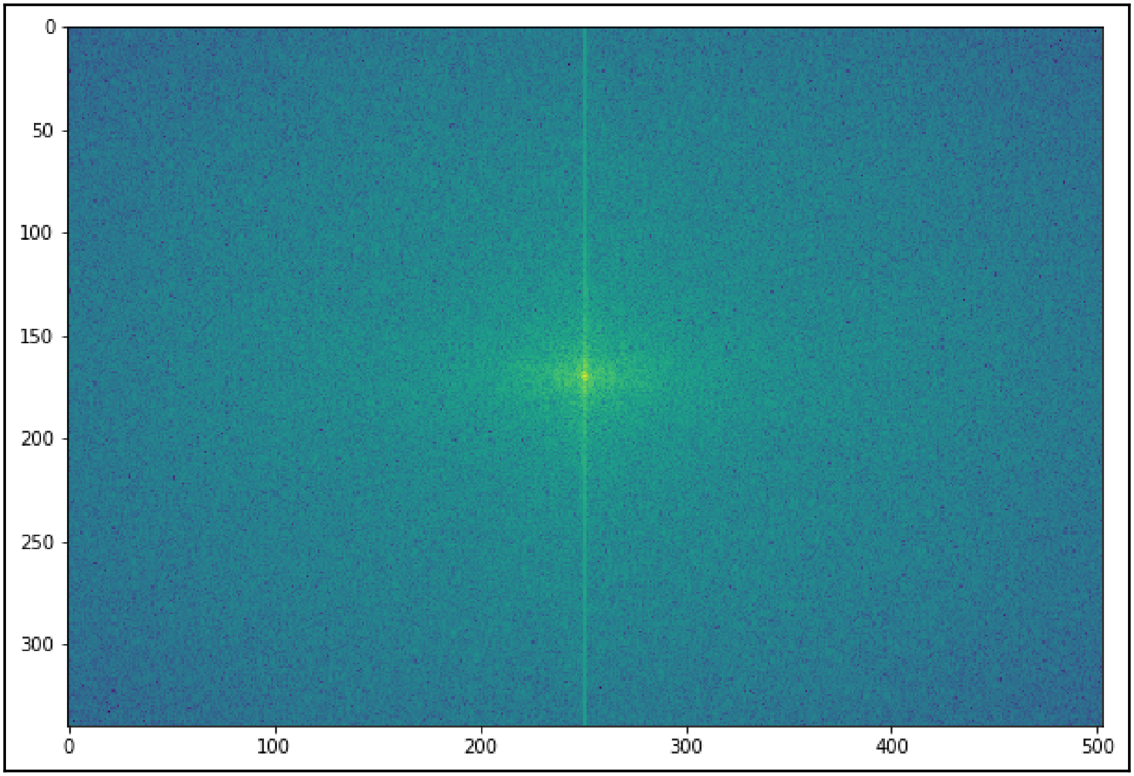

Original
Gaussian kernel
Blurred

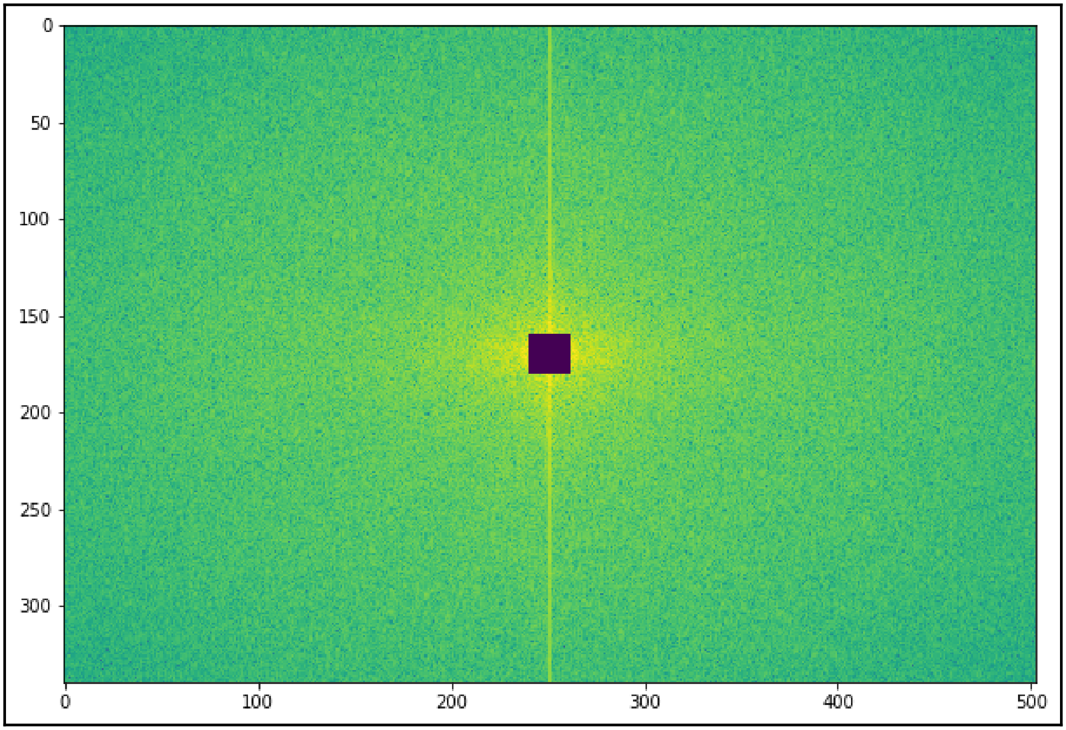

Original Image Spectrum
Blurred Image Spectrum
0
25
50
75
100
125
150
175
200

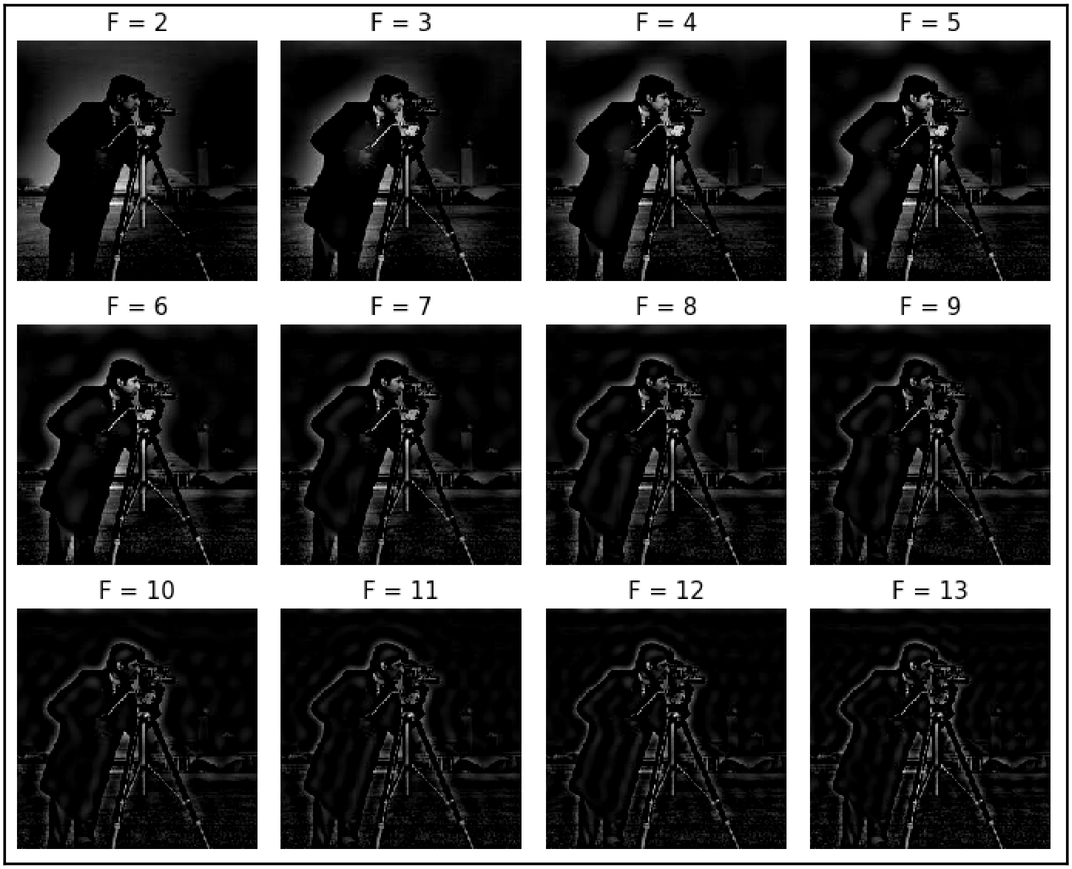

Original
Gaussian kernel
Blurred

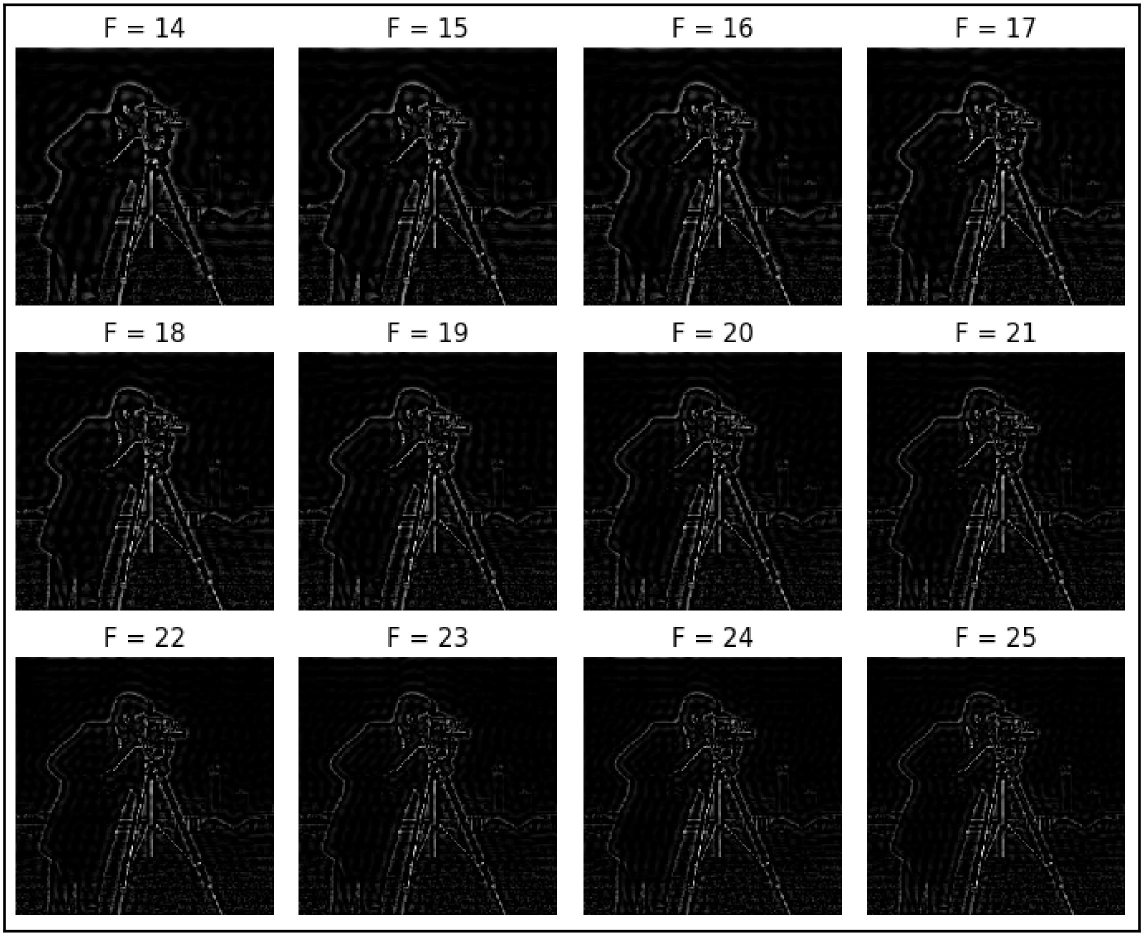

Original Image
convolve Output
ffconvolve Output

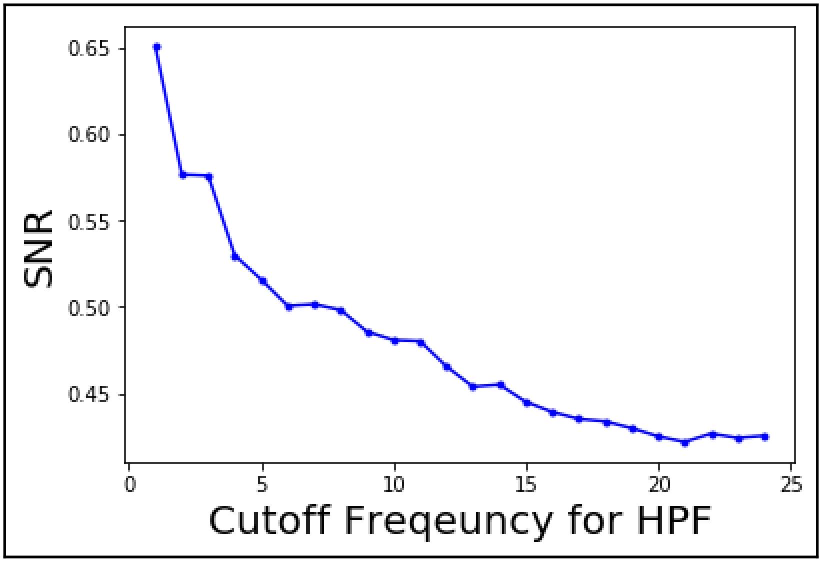

0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
time taken to run
convolve
fftconvolve
scipy.signal convolution methods


0
50
100
150
200
250
300
0
100
200
300
400
500

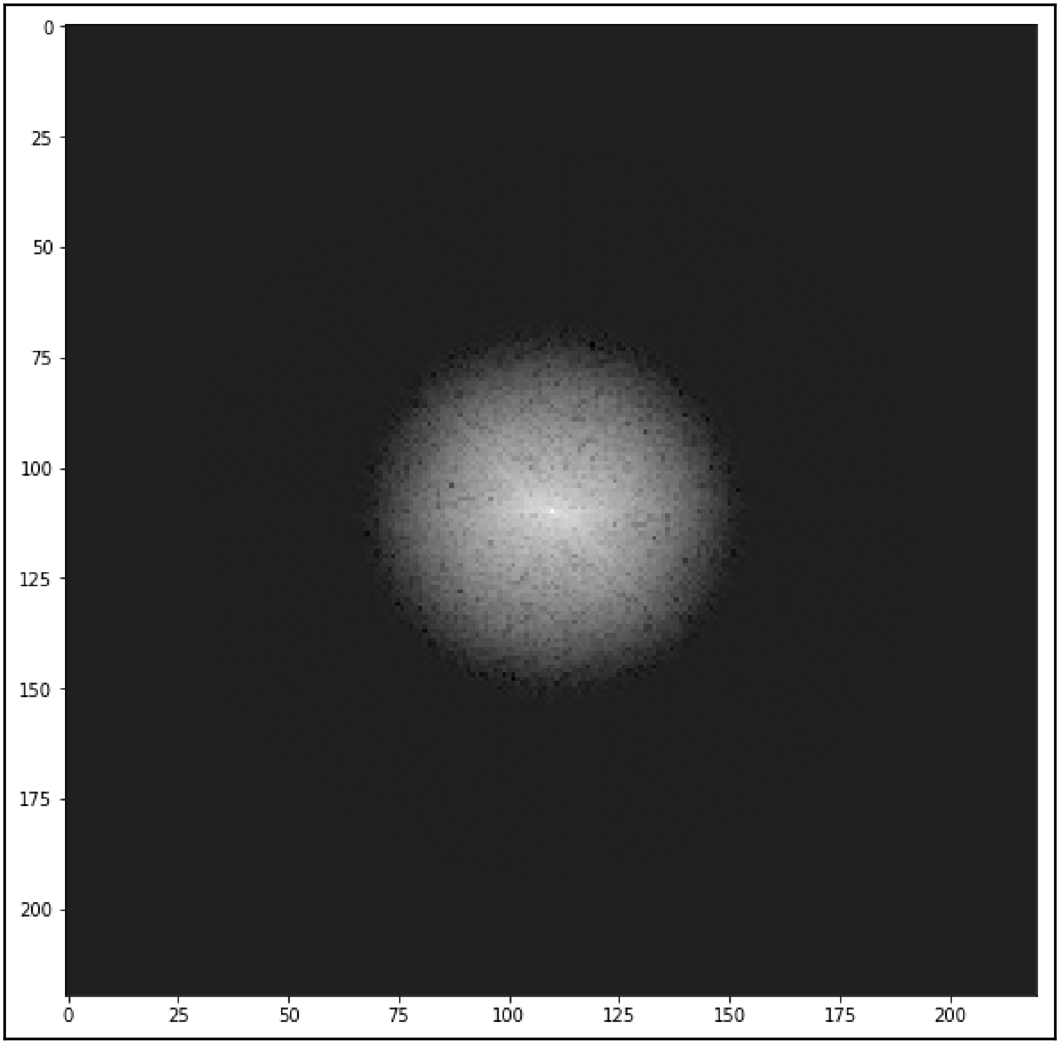

0
50
100
150
200
250
300
0
100
200
300
400
500

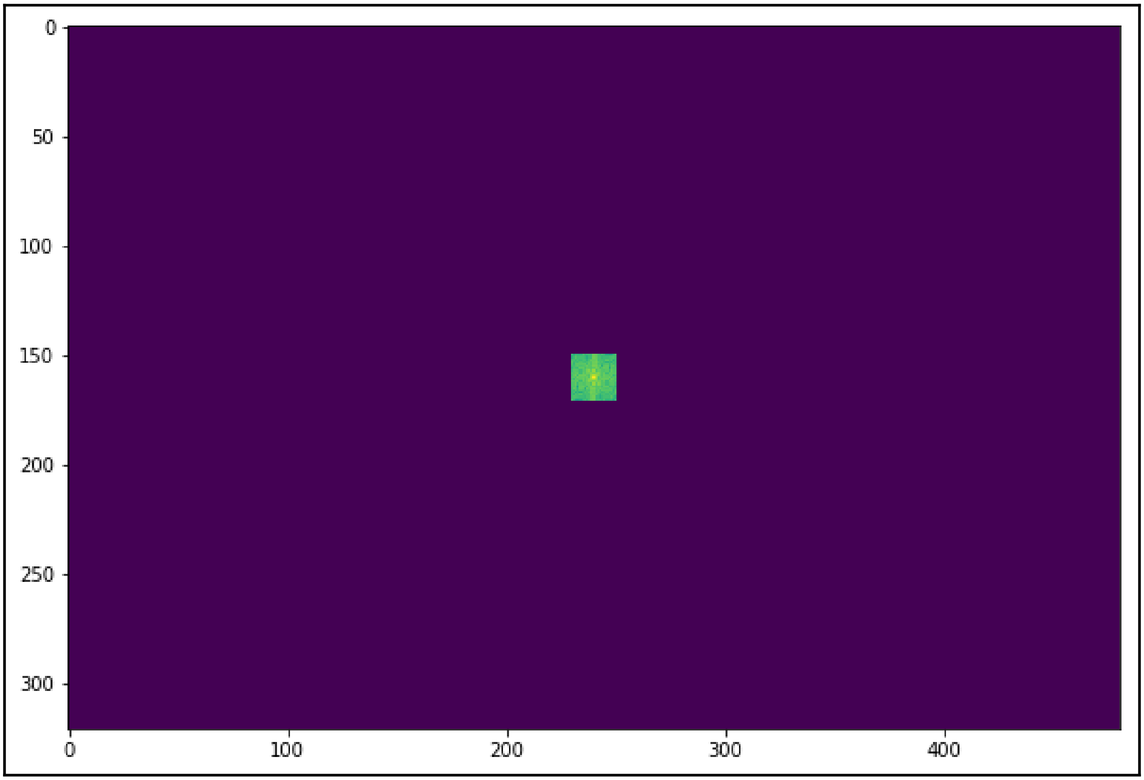

F = 2
F = 3
F = 4
F = 5
F = 6
F = 7
F = 8
F = 9
F = 10
F = 11
F = 12
F = 13

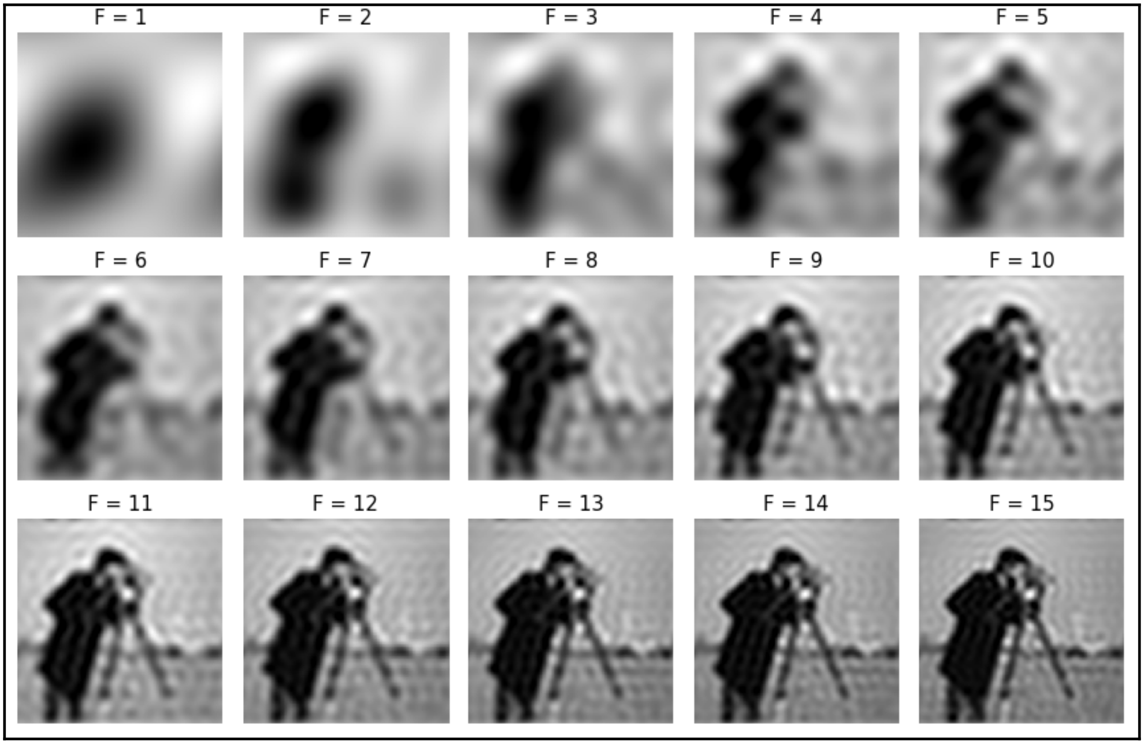

F = 14
F = 15
F = 16
F = 17
F = 18
F = 19
F = 20
F = 21
F = 22
F = 23
F = 24
F = 25

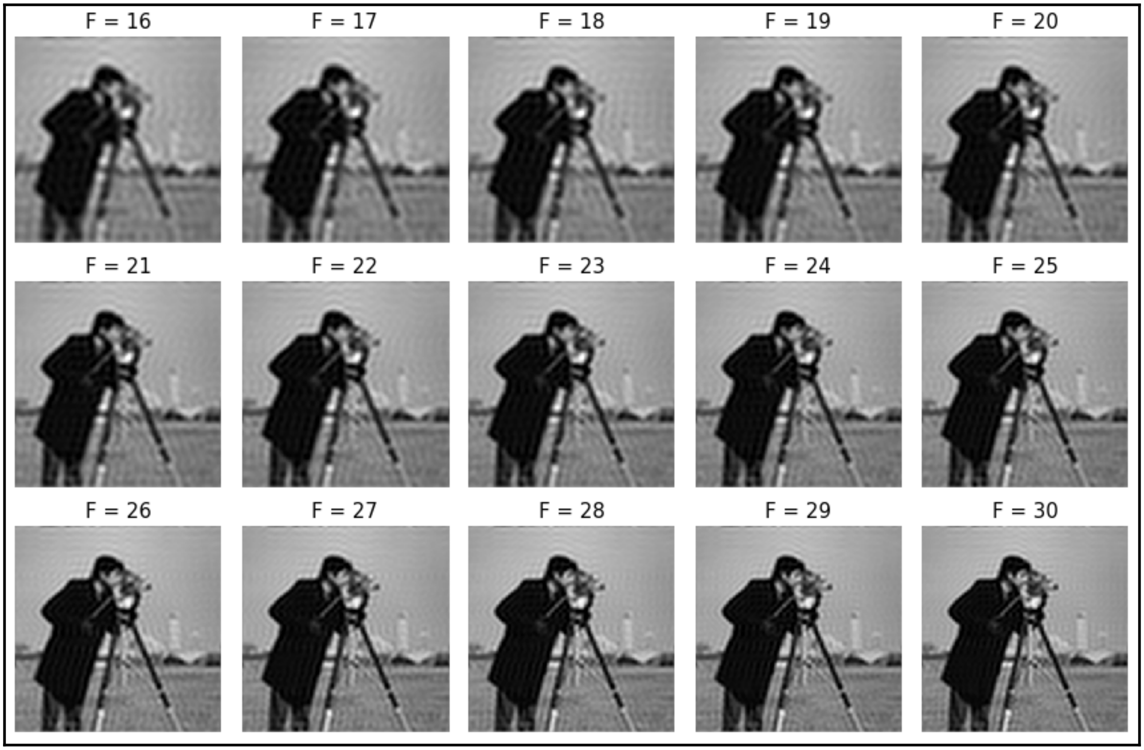

SNR
0.65
0.60
0.55
0.50
0.45
0
5
10
15
20
25
Cutoff Freqeuncy for HPF

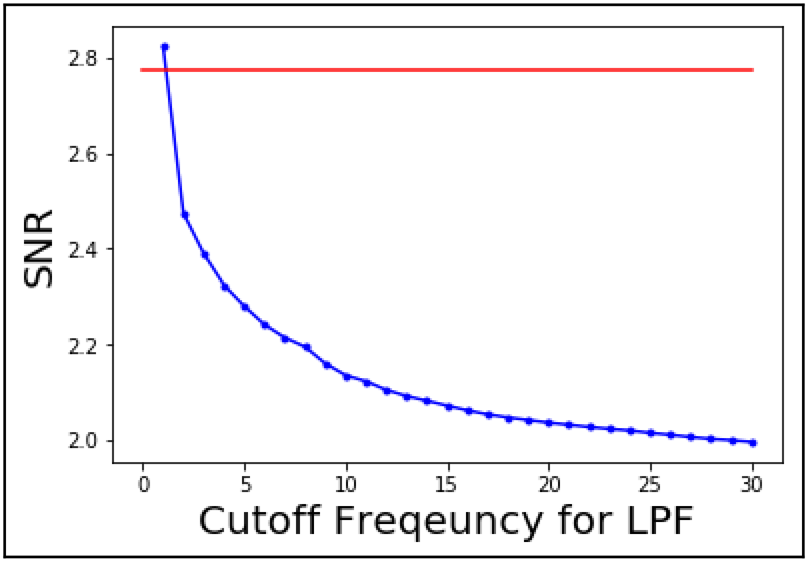

Original Image
Blurred Image with LPF

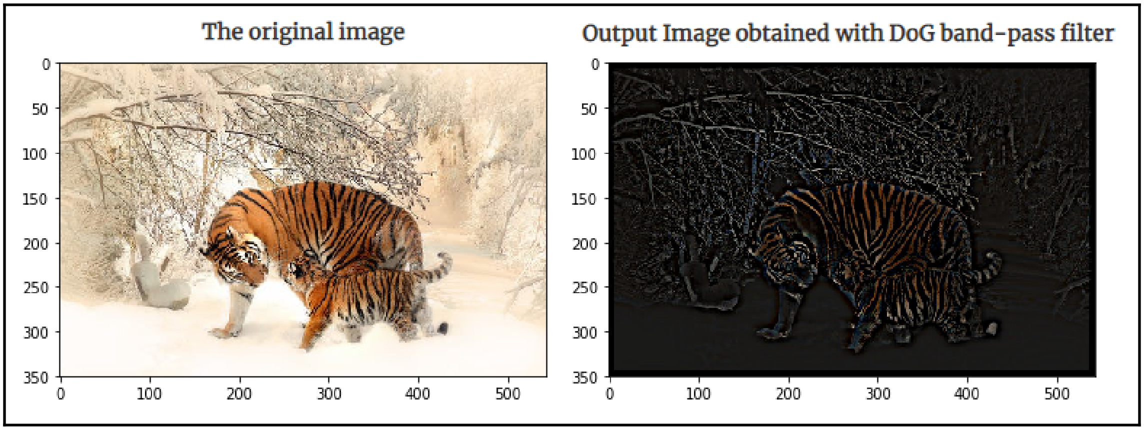

0
25
50
75
100
125
150
175
200
0
25
50
75
100
125
150
175
200

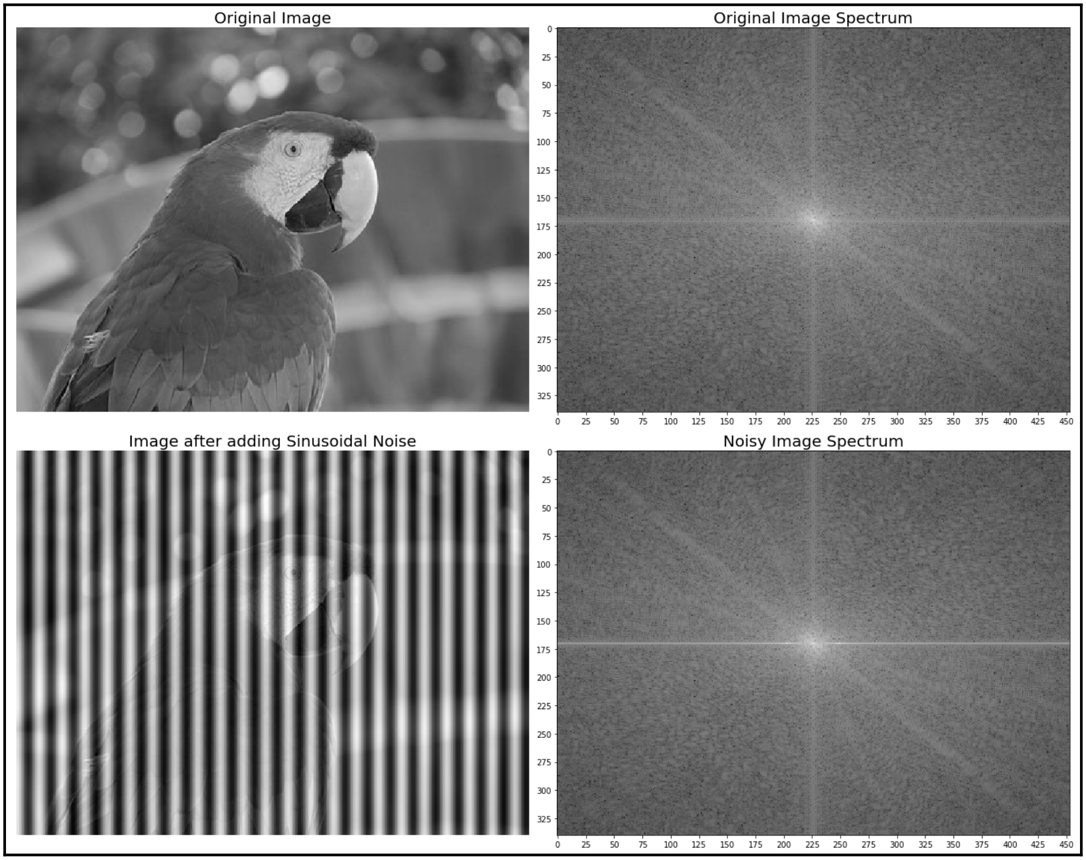

0
50
100
150
200
250
300
0
100
200
300
400

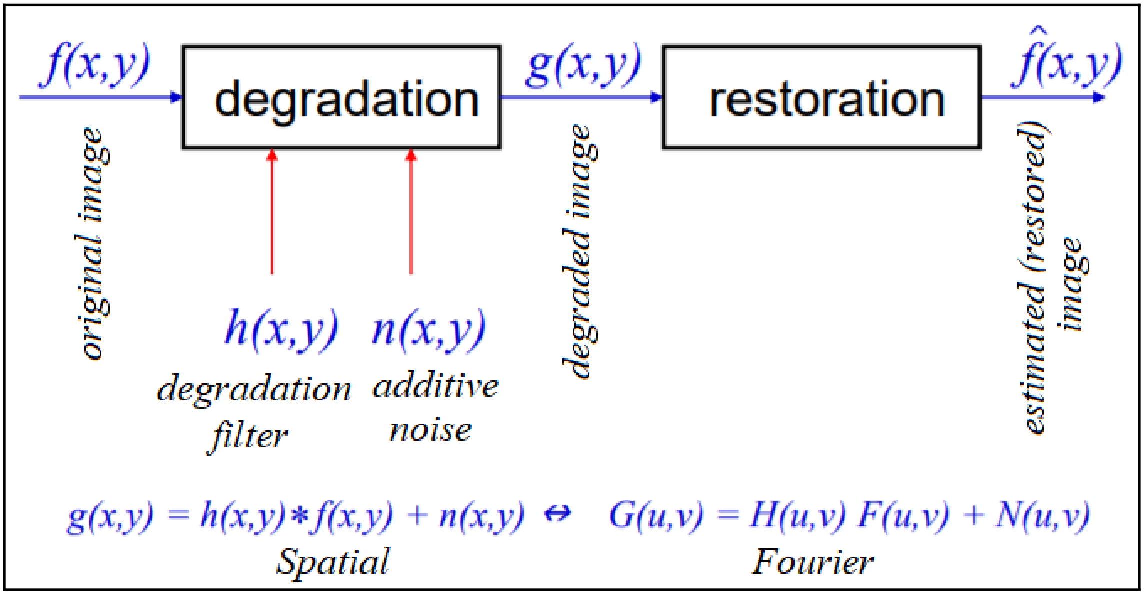

F = 1
F = 2
F = 3
F = 4
F = 5
F = 6
F = 7
F = 8
F = 9
F = 10
F = 11
F = 12
F = 13
F = 14
F = 15

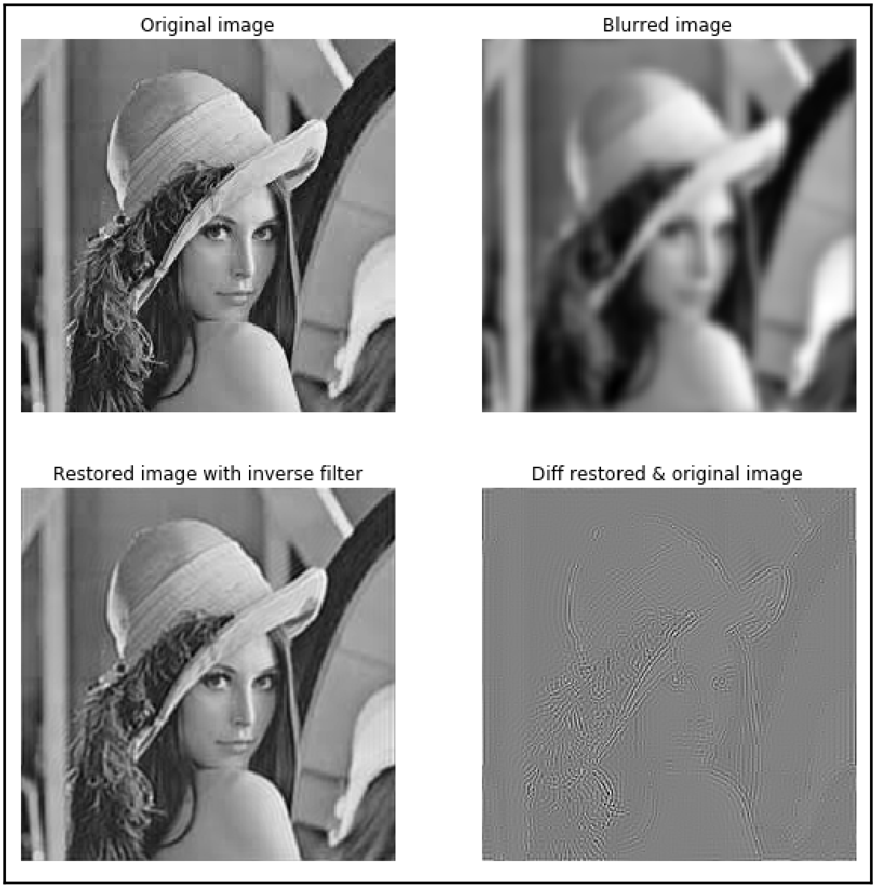

F = 16
F = 17
F = 18
F = 19
F = 20
F = 21
F = 22
F = 23
F = 24
F = 25
F = 26
F = 27
F = 28
F = 29
F = 30

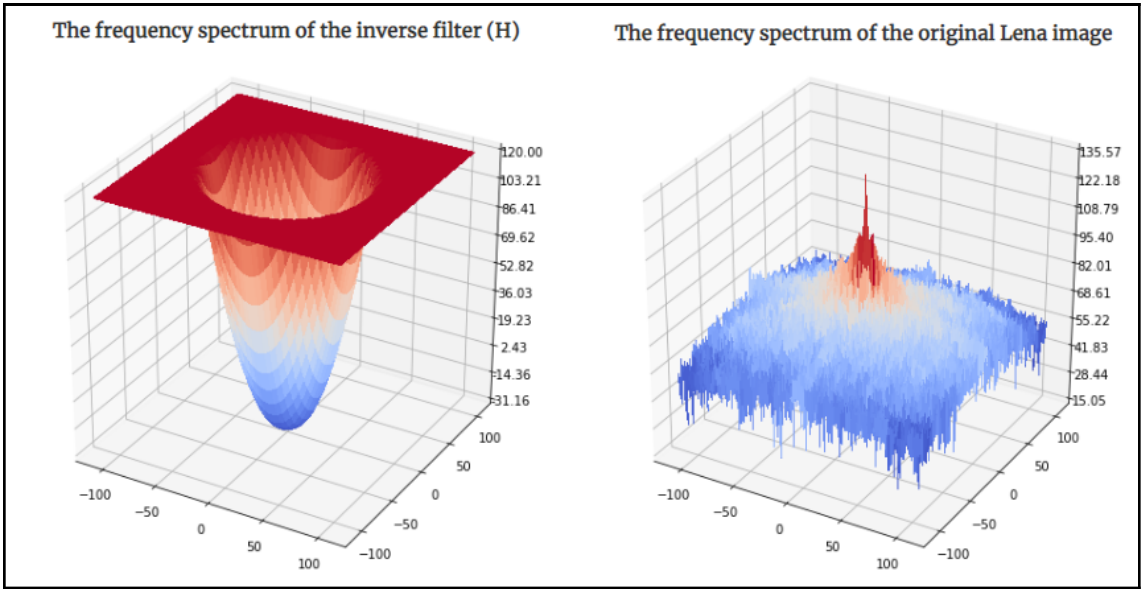

SNR
2.8
2.6
2.4
2.2
2.0
0
5
10
15
20
25
30
Cutoff Freqeuncy for LPF

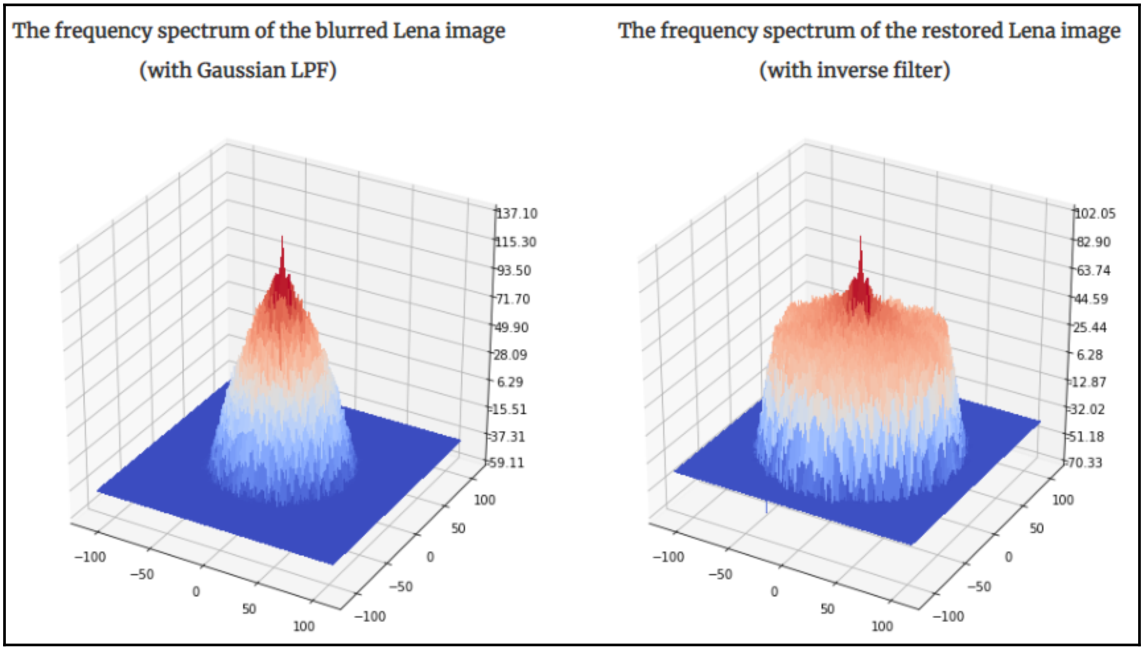

The original image
Output Image obtained with DoG band-pass filter

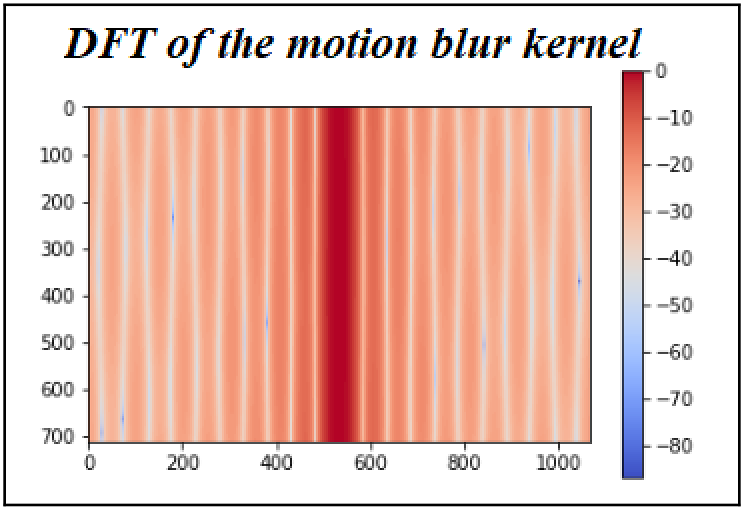

Original Image
Original Image Spectrum
Image after adding Sinusoidal Noise
Noisy Image Spectrum

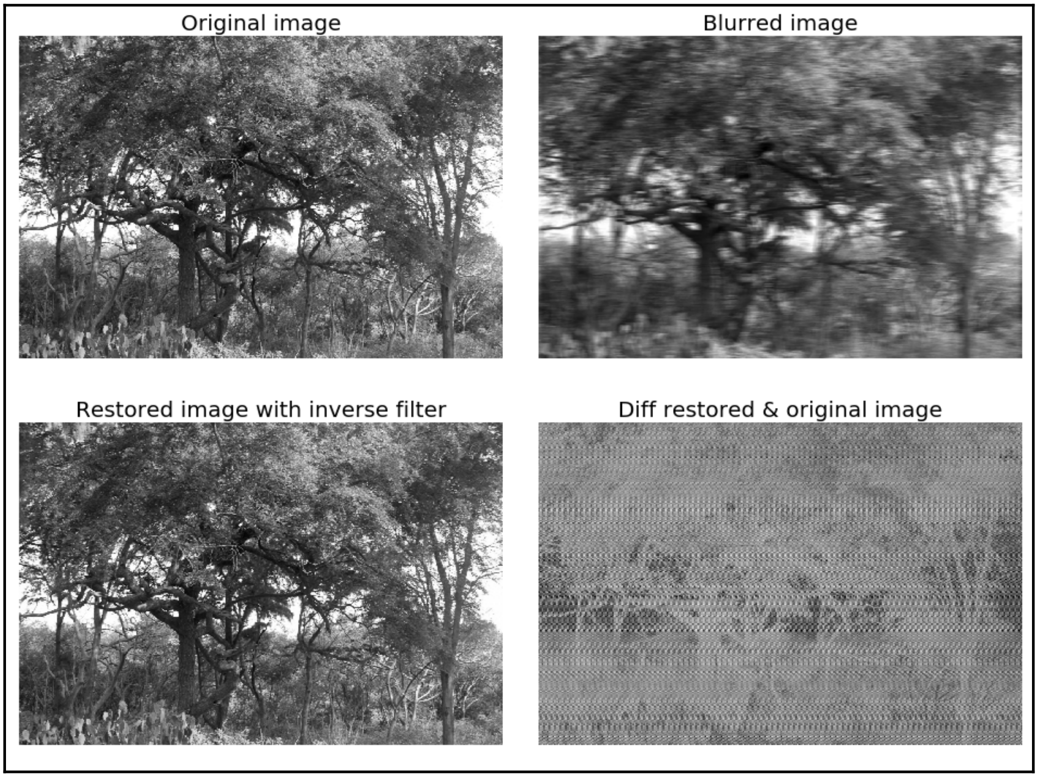

f(x,y)
degradation
g(x,y)
restoration
f̂(x,y)
original image
degraded image
estimated (restored) image
h(x,y)
degradation filter
n(x,y)
additive noise
g(x,y) = h(x,y)∗f(x,y) + n(x,y) ↔ G(u,v) = H(u,v) F(u,v) + N(u,v)
Spatial
Fourier

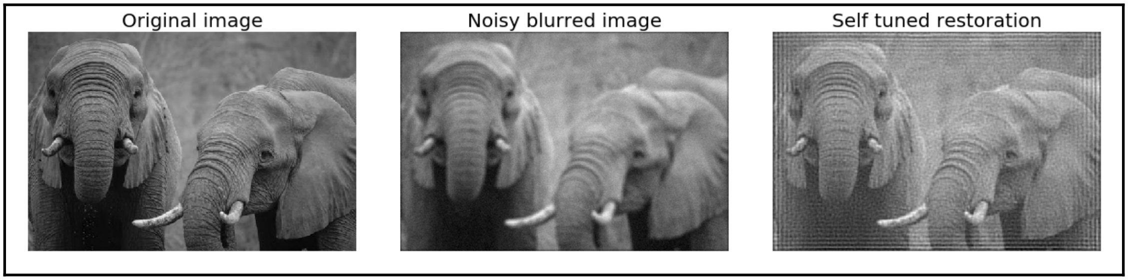

Original image
Blurred image
Restored image with inverse filter
Diff restored & original image

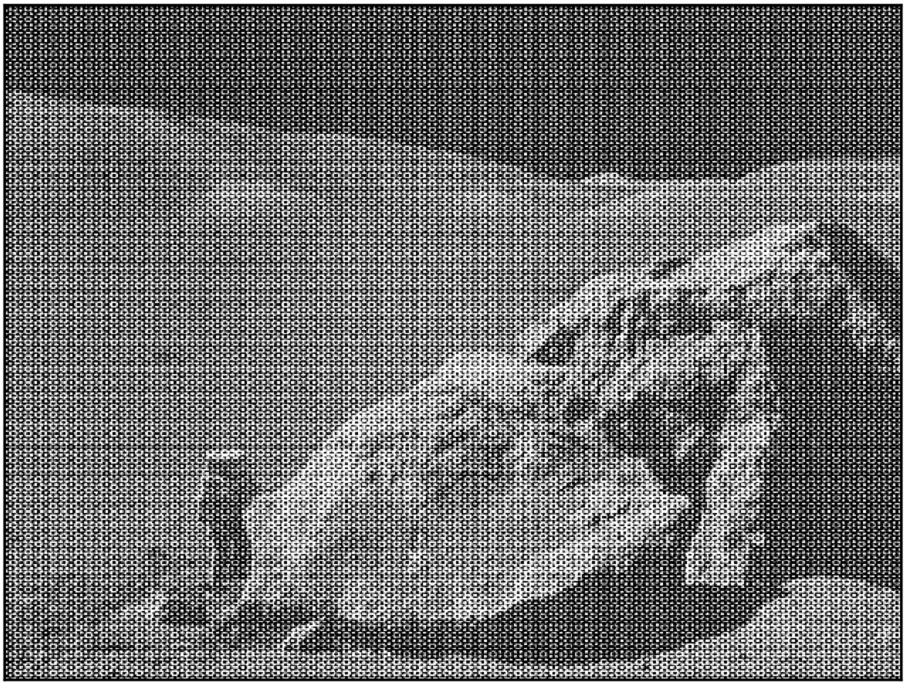

The frequency spectrum of the inverse filter (H)
120.00
103.21
86.41
69.62
52.82
36.03
19.23
2.43
-14.36
-31.16
-100
-50
0
50
100
The frequency spectrum of the original Lena image
135.57
122.18
108.79
95.40
82.01
68.61
55.22
41.83
28.44
15.05

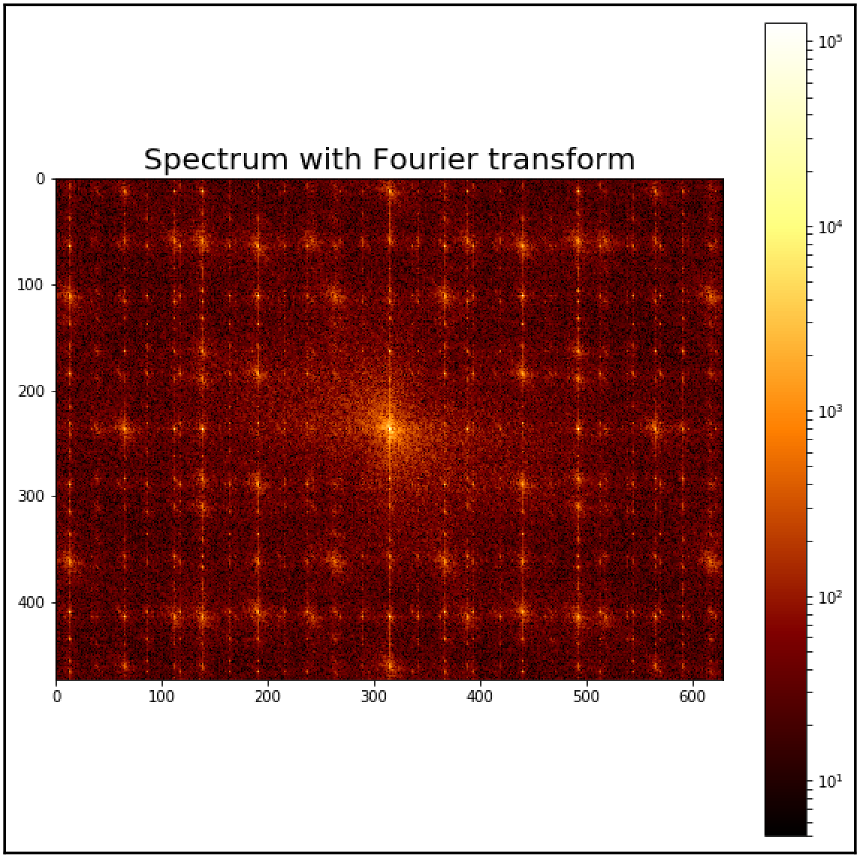

The frequency spectrum of the blurred Lena image
(with Gaussian LPF)
137.10
115.30
93.50
71.70
49.90
28.09
6.29
-15.51
-37.31
-59.11
-100
-50
0
50
100
The frequency spectrum of the restored Lena image
(with inverse filter)
102.05
82.90
63.74
44.59
25.44
6.28
-12.87
-32.02
-51.18
-70.33

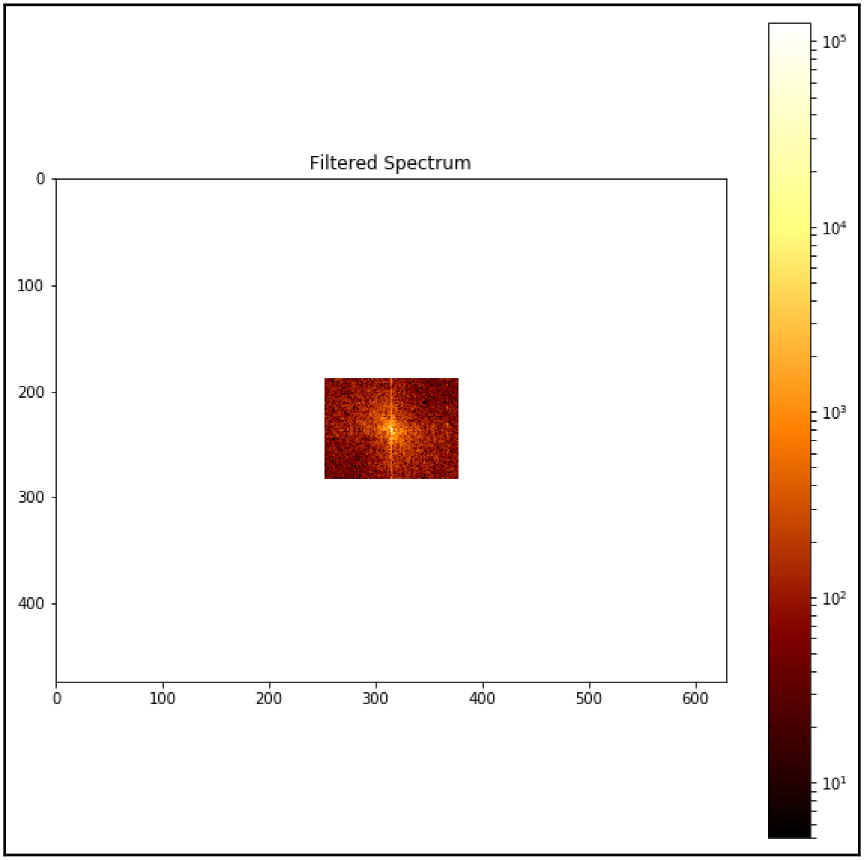

DFT of the motion blur kernel
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
0
−10
−20
−30
−40
−50
−60
−70
−80


Original image
Blurred image
Restored image with inverse filter
Diff restored & original image


Original image
Noisy blurred image
Self tuned restoration

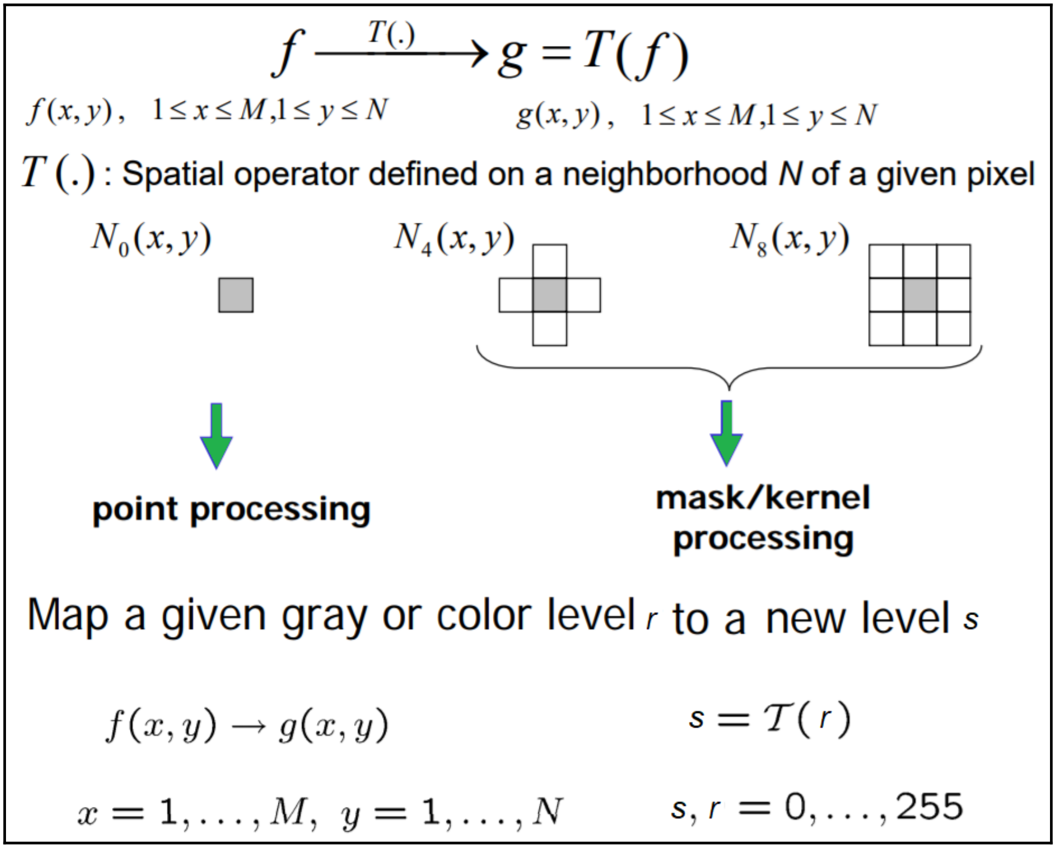


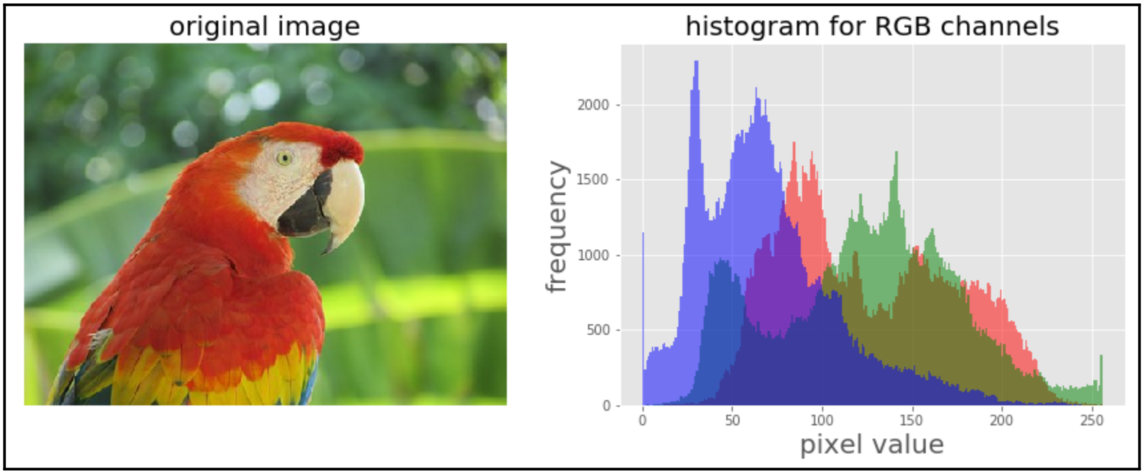

Spectrum with Fourier transform
0
100
200
300
400
0
100
200
300
400
500
600
10⁵
10⁴
10³
10²
10¹


Filtered Spectrum
0
100
200
300
400
0
100
200
300
400
500
600
10^5
10^4
10^3
10^2
10^1


Reconstructed Image

Chapter 4: Image Enhancement




original image
histogram for RGB channels
frequency
pixel value
0
500
1000
1500
2000
50
100
150
200
250

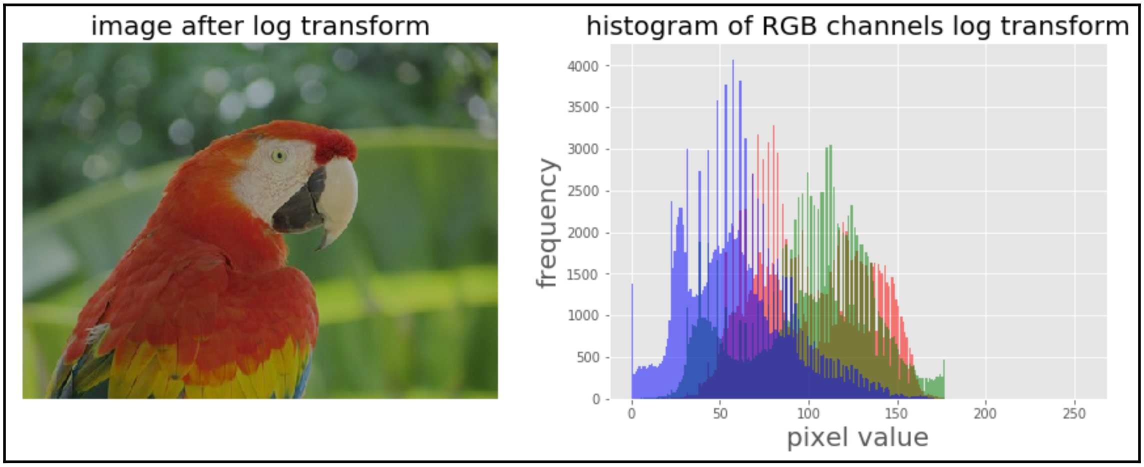
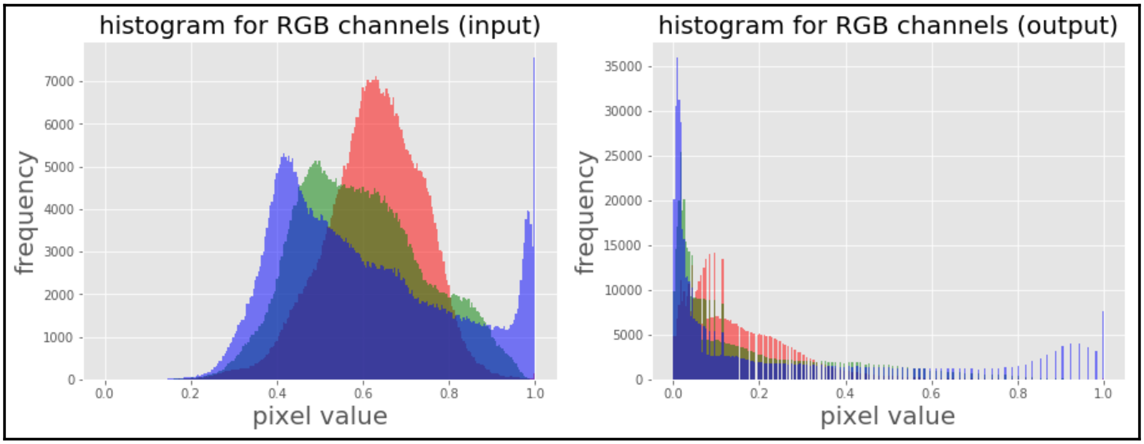
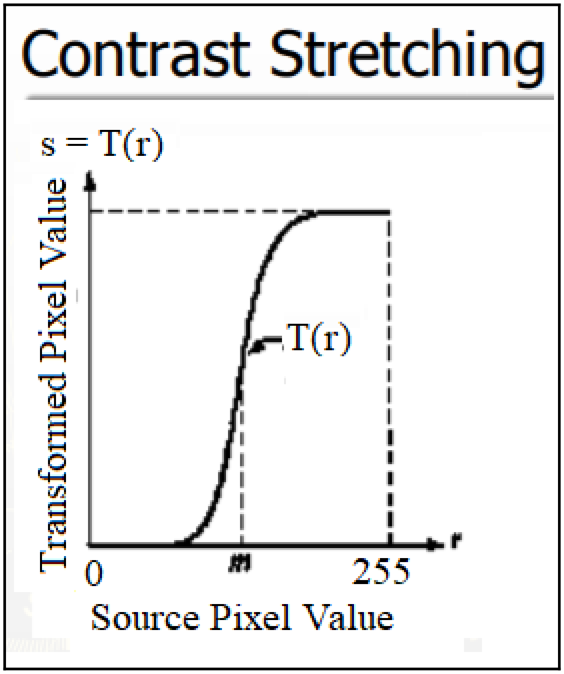
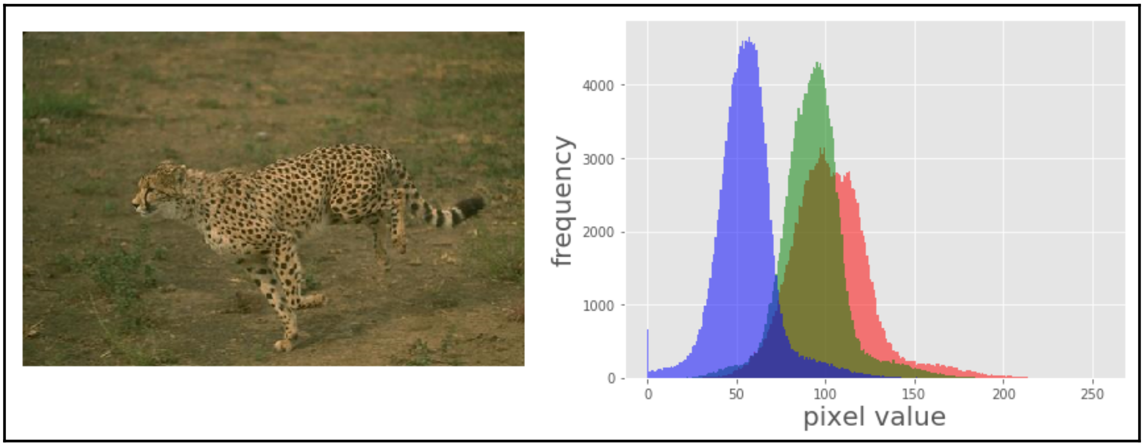
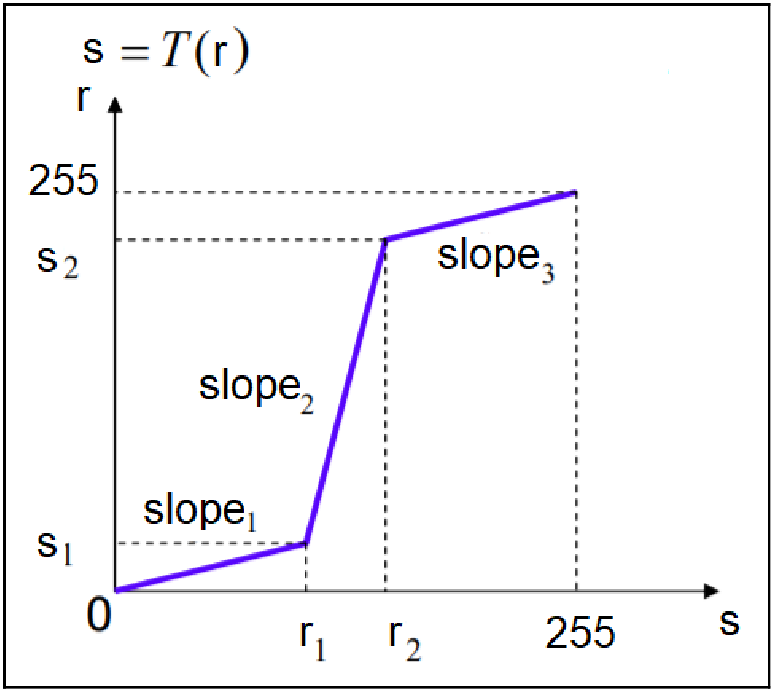


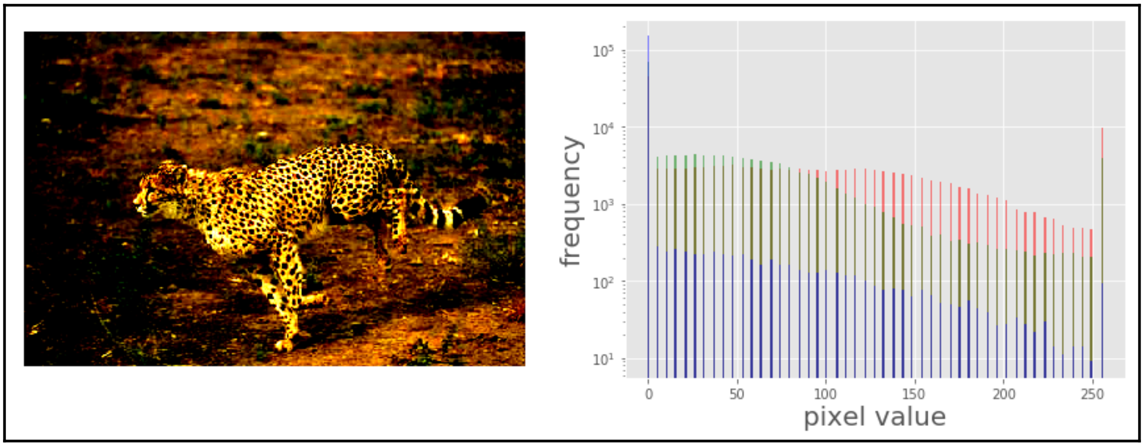
image after log transform
histogram of RGB channels log transform
frequency
pixel value
0
500
1000
1500
2000
2500
3000
3500
4000
50
100
150
200
250

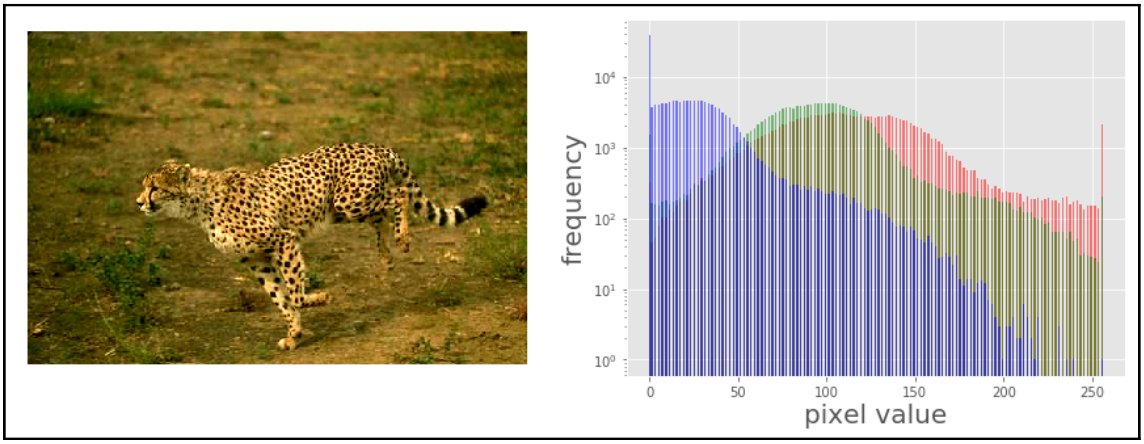

histogram for RGB channels (input)
frequency
pixel value
histogram for RGB channels (output)
frequency
pixel value


Contrast Stretching
s = T(r)
Transformed Pixel Value
T(r)
r
0
m
255
Source Pixel Value

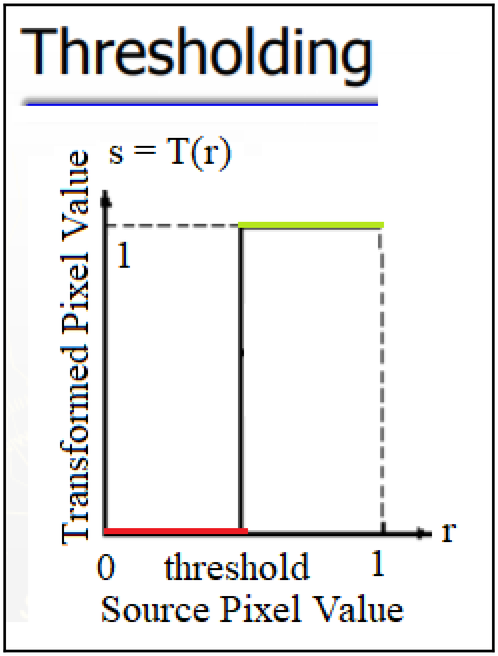

frequency
4000
3000
2000
1000
0
0
50
100
150
200
250
pixel value


s = T(r)
r
255
s2
slope3
slope2
slope1
s1
0
r1
r2
255
s

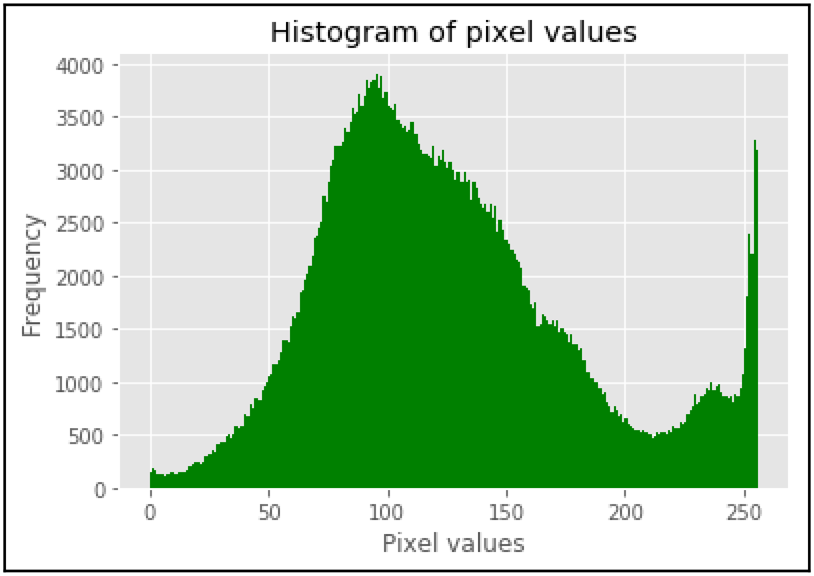

frequency
pixel value
0
50
100
150
200
250

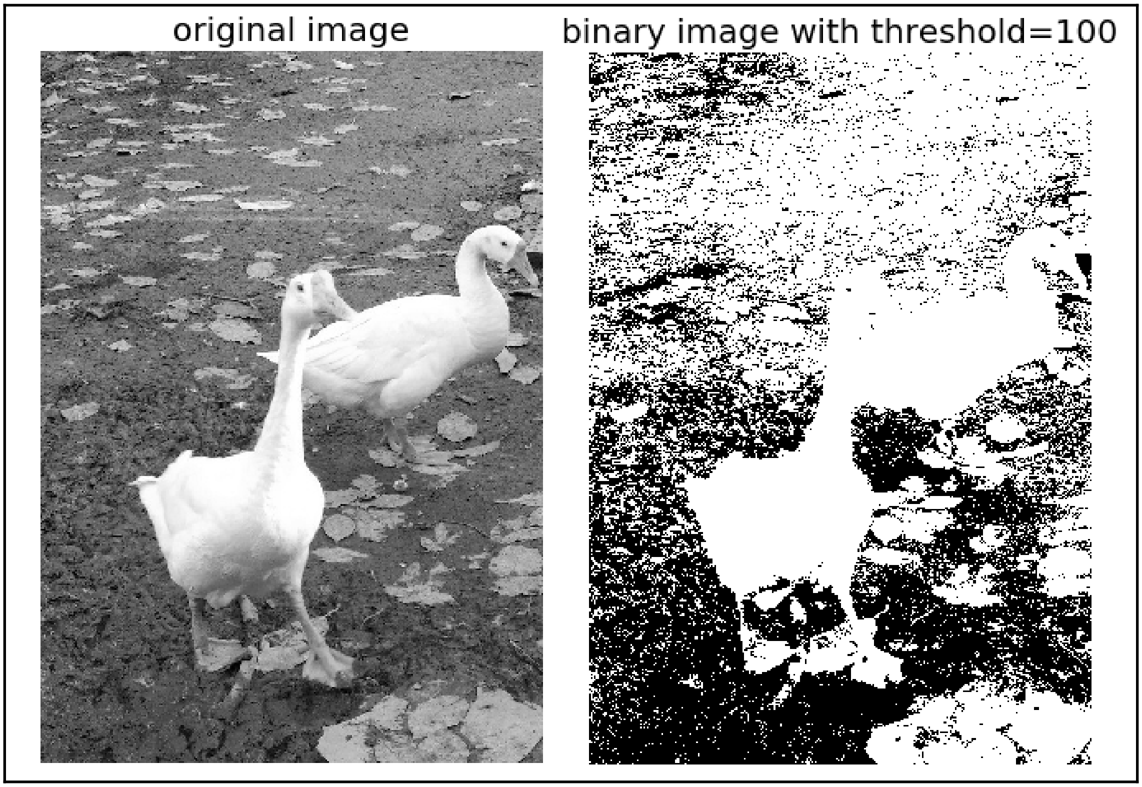


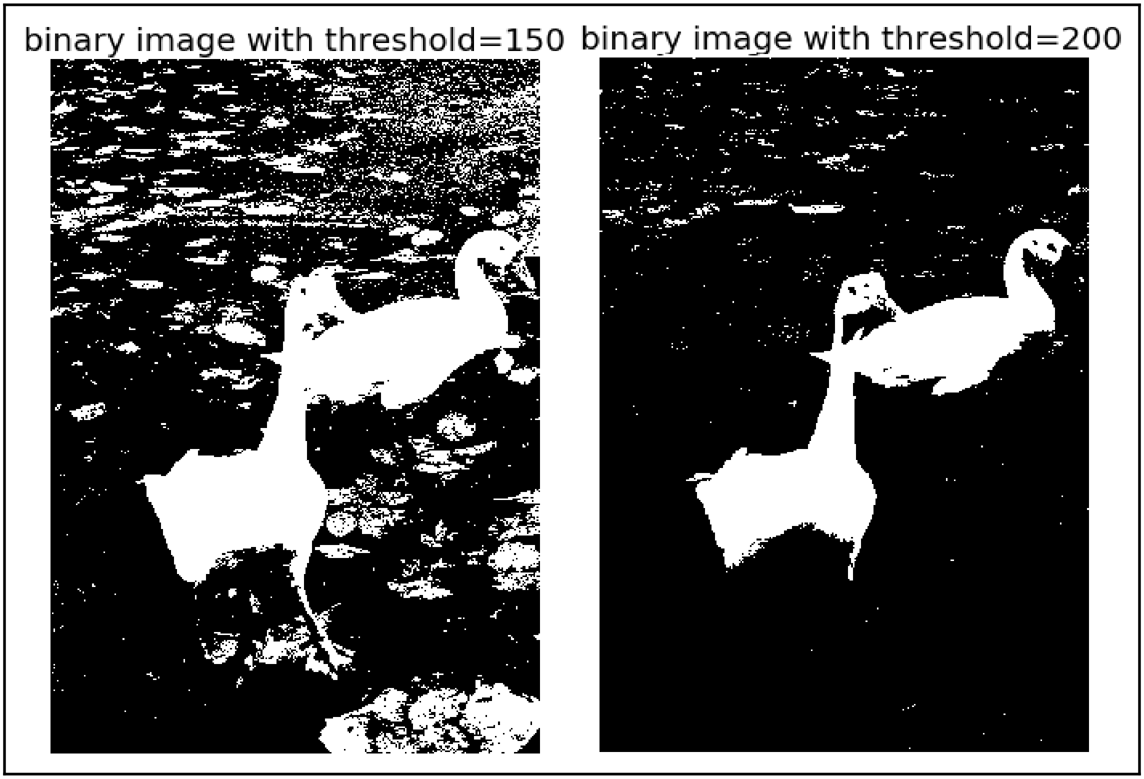
frequency
pixel value

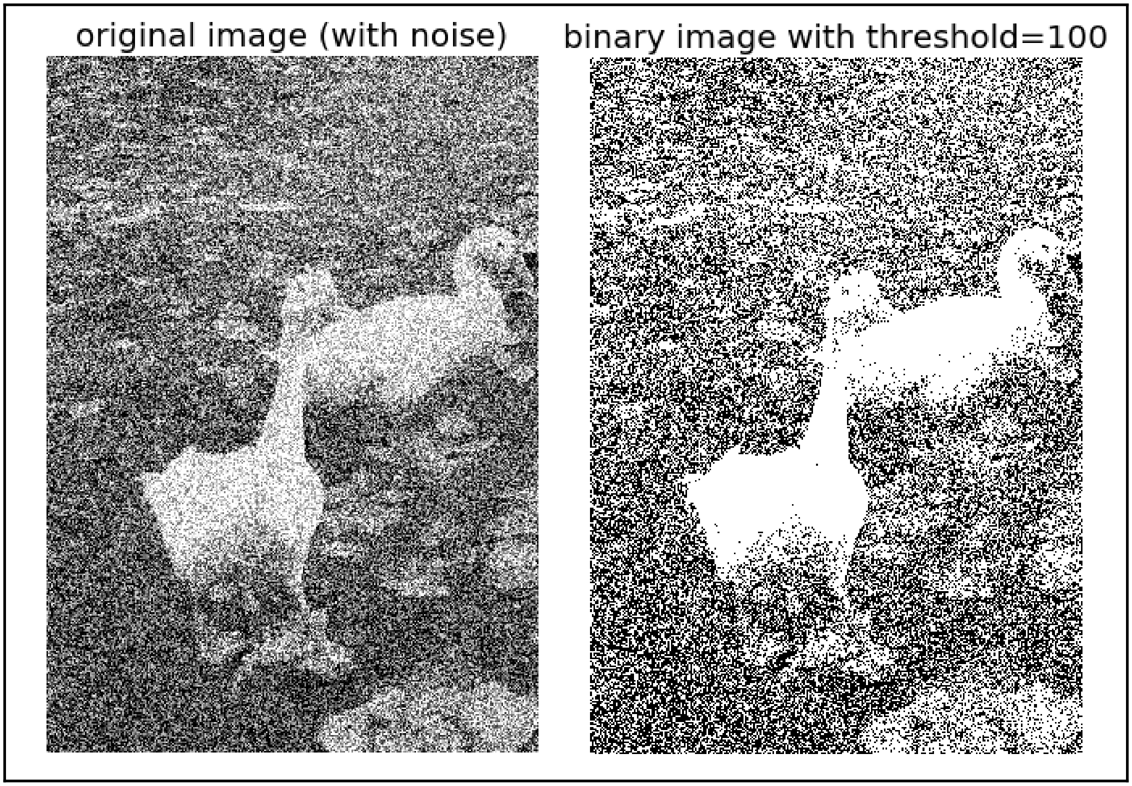


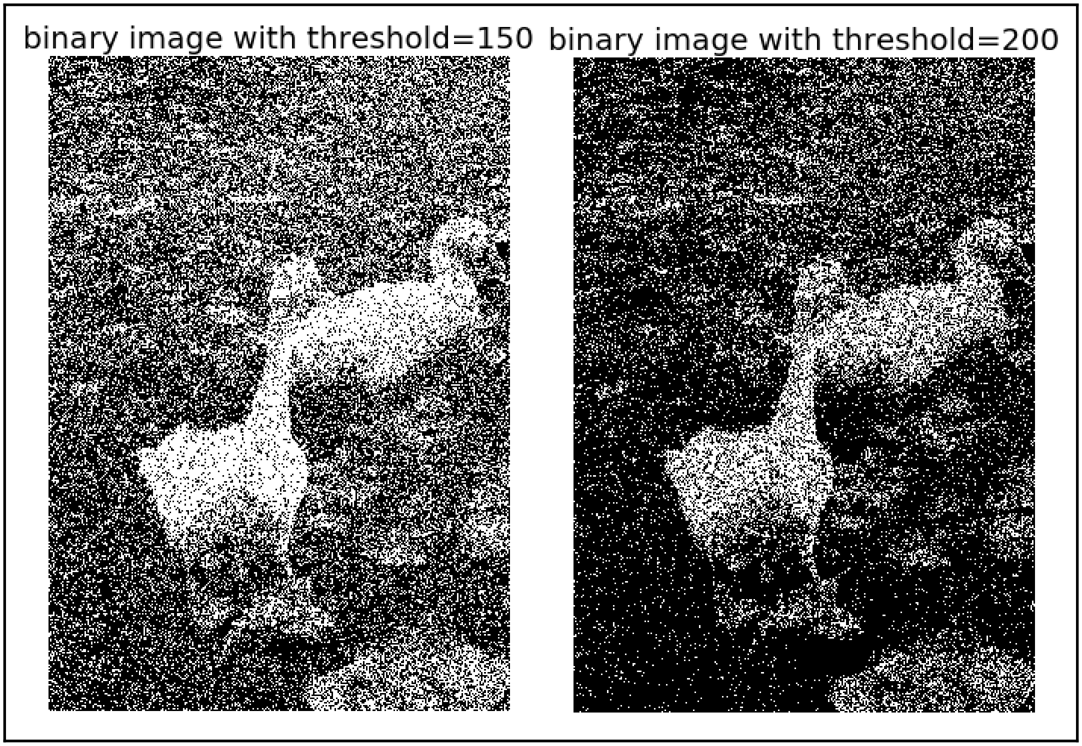
Thresholding
s = T(r)
Transformed Pixel Value
1
r
0
threshold
1
Source Pixel Value


Histogram of pixel values
Frequency
Pixel values
0
500
1000
1500
2000
2500
3000
3500
4000
0
50
100
150
200
250


original image
binary image with threshold=100

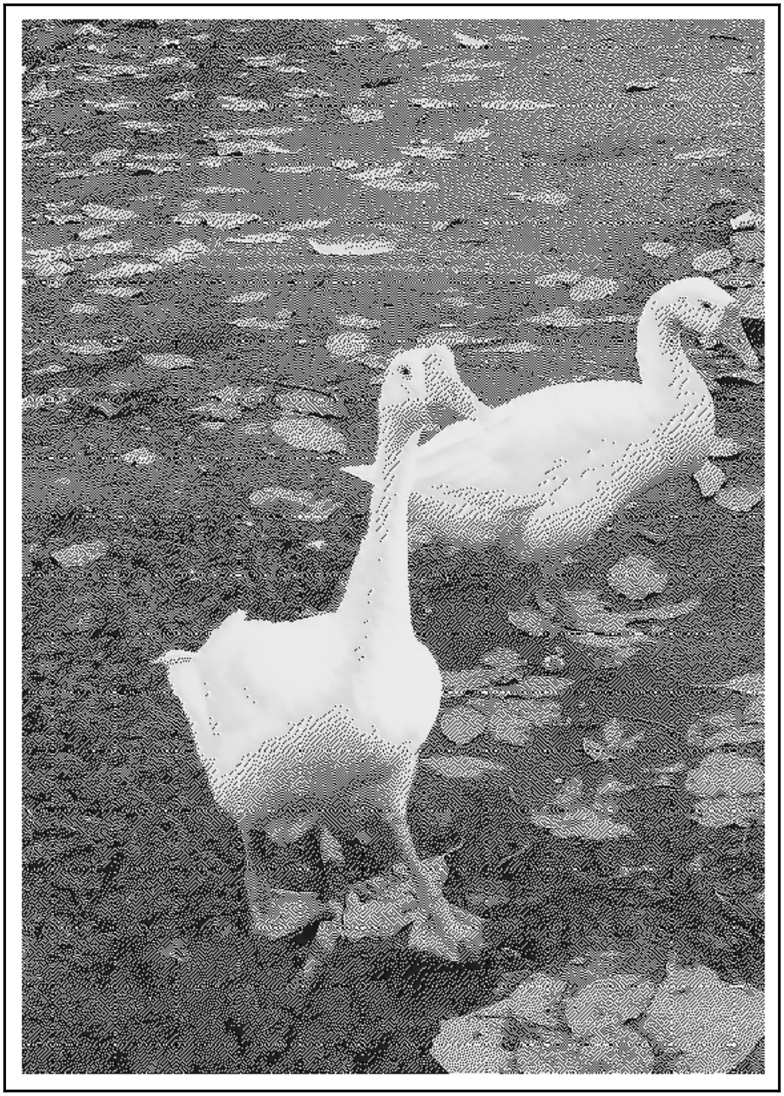

binary image with threshold=150
binary image with threshold=200


original image (with noise)
binary image with threshold=100



$$\begin{bmatrix} & & * & \frac{7}{16} & \cdots \\ \cdots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \cdots \end{bmatrix}$$

```
for each y from top to bottom
   for each x from left to right
      oldpixel  := pixel[x][y]
      newpixel  := find_closest_palette_color(oldpixel)
      pixel[x][y]  := newpixel
      quant_error  := oldpixel - newpixel
      pixel[x + 1][y    ] := pixel[x + 1][y    ] + quant_error * 7 / 16
      pixel[x - 1][y + 1] := pixel[x - 1][y + 1] + quant_error * 3 / 16
      pixel[x    ][y + 1] := pixel[x    ][y + 1] + quant_error * 5 / 16
      pixel[x + 1][y + 1] := pixel[x + 1][y + 1] + quant_error * 1 / 16
```



$$s_k = T(r_k) = \sum_{j=0}^{k} P_r(r_j) = \sum_{j=0}^{k} n_j / N$$

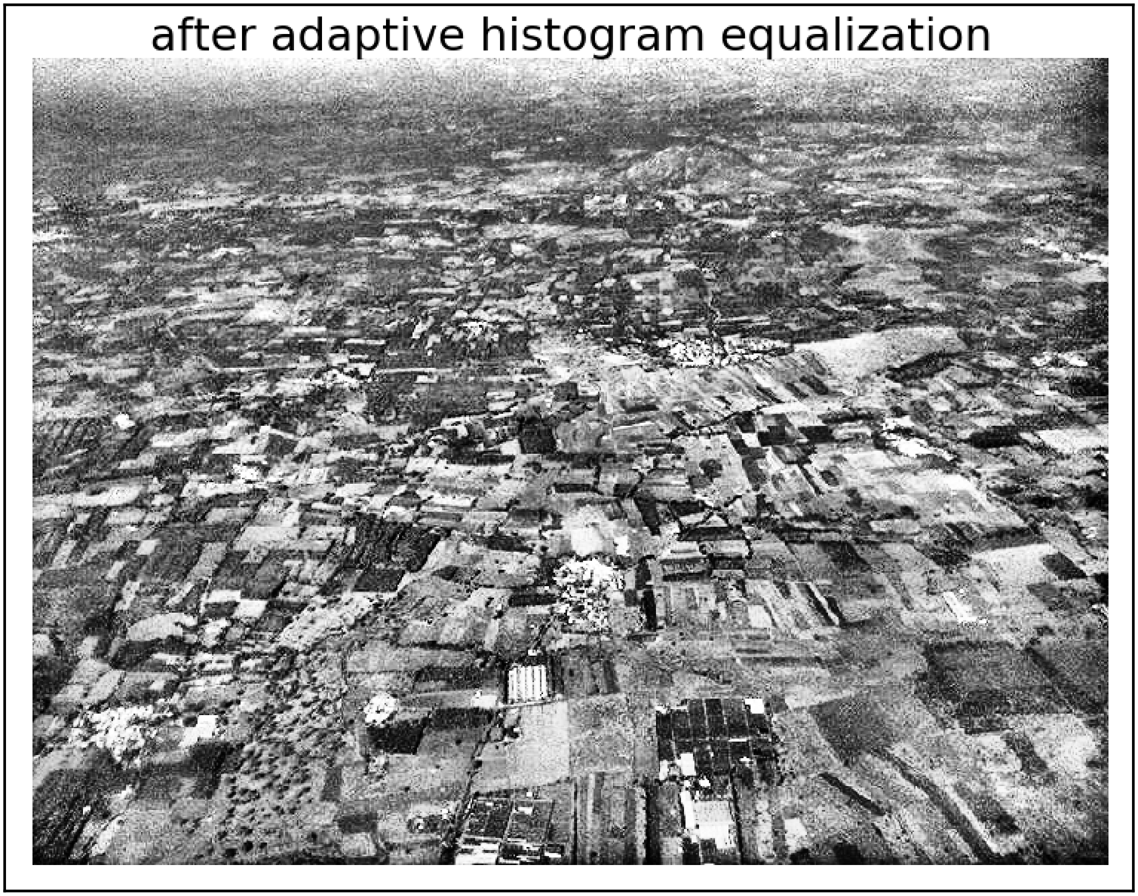
$$0 \le r_k \le 1, \qquad k = 0,1,2,\ldots.,255$$

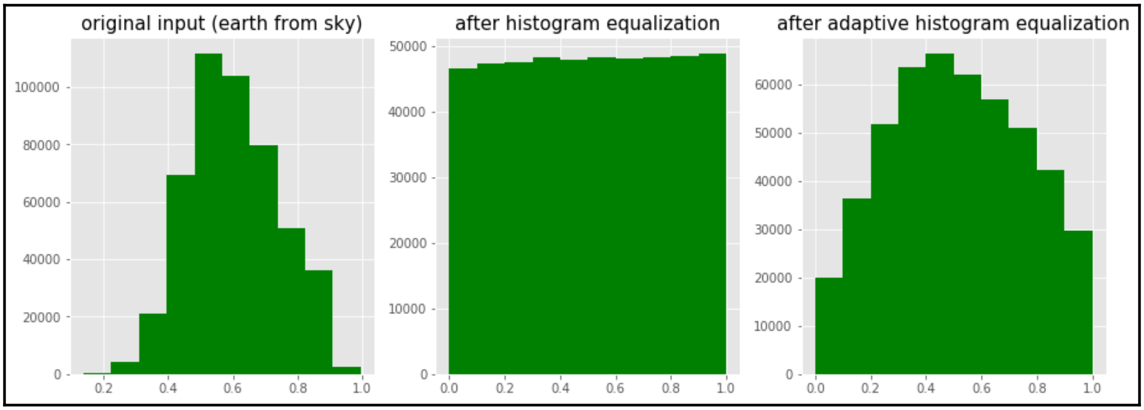
N: total number of pixels

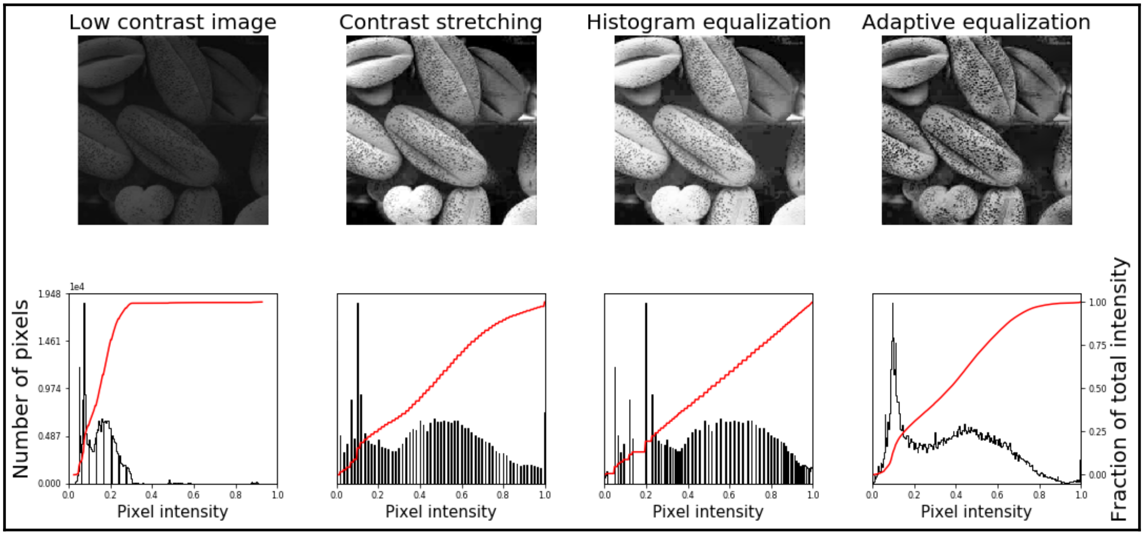
n_j: frequency of pixel with gray-level j

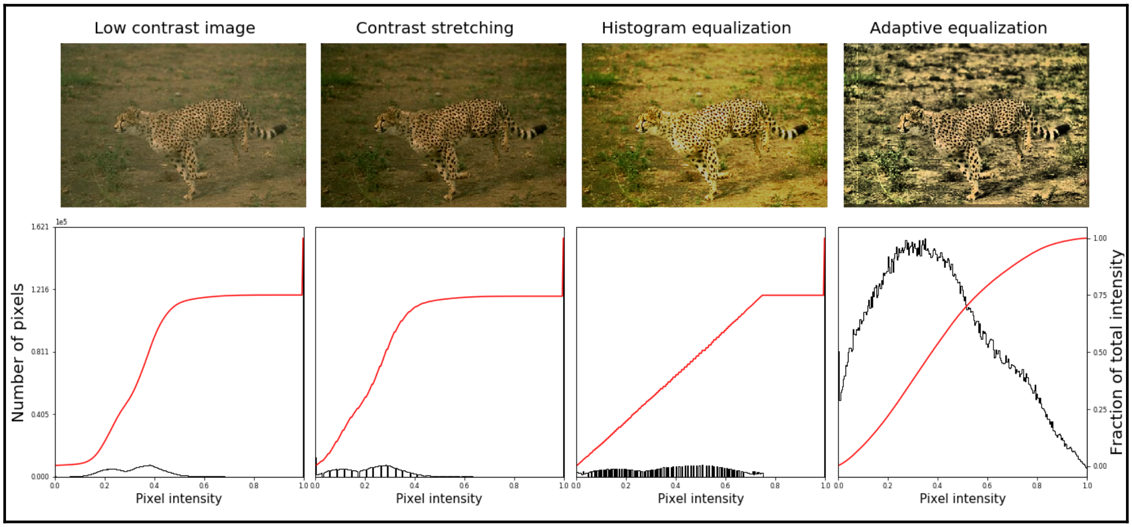
original input (earth from sky)

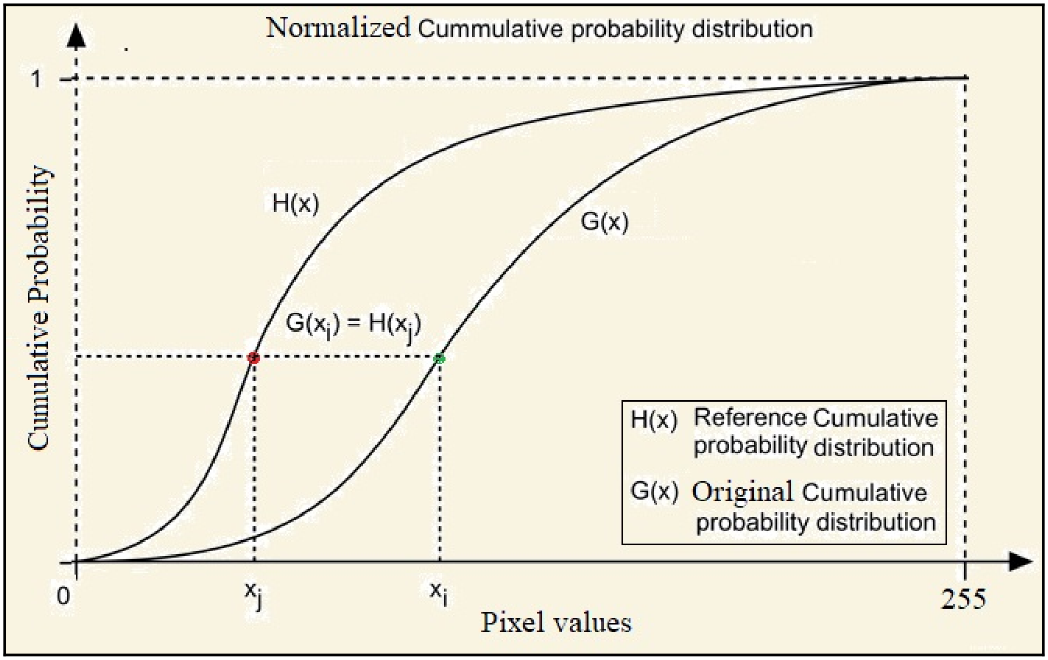

after histogram equalization

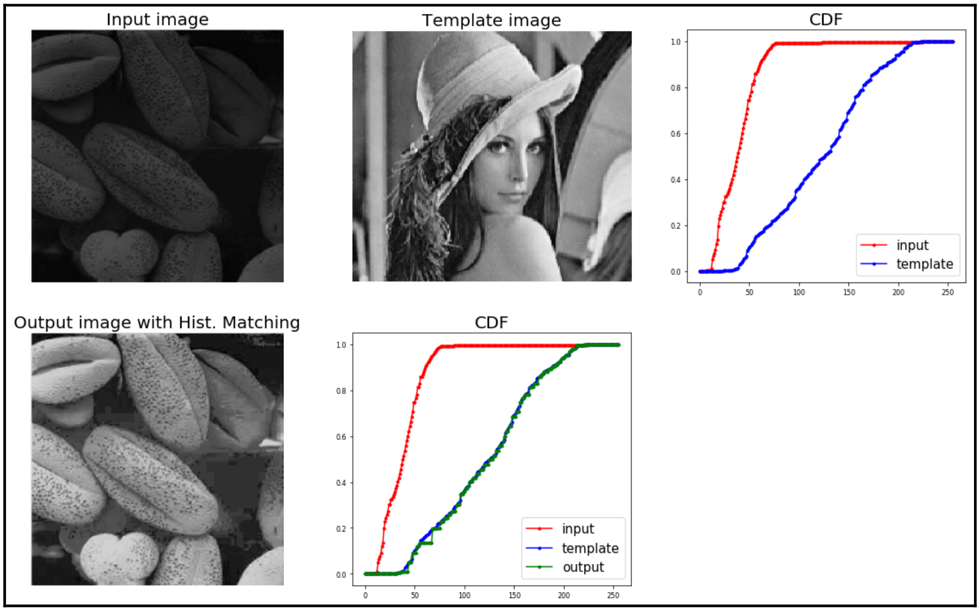

after adaptive histogram equalization


original input (earth from sky)
after histogram equalization
after adaptive histogram equalization
100000
80000
60000
40000
20000
0
0.2
0.4
0.6
0.8
1.0
50000
40000
30000
20000
10000
0.0
60000

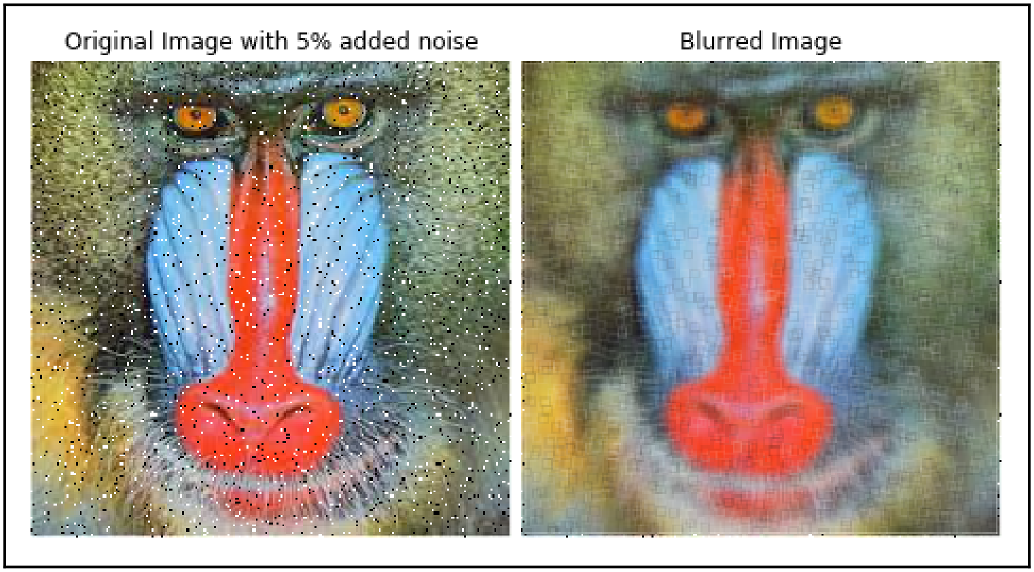

Low contrast image
Contrast stretching
Histogram equalization
Adaptive equalization
Number of pixels
Pixel intensity
Fraction of total intensity

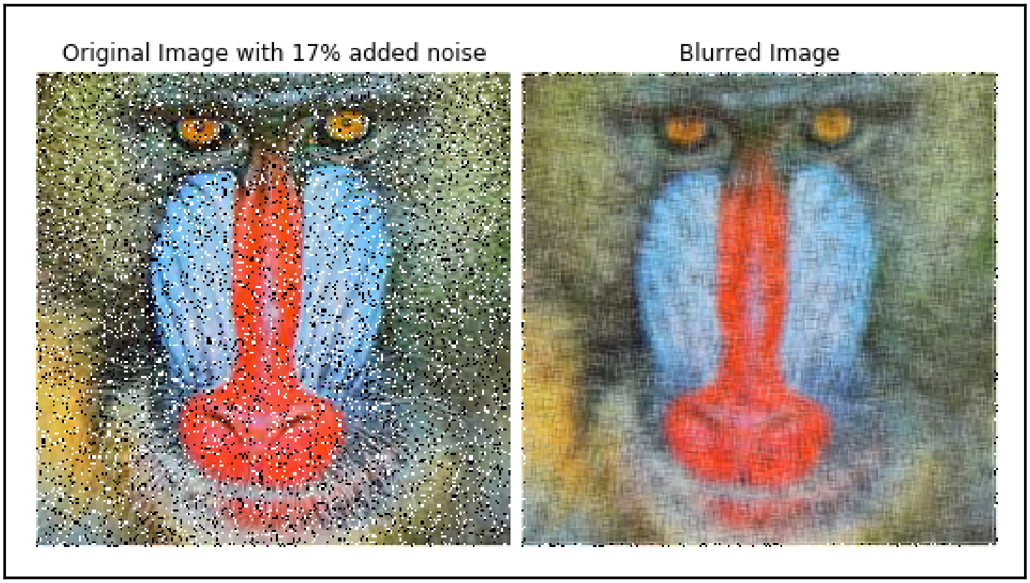

Low contrast image
Contrast stretching
Histogram equalization
Adaptive equalization
Number of pixels
Pixel intensity
Fraction of total intensity

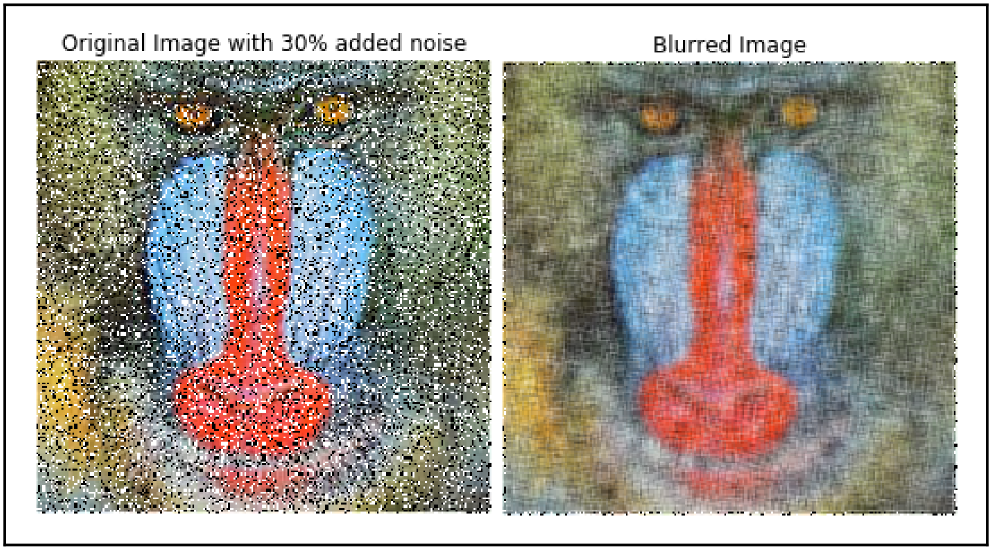

Normalized Cummulative probability distribution
Cumulative Probability
1
H(x)
G(x)
$G(x_i) = H(x_j)$
H(x) Reference Cumulative probability distribution
G(x) Original Cumulative probability distribution
0
x_j
x_i
255
Pixel values

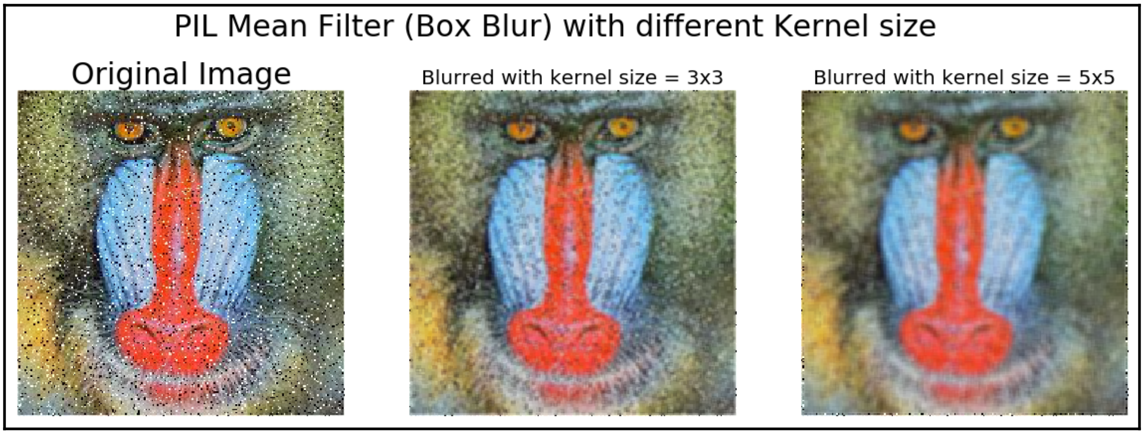

Input image
Template image
CDF
input
template
Output image with Hist. Matching
CDF
input
template
output

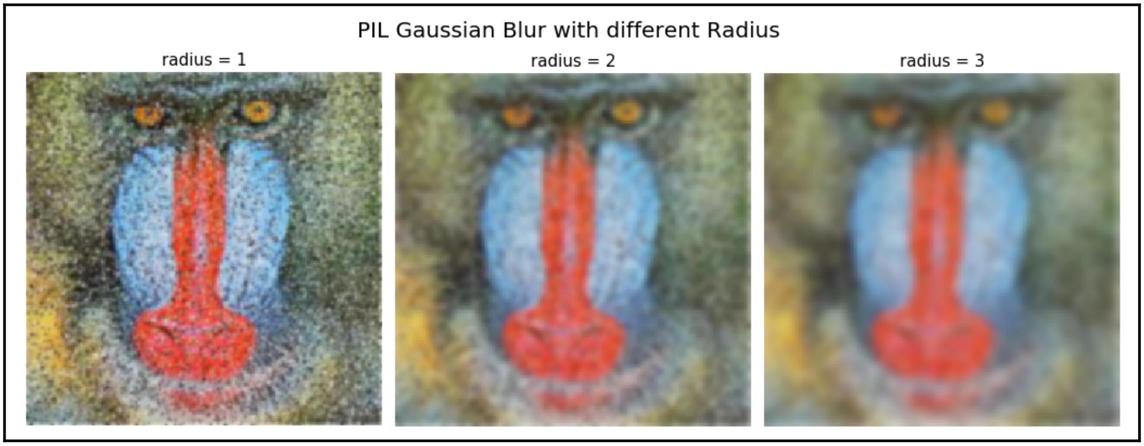

Input Image
Template Image

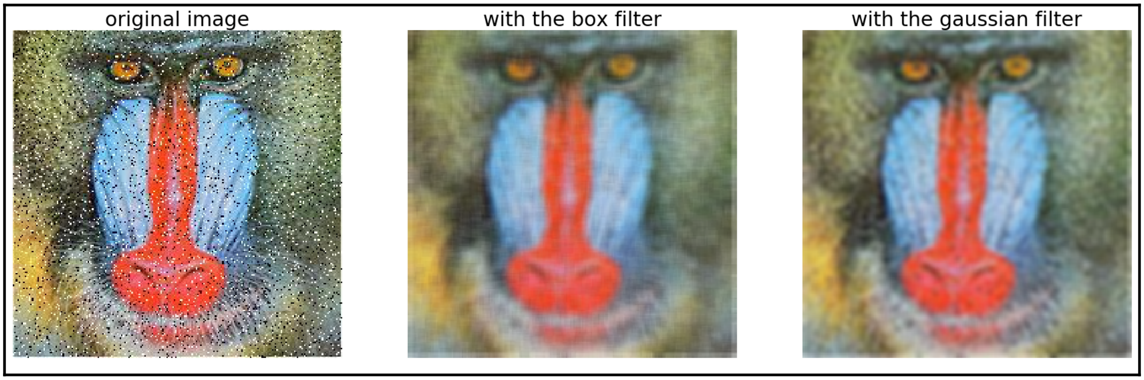

Original Image with 5% added noise
Blurred Image

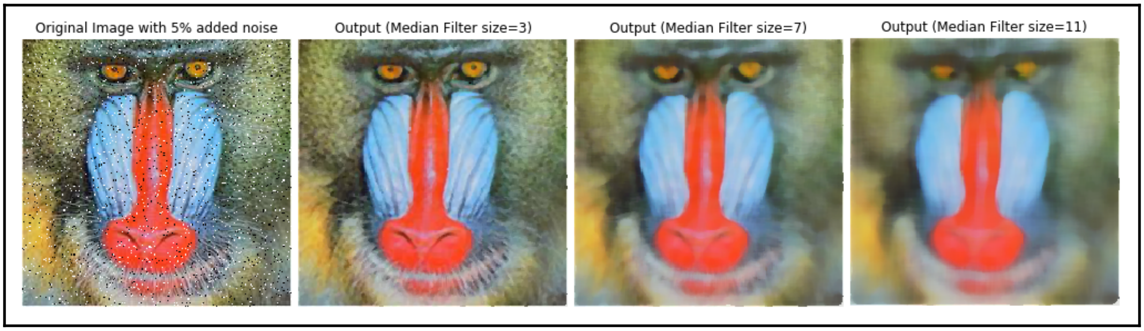

Original Image with 17% added noise
Blurred Image

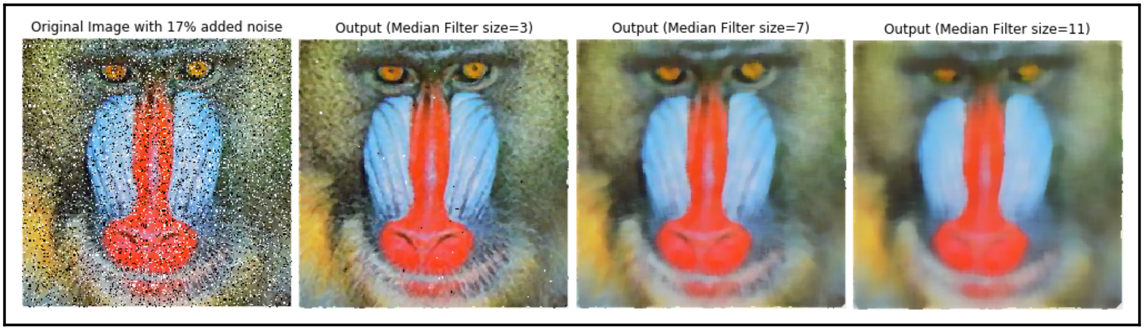

Original Image with 30% added noise
Blurred Image

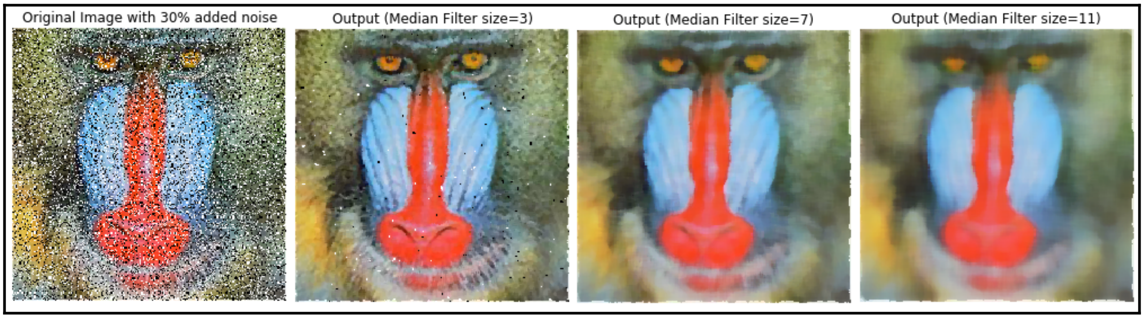

PIL Mean Filter (Box Blur) with different Kernel size
Original Image
Blurred with kernel size = 3x3
Blurred with kernel size = 5x5

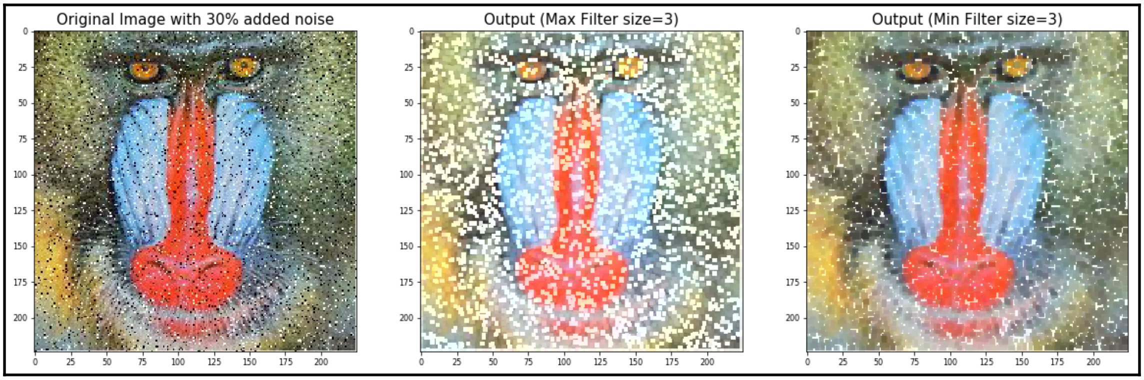

PIL Gaussian Blur with different Radius
radius = 1
radius = 2
radius = 3

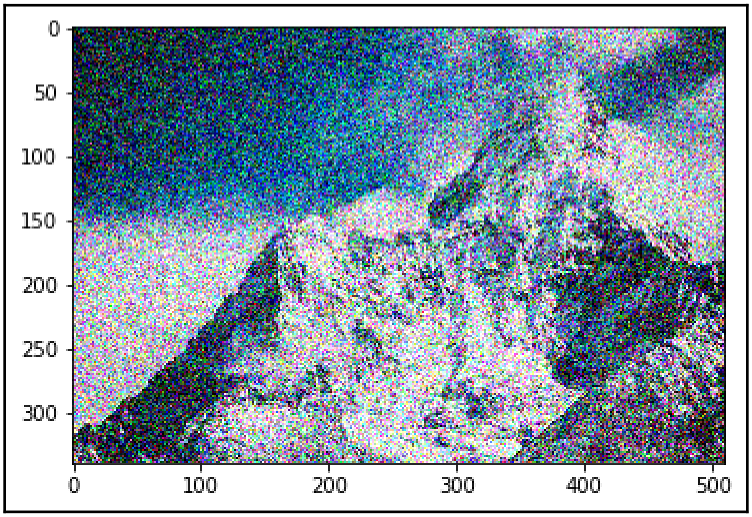

original image
with the box filter
with the gaussian filter

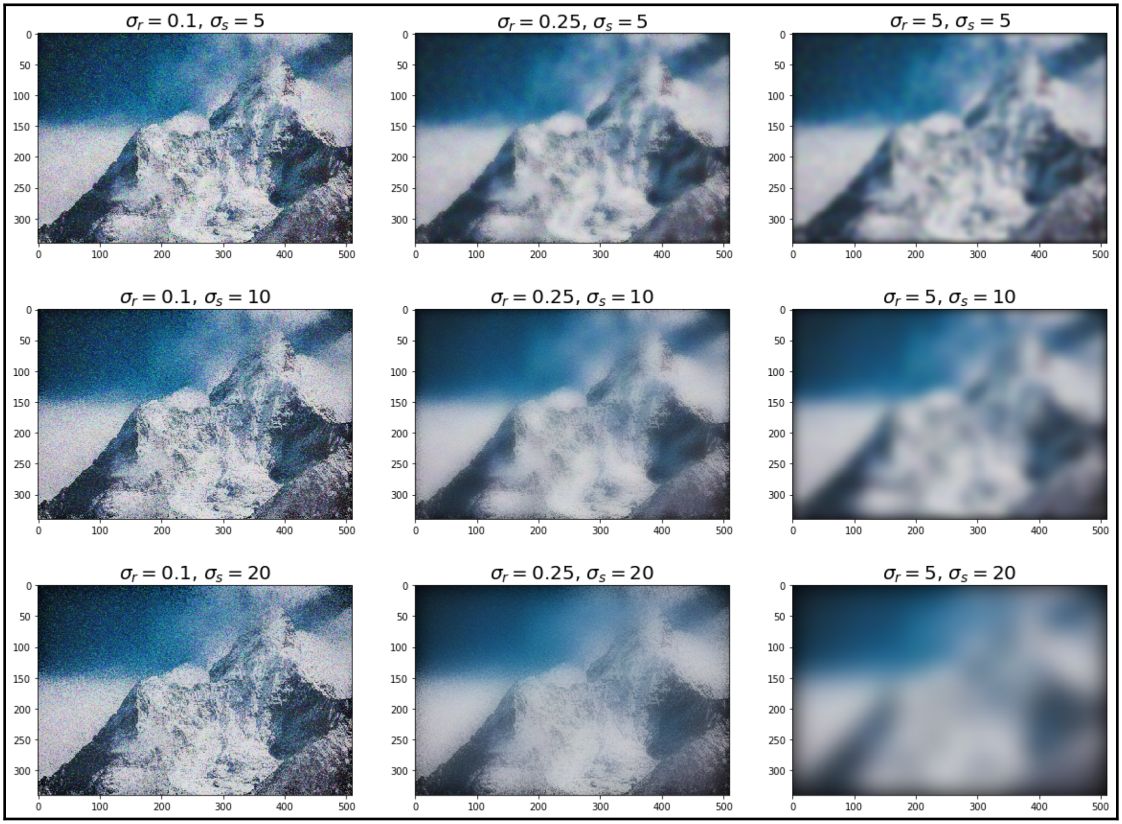

Original Image with 5% added noise
Output (Median Filter size=3)
Output (Median Filter size=7)
Output (Median Filter size=11)

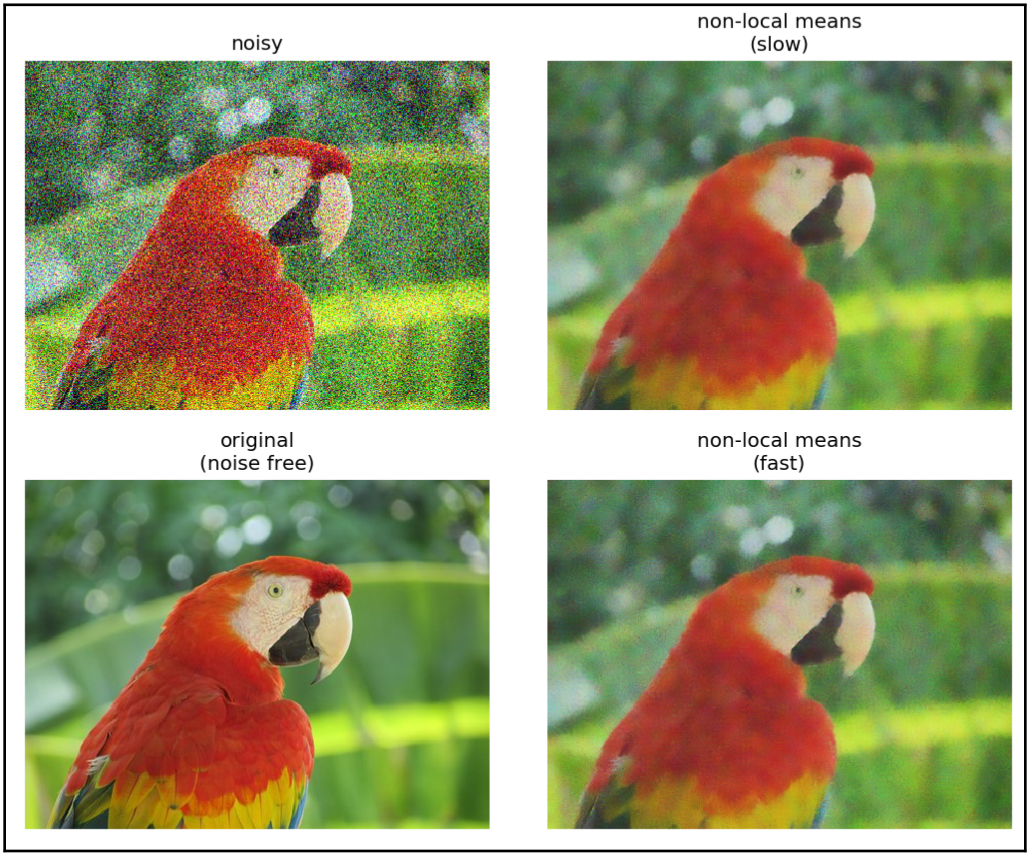

Original Image with 17% added noise
Output (Median Filter size=3)
Output (Median Filter size=7)
Output (Median Filter size=11)

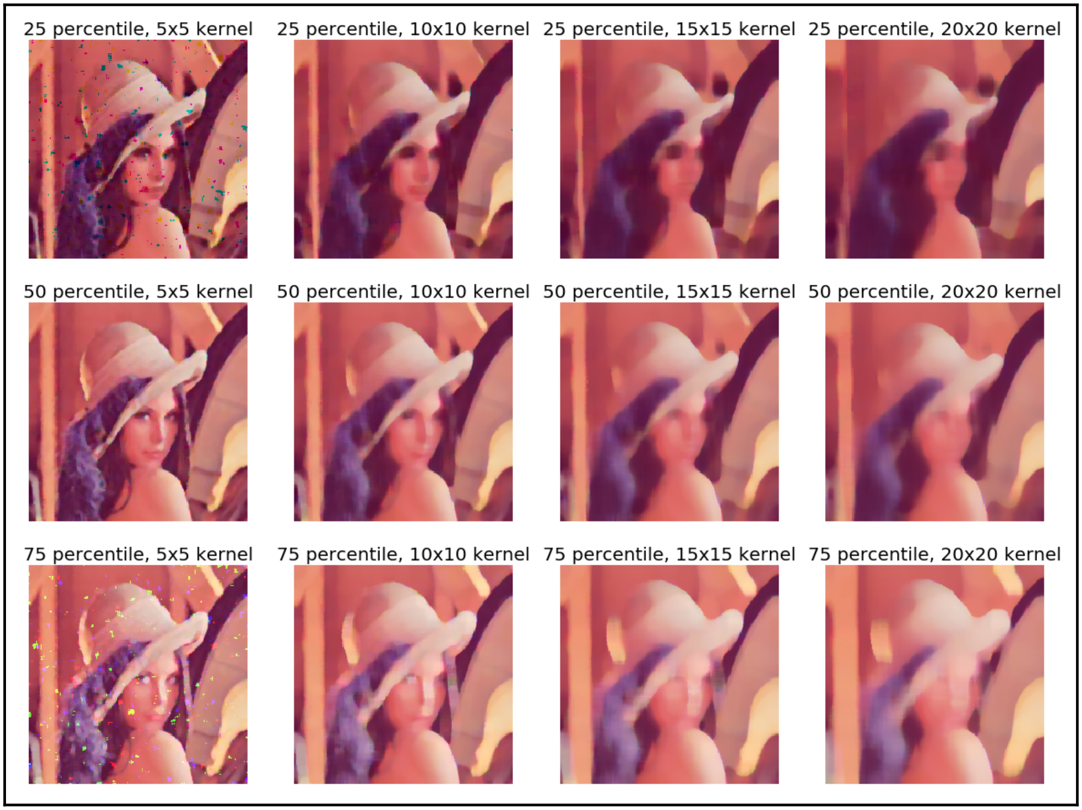

Original Image with 30% added noise
Output (Median Filter size=3)
Output (Median Filter size=7)
Output (Median Filter size=11)


Original Image with 30% added noise
Output (Max Filter size=3)
Output (Min Filter size=3)

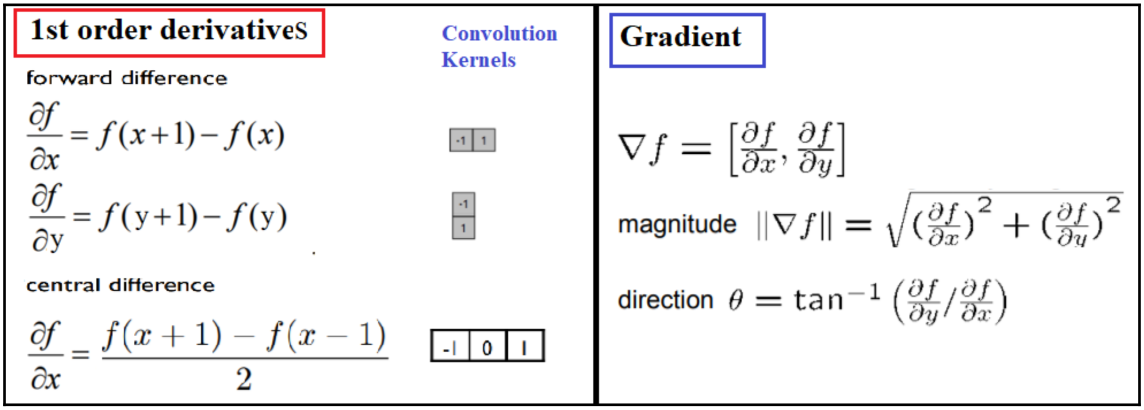

0
50
100
150
200
250
300
0
100
200
300
400
500

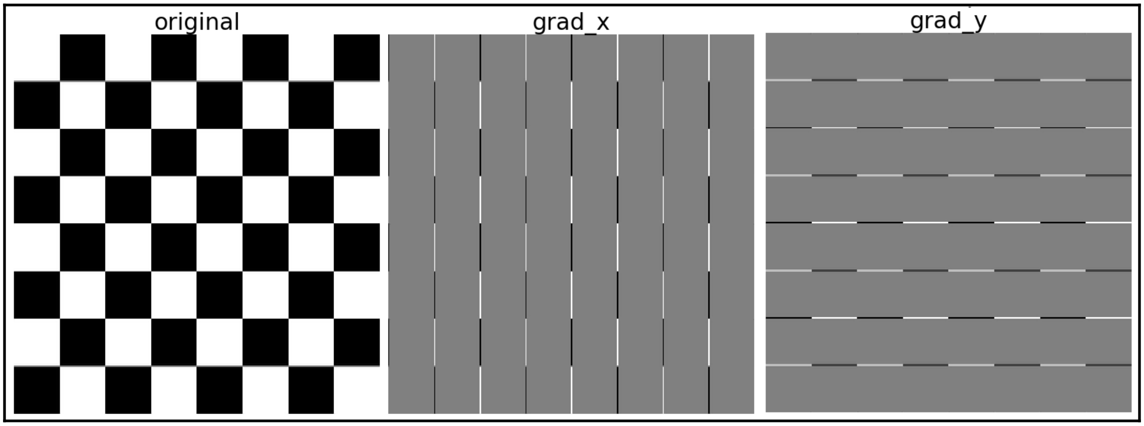

$\sigma_r = 0.1$, $\sigma_s = 5$
$\sigma_r = 0.25$, $\sigma_s = 5$
$\sigma_r = 5$, $\sigma_s = 5$
$\sigma_r = 0.1$, $\sigma_s = 10$
$\sigma_r = 0.25$, $\sigma_s = 10$
$\sigma_r = 5$, $\sigma_s = 10$
$\sigma_r = 0.1$, $\sigma_s = 20$
$\sigma_r = 0.25$, $\sigma_s = 20$
$\sigma_r = 5$, $\sigma_s = 20$

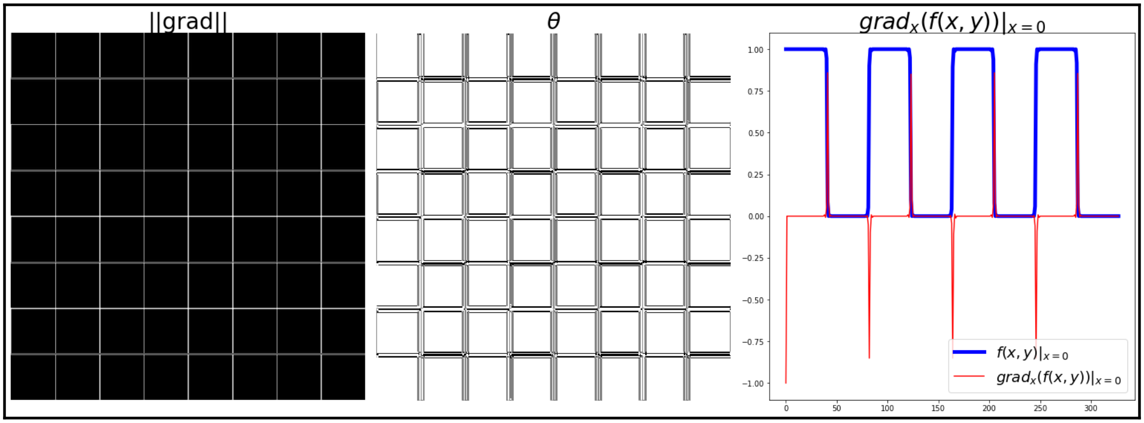

noisy
non-local means
(slow)
original
(noise free)
non-local means
(fast)

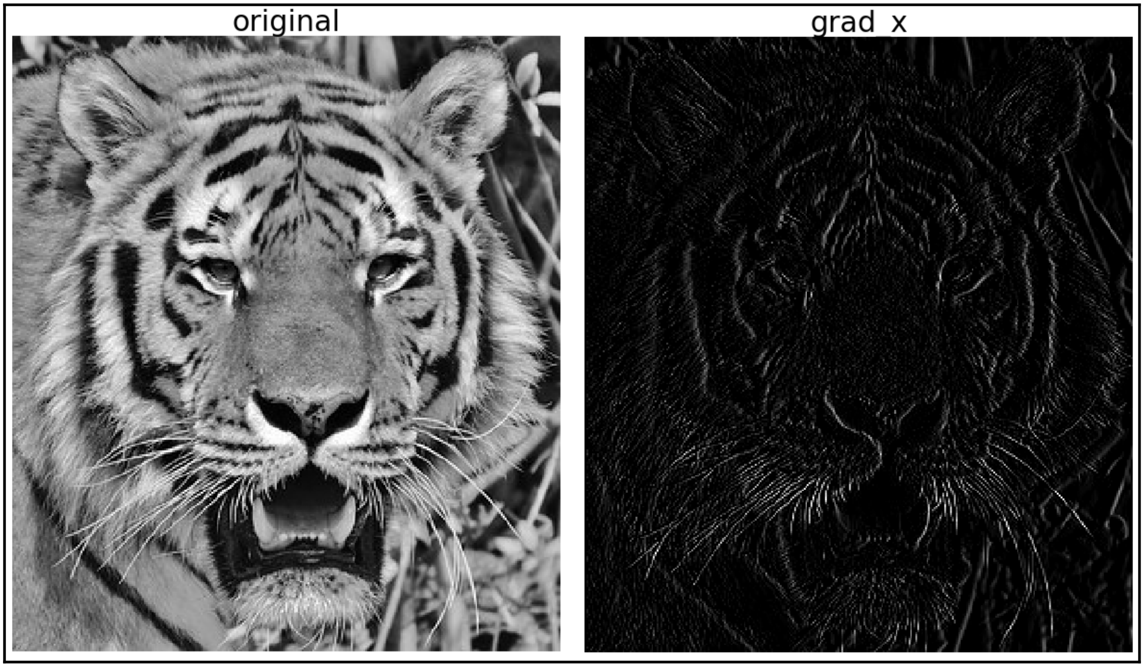

25 percentile, 5x5 kernel
25 percentile, 10x10 kernel
25 percentile, 15x15 kernel
25 percentile, 20x20 kernel
50 percentile, 5x5 kernel
50 percentile, 10x10 kernel
50 percentile, 15x15 kernel
50 percentile, 20x20 kernel
75 percentile, 5x5 kernel
75 percentile, 10x10 kernel
75 percentile, 15x15 kernel
75 percentile, 20x20 kernel

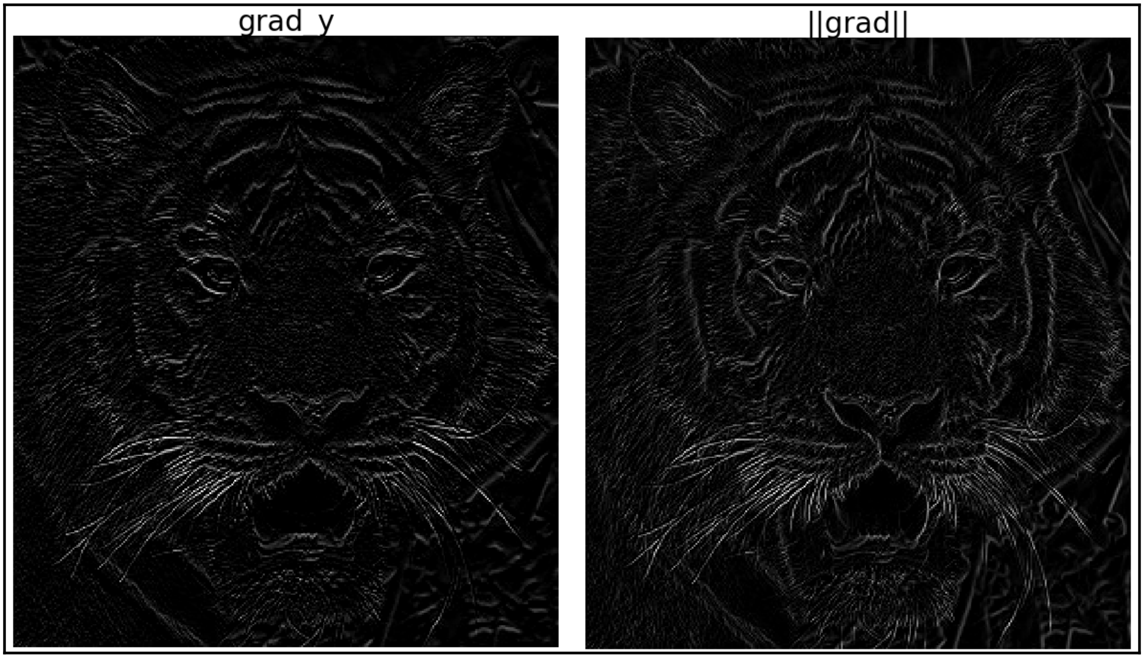
Chapter 5: Image Enhancement Using Derivatives

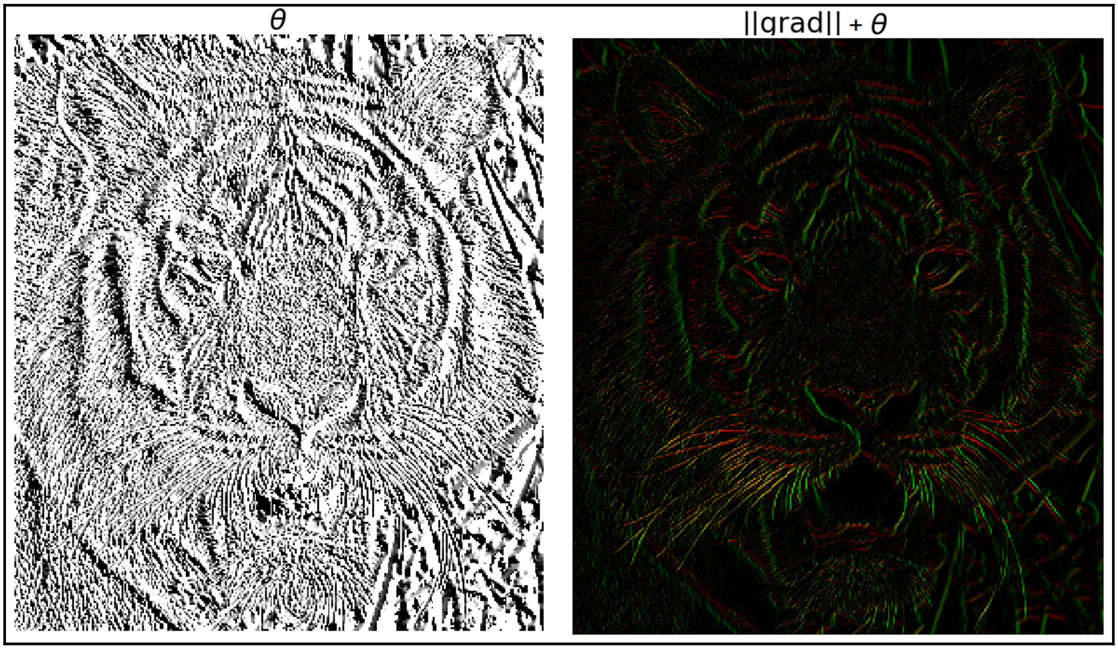





||grad||
θ
$grad_x(f(x, y))|_{x=0}$
$f(x, y)|_{x=0}$
$grad_x(f(x, y))|_{x=0}$


original
grad x


grad_y
||grad||


θ
||grad|| + θ

2nd order derivative

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

$$\frac{\partial^2 f}{\partial y^2} = f(y+1) + f(y-1) - 2f(y)$$

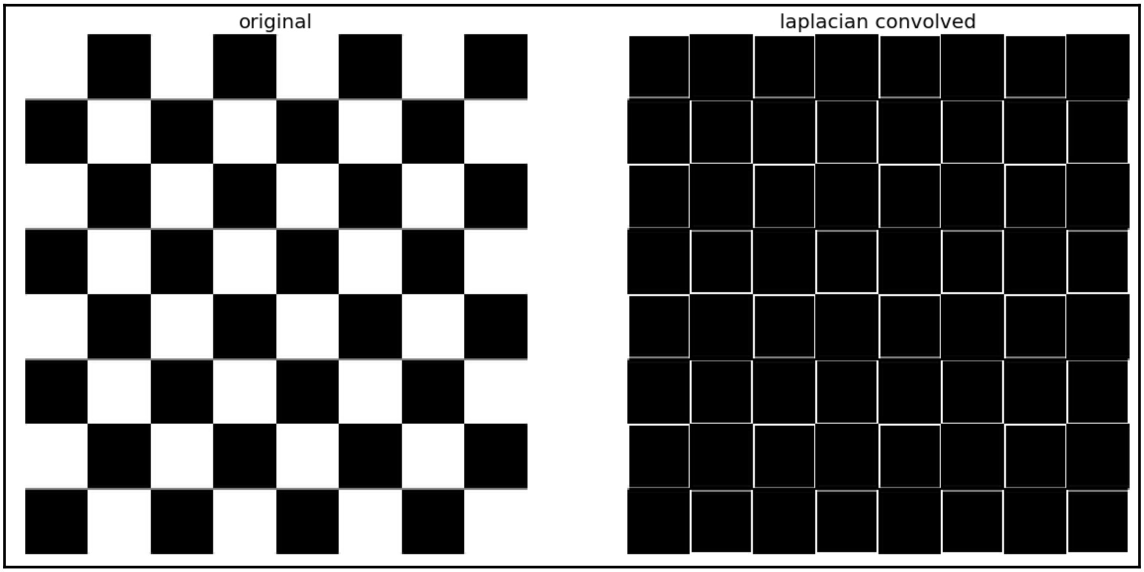
Convolution Kernel

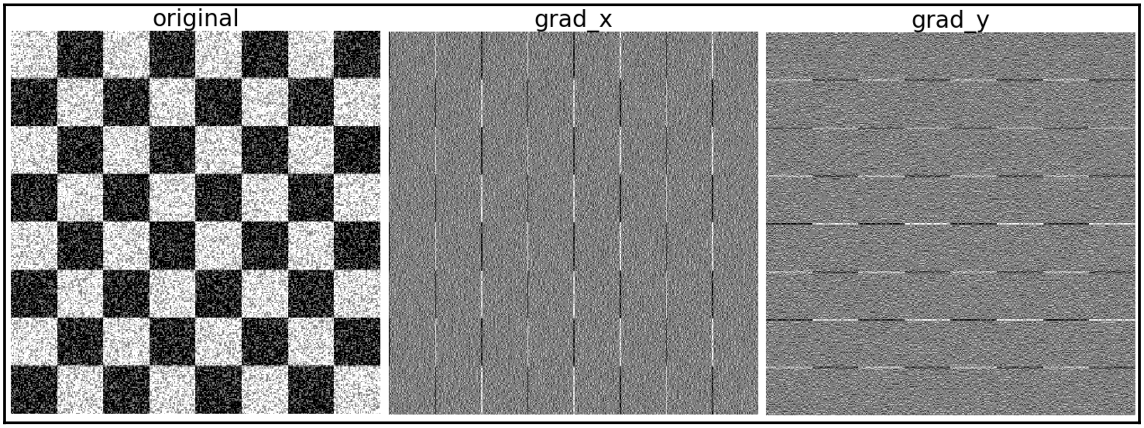
	1	
1	-4	1
	1	

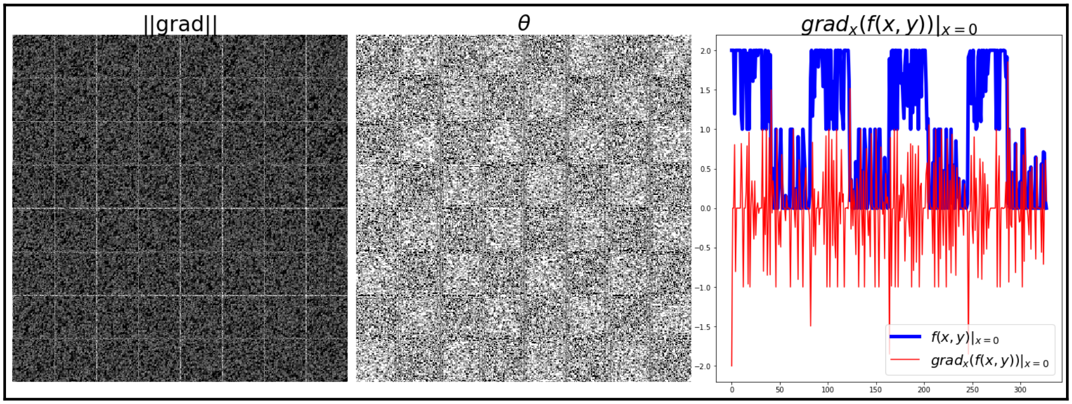
Laplacian

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x^2}$$

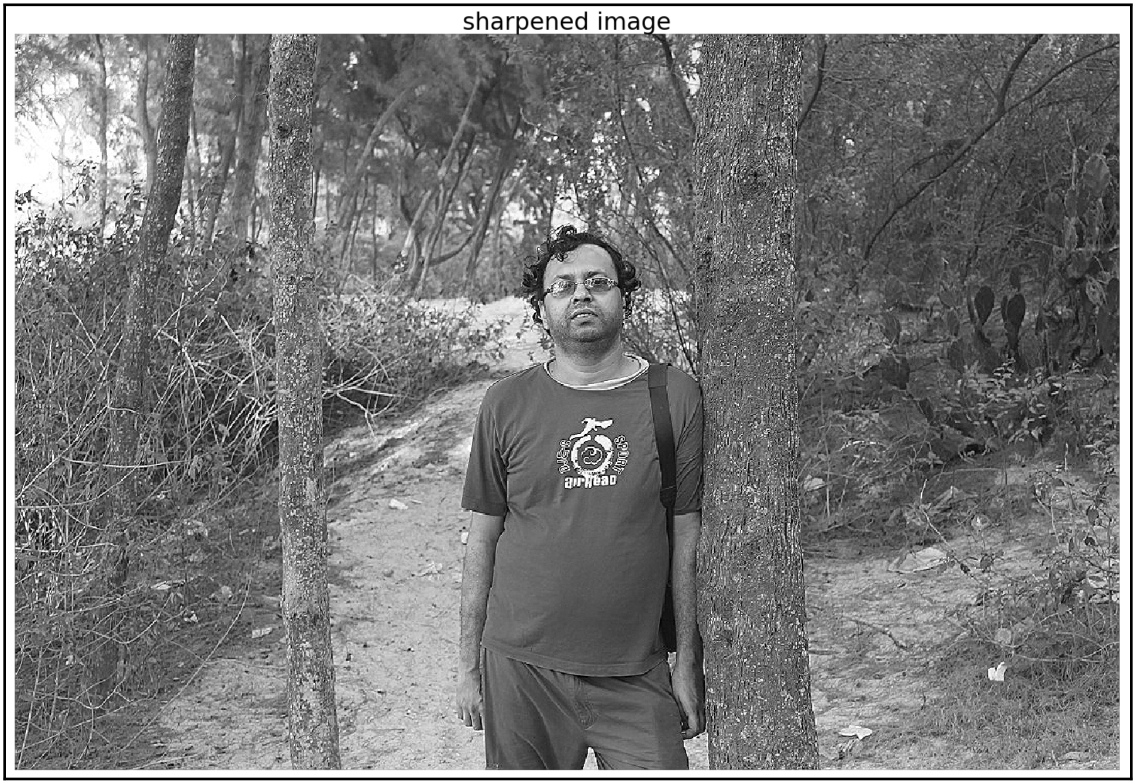
$$= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$




original
grad_x
grad_y


||grad||
θ
$grad_x(f(x,y))|_{x=0}$
$f(x,y)|_{x=0}$
$grad_x(f(x,y))|_{x=0}$

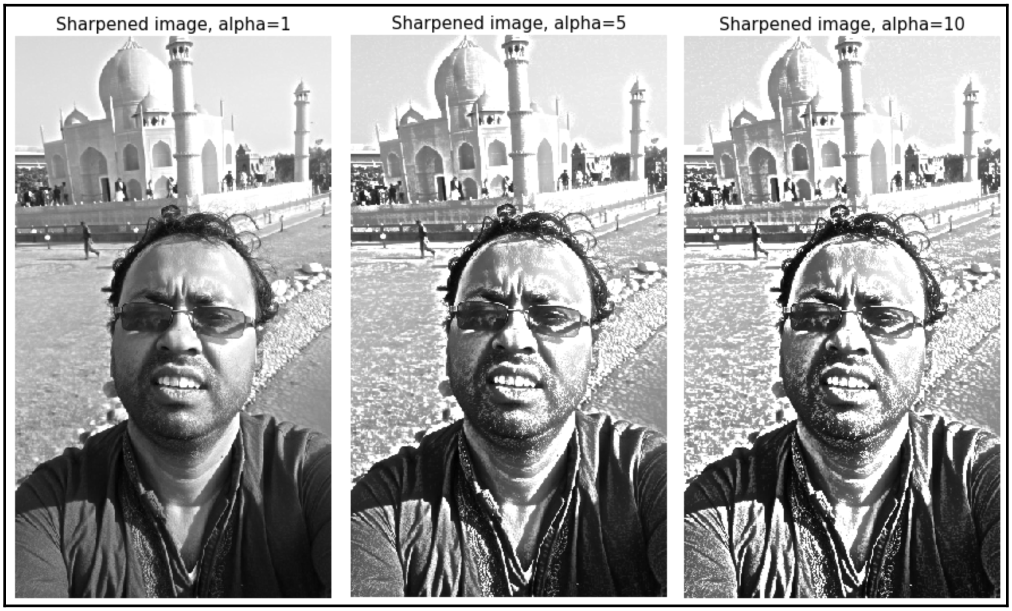

original image

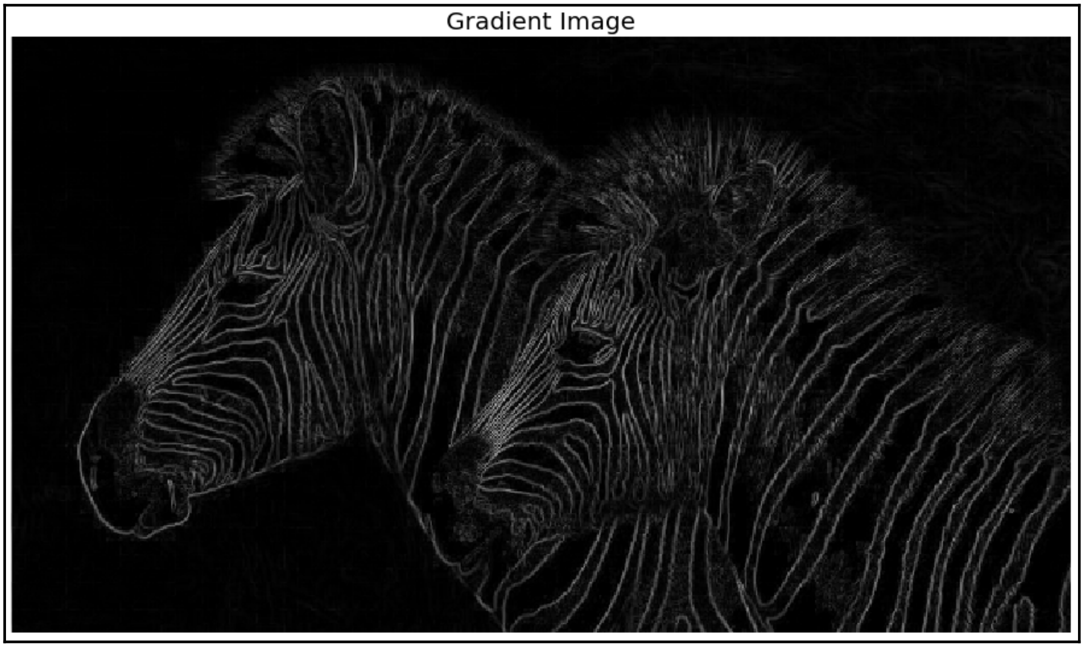


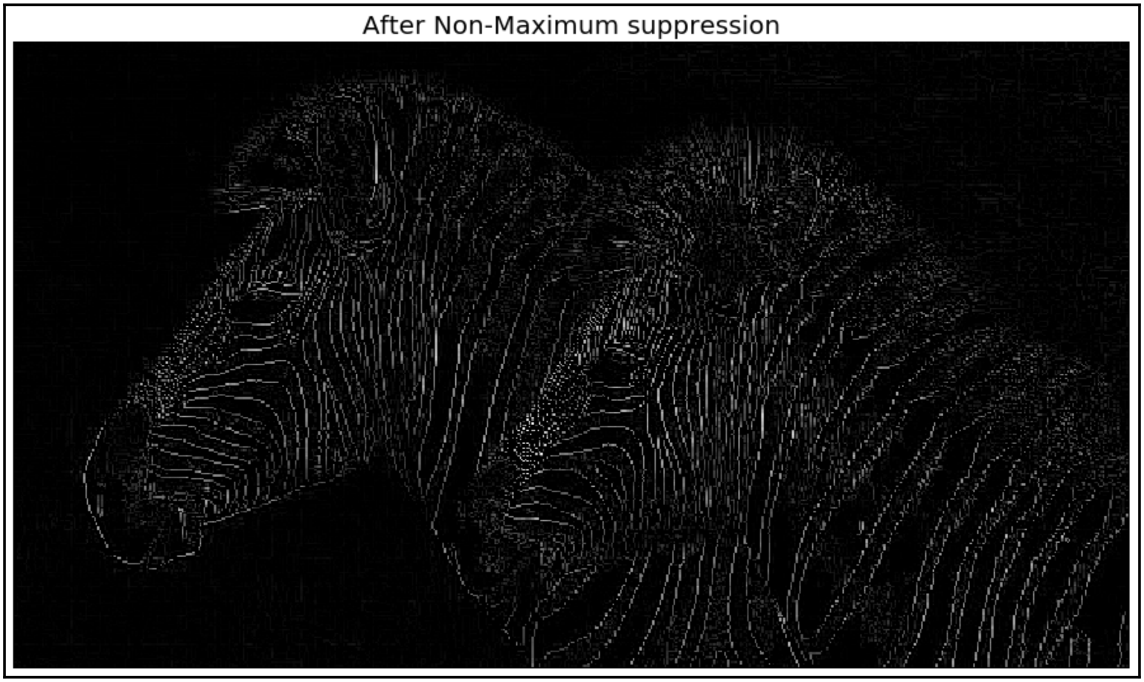
Original Image - Smoothed Image = Detail Image
(with Gaussian Filter)

Original Image + α.(Detail Image) = Sharpened Image

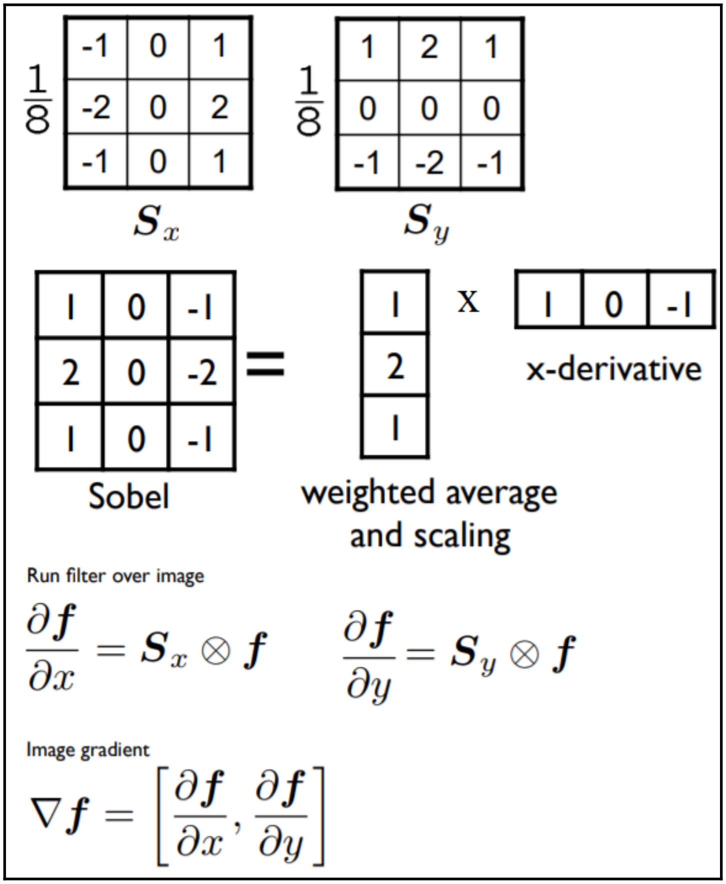

Original image
Blurred image, sigma=5
Detail image

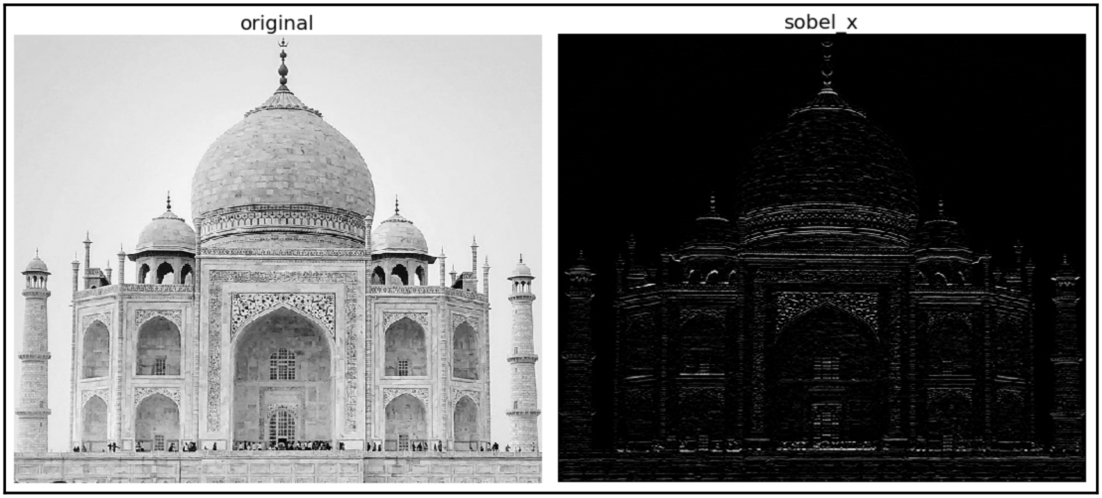

Sharpened image, alpha=1
Sharpened image, alpha=5
Sharpened image, alpha=10

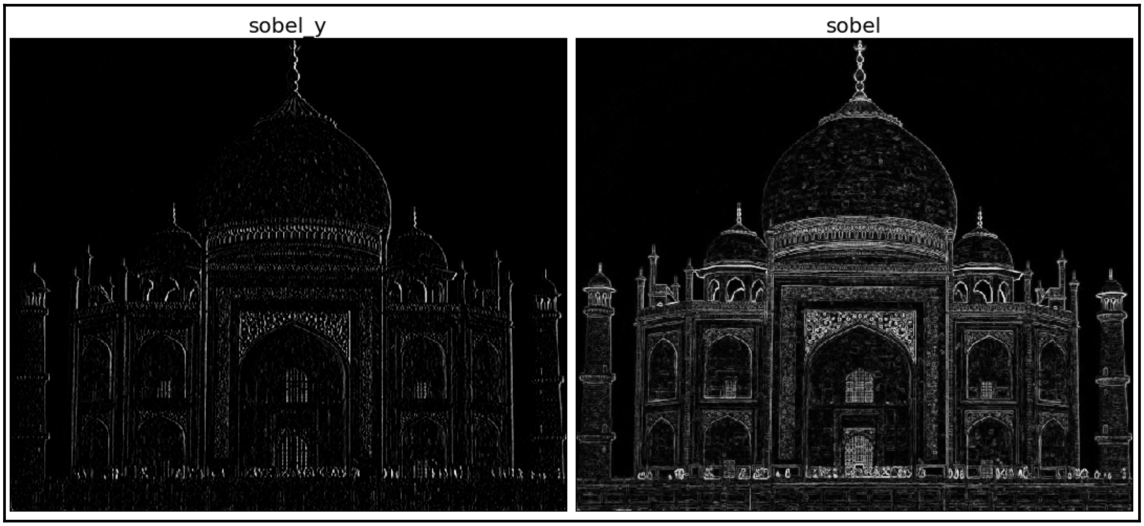

Gradient Image

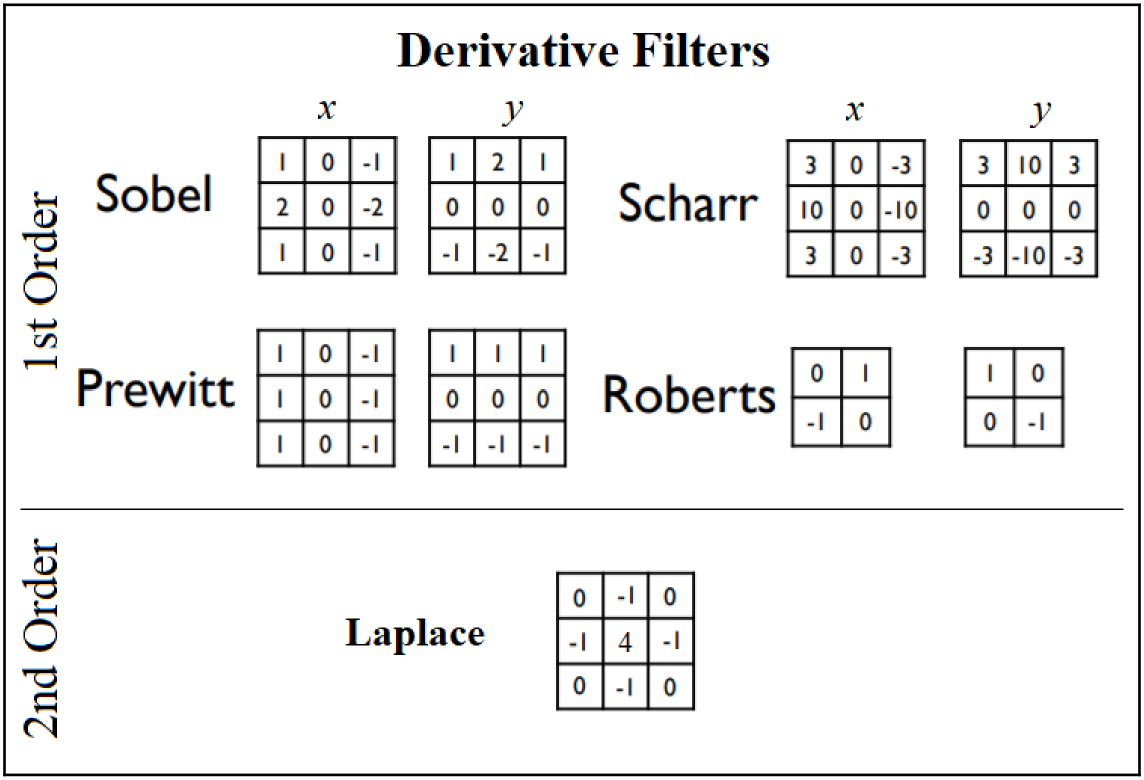


edge direction	pixels to compare
horizontal	top and bottom
vertical	left and right
north west or south east	top right and bottom left
north east or south west	bottom right and top left


1/8
-1 0 1
-2 0 2
-1 0 1
S_x
1/8
1 2 1
0 0 0
-1 -2 -1
S_y
1 0 -1
2 0 -2
1 0 -1
Sobel
=
1
2
1
weighted average and scaling
x
1 0 -1
x-derivative
Run filter over image
$\frac{\partial f}{\partial x} = S_x \otimes f$
$\frac{\partial f}{\partial y} = S_y \otimes f$
Image gradient
$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

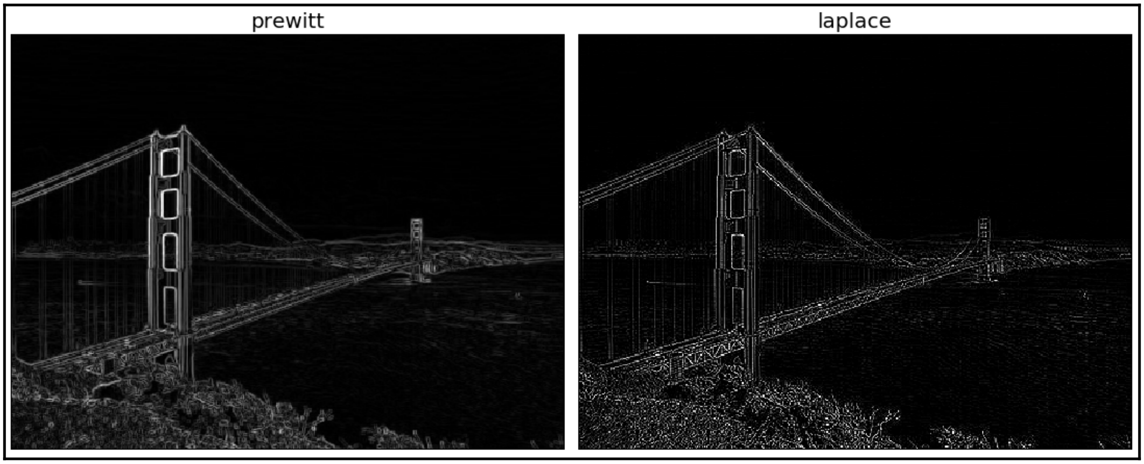

original
sobel_x

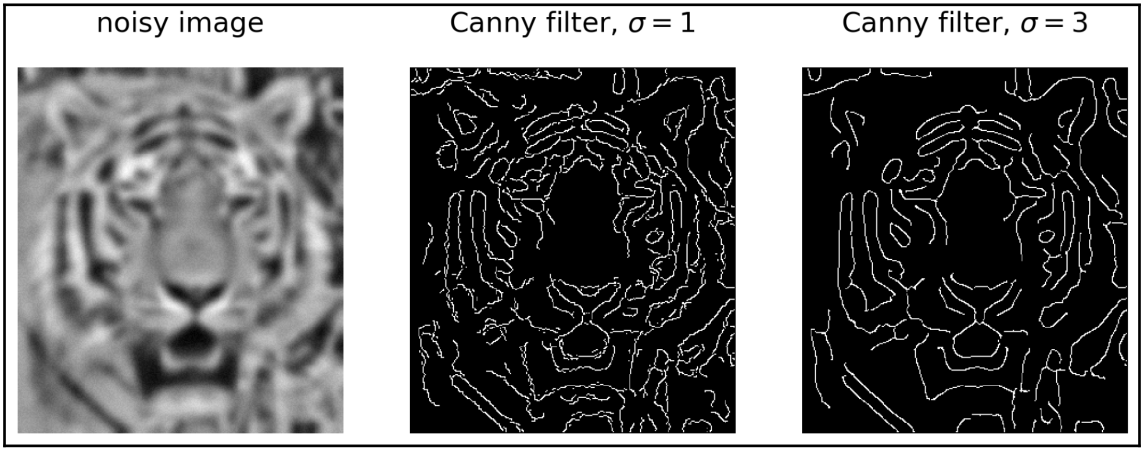

sobel_y
sobel


Derivative Filters
1st Order
x
y
x
y
Sobel
1 0 -1
2 0 -2
1 0 -1
1 2 1
0 0 0
-1 -2 -1
Scharr
3 0 -3
10 0 -10
3 0 -3
3 10 3
0 0 0
-3 -10 -3
Prewitt
1 0 -1
1 0 -1
1 0 -1
1 1 1
0 0 0
-1 -1 -1
Roberts
0 1
-1 0
1 0
0 -1
2nd Order
Laplace
0 -1 0
-1 4 -1
0 -1 0


original
roberts


scharr
sobel


prewitt
laplace



LoG

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

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

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

$$\text{LoG}(x,y) = \nabla^2 G_\sigma(x,y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}$$

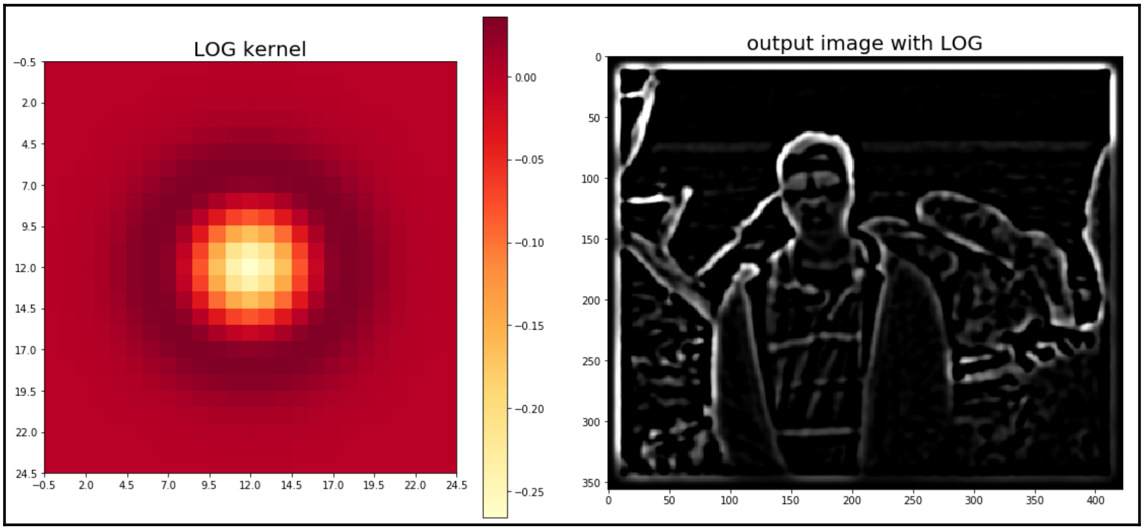
$$= -\frac{1}{\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(1 - \frac{x^2+y^2}{2\sigma^2}\right)$$

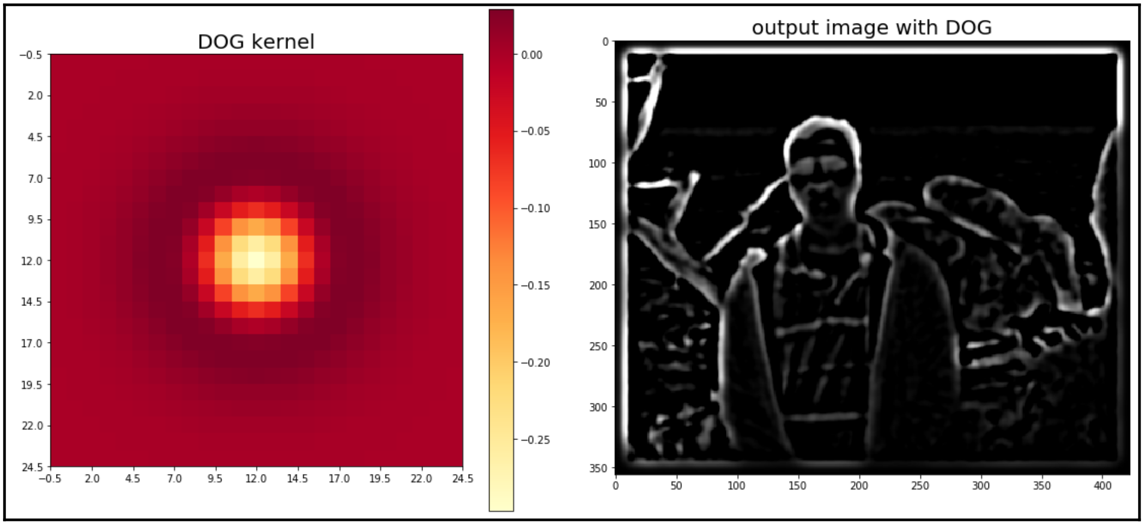
Approximation with DoG

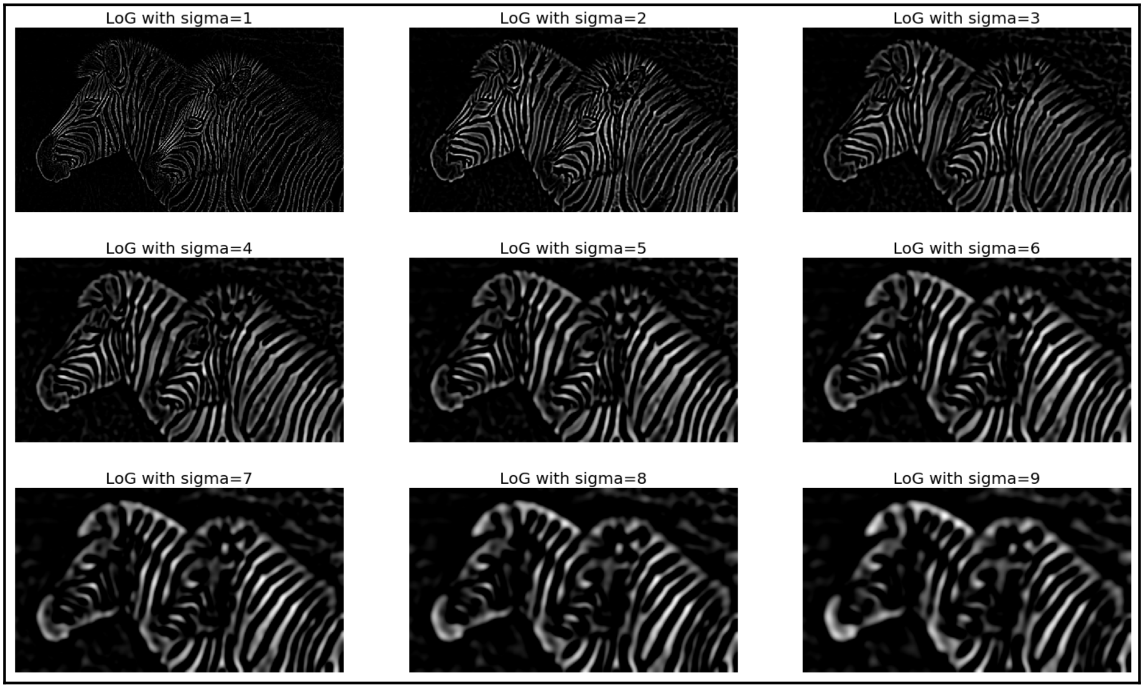
$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

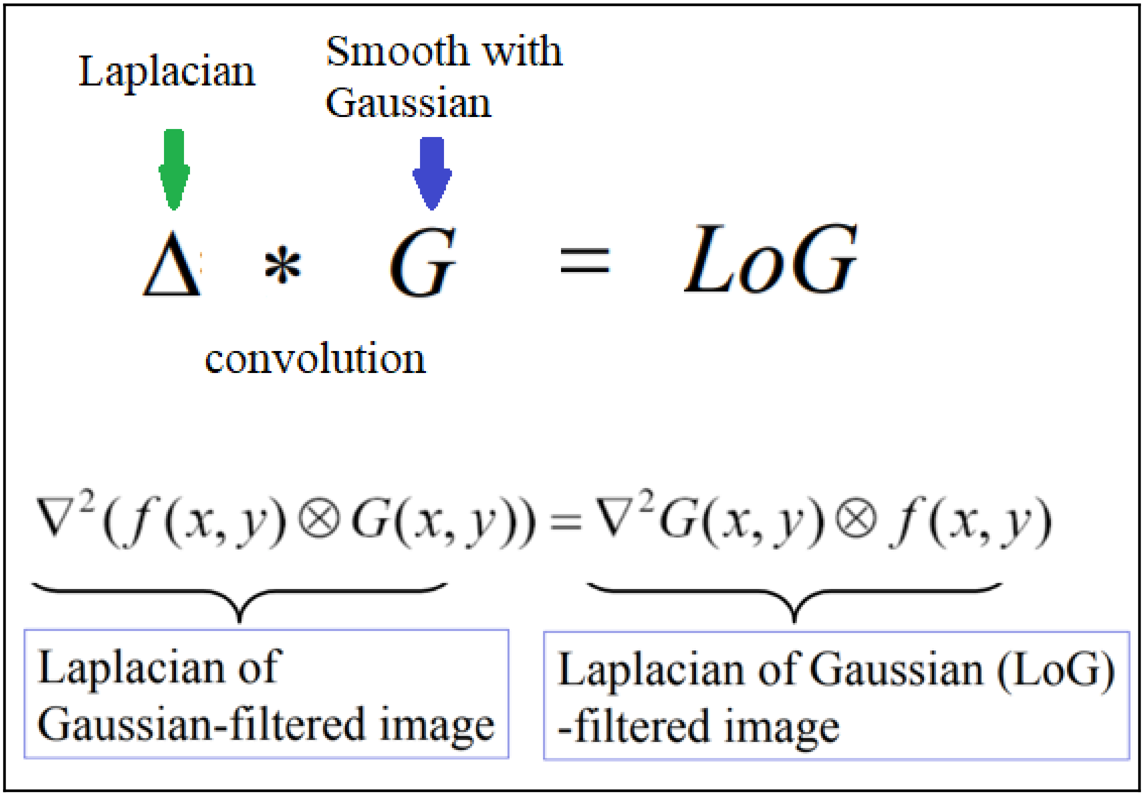
Best approximation with

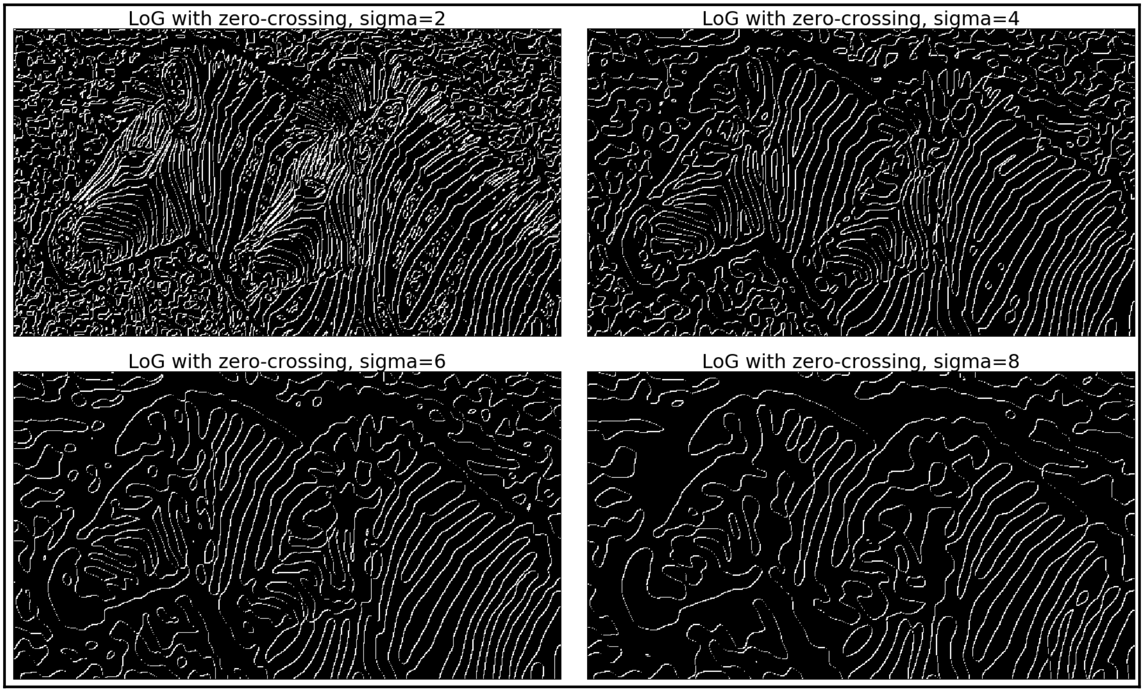
$$\sigma_1 = \sqrt{2}\sigma \, , \; \sigma_2 = \frac{\sigma}{\sqrt{2}}$$

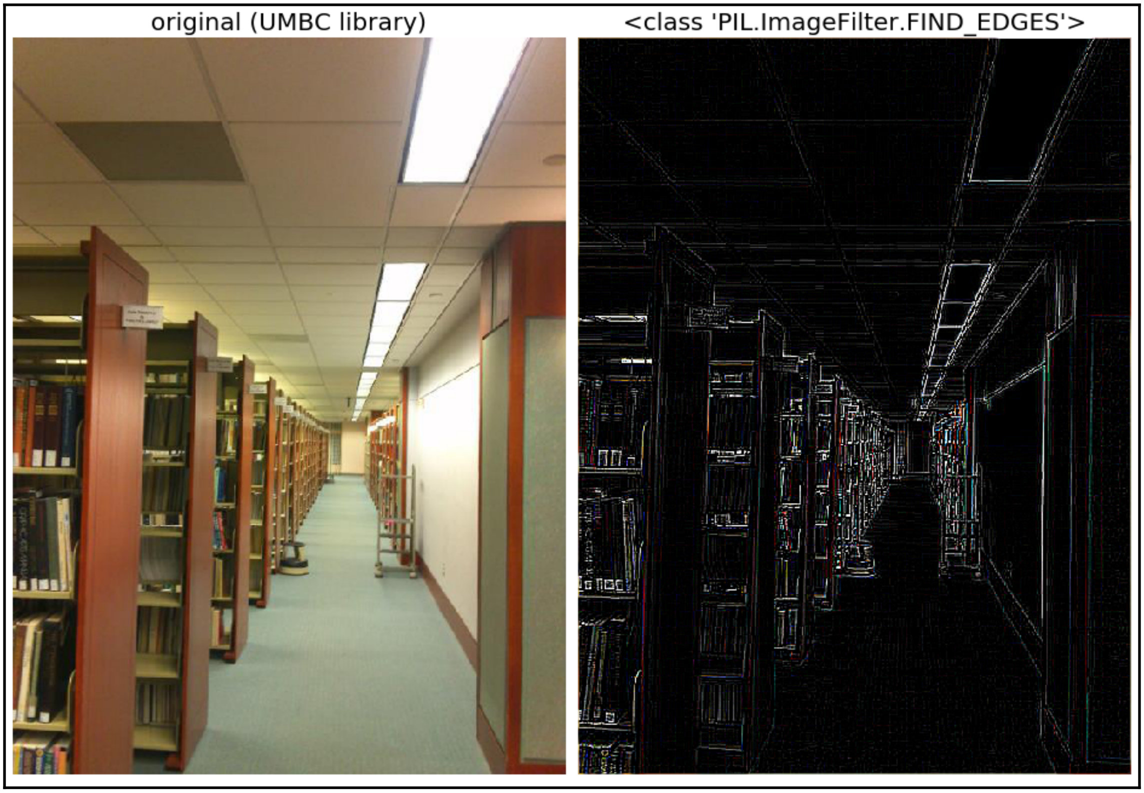

LOG kernel
output image with LOG


DOG kernel
−0.5 2.0 4.5 7.0 9.5 12.0 14.5 17.0 19.5 22.0 24.5
0.00 −0.05 −0.10 −0.15 −0.20 −0.25
output image with DOG
0 50 100 150 200 250 300 350
0 50 100 150 200 250 300 350 400


LoG with sigma=1
LoG with sigma=2
LoG with sigma=3
LoG with sigma=4
LoG with sigma=5
LoG with sigma=6
LoG with sigma=7
LoG with sigma=8
LoG with sigma=9


Laplacian
Smooth with Gaussian
$\Delta * G = LoG$
convolution
$\nabla^2(f(x,y) \otimes G(x,y)) = \nabla^2 G(x,y) \otimes f(x,y)$
Laplacian of Gaussian-filtered image
Laplacian of Gaussian (LoG) -filtered image

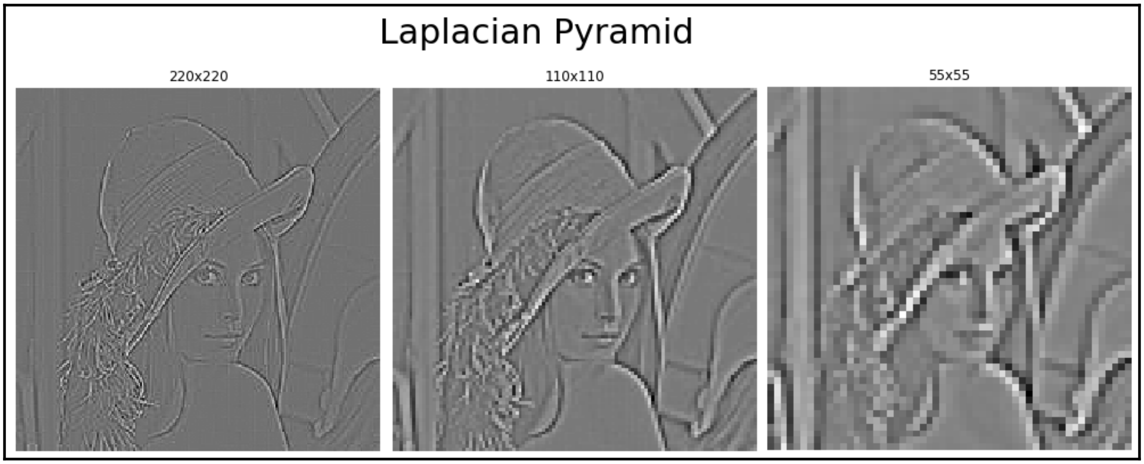

LoG with zero-crossing, sigma=2
LoG with zero-crossing, sigma=4
LoG with zero-crossing, sigma=6
LoG with zero-crossing, sigma=8

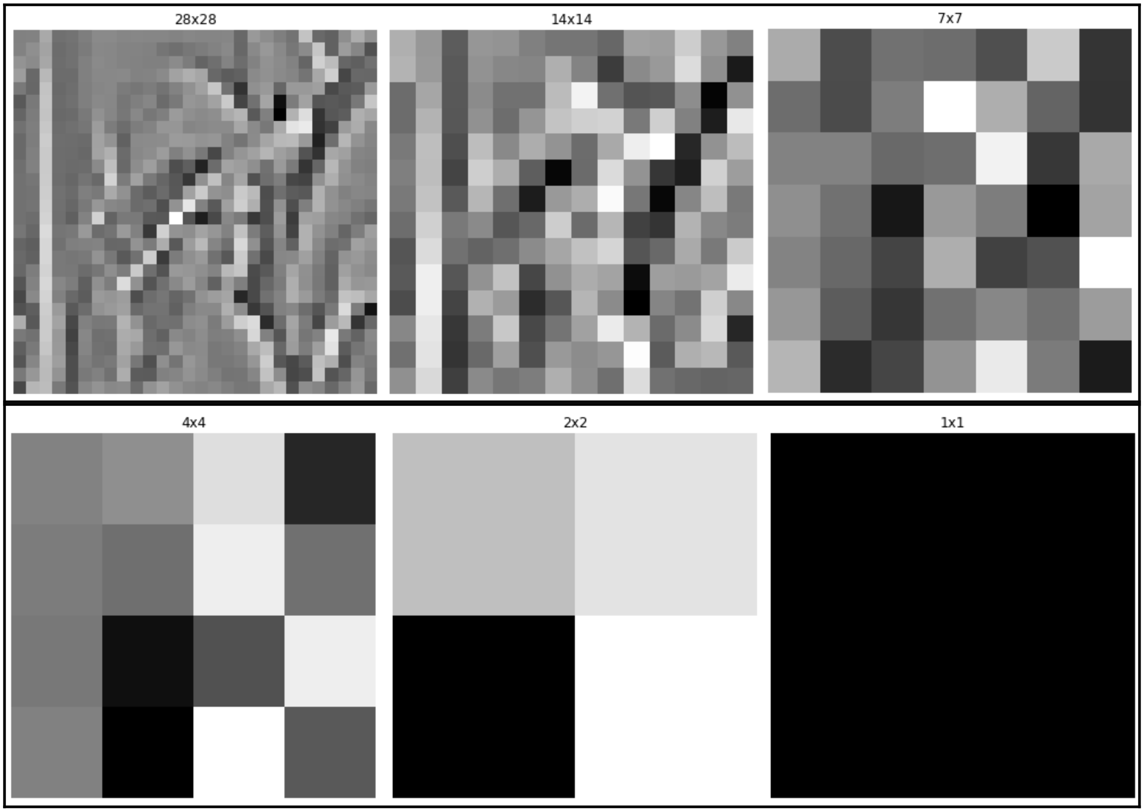

original (UMBC library)
<class 'PIL.ImageFilter.FIND_EDGES'>


<class 'PIL.ImageFilter.EDGE_ENHANCE'>
<class 'PIL.ImageFilter.EDGE_ENHANCE_MORE'>

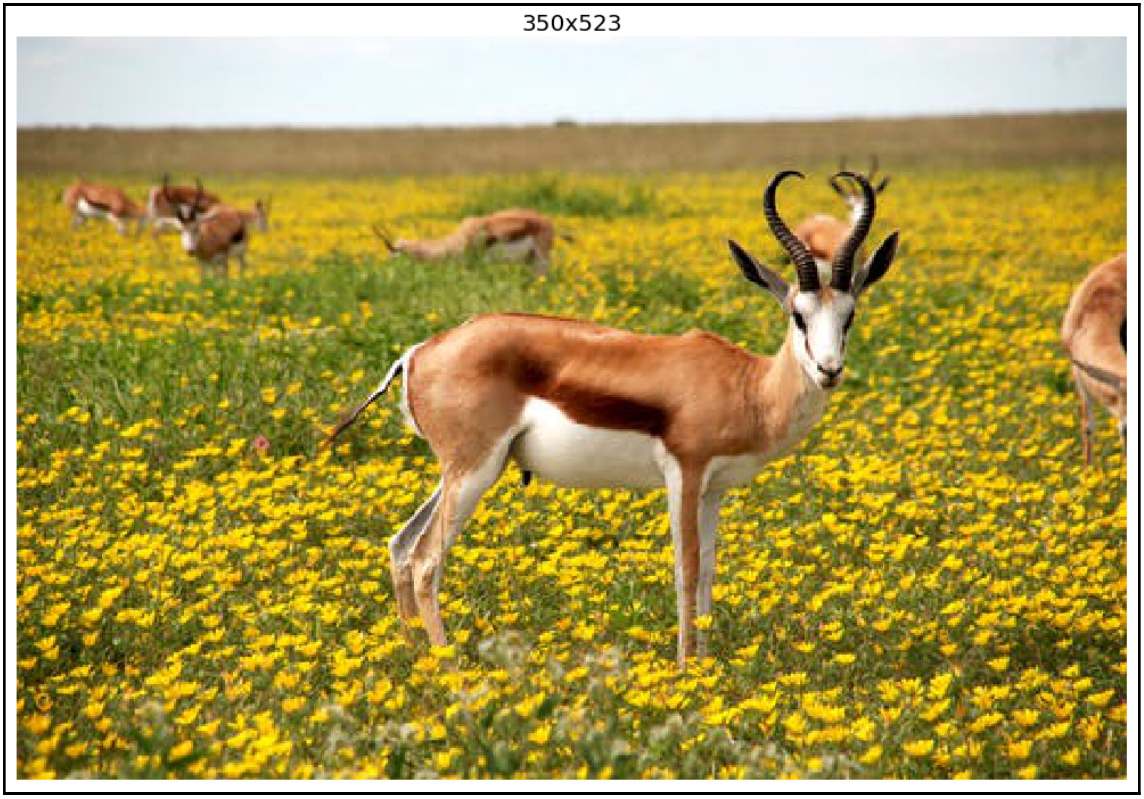

Gaussian Pyramid
220x220
110x110
55x55
28x28
14x14
7x7
4x4
2x2
1x1

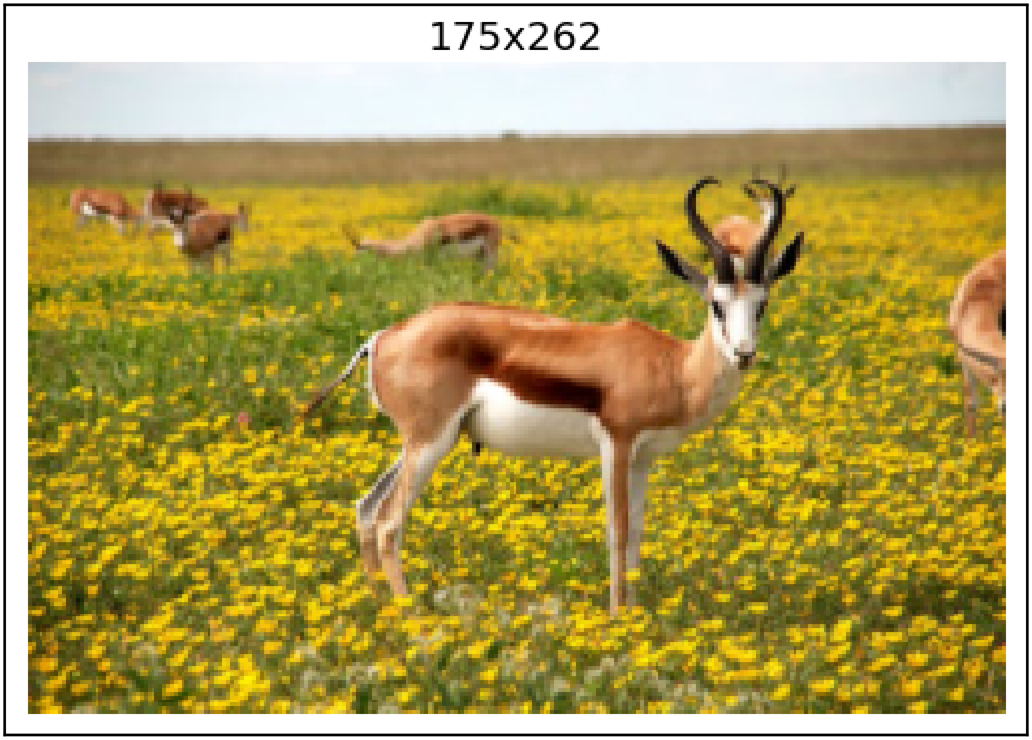

0
25
50
75
100
125
150
175
200
0
50
100
150
200
250
300

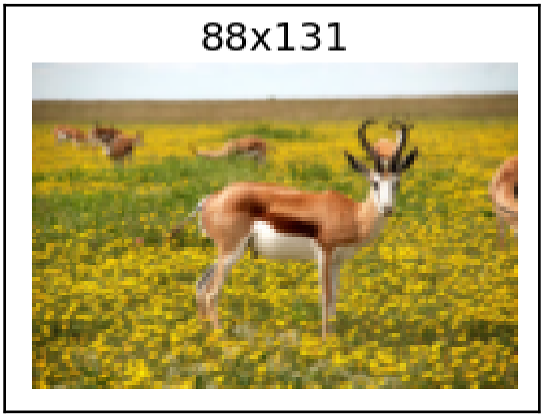

Laplacian Pyramid
220x220
110x110
55x55

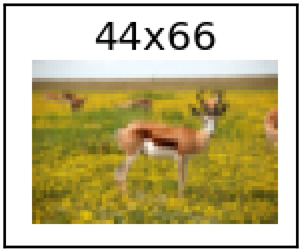

28x28
14x14
7x7
4x4
2x2
1x1

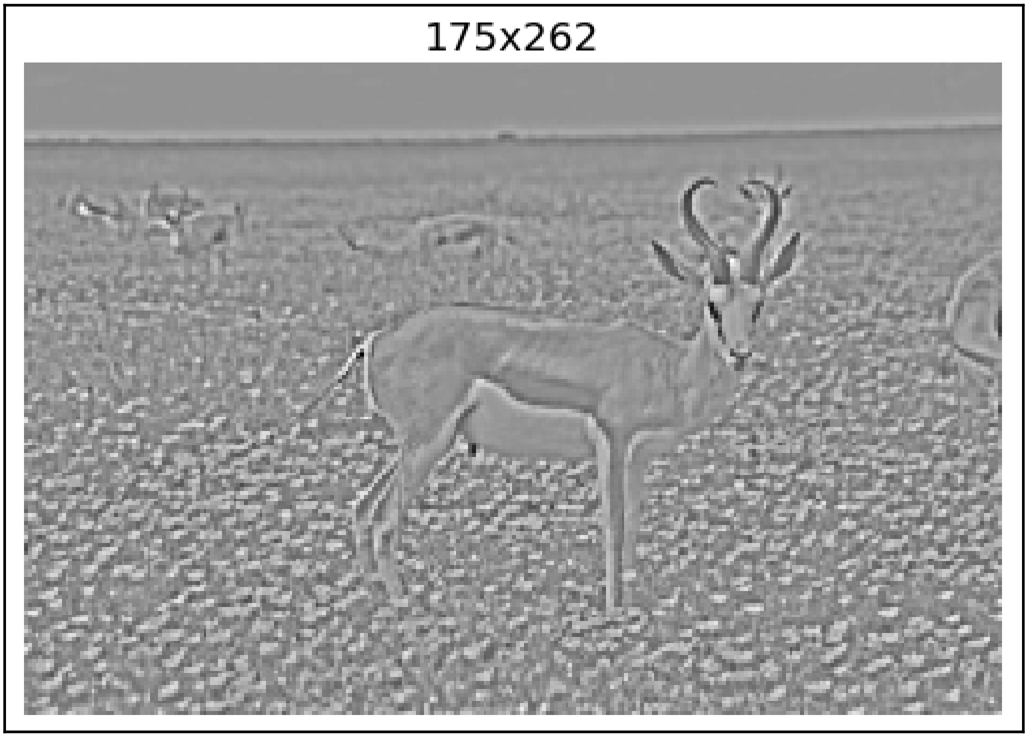

0
25
50
75
100
125
150
175
200
0
50
100
150
200
250
300

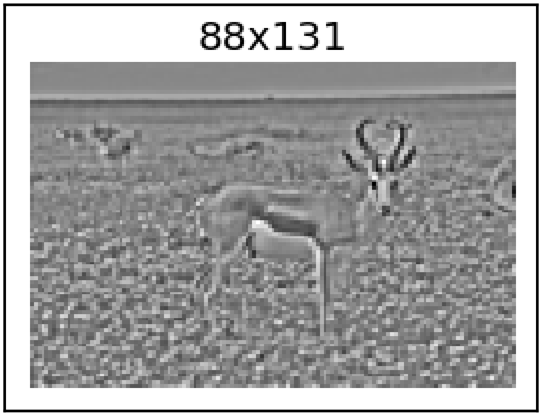

350x523

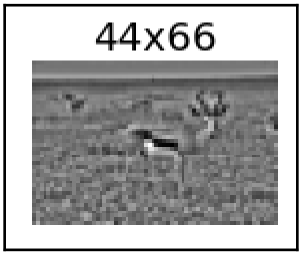

175x262


88x131


44x66


175x262


88x131


44x66

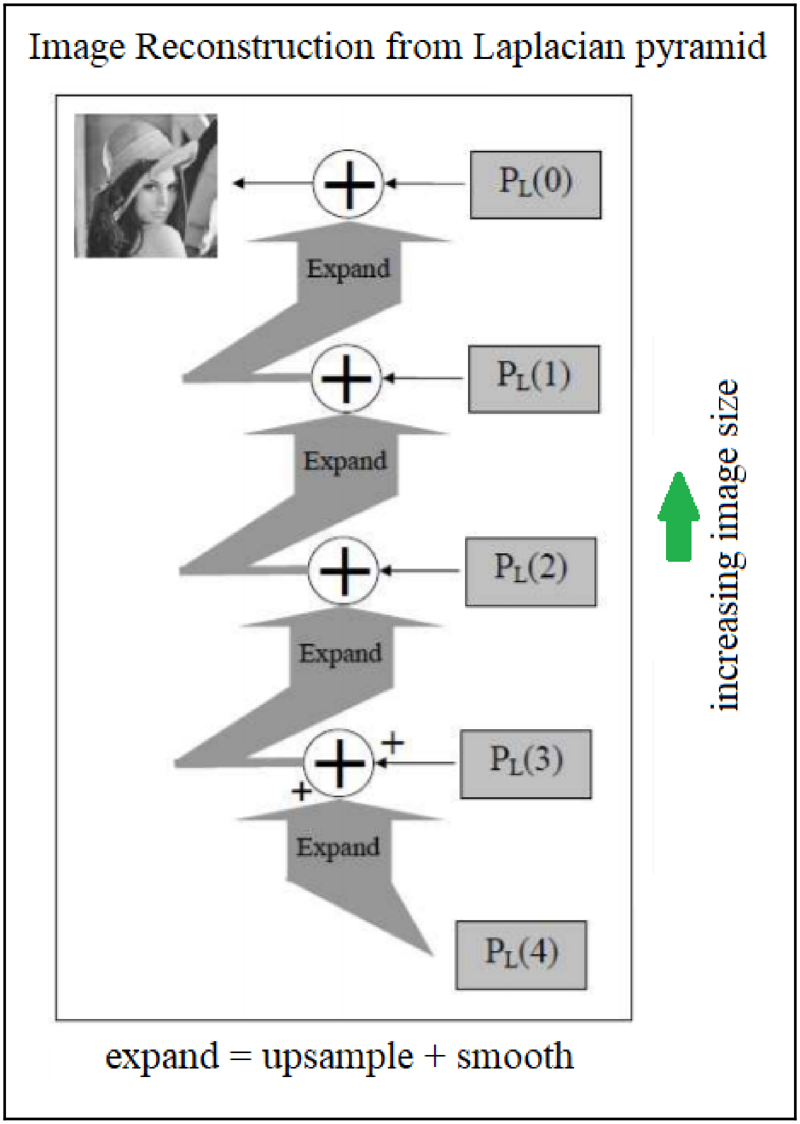
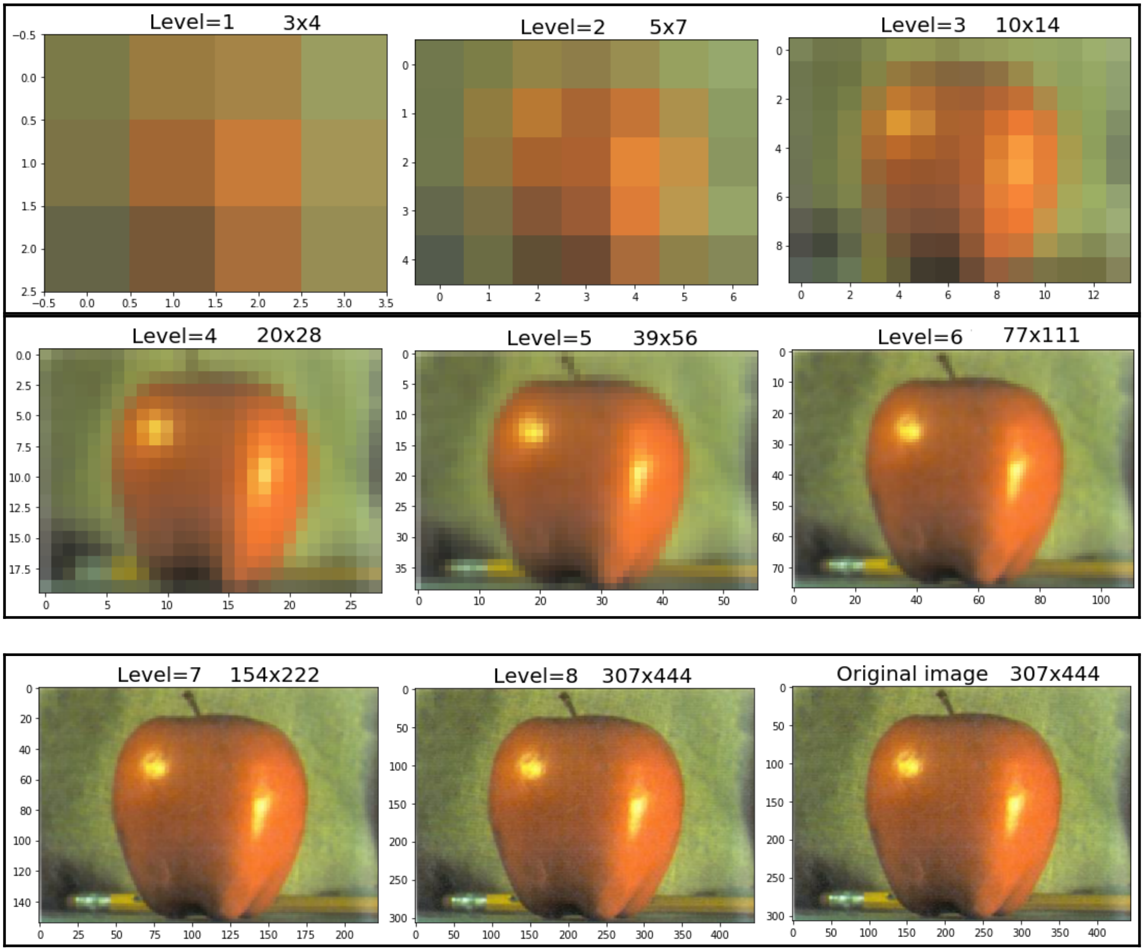
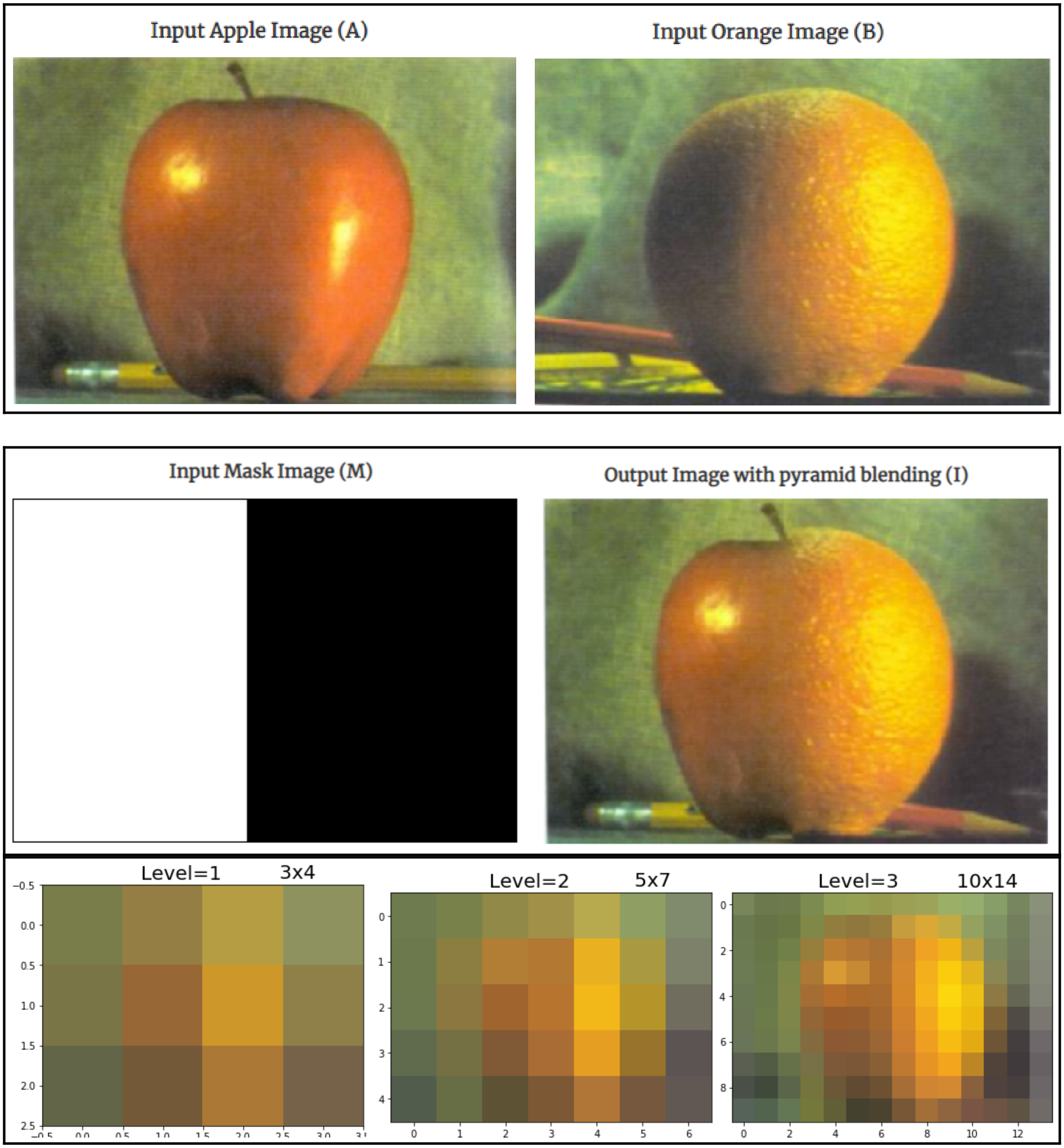
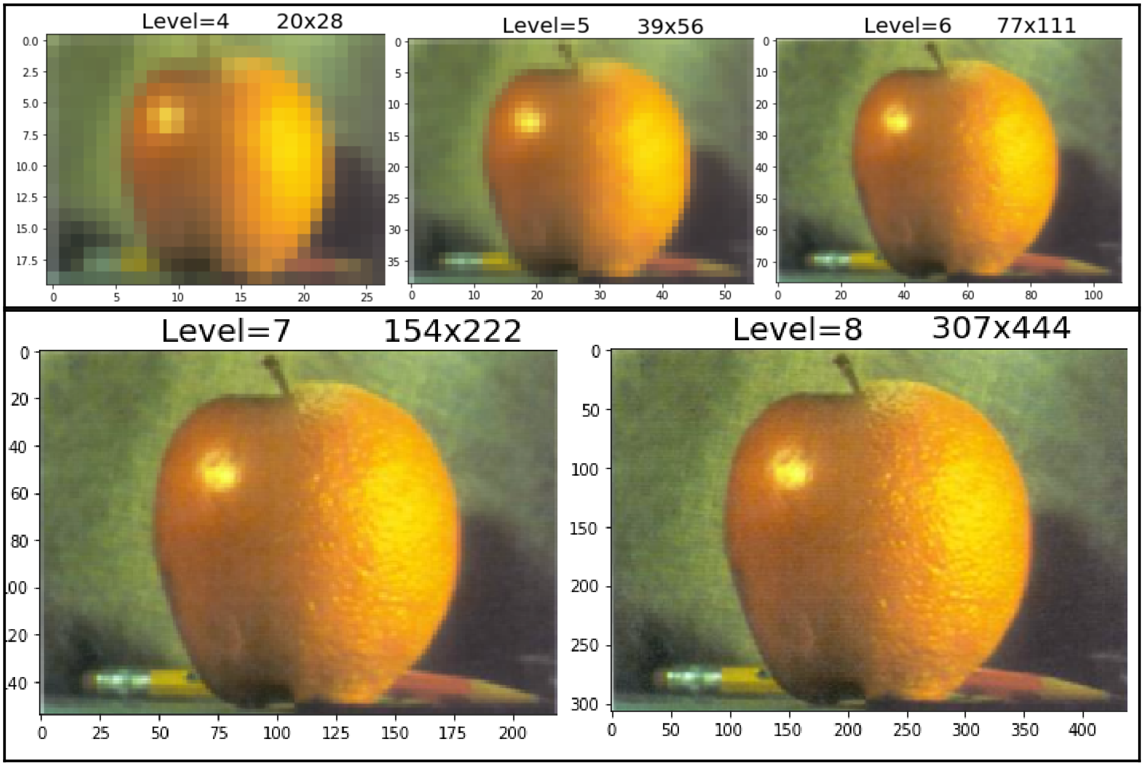


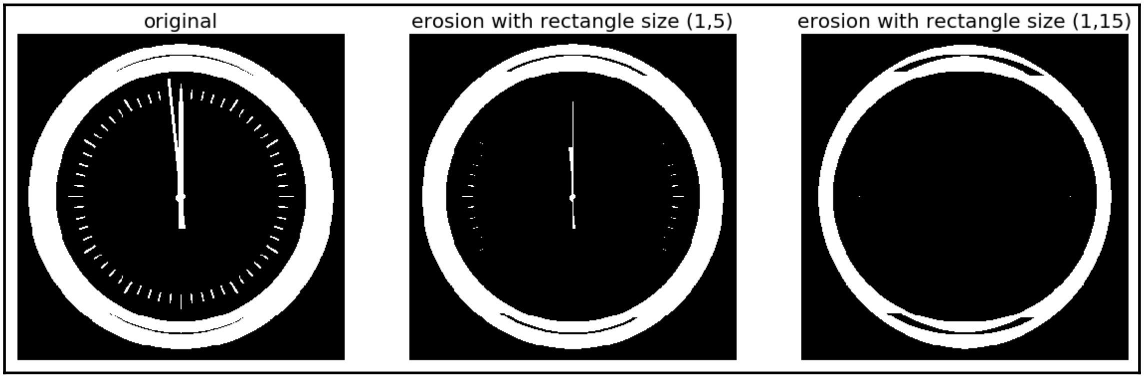
Image Reconstruction from Laplacian pyramid
P_L(0)
Expand
P_L(1)
Expand
P_L(2)
Expand
+
+
P_L(3)
Expand
P_L(4)
increasing image size
expand = upsample + smooth

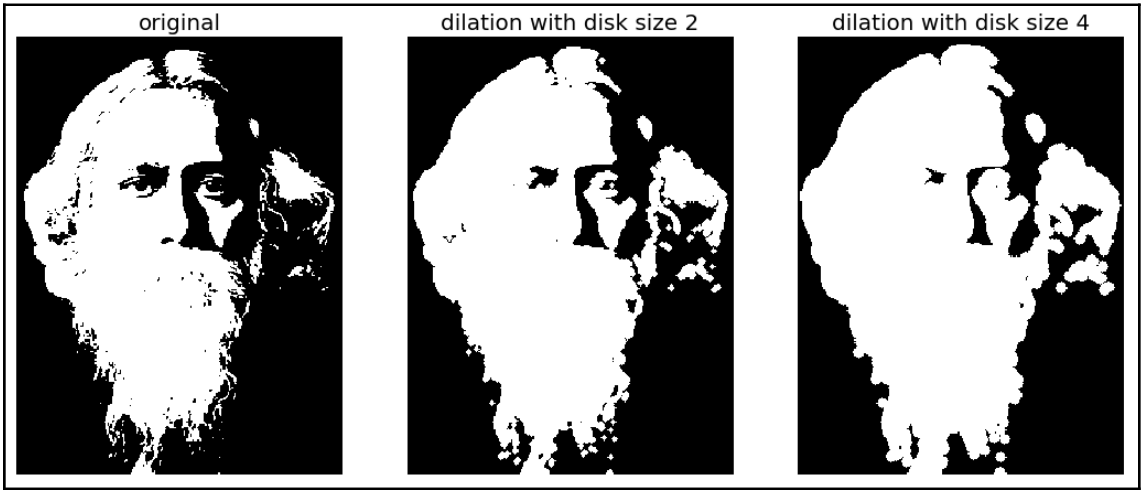

Level=1 3x4
Level=2 5x7
Level=3 10x14
Level=4 20x28
Level=5 39x56
Level=6 77x111
Level=7 154x222
Level=8 307x444
Original image 307x444

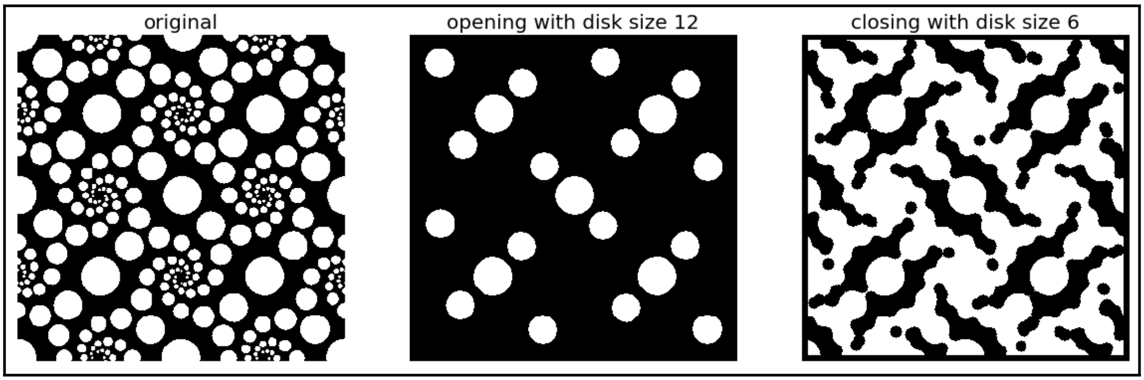

Input Apple Image (A)
Input Orange Image (B)
Input Mask Image (M)
Output Image with pyramid blending (I)
Level=1
3x4
Level=2
5x7
Level=3
10x14

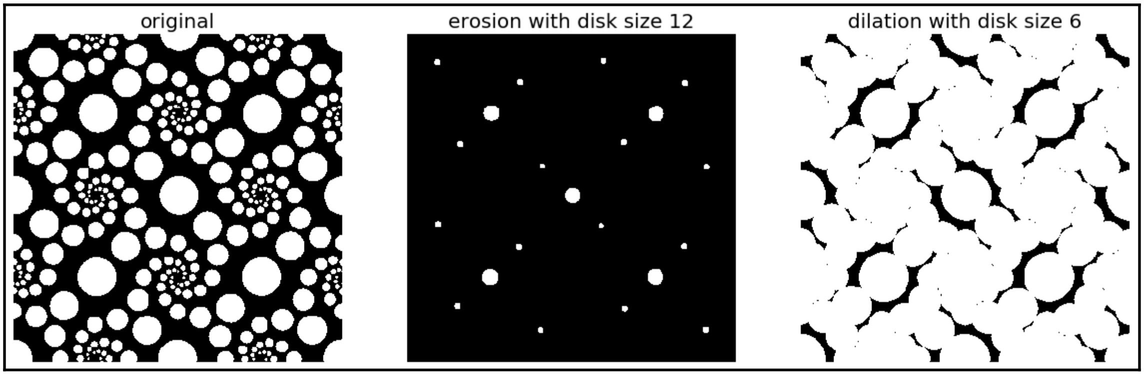

Level=4 20x28
Level=5 39x56
Level=6 77x111
Level=7 154x222
Level=8 307x444

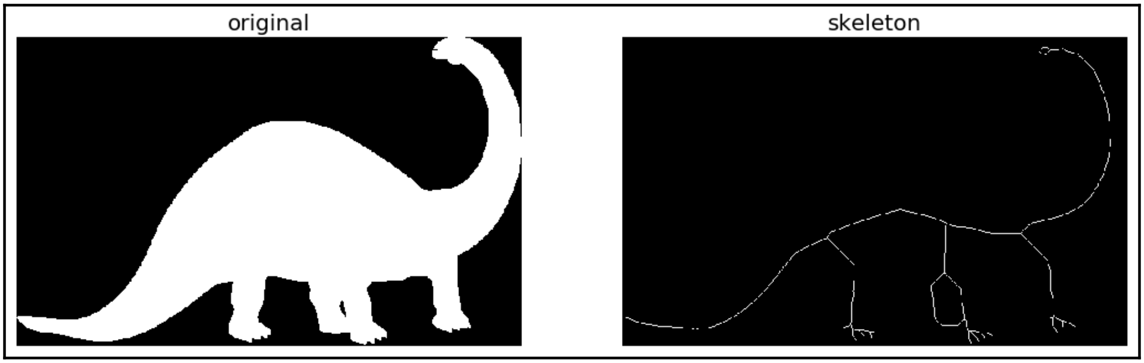
Chapter 6: Morphological Image Processing

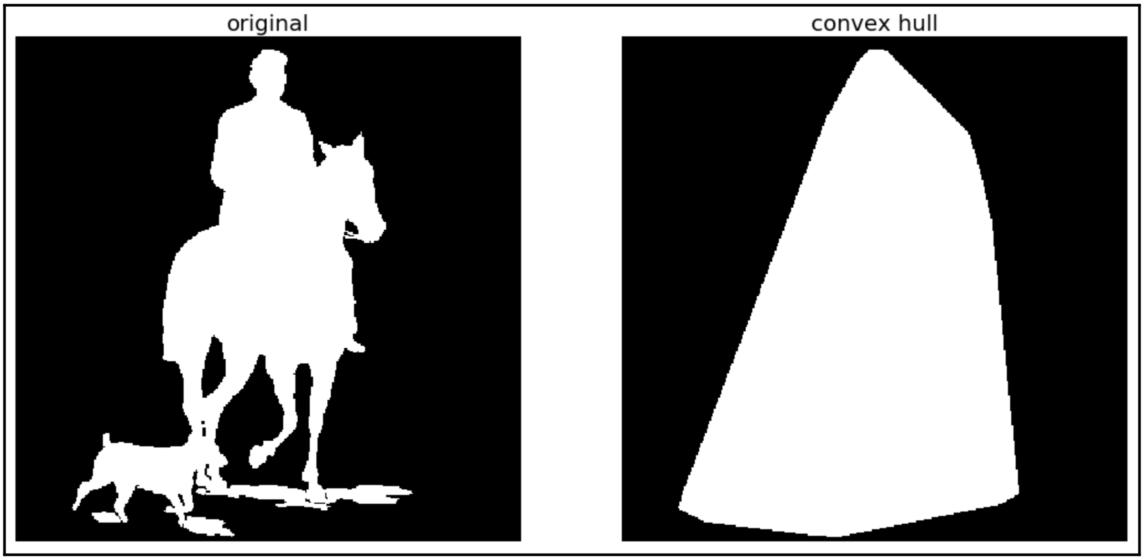


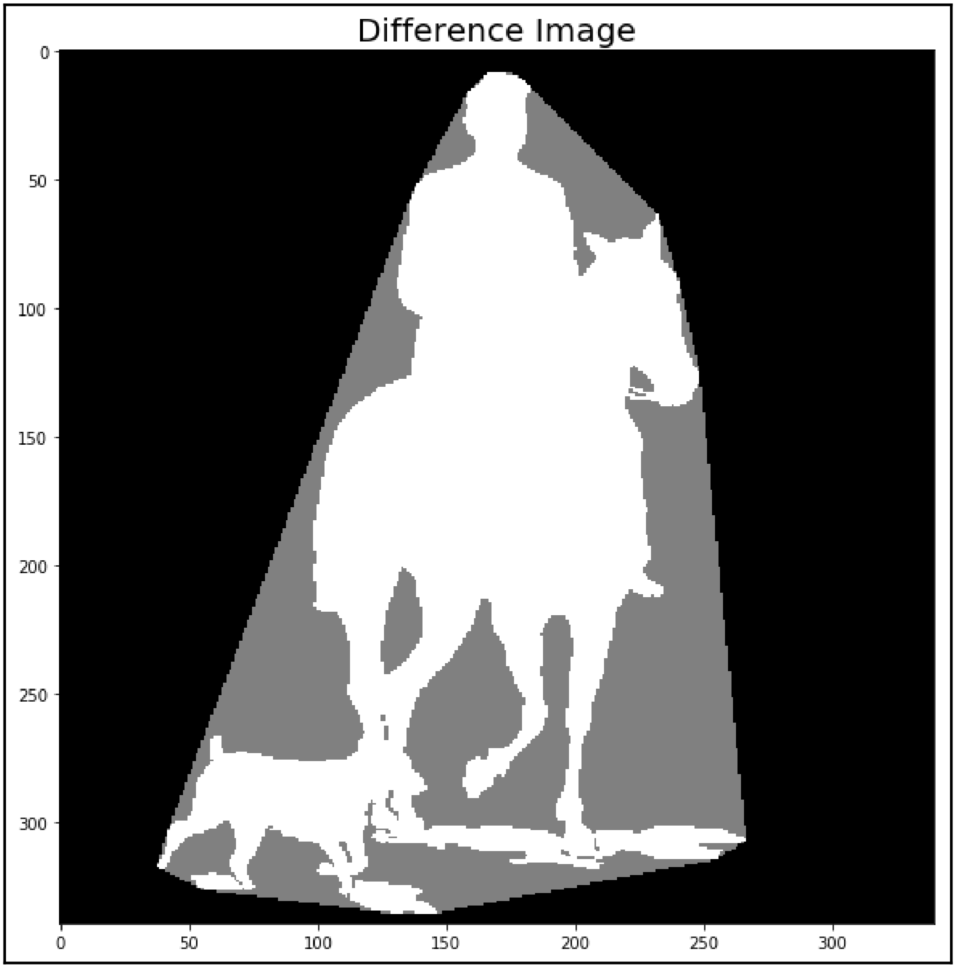


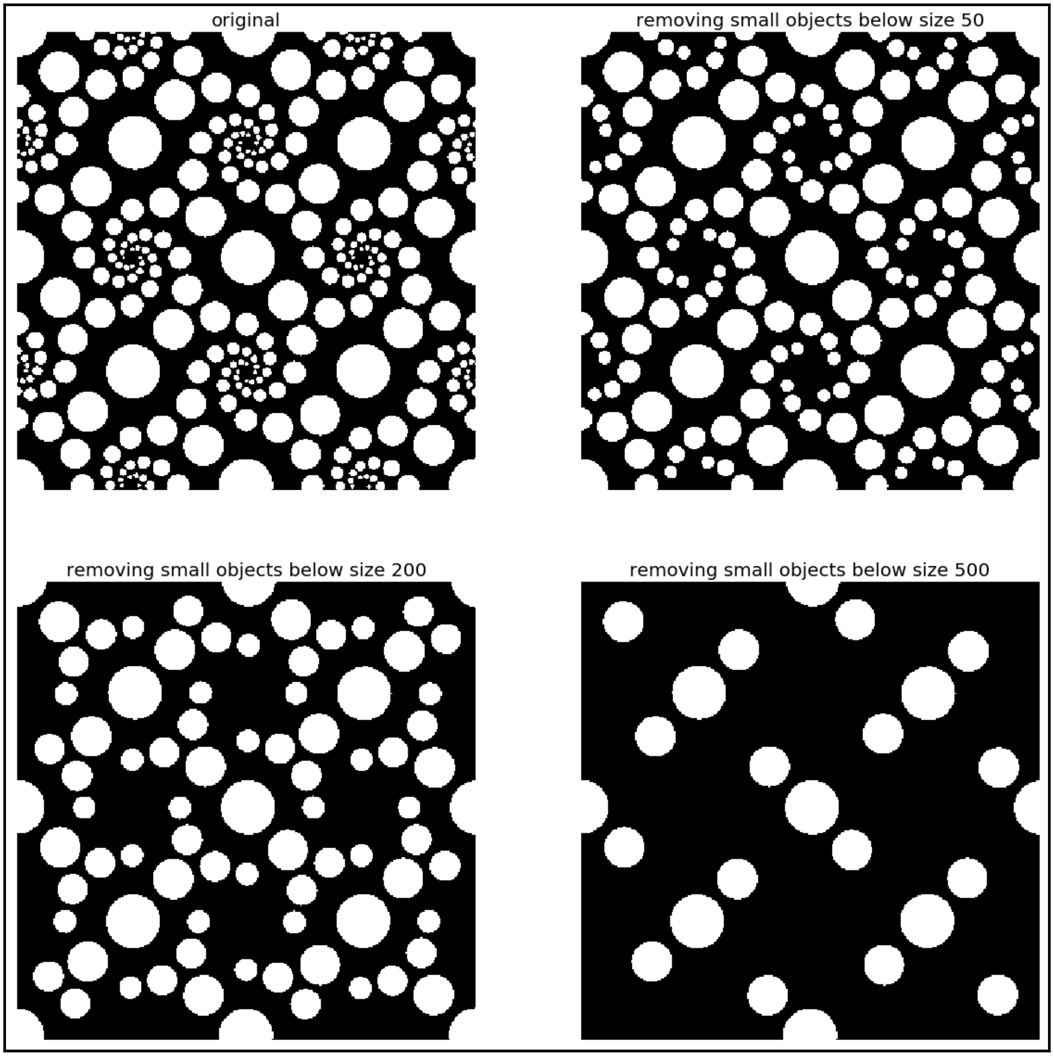


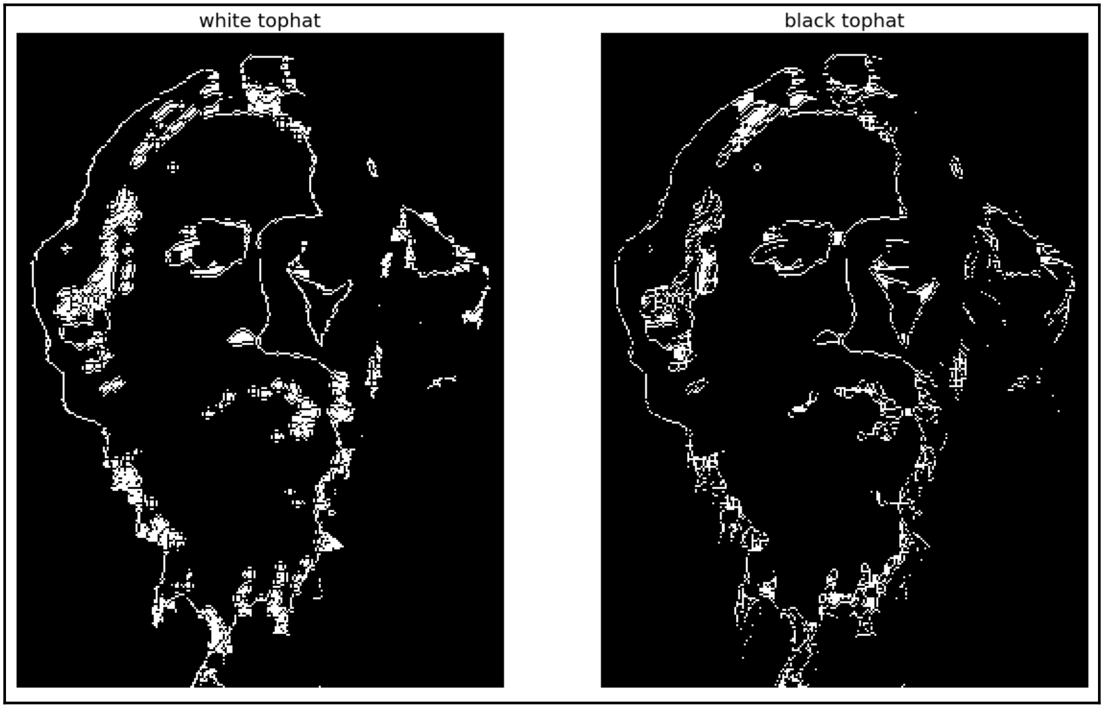

original
erosion with disk size 12
dilation with disk size 6

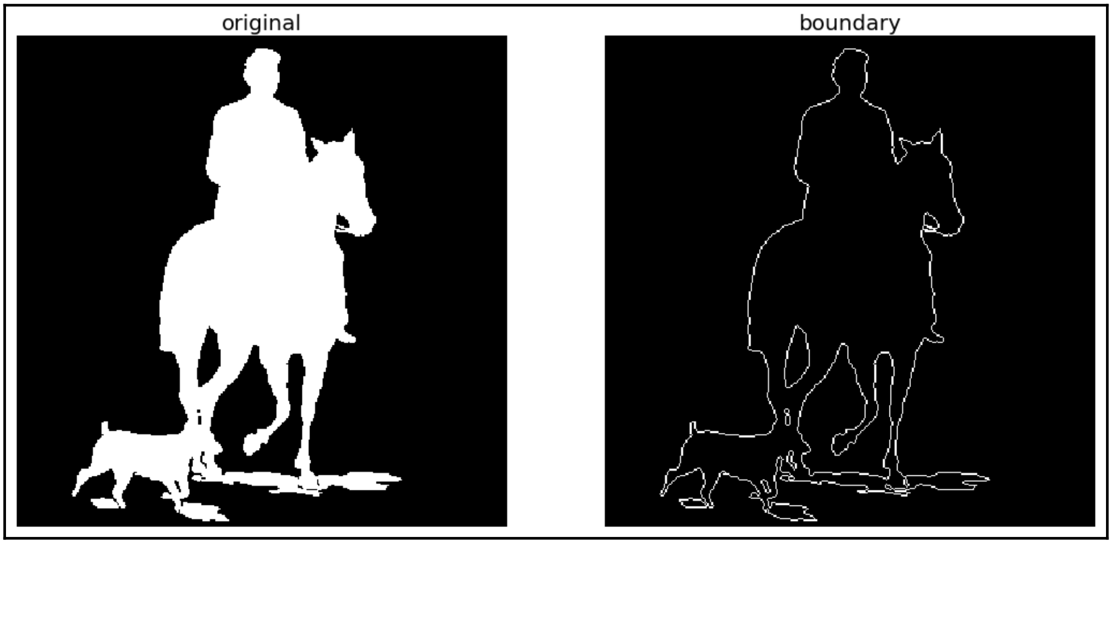

original
skeleton

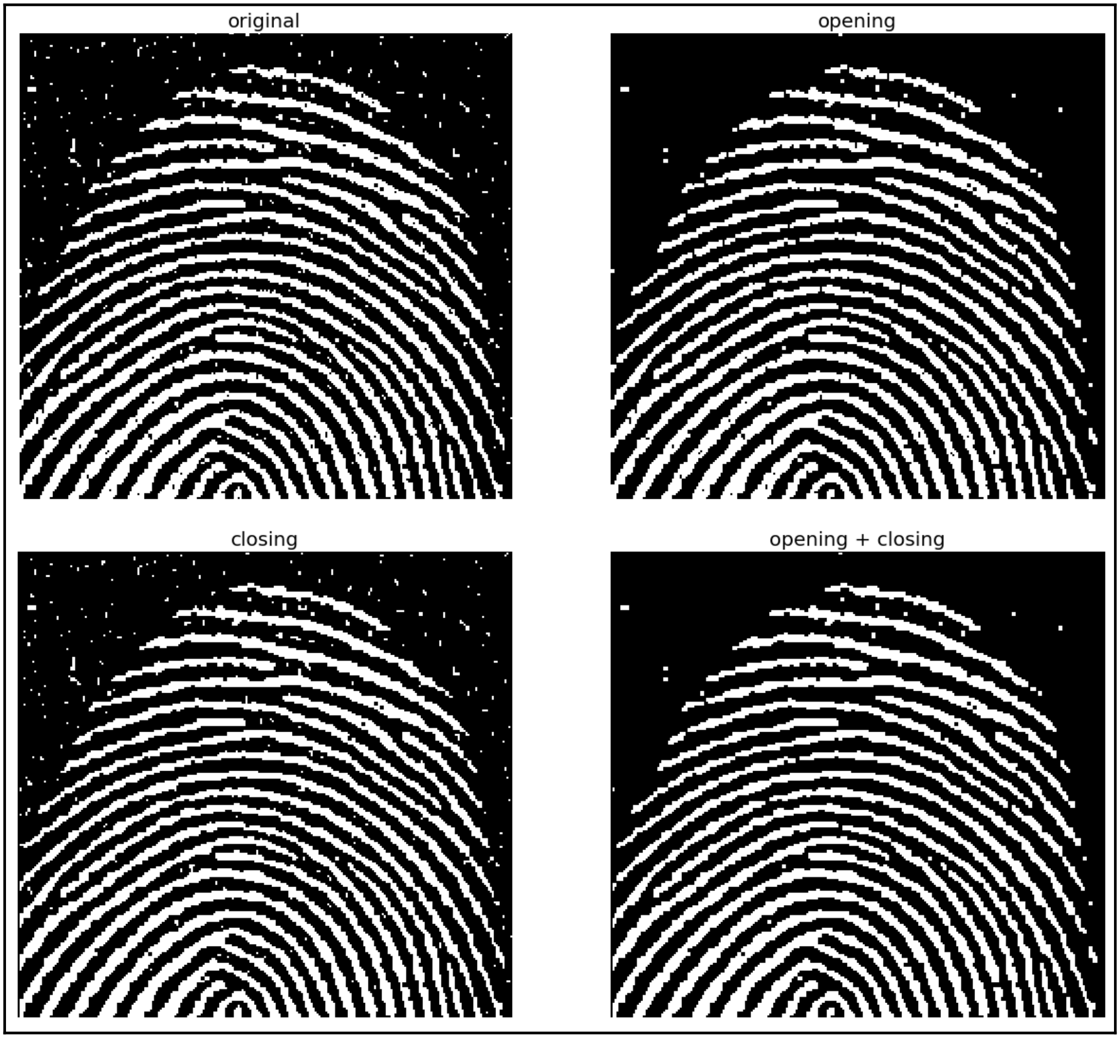

original
convex hull

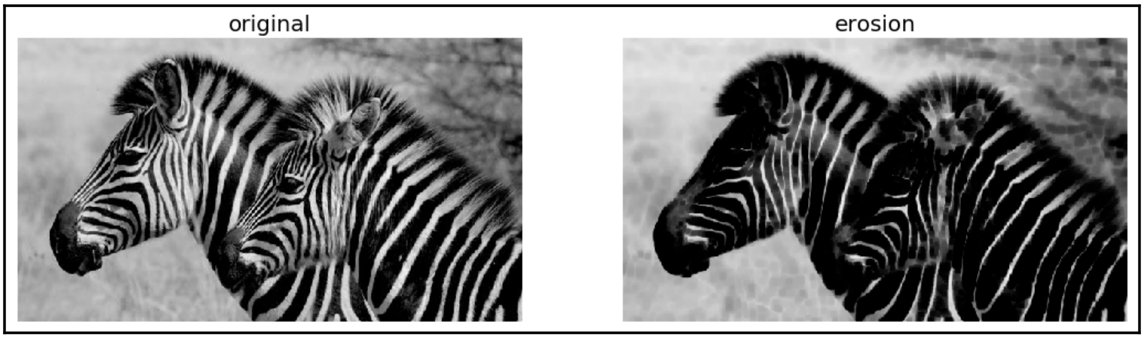

Difference Image
0
50
100
150
200
250
300
0
50
100
150
200
250
300

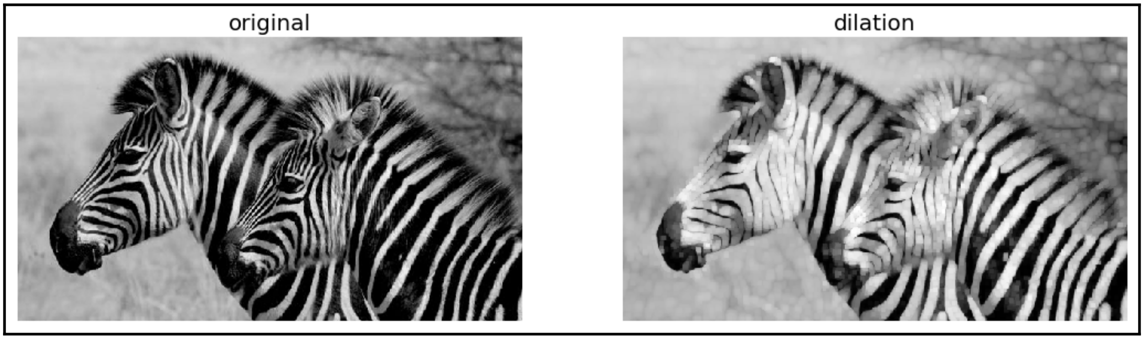

original
removing small objects below size 50
removing small objects below size 200
removing small objects below size 500

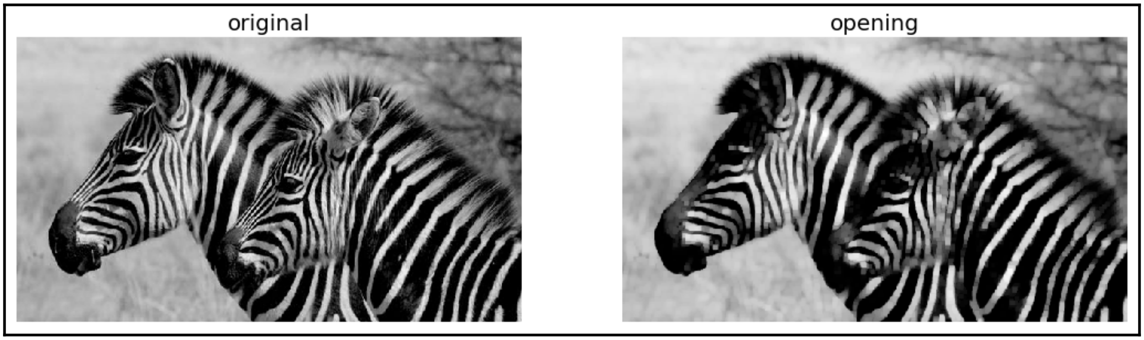

white tophat
black tophat

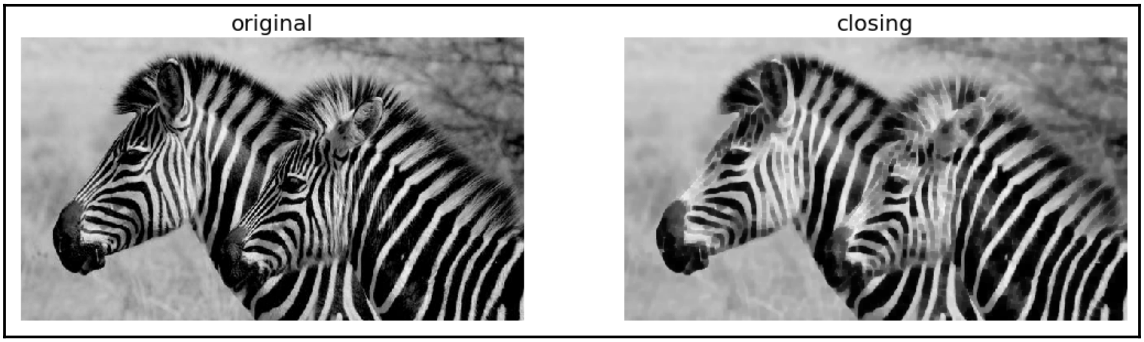

original
boundary


original
opening
closing
opening + closing

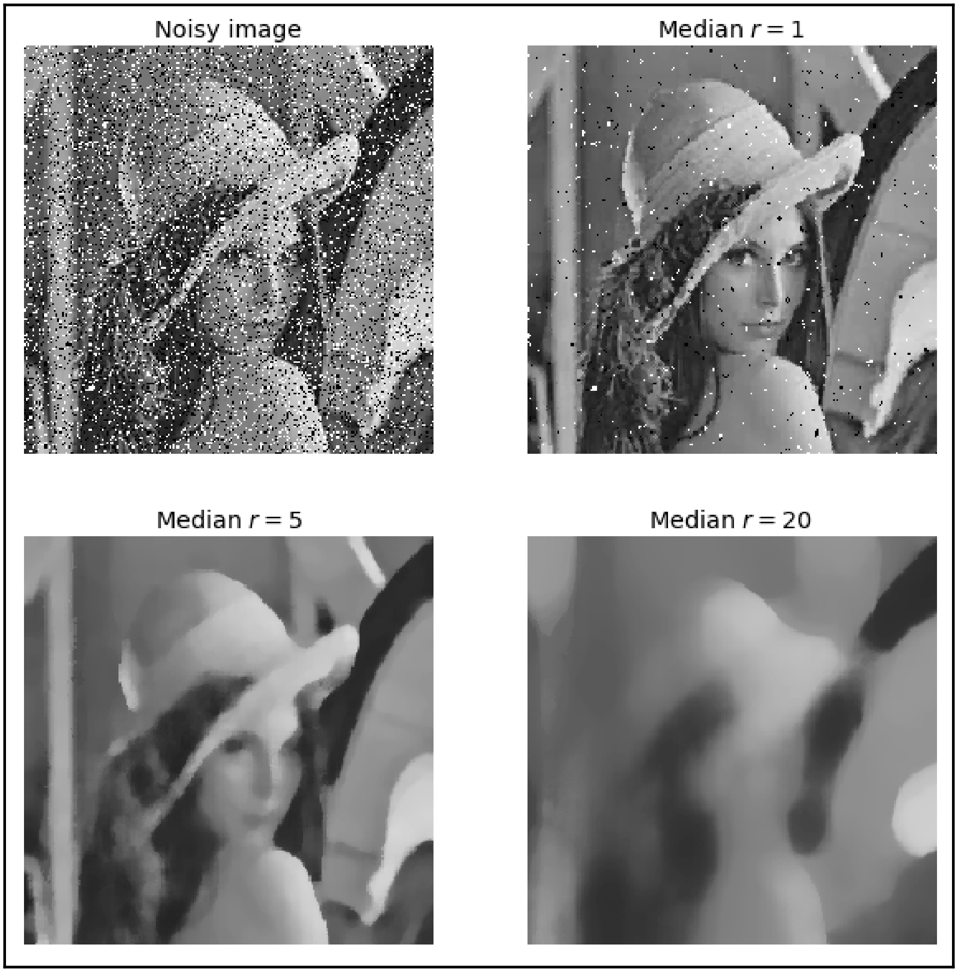

original
erosion


original
dilation


original
opening


original
closing


Original
Local morphological contrast enhancement
Adaptive Histogram equalization


Noisy image
Median r = 1
Median r = 5
Median r = 20

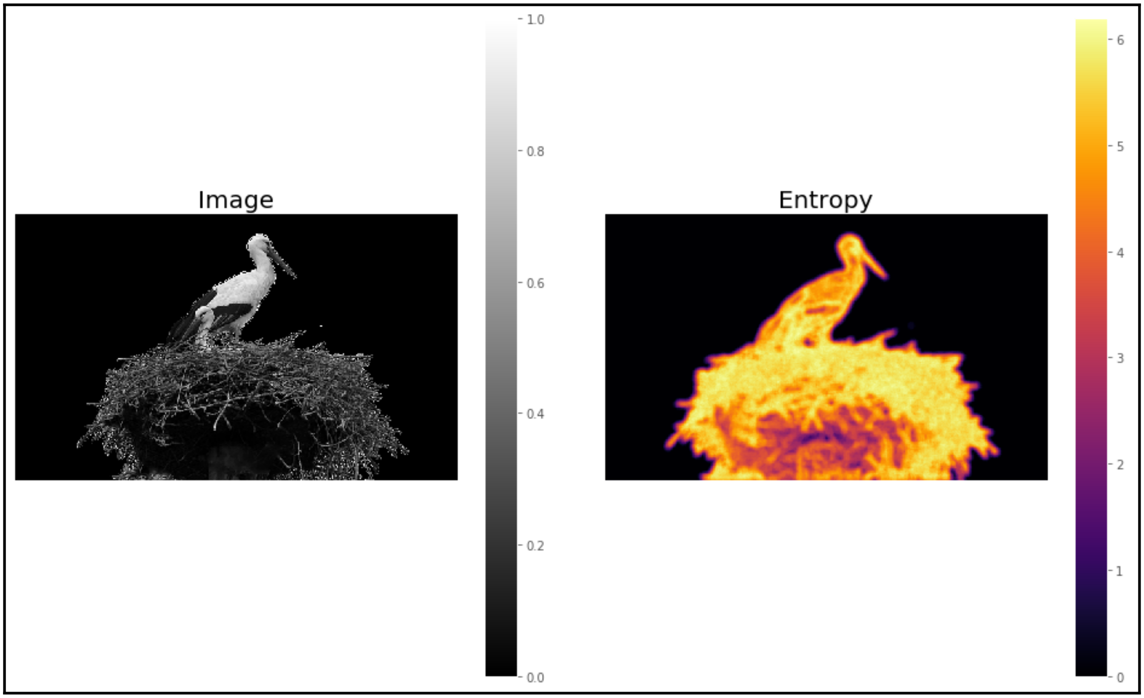
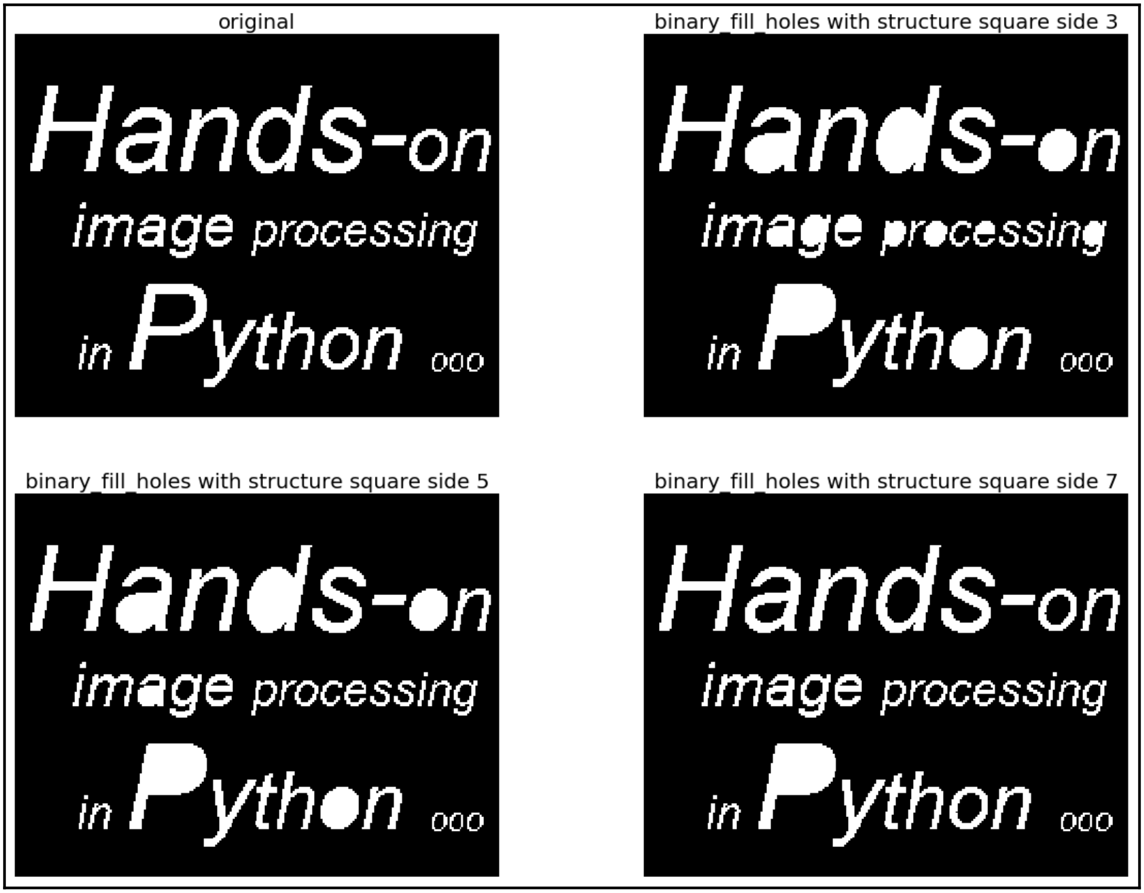
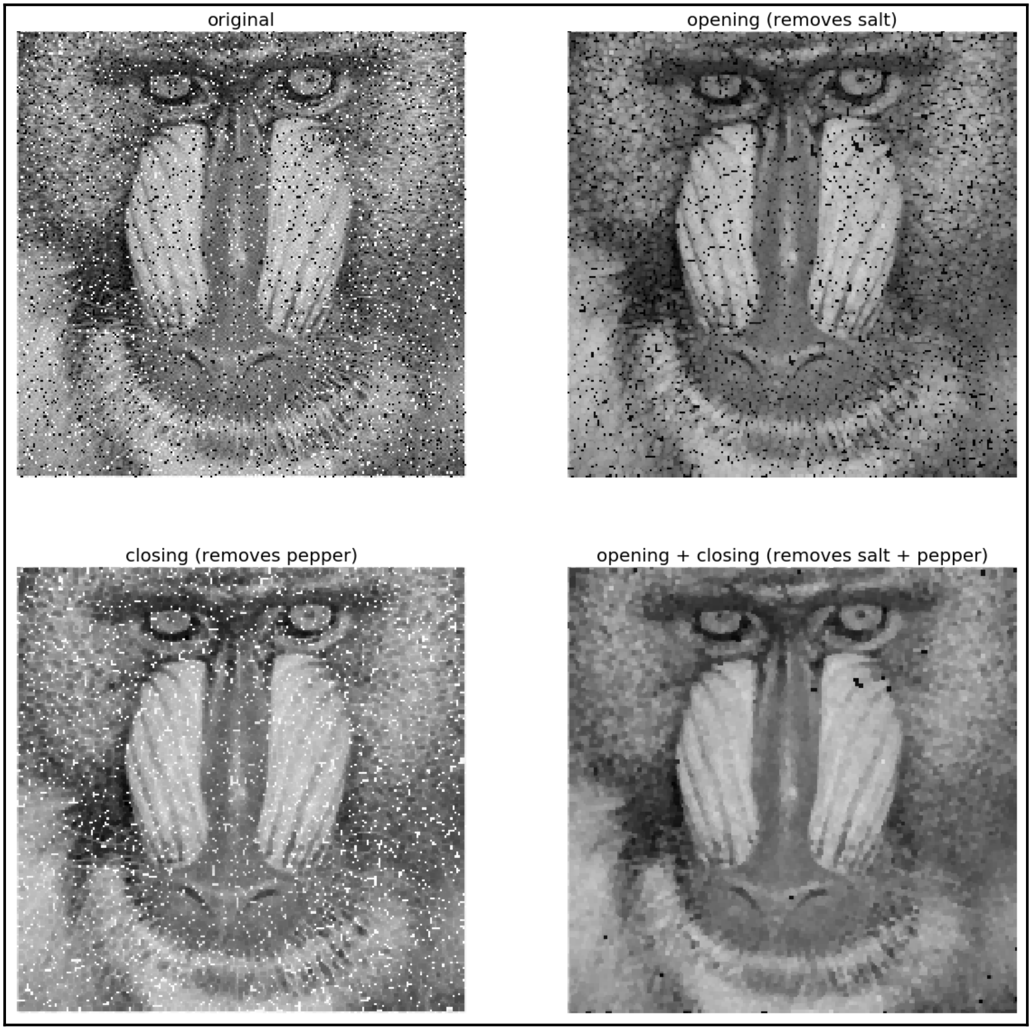
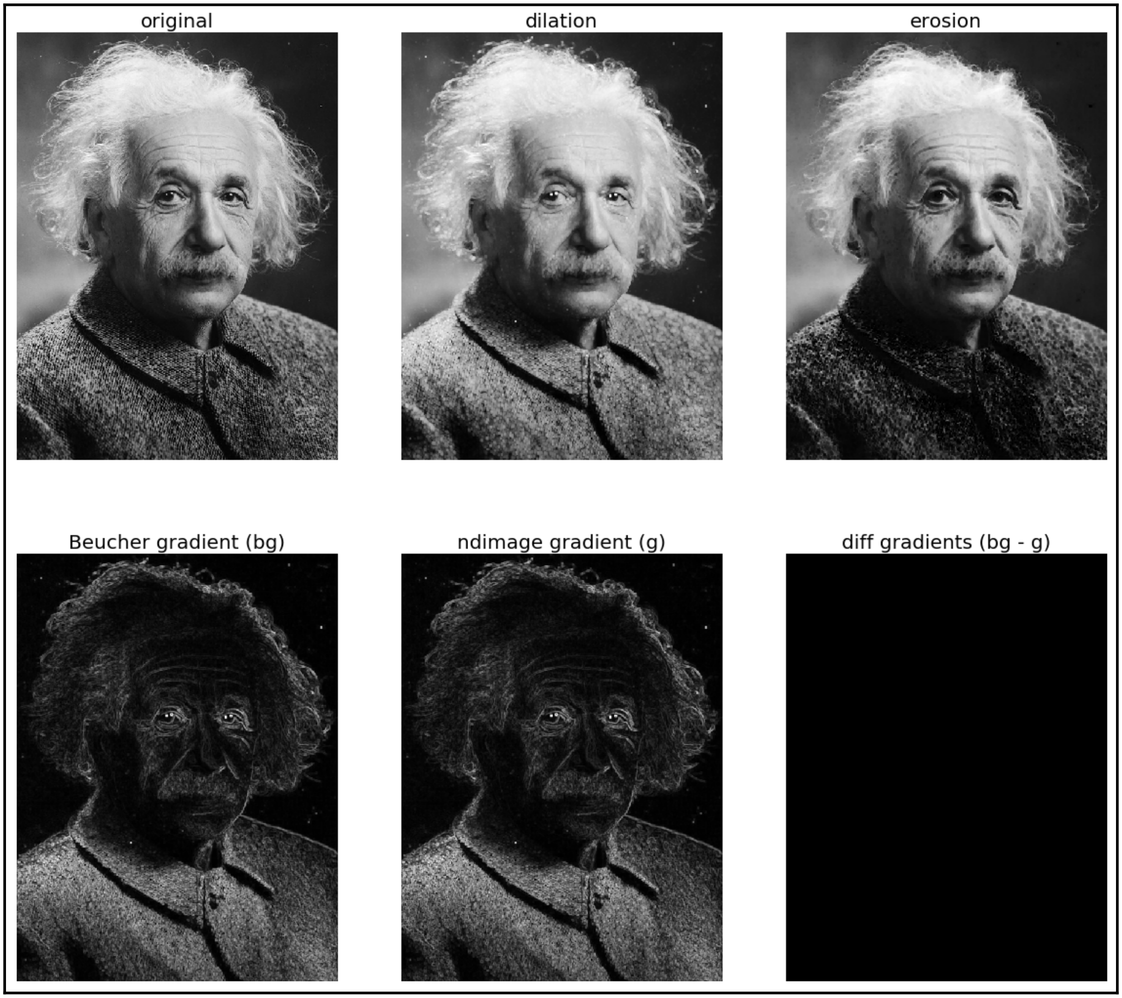
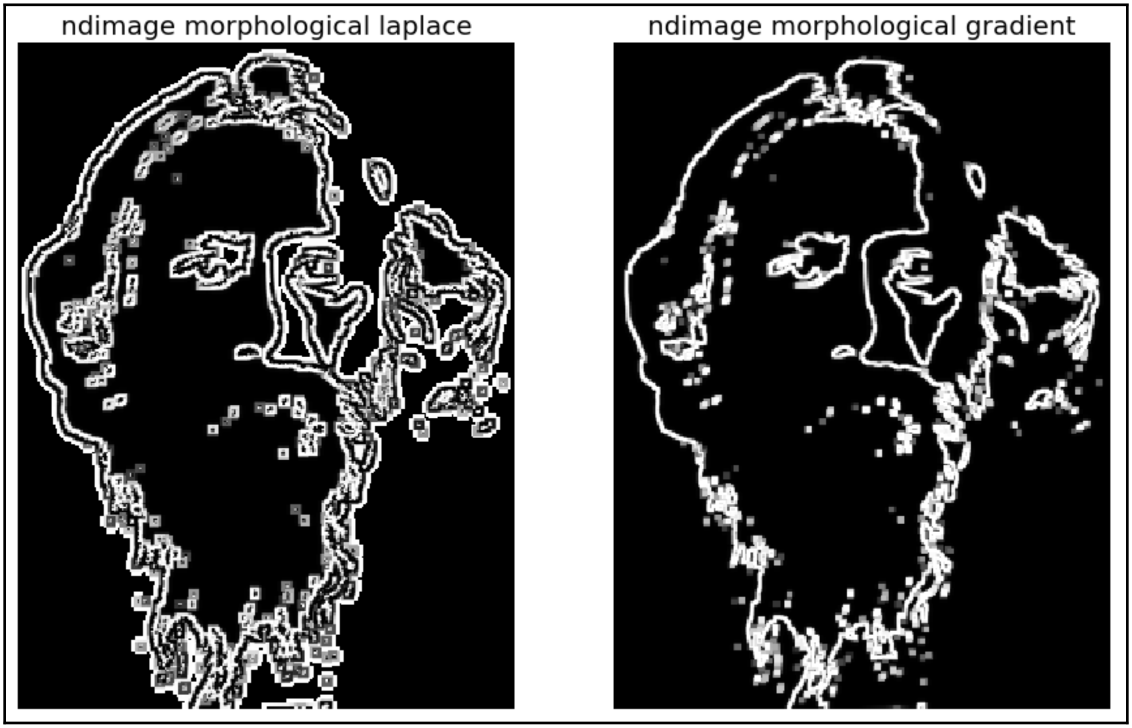


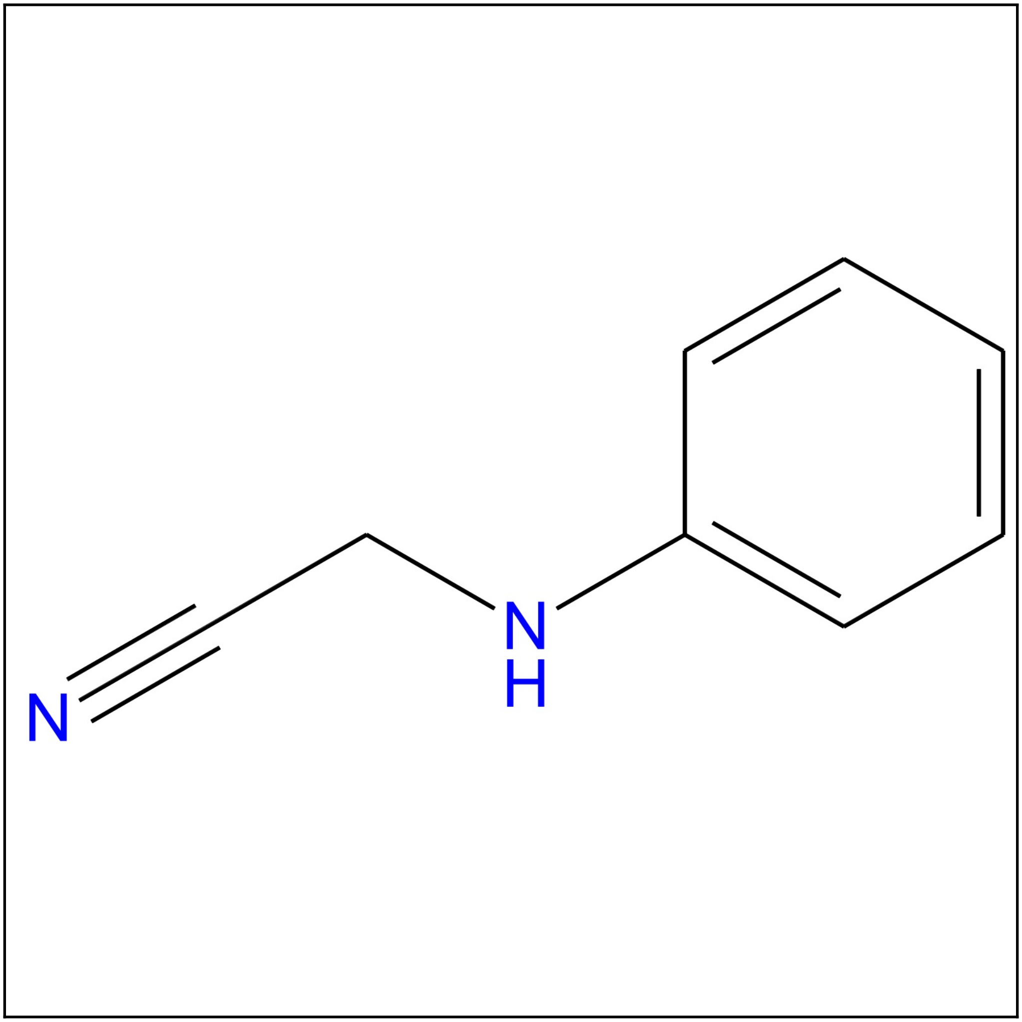
Image
1.0
0.8
0.6
0.4
0.2
0.0
Entropy
6
5
4
3
2
1
0

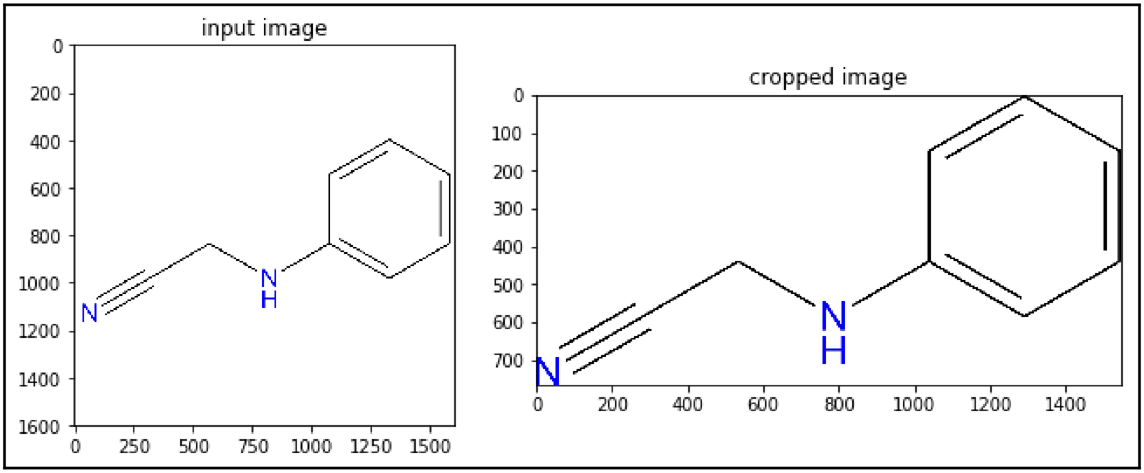

original
binary_fill_holes with structure square side 3
binary_fill_holes with structure square side 5
binary_fill_holes with structure square side 7
Hands-on
image processing
in Python
ooo


original
opening (removes salt)
closing (removes pepper)
opening + closing (removes salt + pepper)

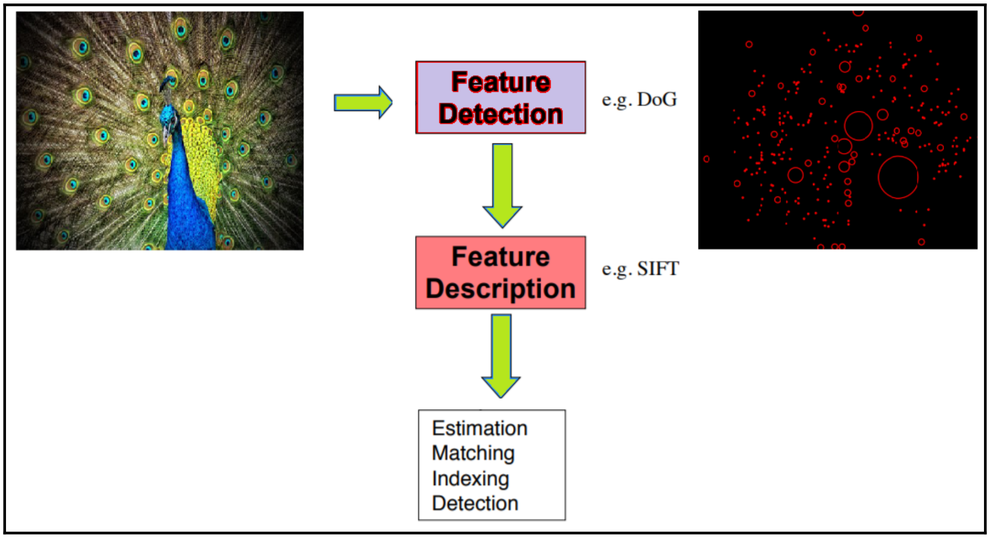

original
dilation
erosion
Beucher gradient (bg)
ndimage gradient (g)
diff gradients (bg - g)

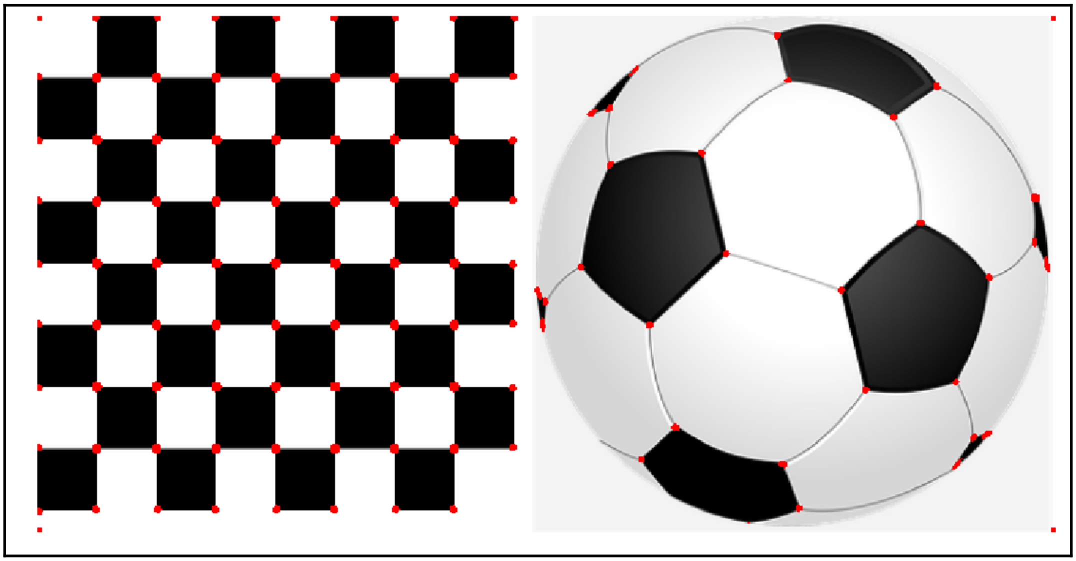

ndimage morphological laplace
ndimage morphological gradient

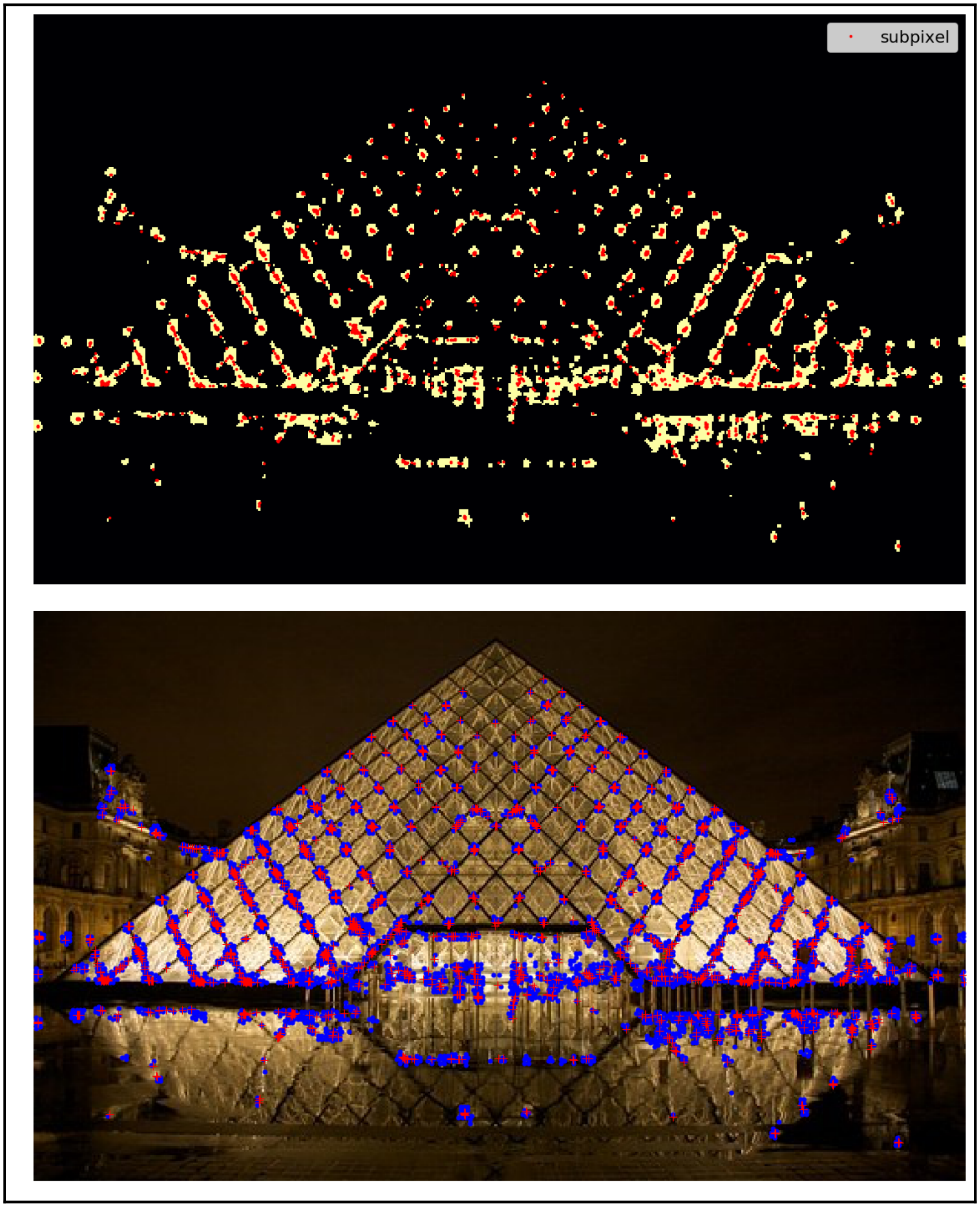

N
N
H

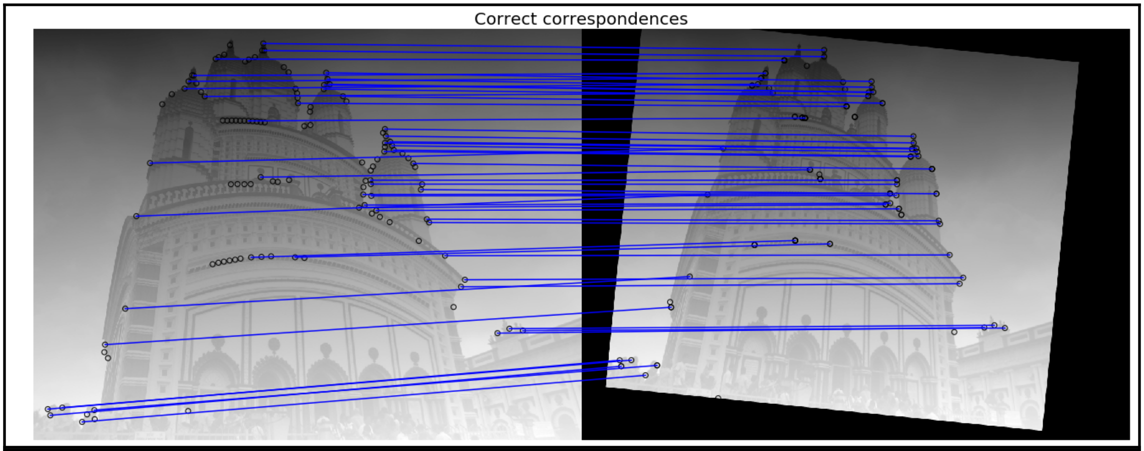

input image
0
200
400
600
800
1000
1200
1400
1600
0
250
500
750
1000
1250
1500
N
N
H
cropped image
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200
1400
N
N
H

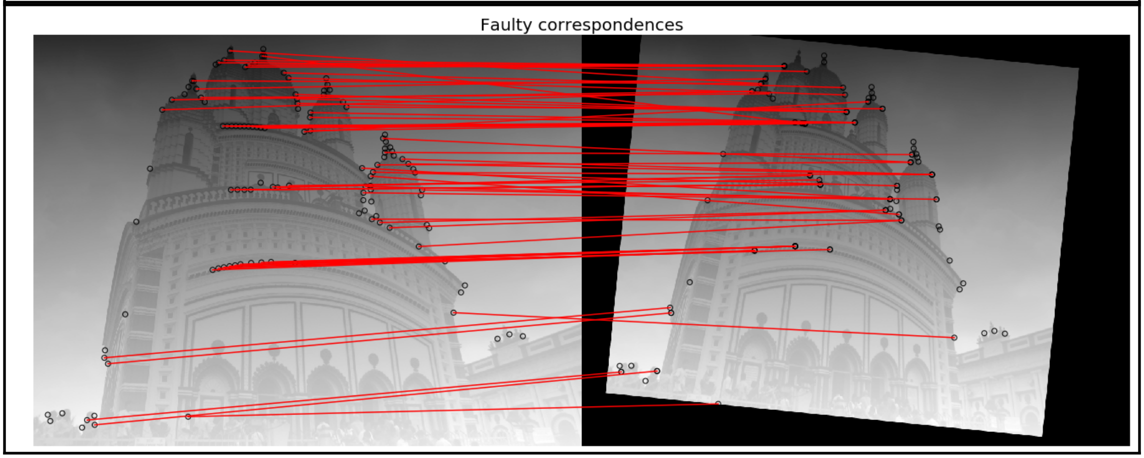
Chapter 7: Extracting Image Features and Descriptors

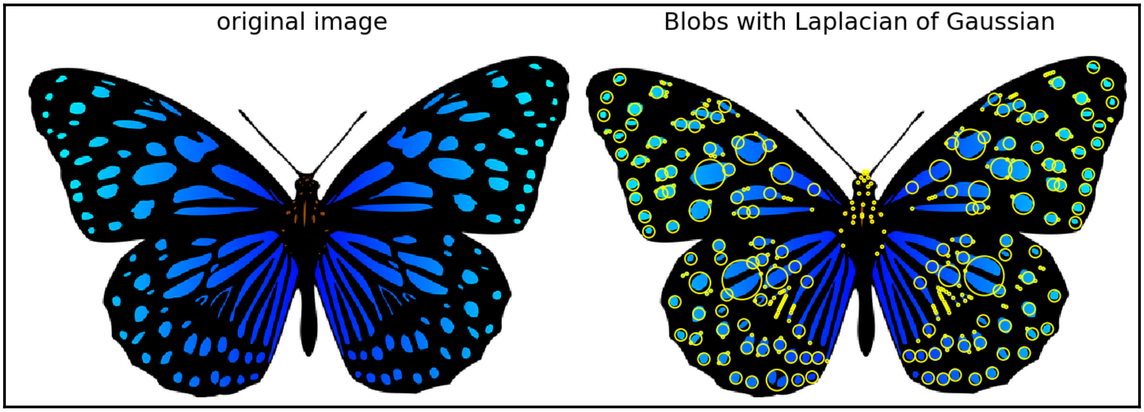


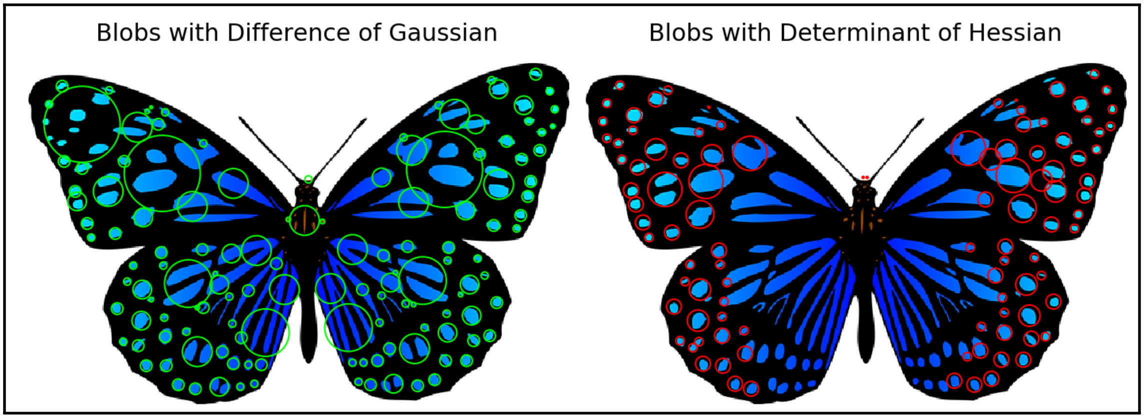


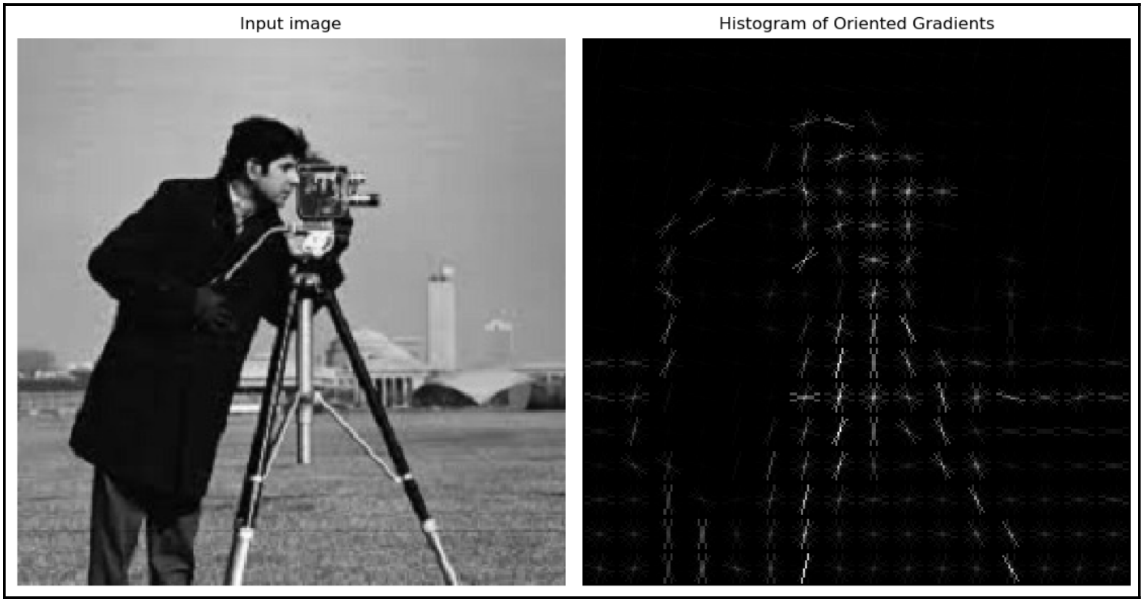

subpixel

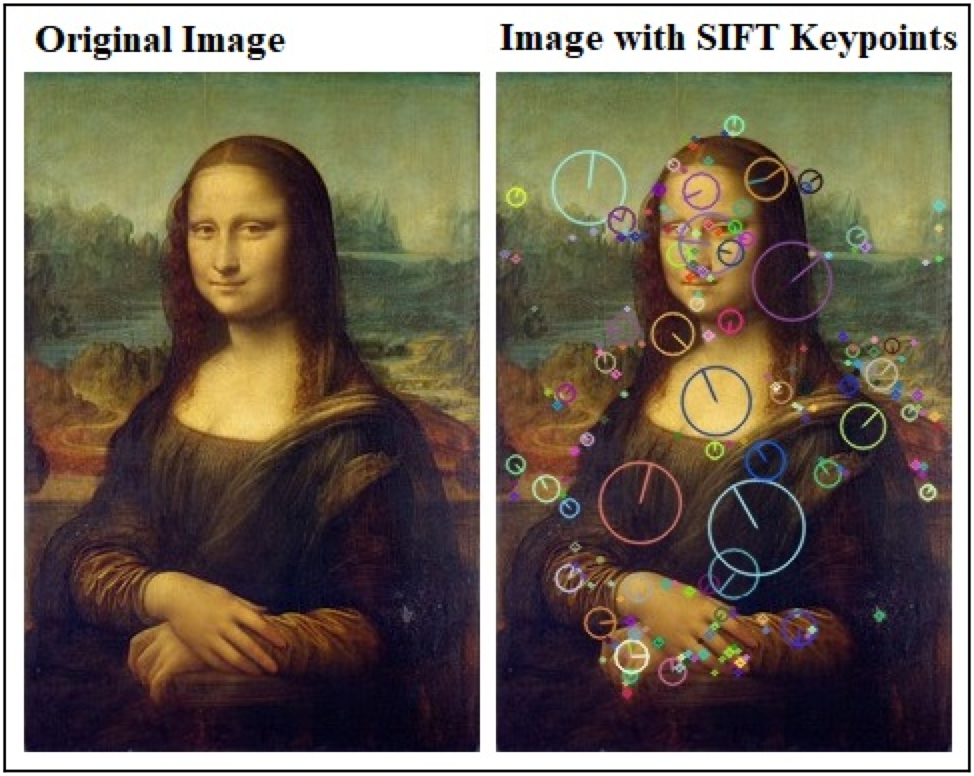

Correct correspondences

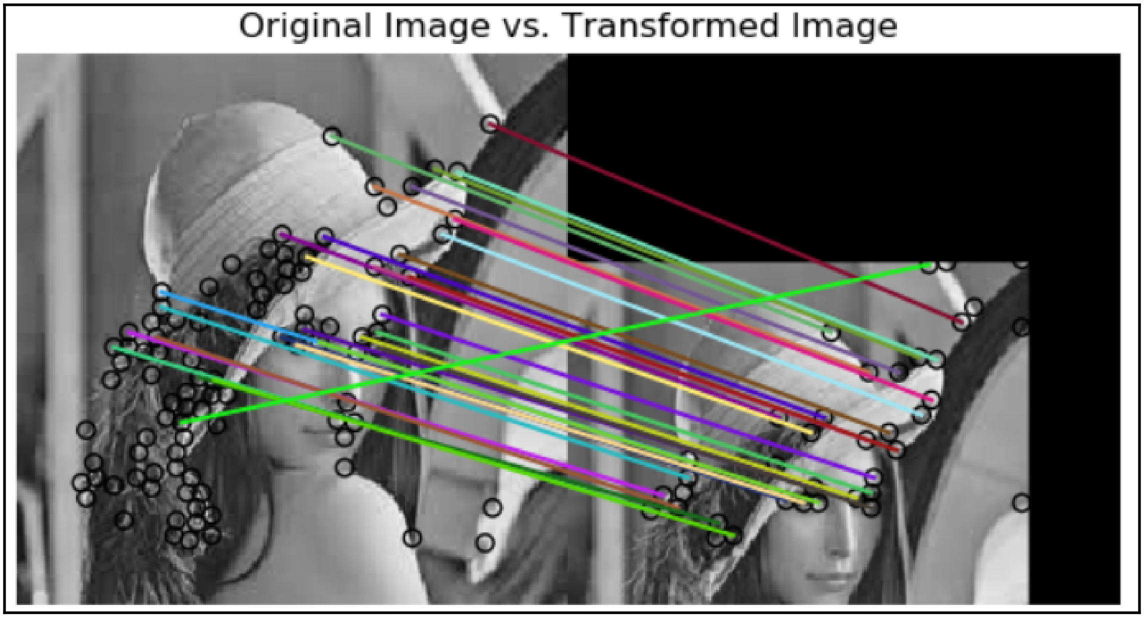

Faulty correspondences


original image
Blobs with Laplacian of Gaussian


Blobs with Difference of Gaussian
Blobs with Determinant of Hessian


Input image
Histogram of Oriented Gradients


Original Image
Image with SIFT Keypoints


Original Image vs. Transformed Image

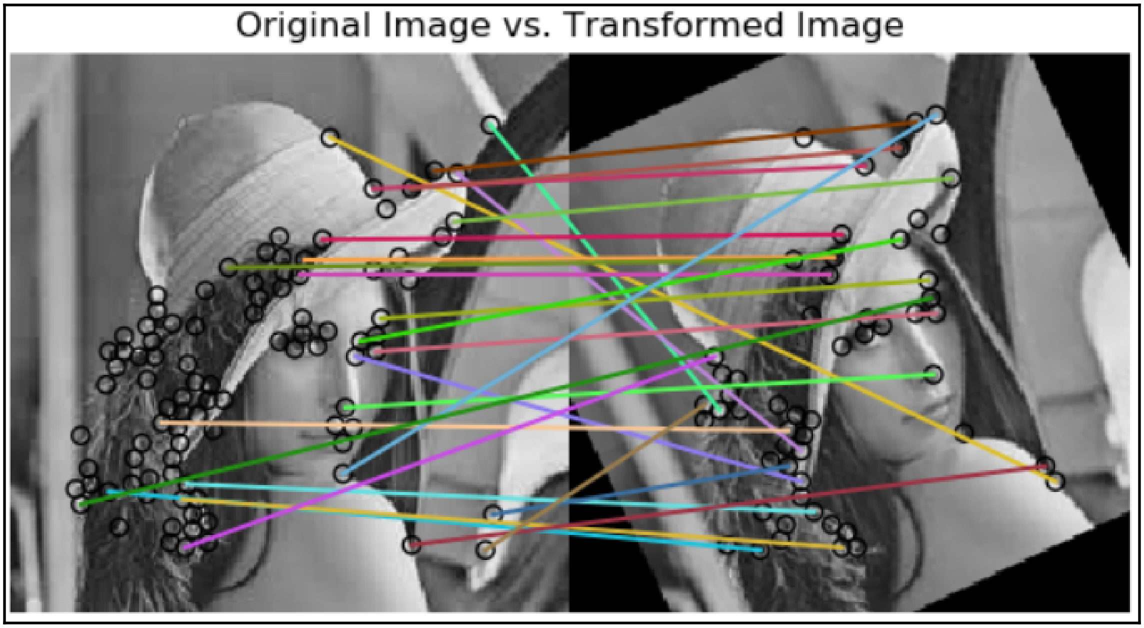
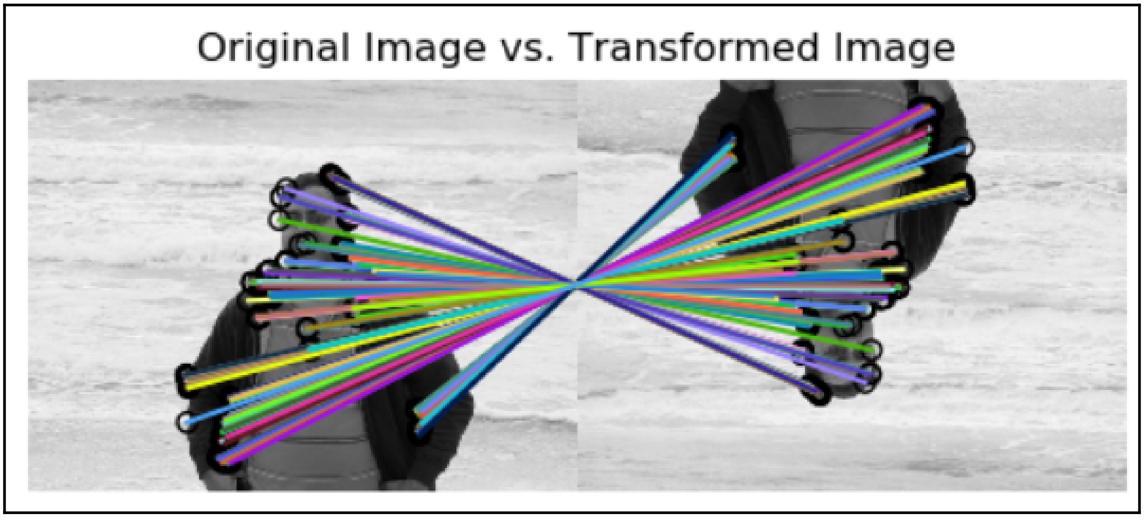
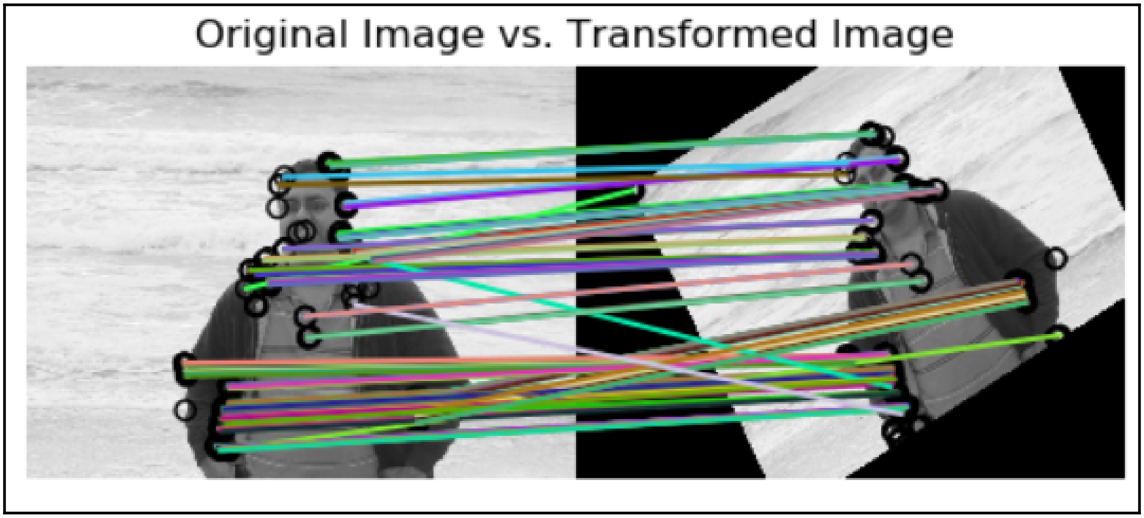

Original Image vs. Transformed Image


Image1
Image2


Original Image vs. Transformed Image


Original Image vs. Transformed Image

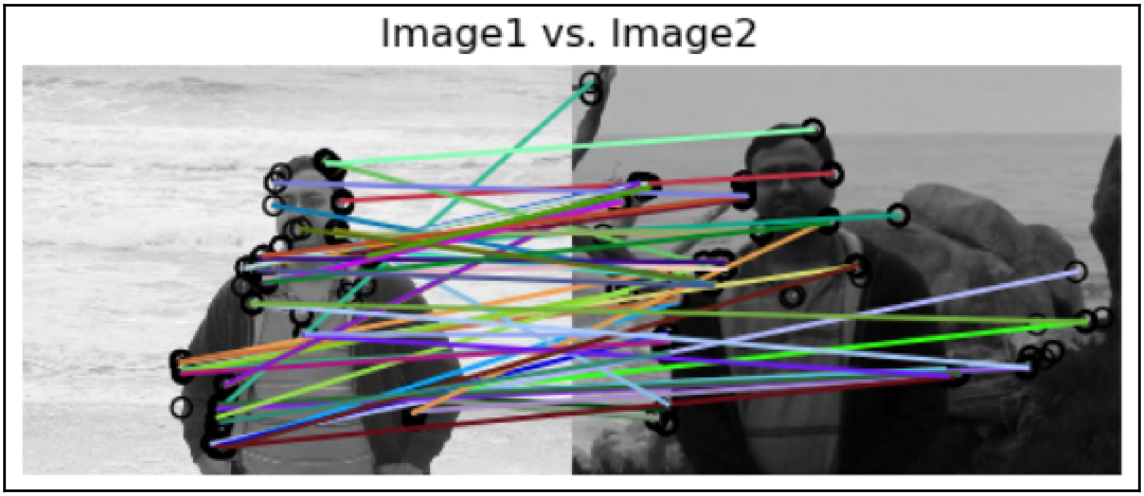
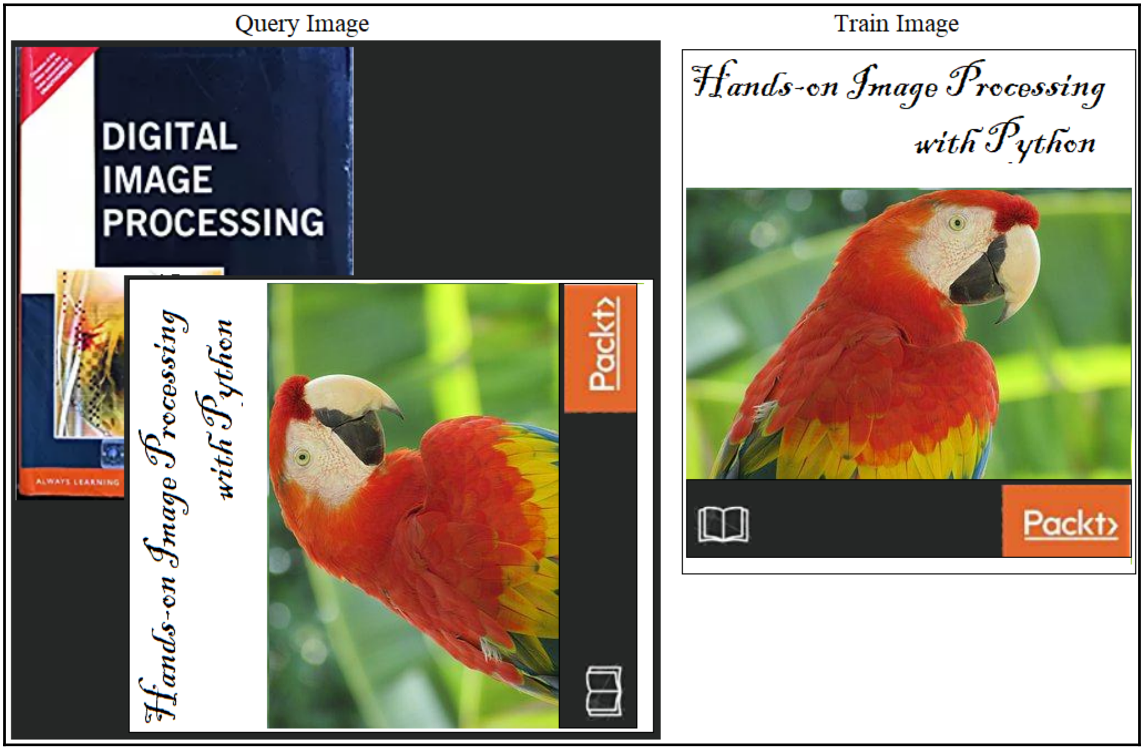
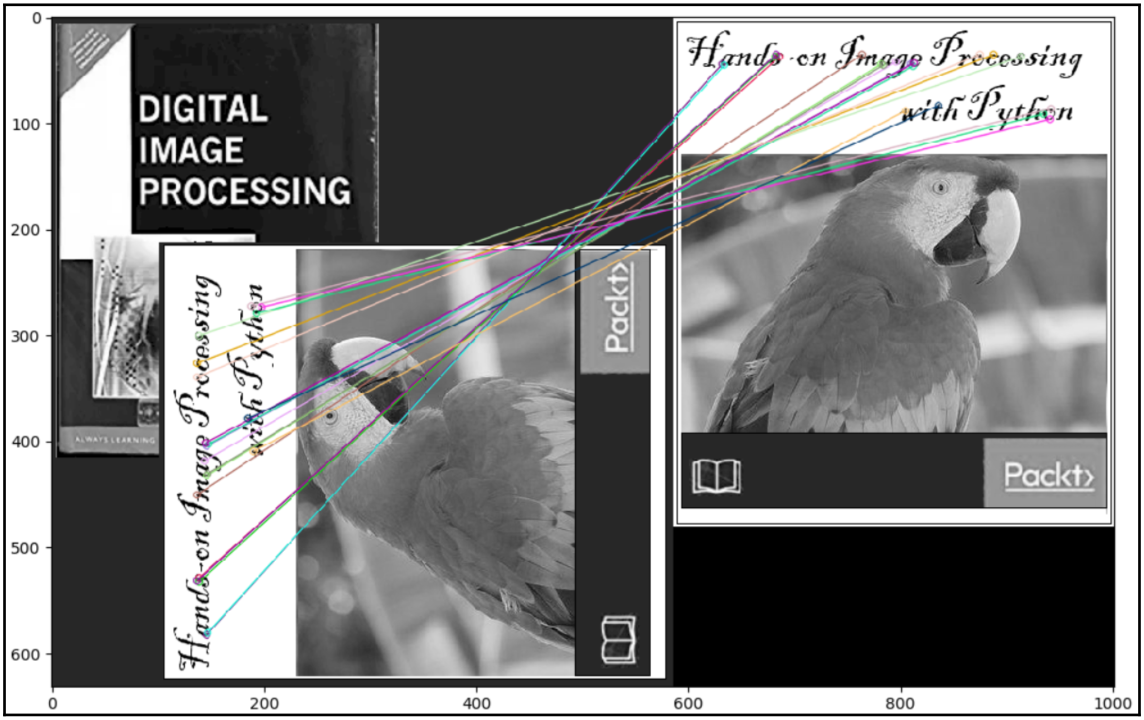
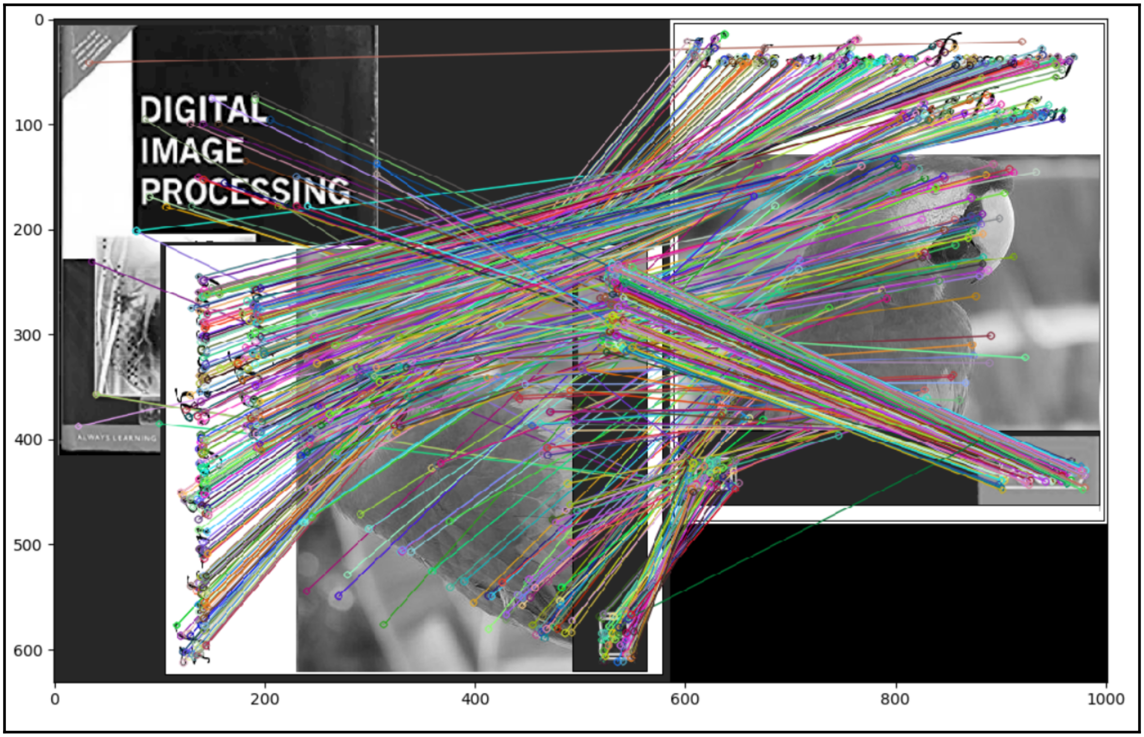
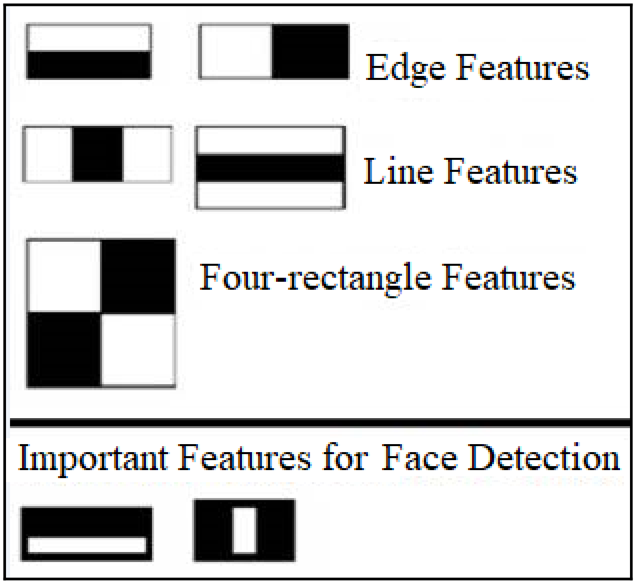


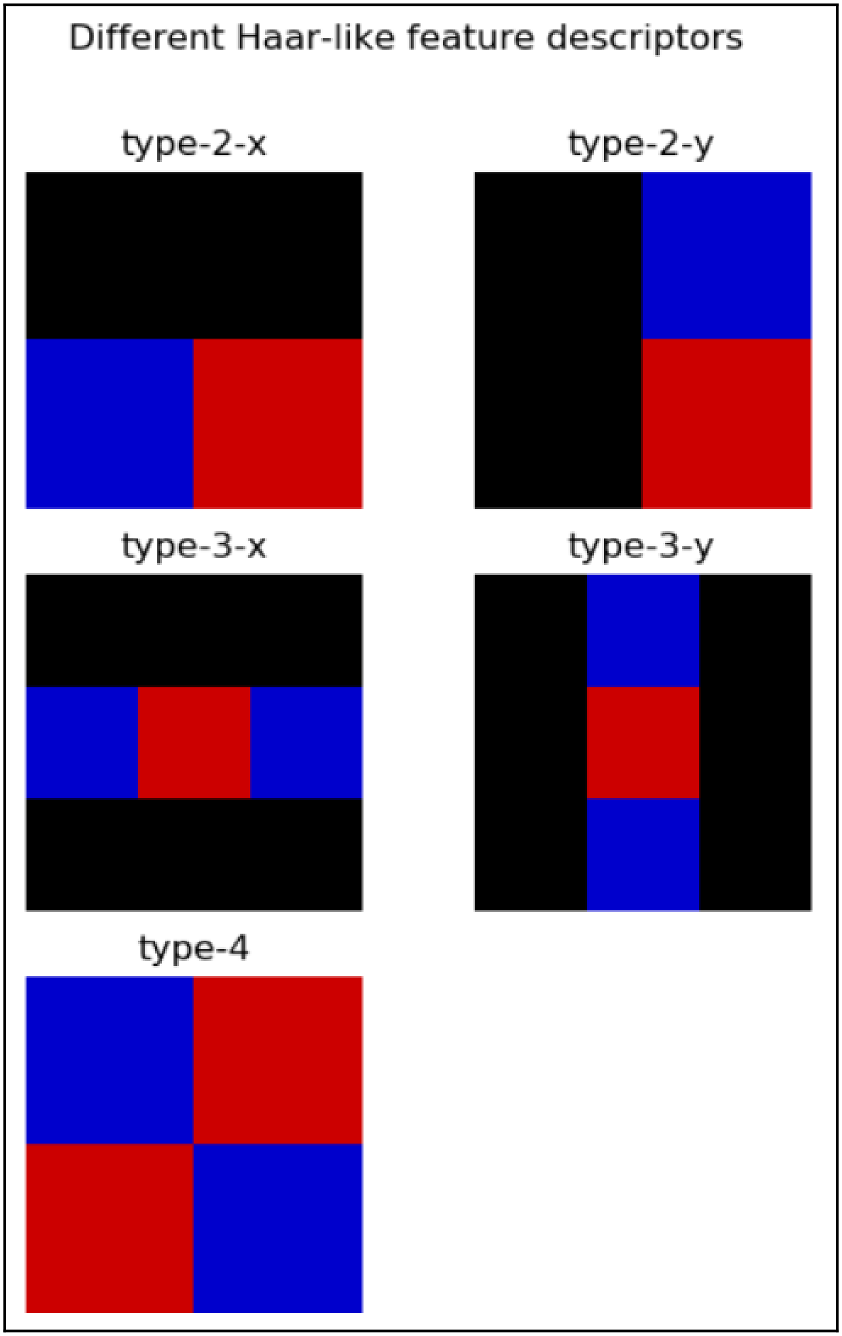
Image1 vs. Image2

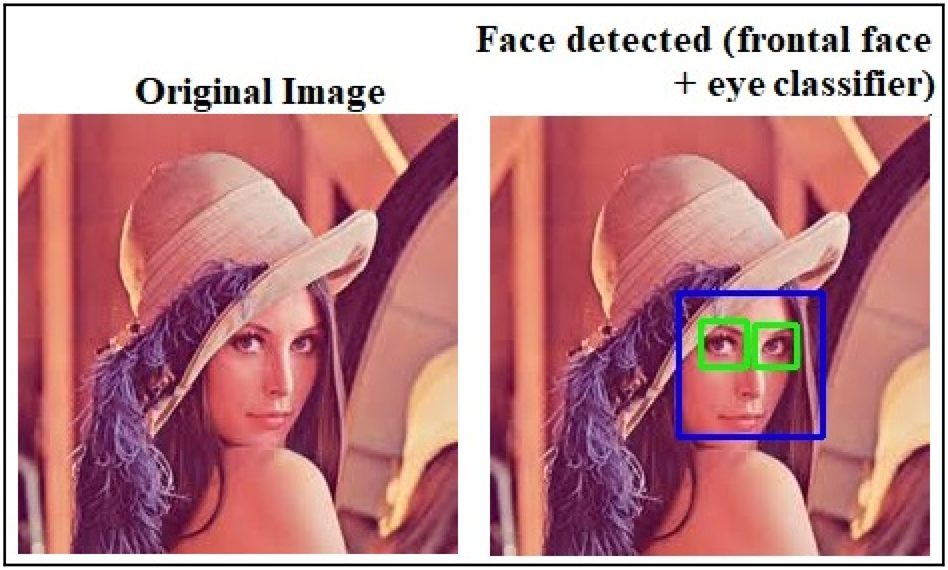

Query Image
Train Image
DIGITAL IMAGE PROCESSING
ALWAYS LEARNING
Hands-on Image Processing with Python
Packt>
Hands-on Image Processing with Python
Packt>

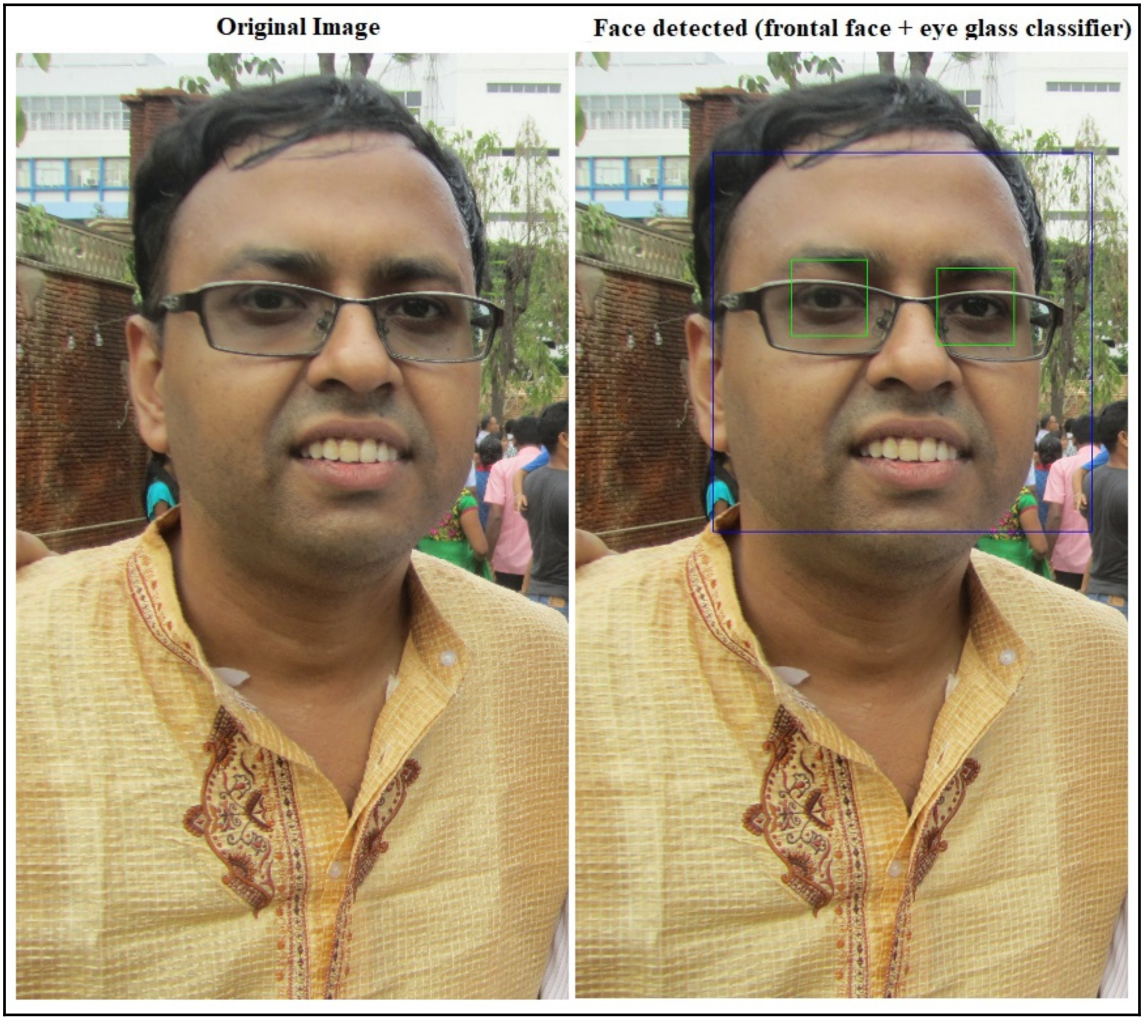

DIGITAL
IMAGE
PROCESSING
Hands-on Image Processing
with Python
Packt>
0
100
200
300
400
500
600
0
200
400
600
800
1000


DIGITAL
IMAGE
PROCESSING
0
100
200
300
400
500
600
0
200
400
600
800
1000

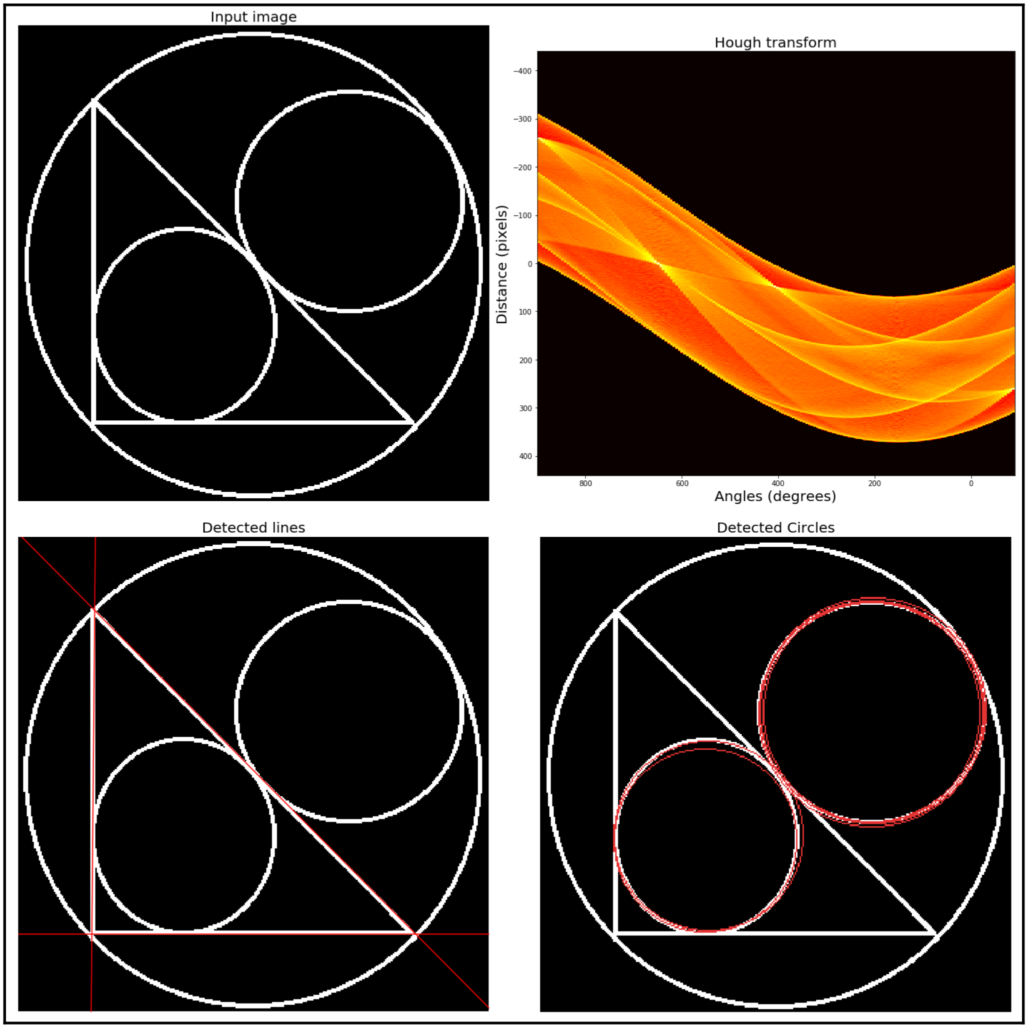

Edge Features
Line Features
Four-rectangle Features
Important Features for Face Detection

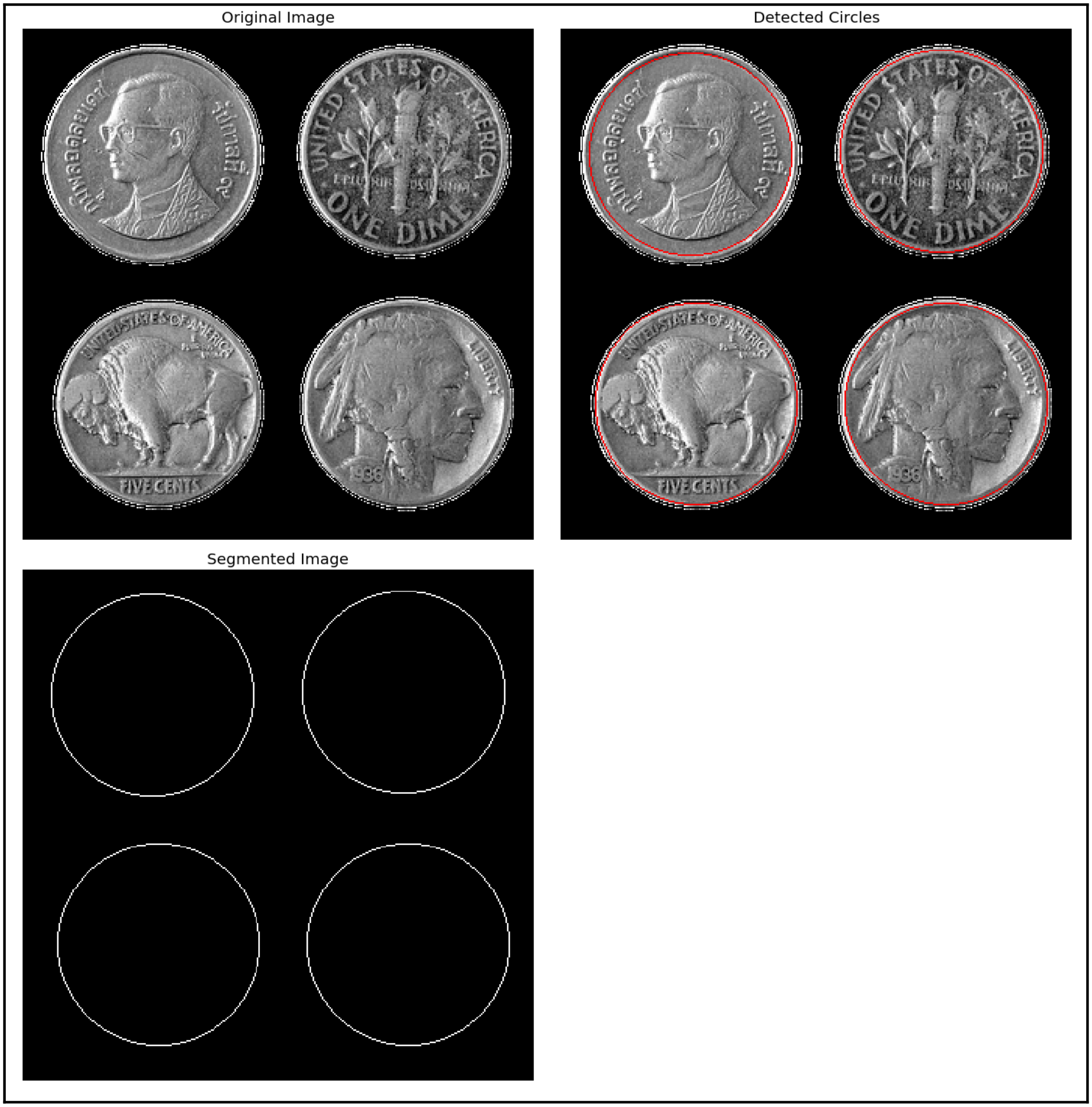

Different Haar-like feature descriptors
type-2-x
type-2-y
type-3-x
type-3-y
type-4

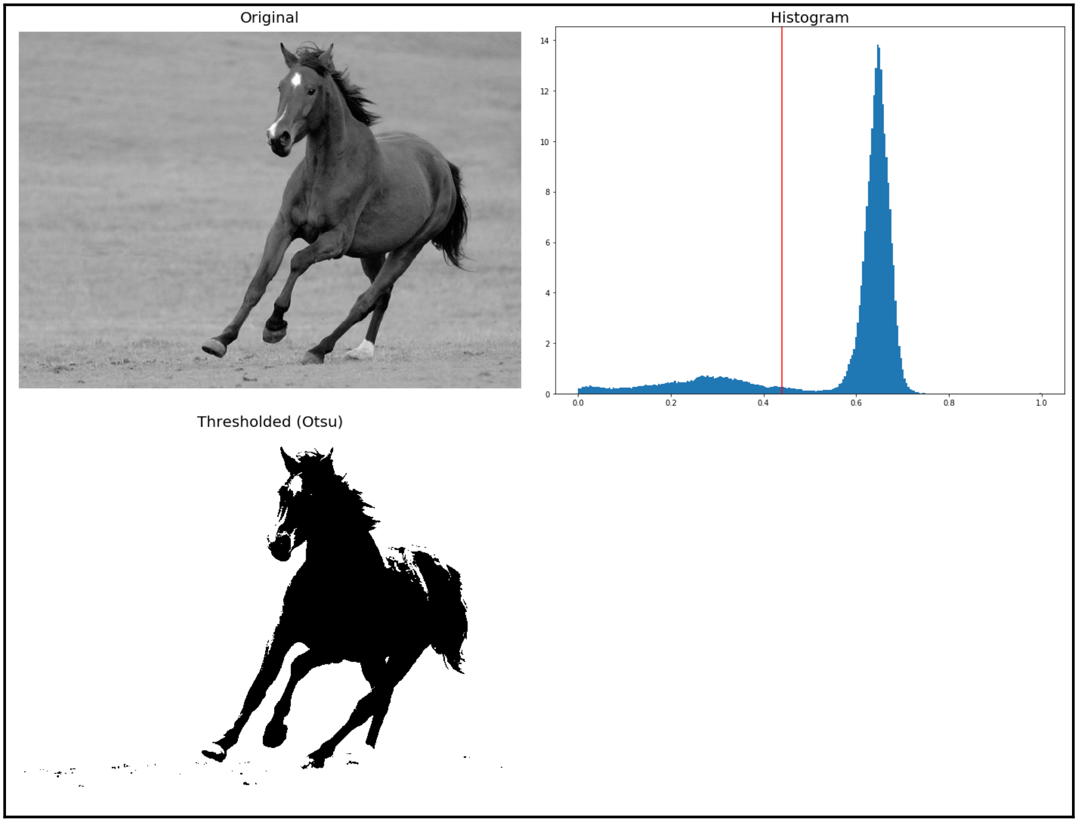

Original Image
Face detected (frontal face + eye classifier)

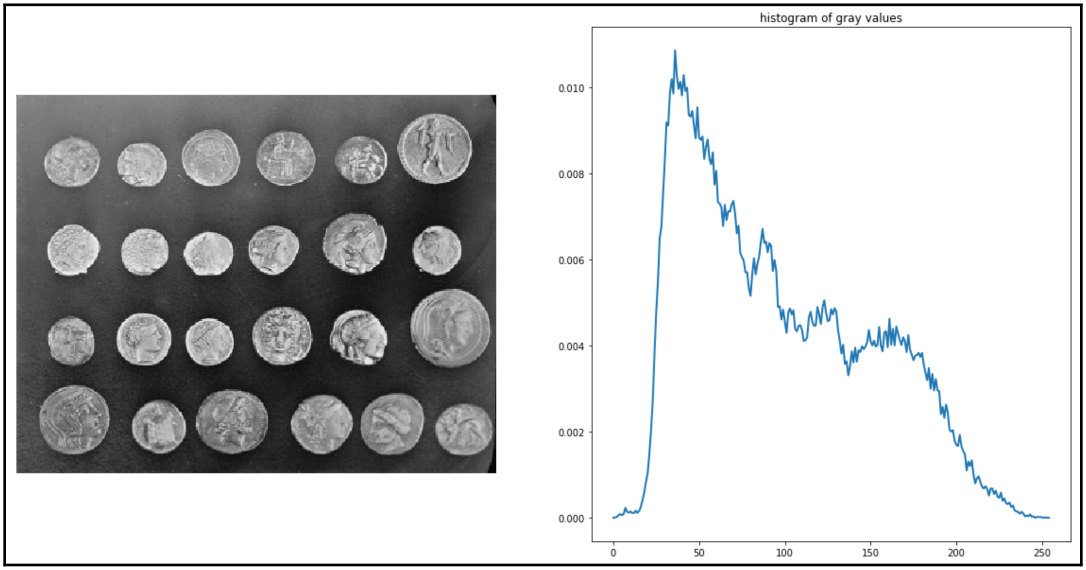

Original Image
Face detected (frontal face + eye glass classifier)

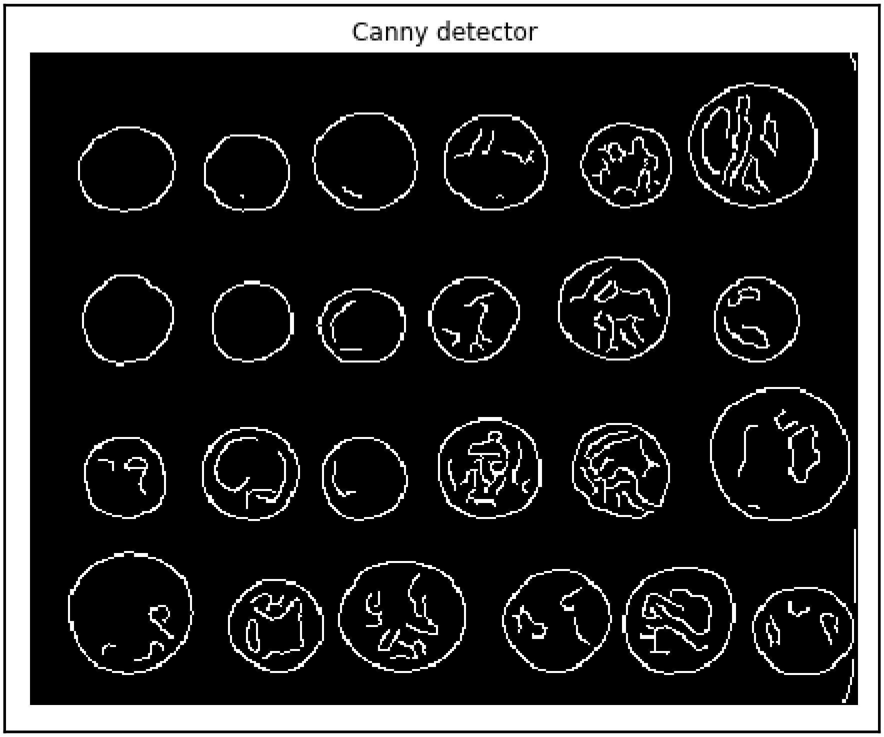
Chapter 8: Image Segmentation

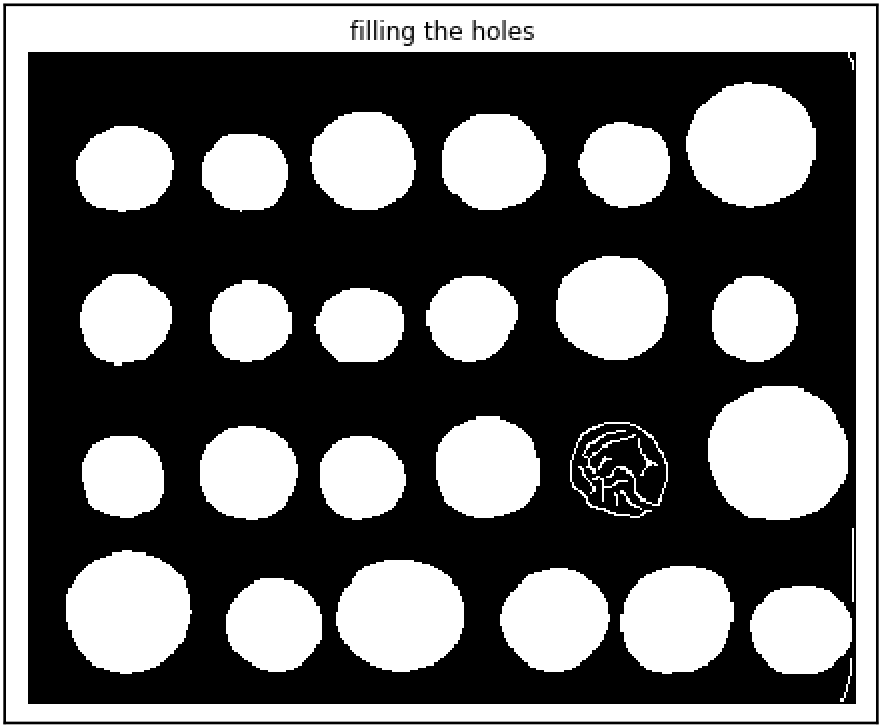


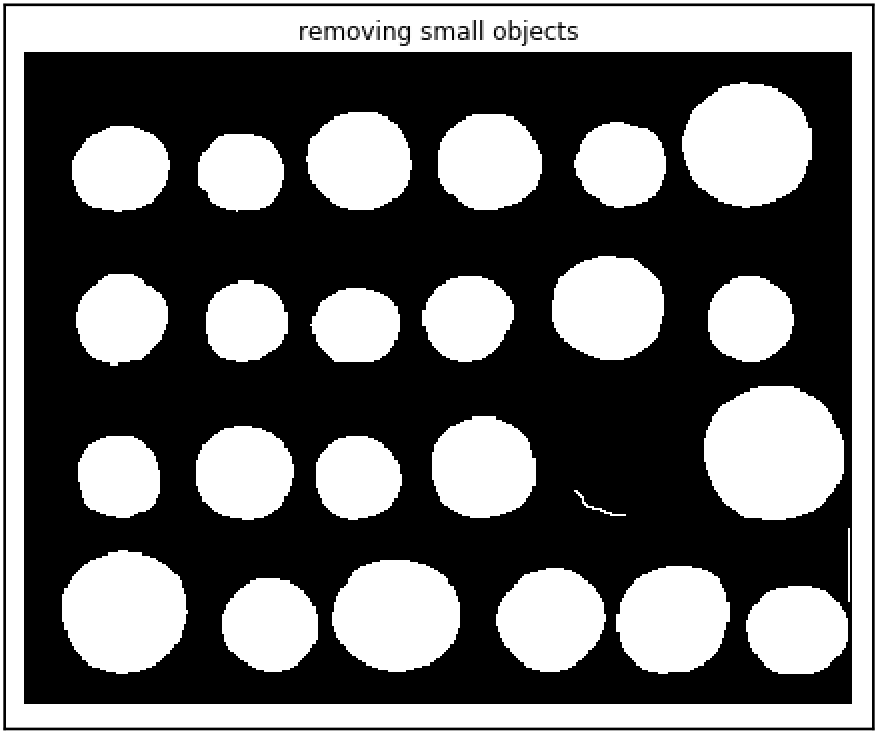

Original Image
Detected Circles
Segmented Image

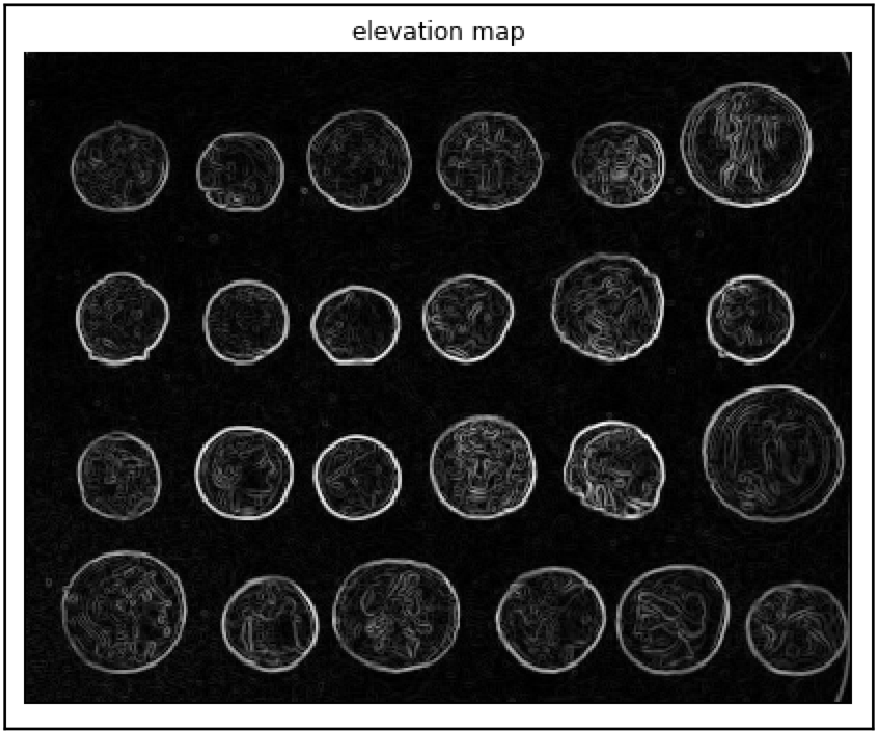

Original
Histogram
Thresholded (Otsu)
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
14

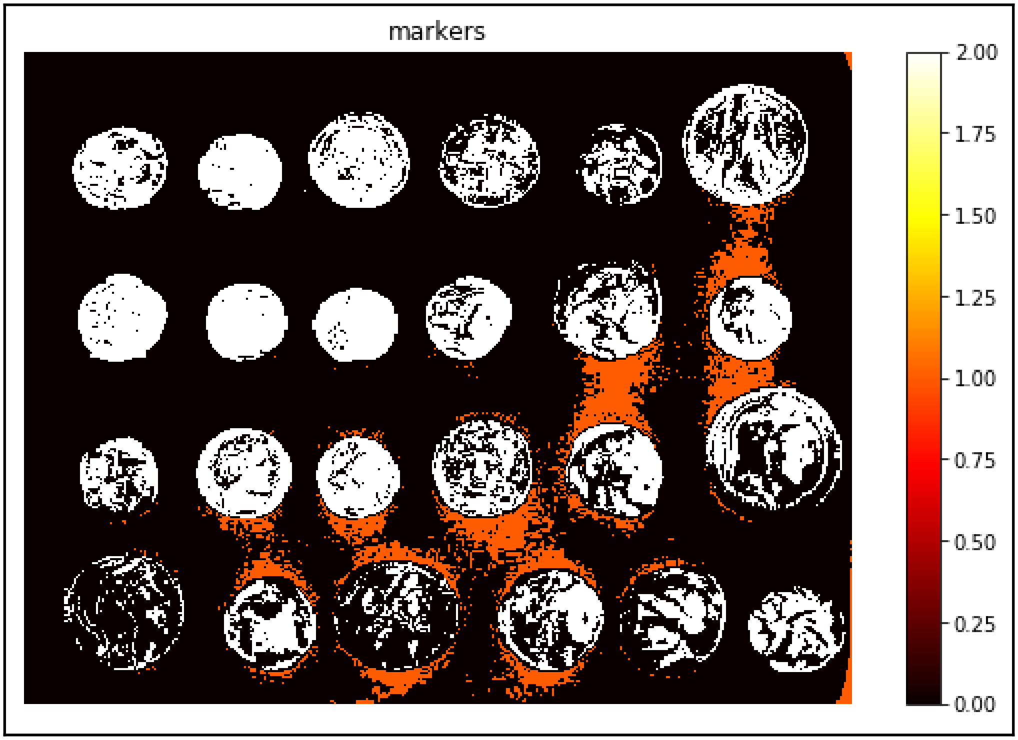

histogram of gray values
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250

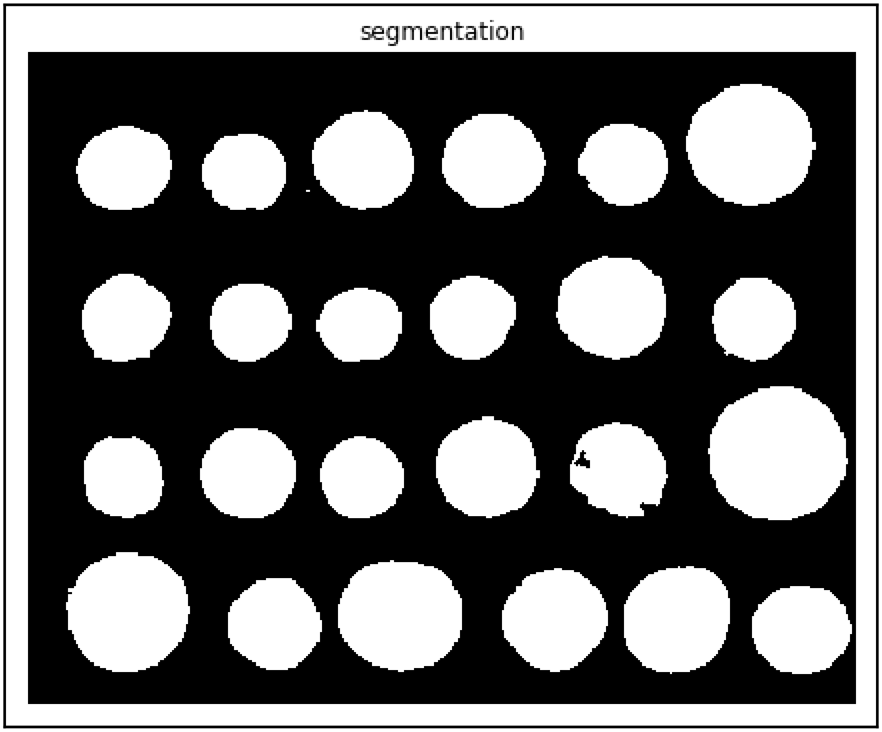

Canny detector

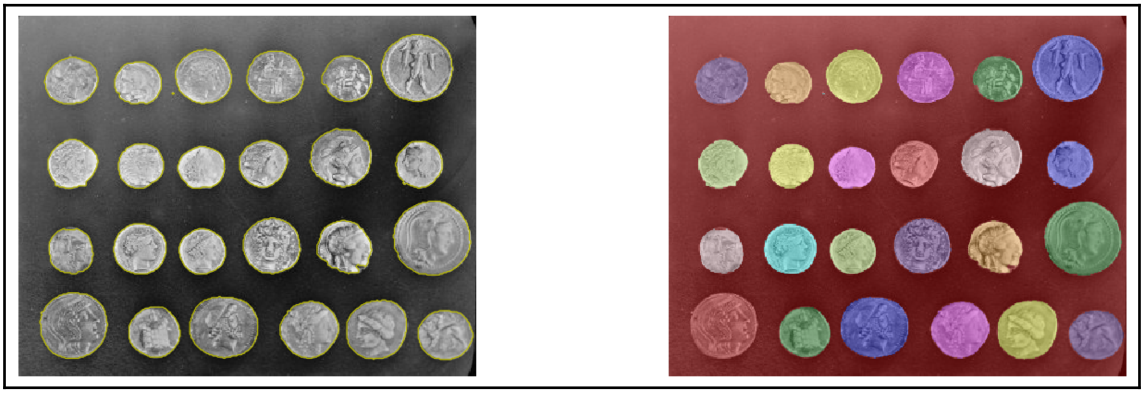

filling the holes

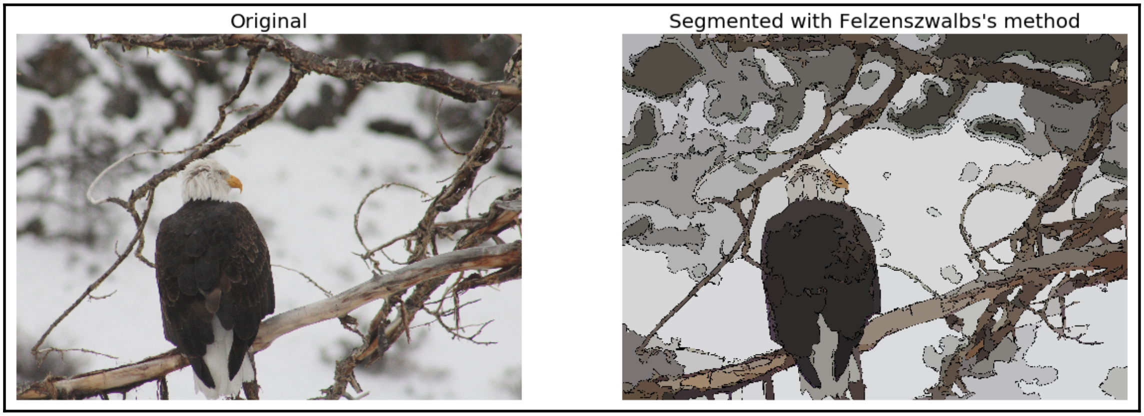

removing small objects

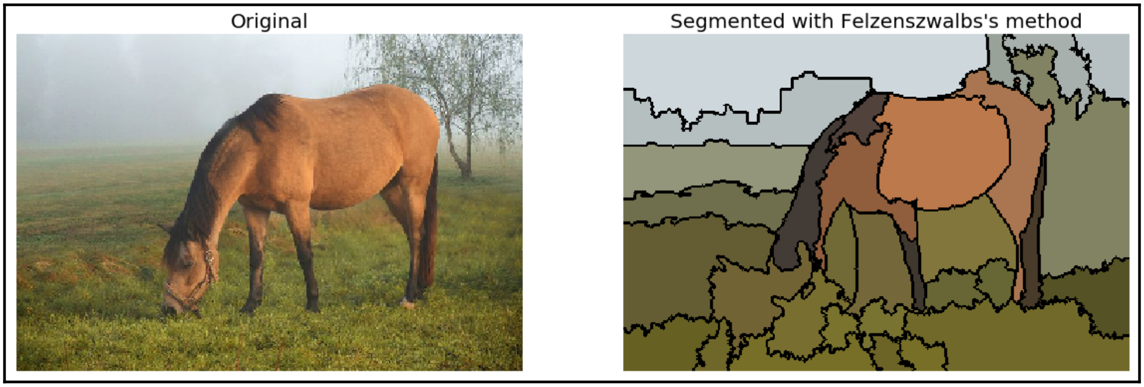

elevation map

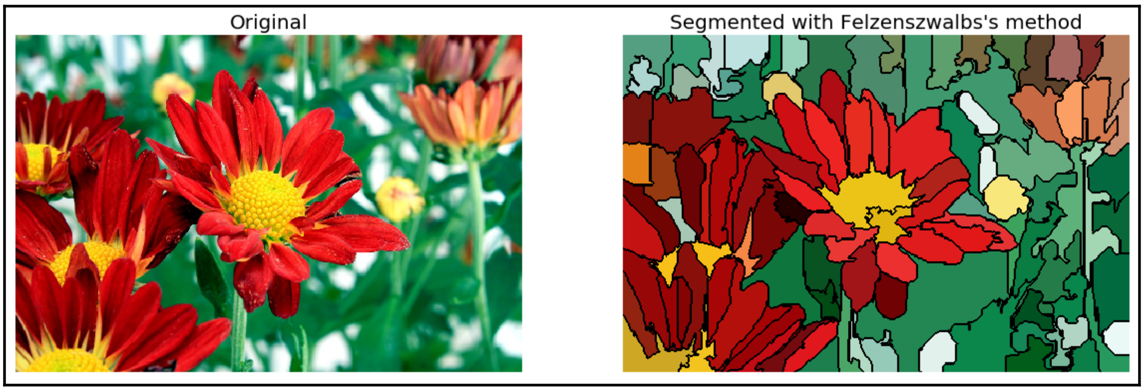

markers
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

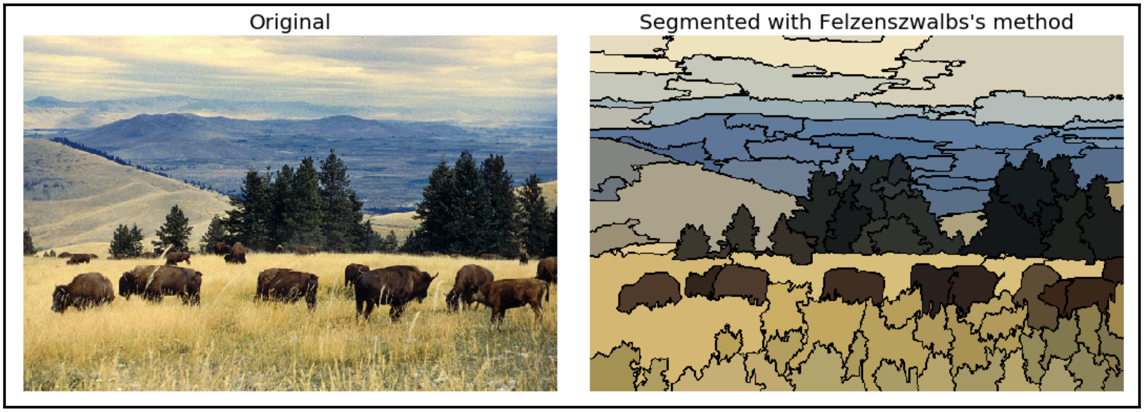

segmentation

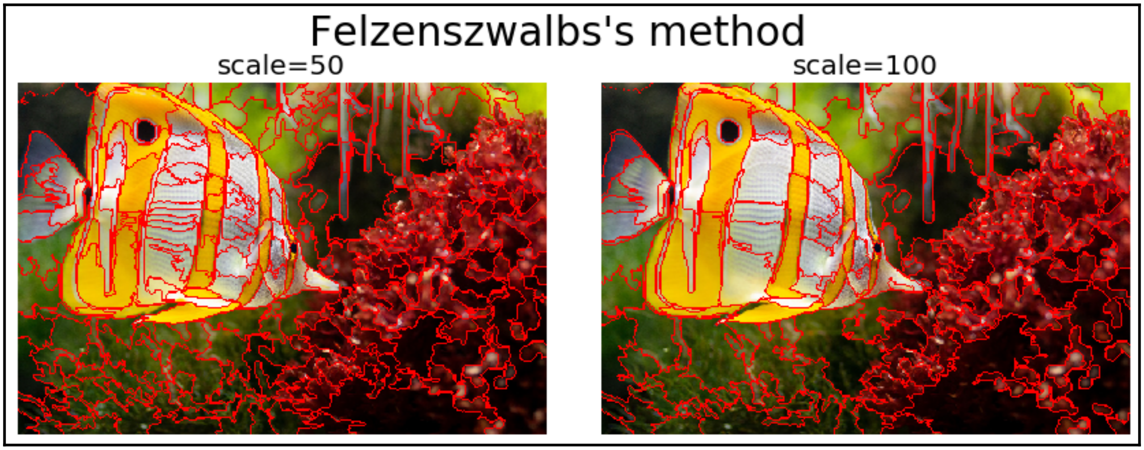



Original
Segmented with Felzenszwalbs's method

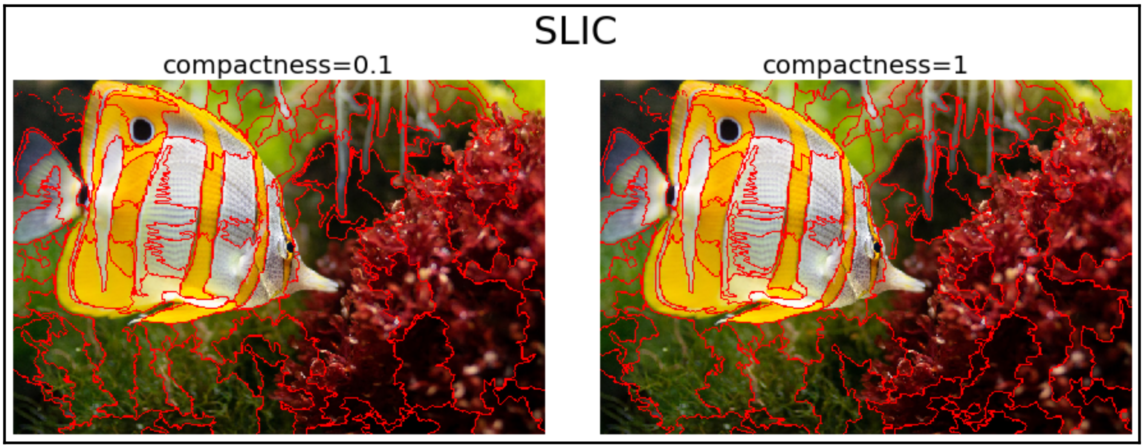

Original
Segmented with Felzenszwalbs's method

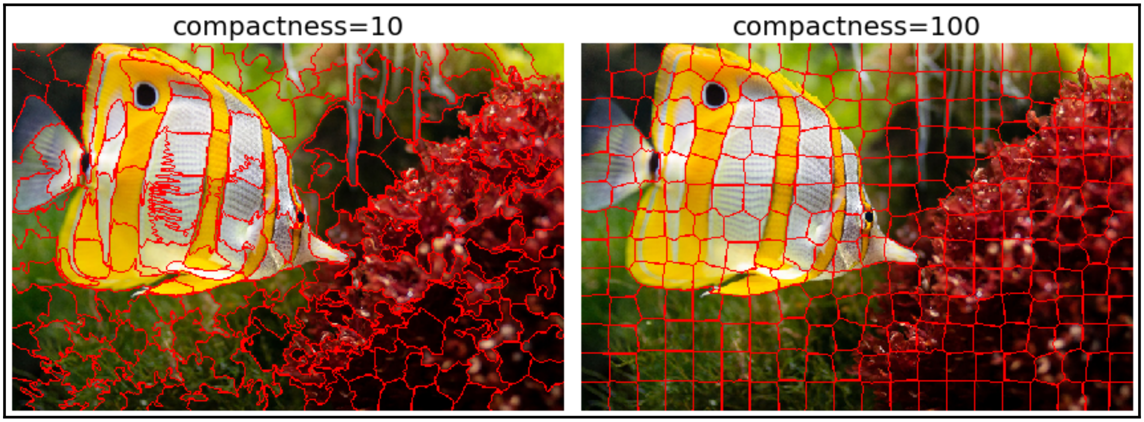

Original
Segmented with Felzenszwalbs's method

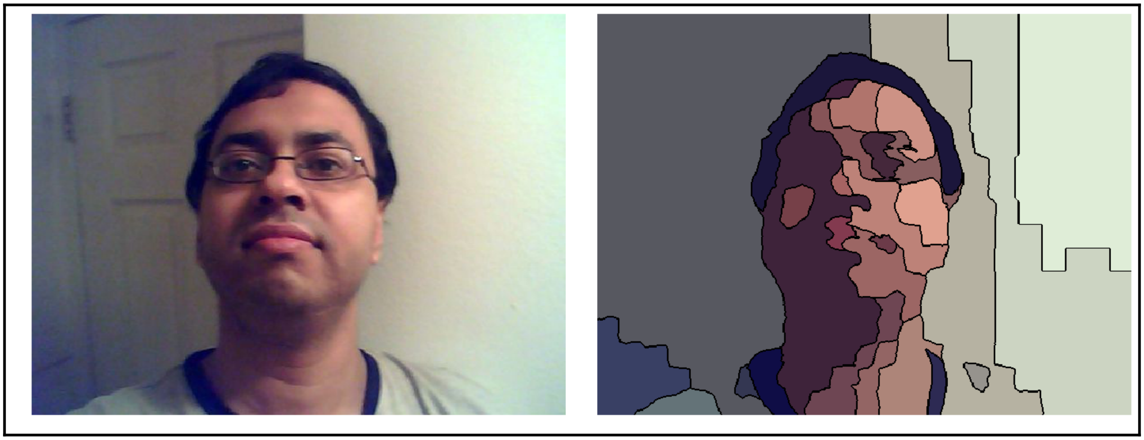

Original
Segmented with Felzenszwalbs's method

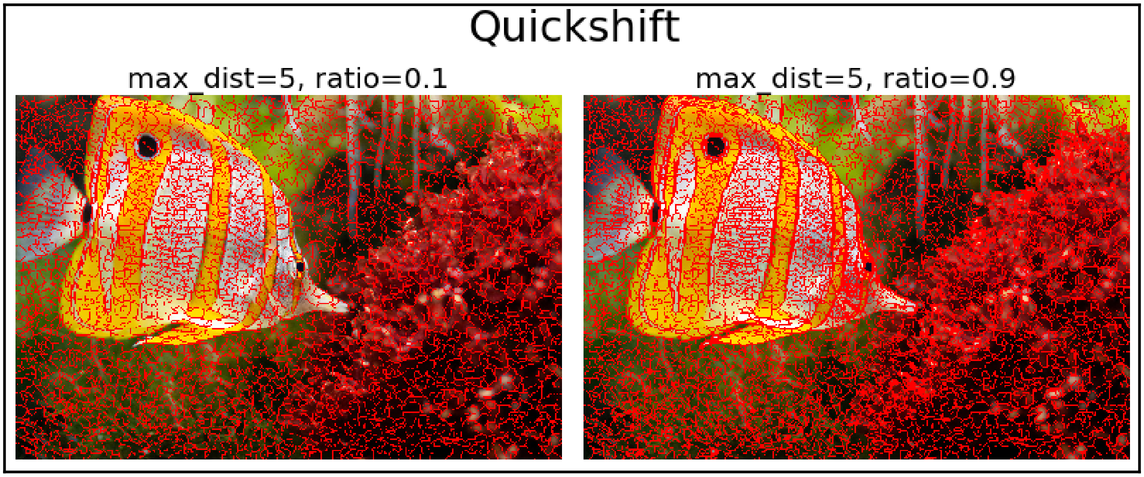

Felzenszwalbs's method
scale=50
scale=100

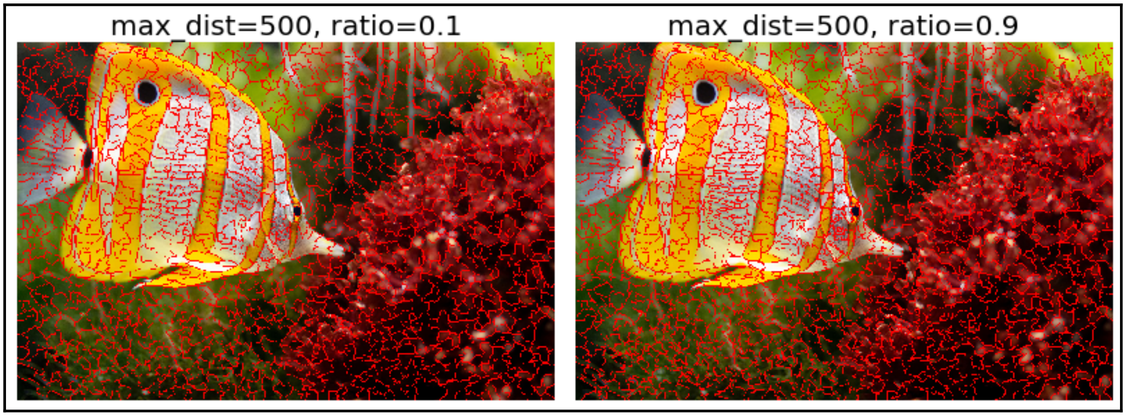

scale=200
scale=400

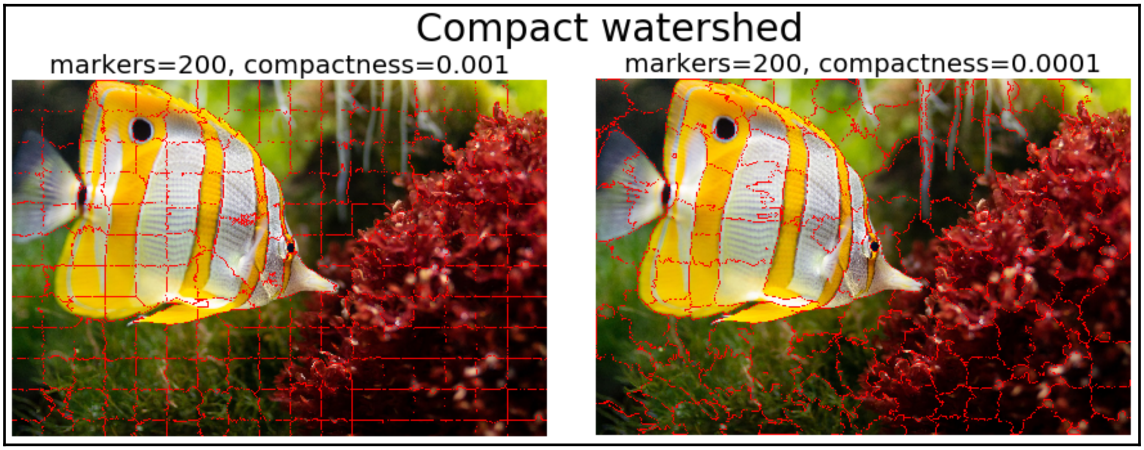

SLIC
compactness=0.1
compactness=1

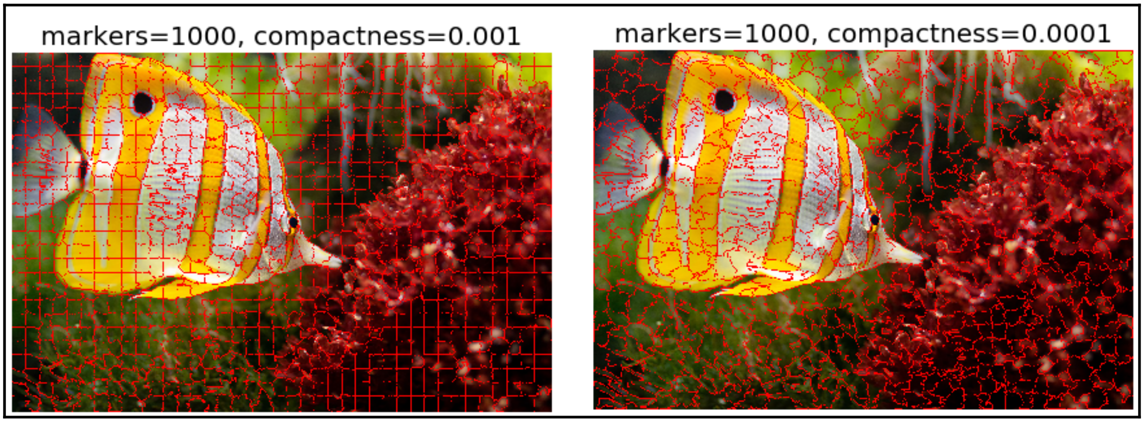

compactness=10
compactness=100

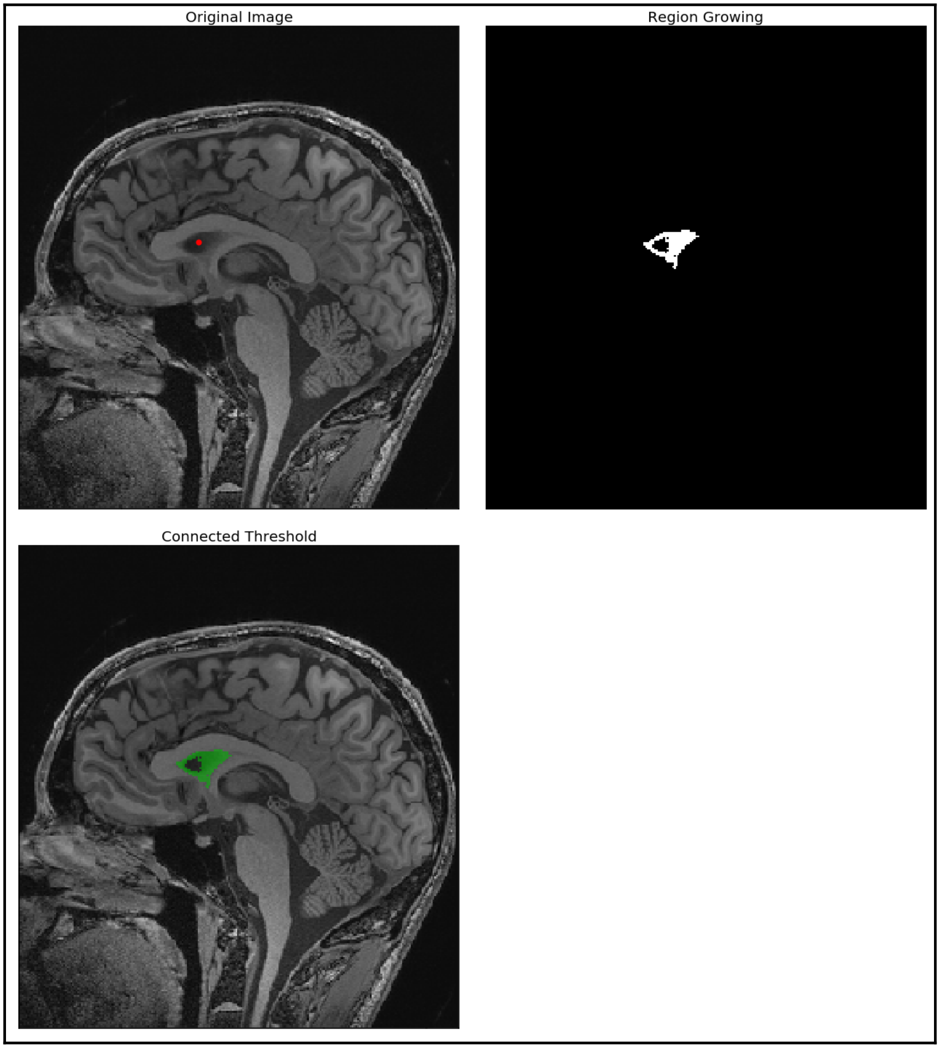


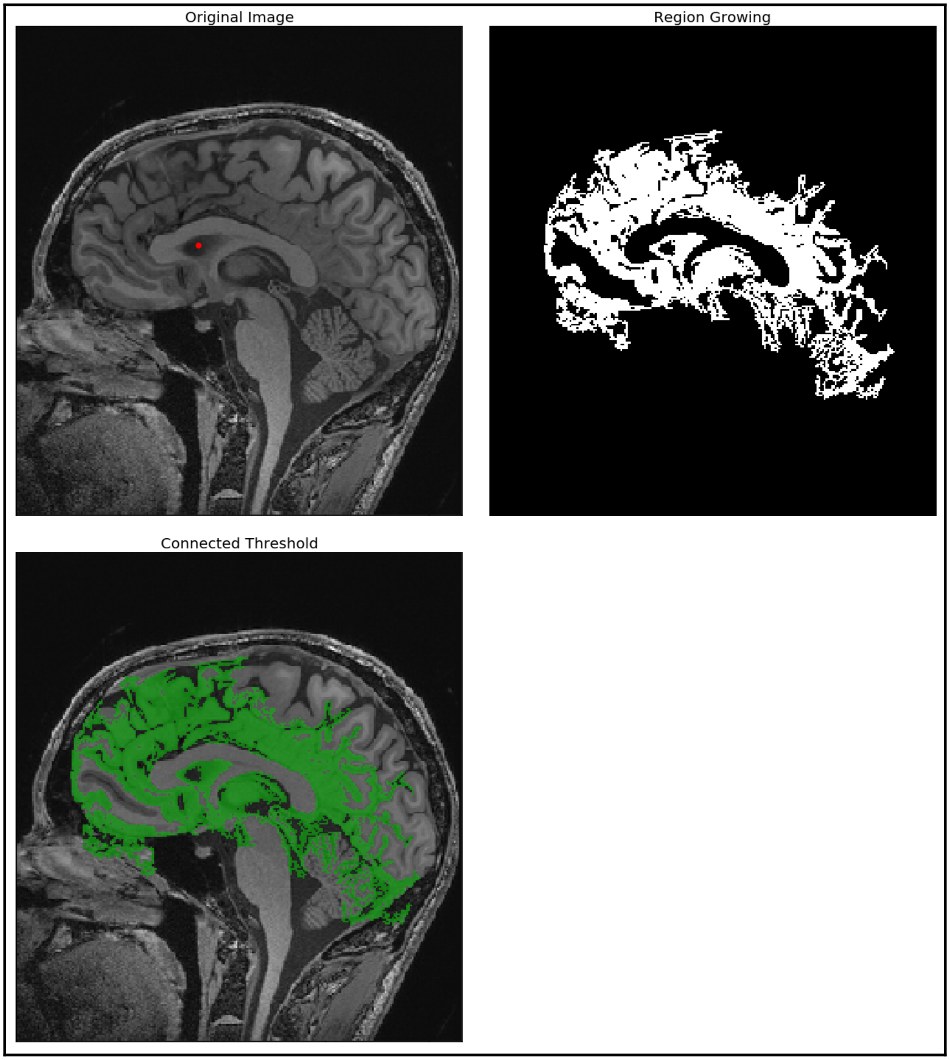

Quickshift
max_dist=5, ratio=0.1
max_dist=5, ratio=0.9

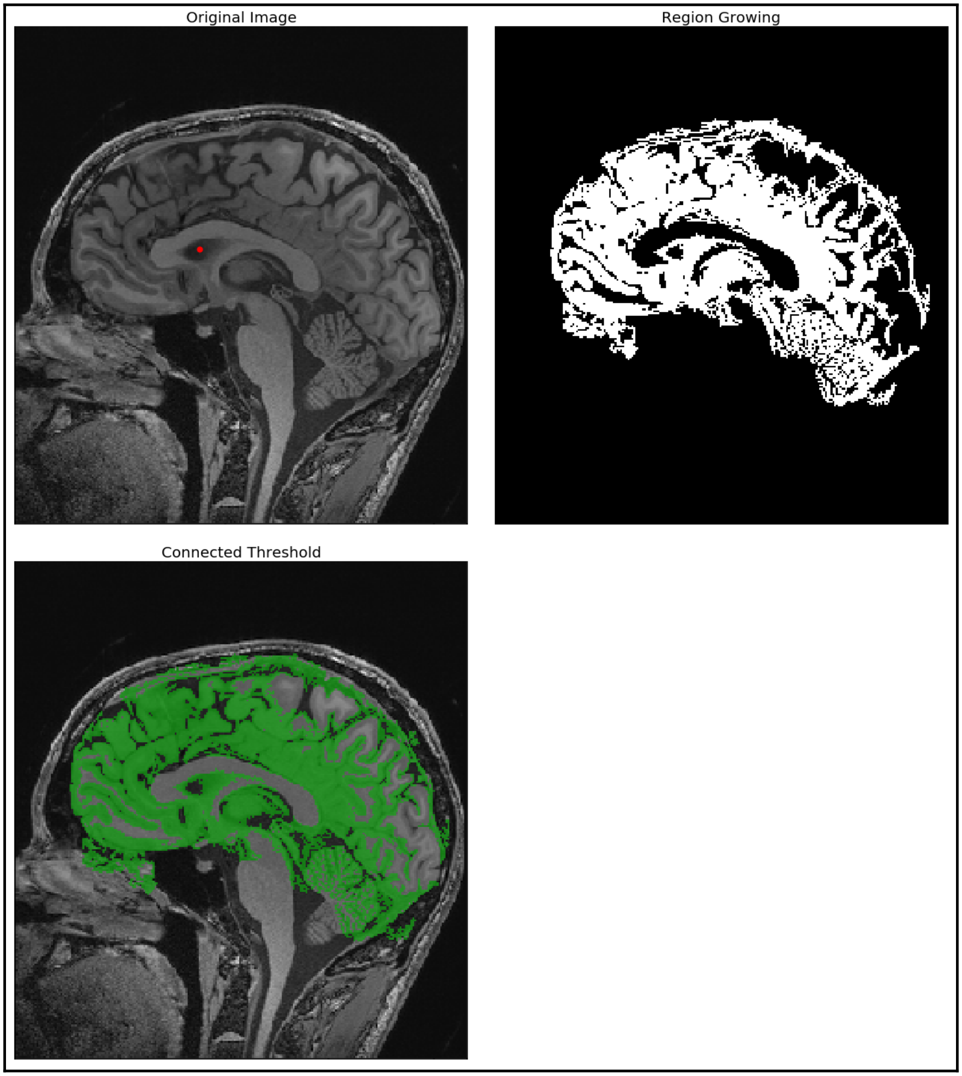

max_dist=500, ratio=0.1
max_dist=500, ratio=0.9

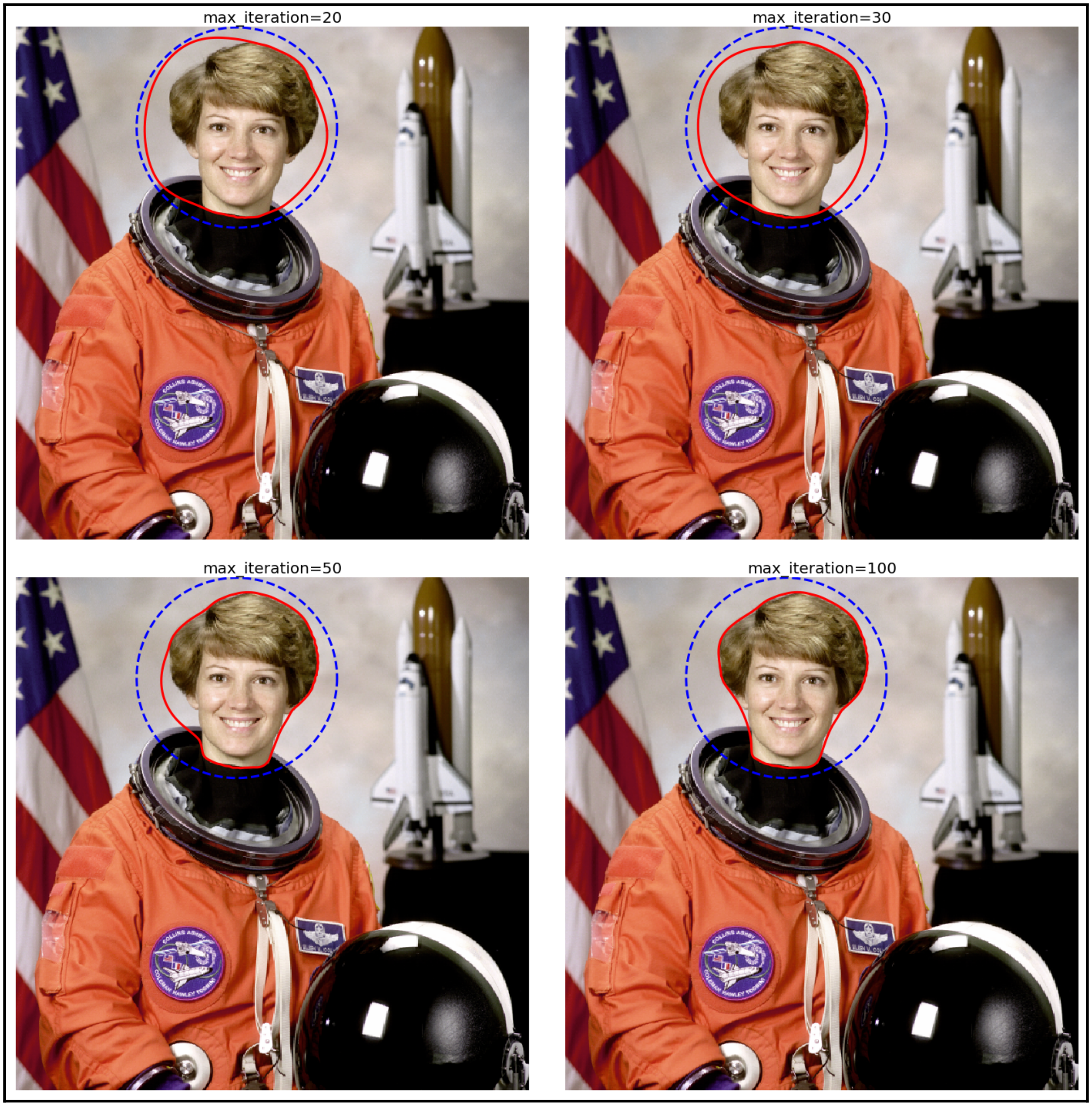

Compact watershed
markers=200, compactness=0.001
markers=200, compactness=0.0001

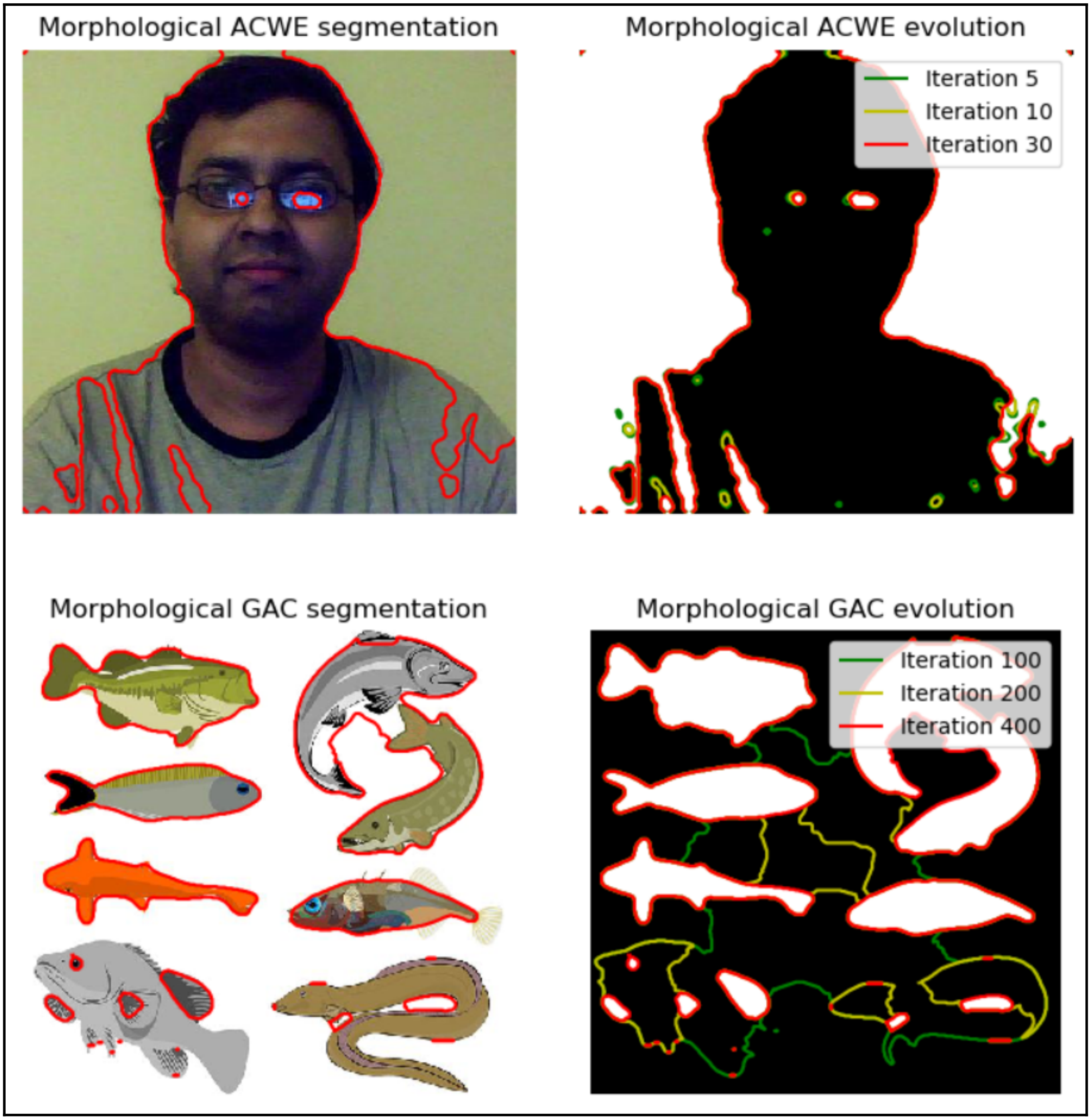

markers=1000, compactness=0.001
markers=1000, compactness=0.0001


Original Image
Region Growing
Connected Threshold

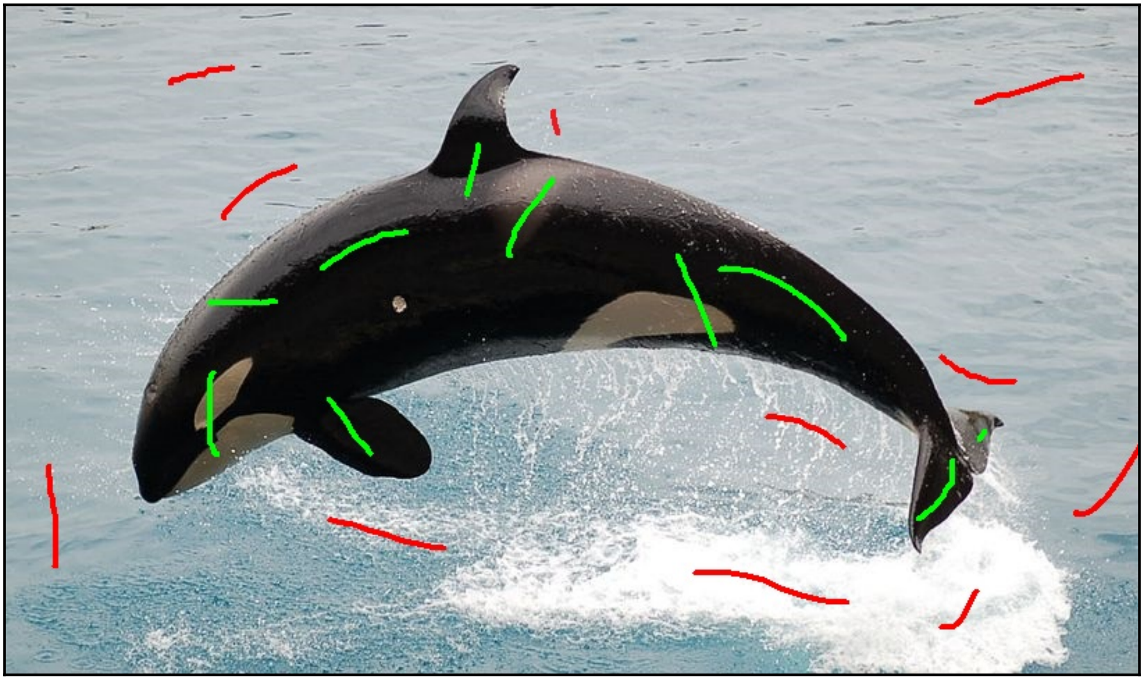

Original Image
Region Growing
Connected Threshold

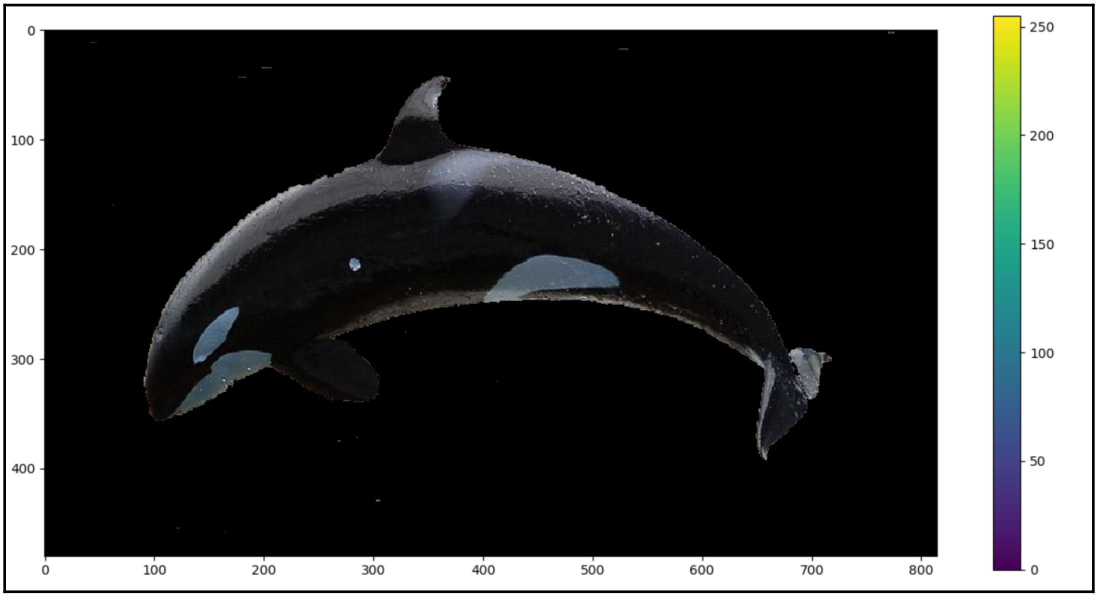

Original Image
Region Growing
Connected Threshold


max_iteration=20
max_iteration=30
max_iteration=50
max_iteration=100

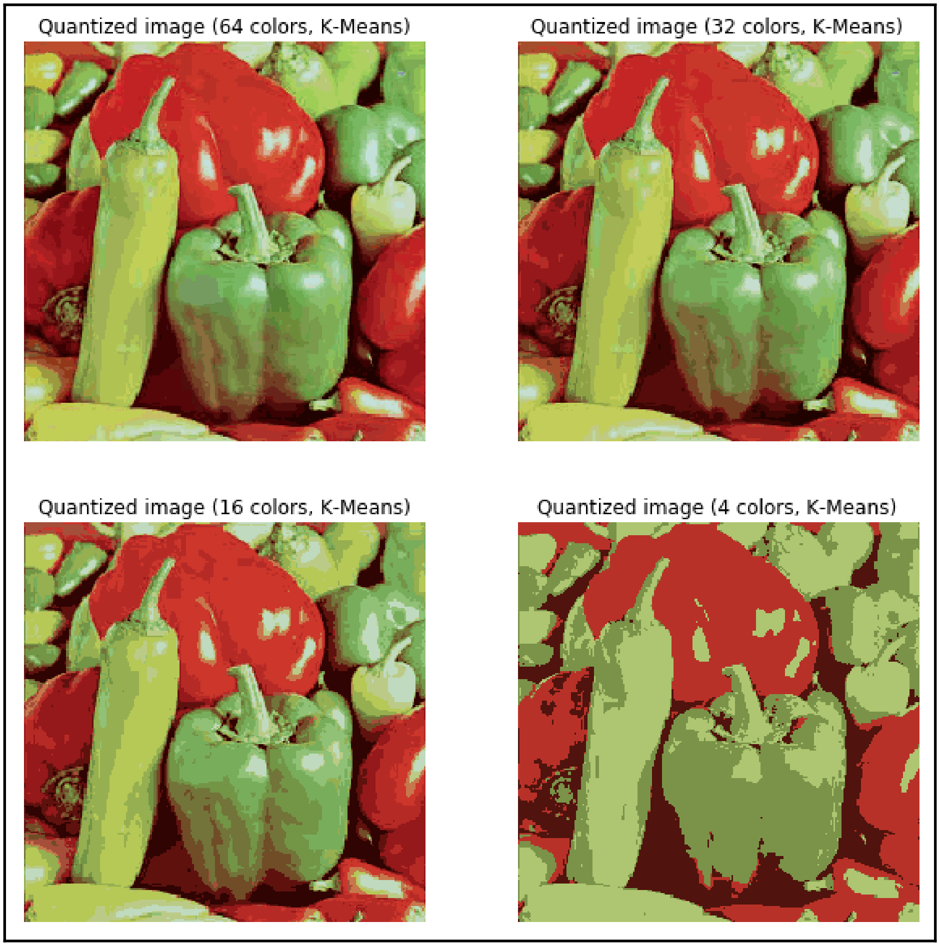

Morphological ACWE segmentation
Morphological ACWE evolution
Iteration 5
Iteration 10
Iteration 30
Morphological GAC segmentation
Morphological GAC evolution
Iteration 100
Iteration 200
Iteration 400

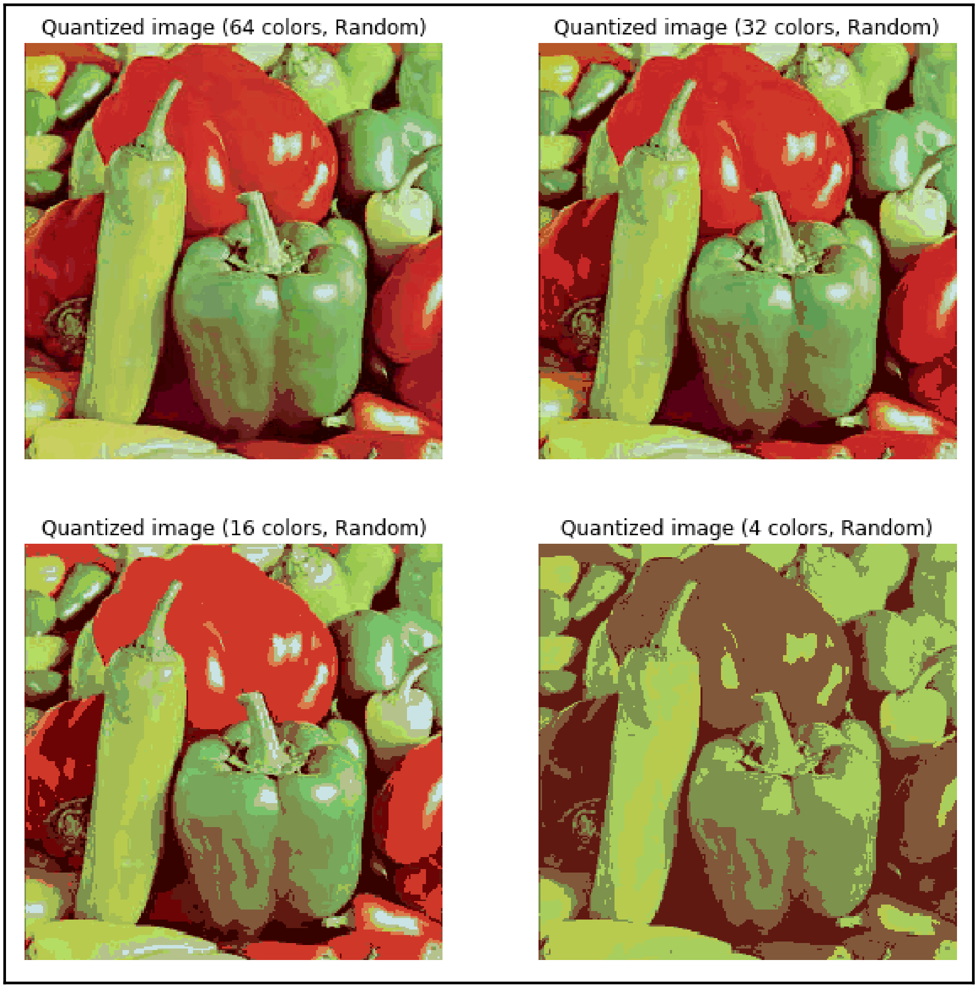

0
100
200
300
400
0
100
200
300
400
500
600
700
800
250
200
150
100
50
0

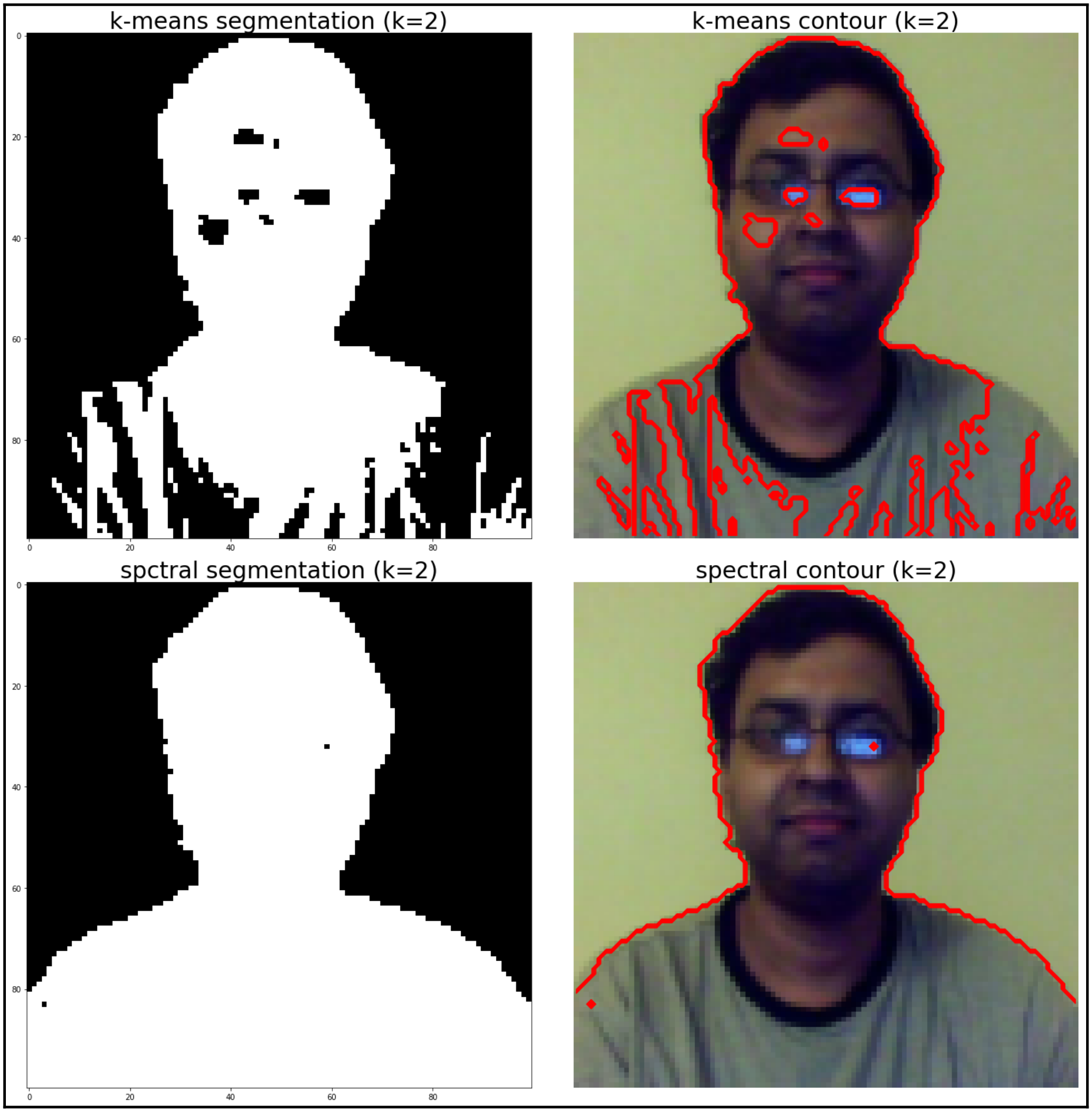


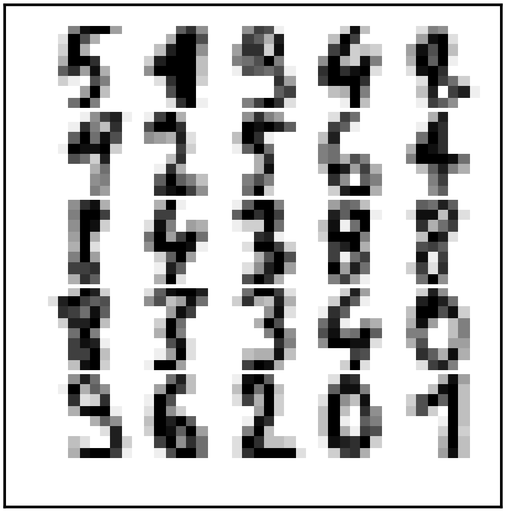

0
100
200
300
400
0
100
200
300
400
500
600
700
800
250
200
150
100
50
0

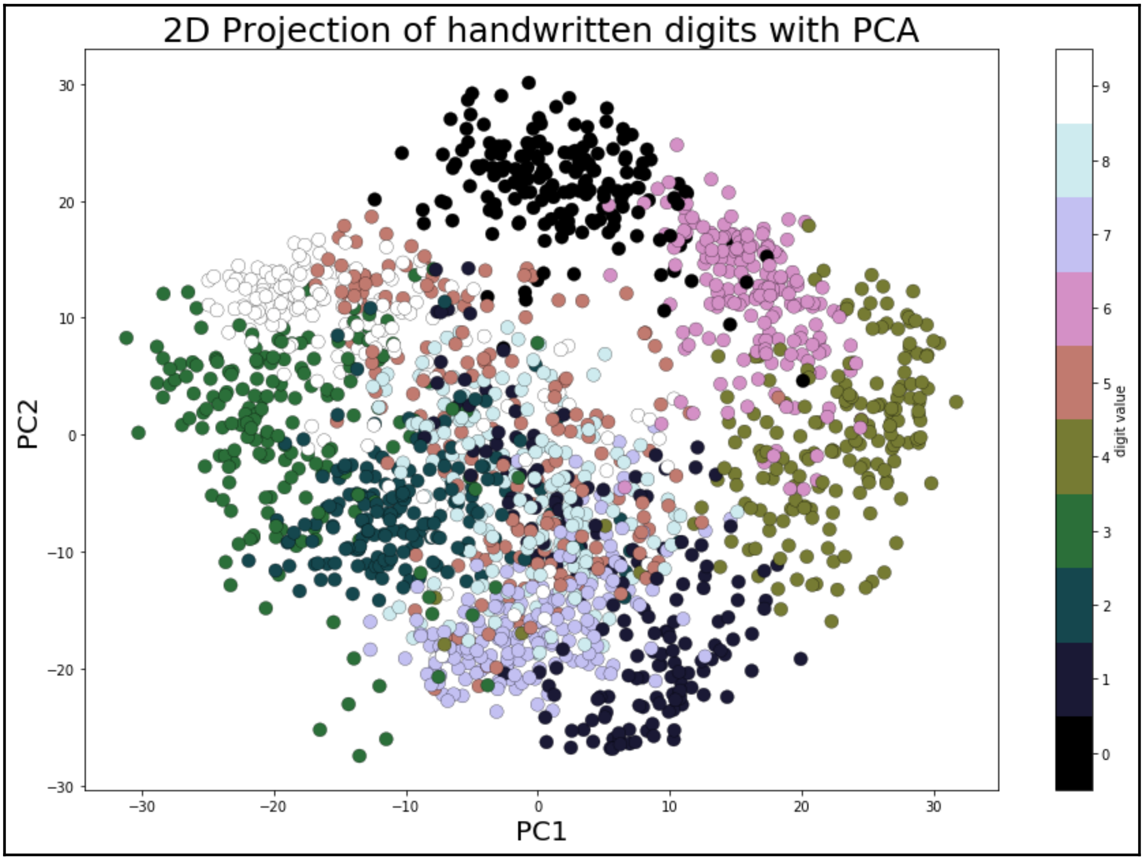
Chapter 9: Classical Machine Learning Methods in Image Processing

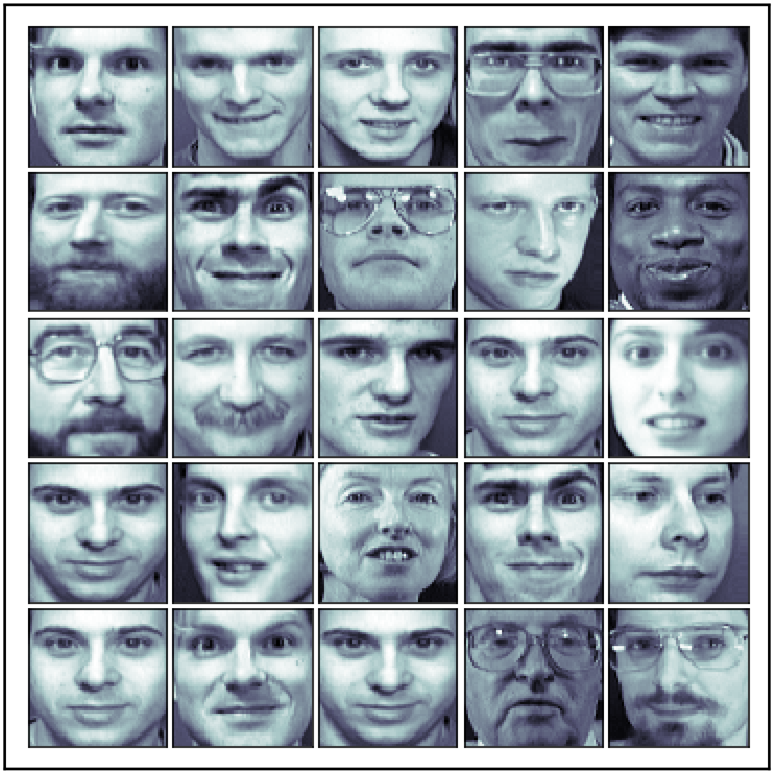

Quantized image (64 colors, K-Means)
Quantized image (32 colors, K-Means)
Quantized image (16 colors, K-Means)
Quantized image (4 colors, K-Means)

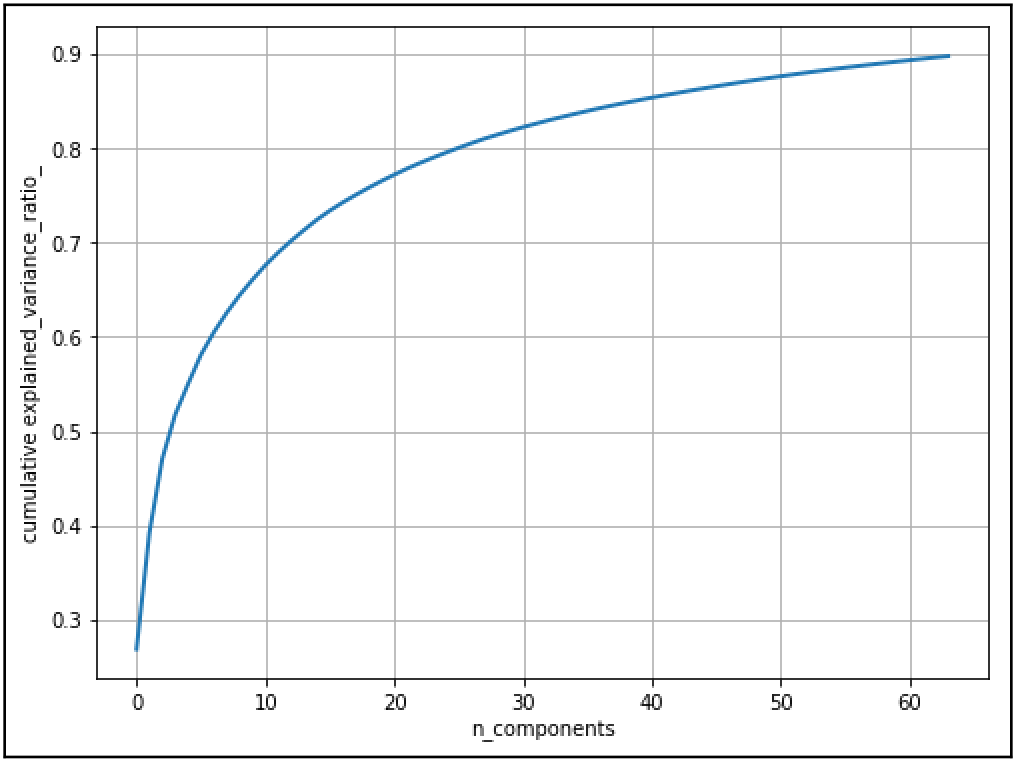

Quantized image (64 colors, Random)
Quantized image (32 colors, Random)
Quantized image (16 colors, Random)
Quantized image (4 colors, Random)

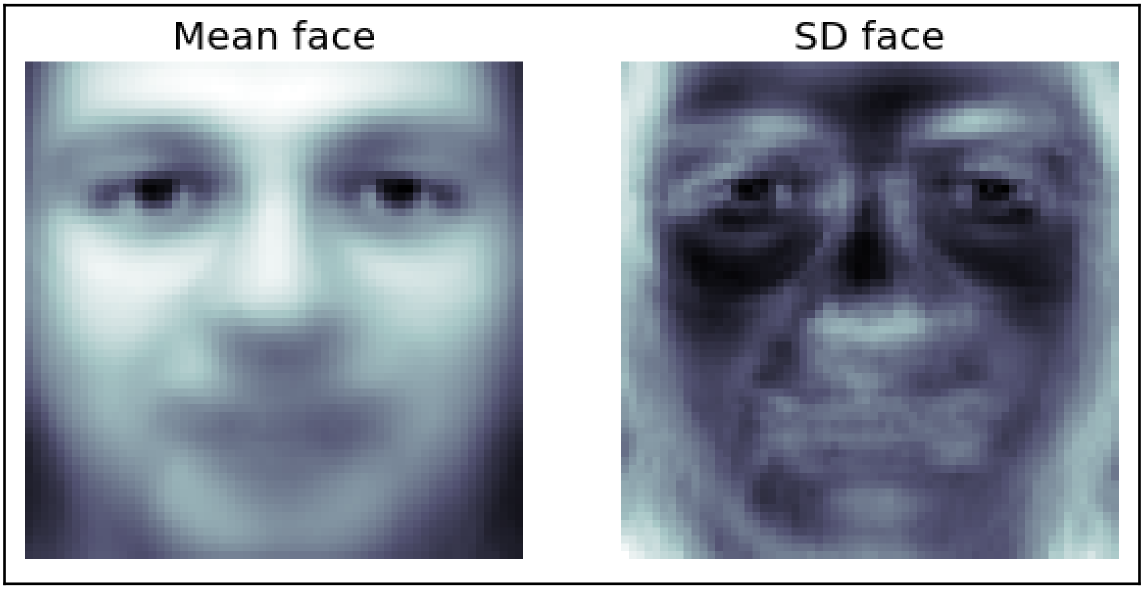

k-means segmentation (k=2)
k-means contour (k=2)
spctral segmentation (k=2)
spectral contour (k=2)
0
20
40
60
80

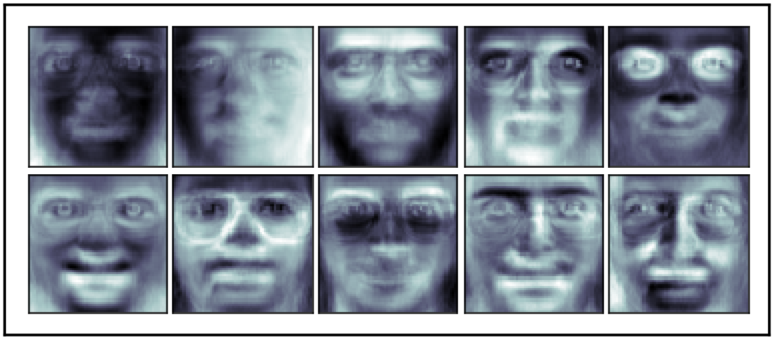


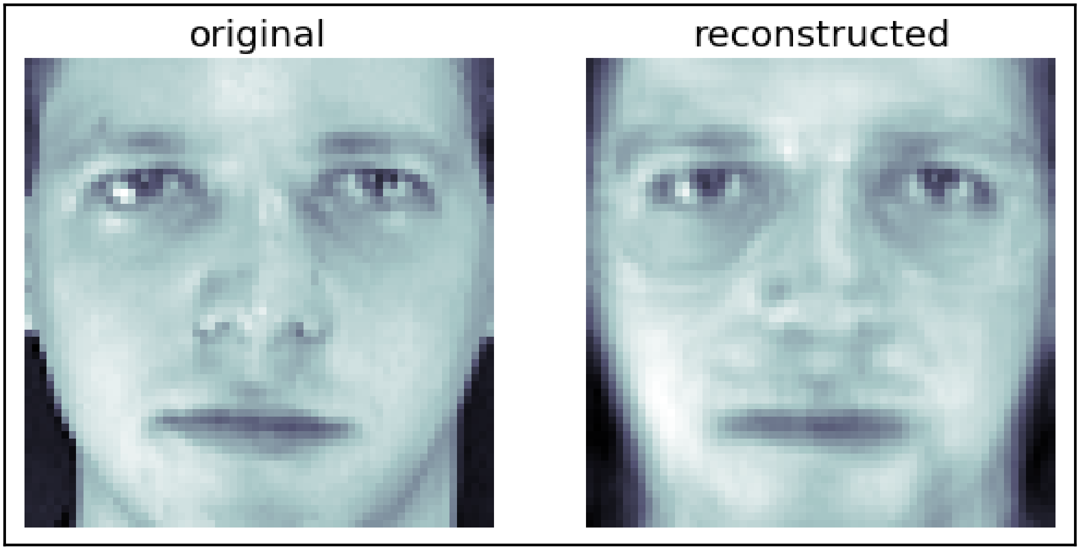

2D Projection of handwritten digits with PCA
PC1
PC2
digit value

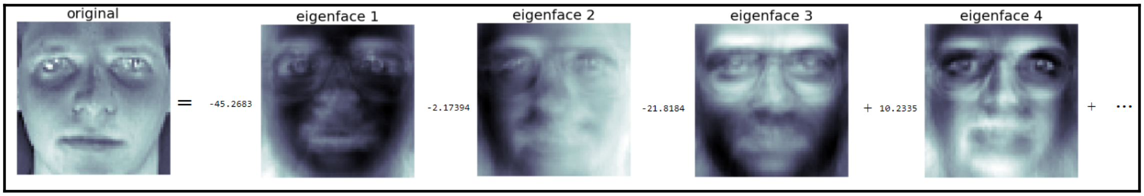


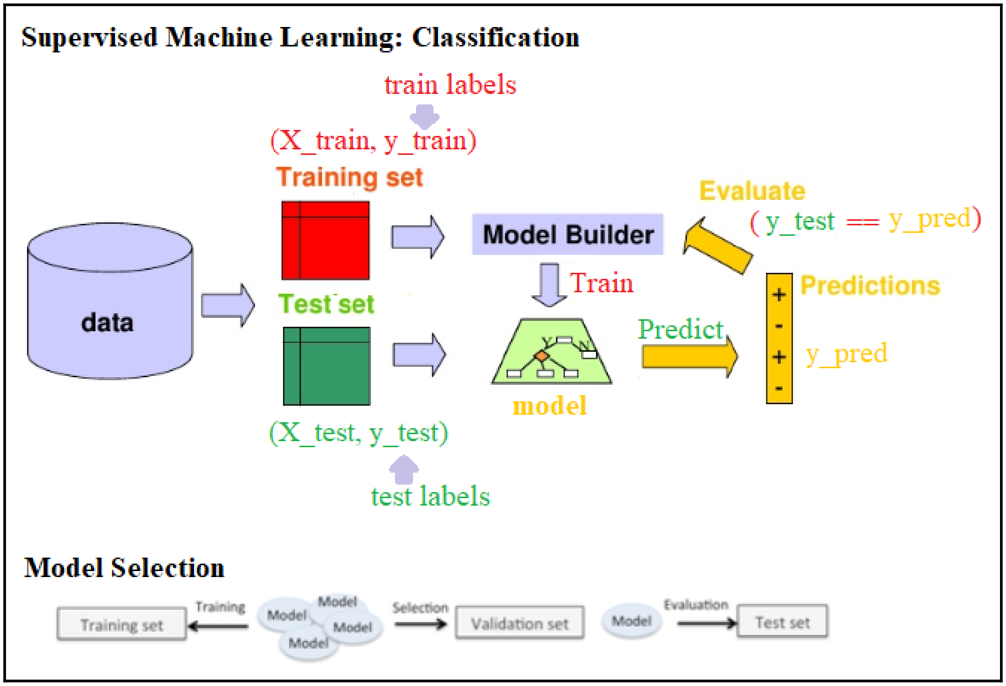

cumulative explained_variance_ratio_
n_components
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50
60

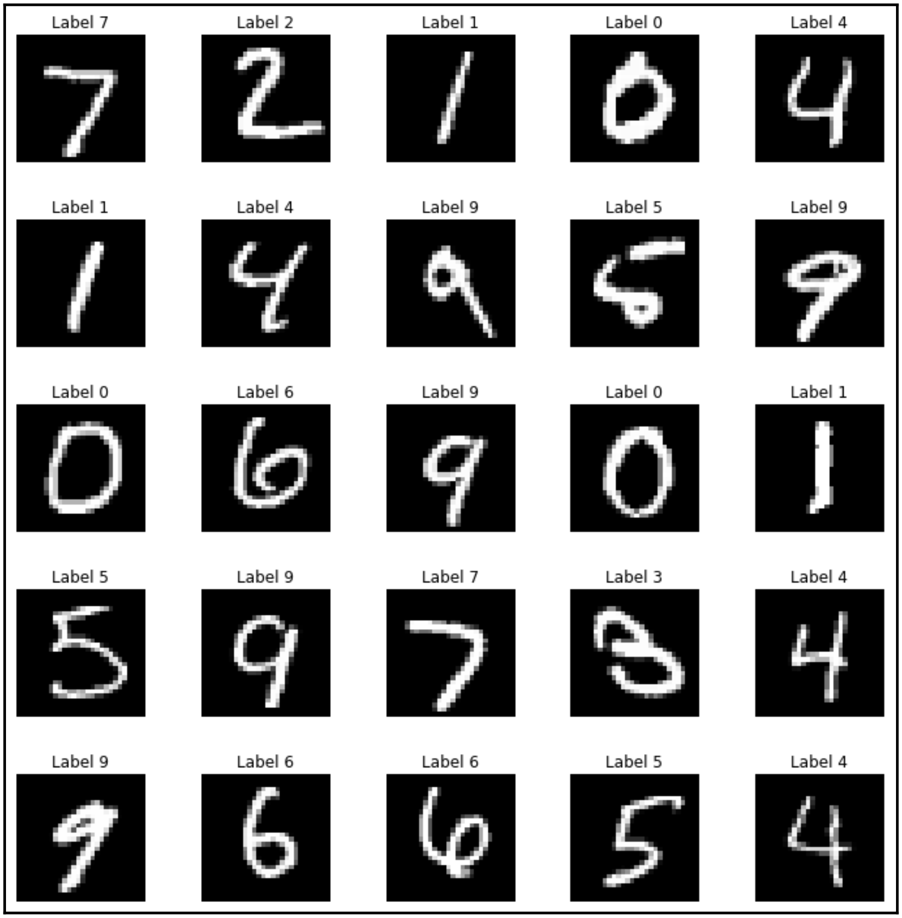

Mean face
SD face

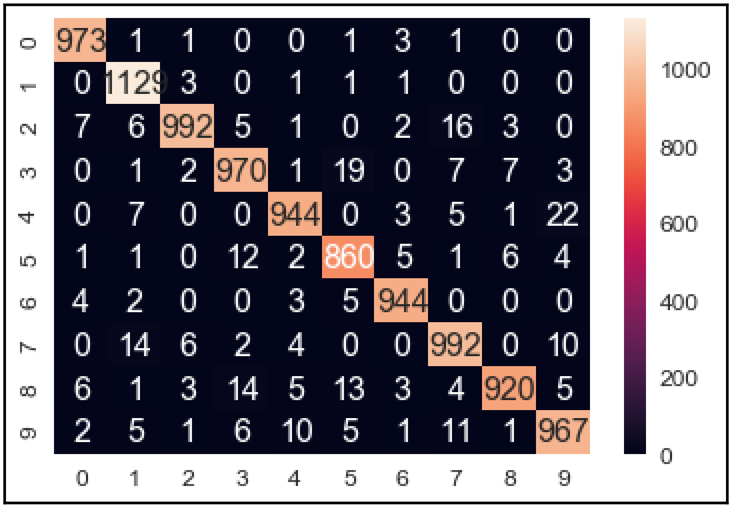


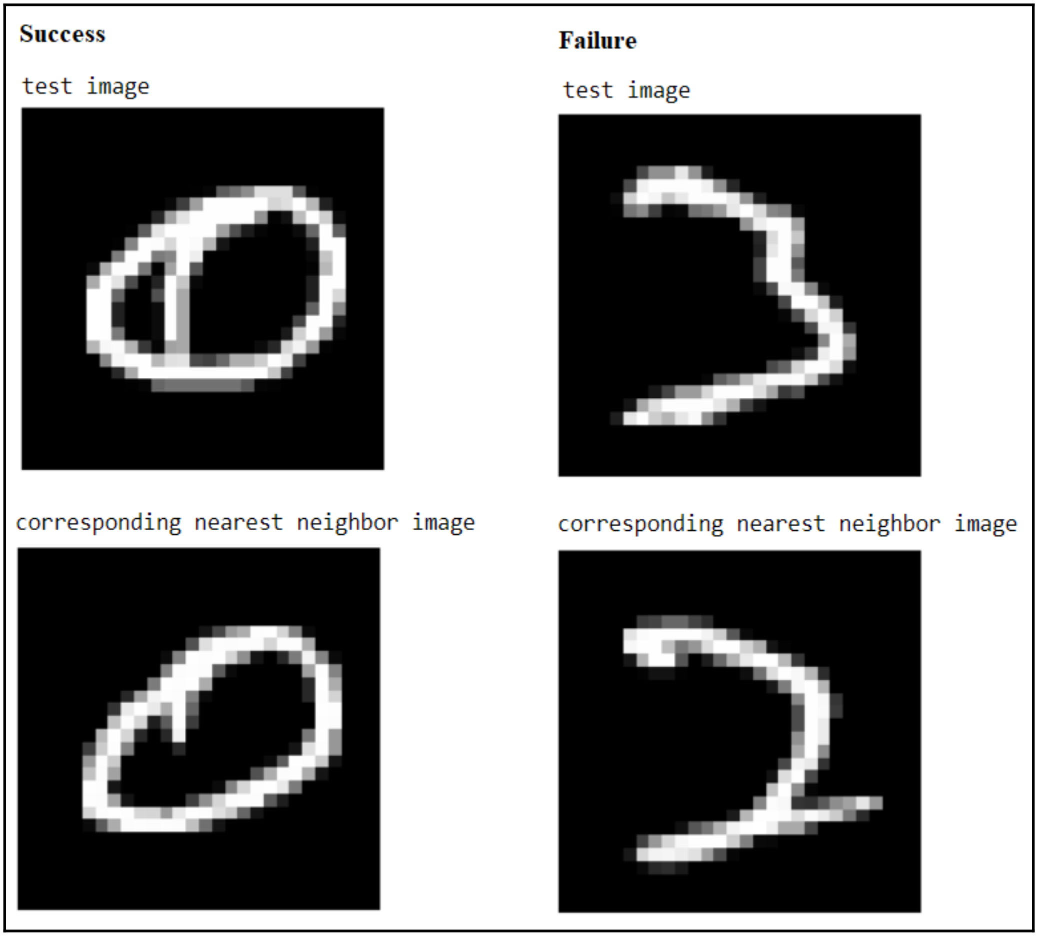

original
reconstructed

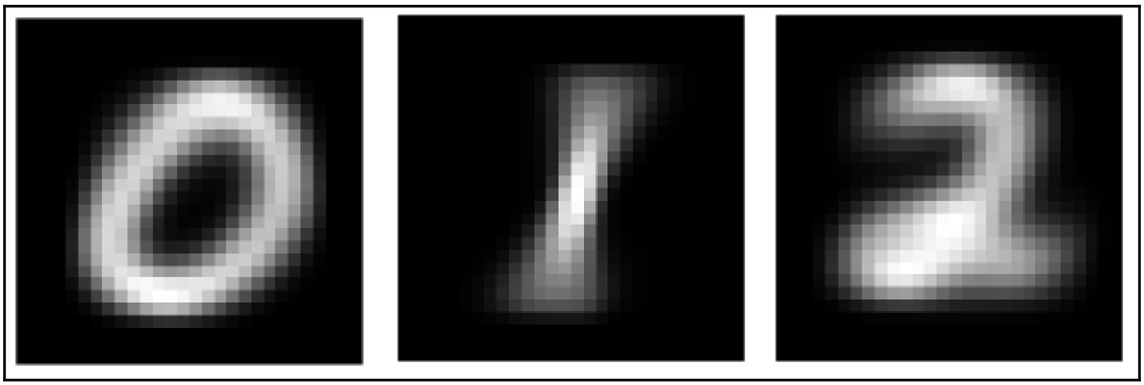

original
=
-45.2683
eigenface 1
-2.17394
eigenface 2
-21.8184
eigenface 3
+ 10.2335
eigenface 4
+ ...

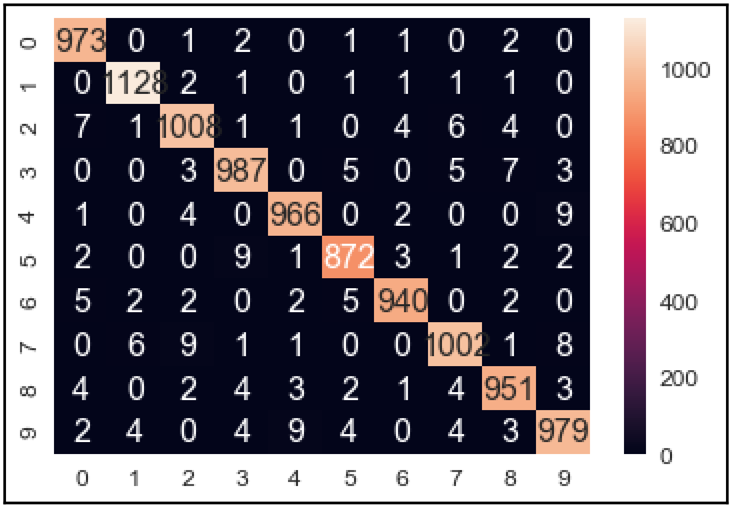

Supervised Machine Learning: Classification
train labels
(X_train, y_train)
Training set
Evaluate
(y_test == y_pred)
Model Builder
data
Train
Test set
Predictions
Predict
y_pred
model
(X_test, y_test)
test labels
Model Selection
Training set
Training
Model
Model
Model
Model
Selection
Validation set
Model
Evaluation
Test set

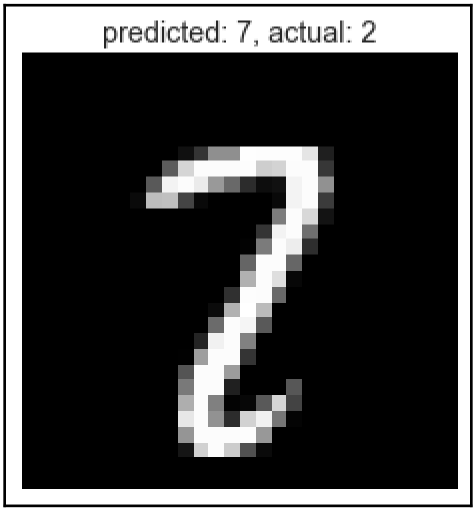

Label 7
Label 2
Label 1
Label 0
Label 4
Label 1
Label 4
Label 9
Label 5
Label 9
Label 0
Label 6
Label 9
Label 0
Label 1
Label 5
Label 9
Label 7
Label 3
Label 4
Label 9
Label 6
Label 6
Label 5
Label 4

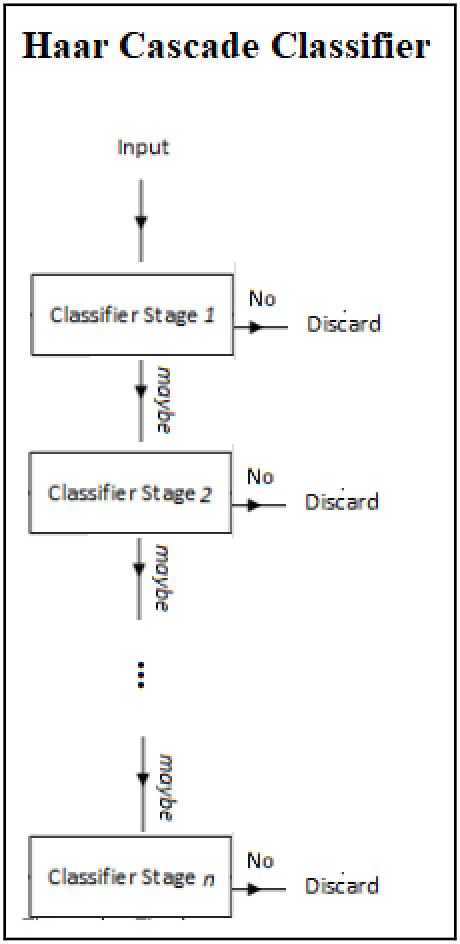

973 1 1 0 0 1 3 1 0 0
0 1129 3 0 1 1 1 0 0 0
7 6 992 5 1 0 2 16 3 0
0 1 2 970 1 19 0 7 7 3
0 7 0 0 944 0 3 5 1 22
1 1 0 12 2 860 5 1 6 4
4 2 0 0 3 5 944 0 0 0
0 14 6 2 4 0 0 992 0 10
6 1 3 14 5 13 3 4 920 5
2 5 1 6 10 5 1 11 1 967
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

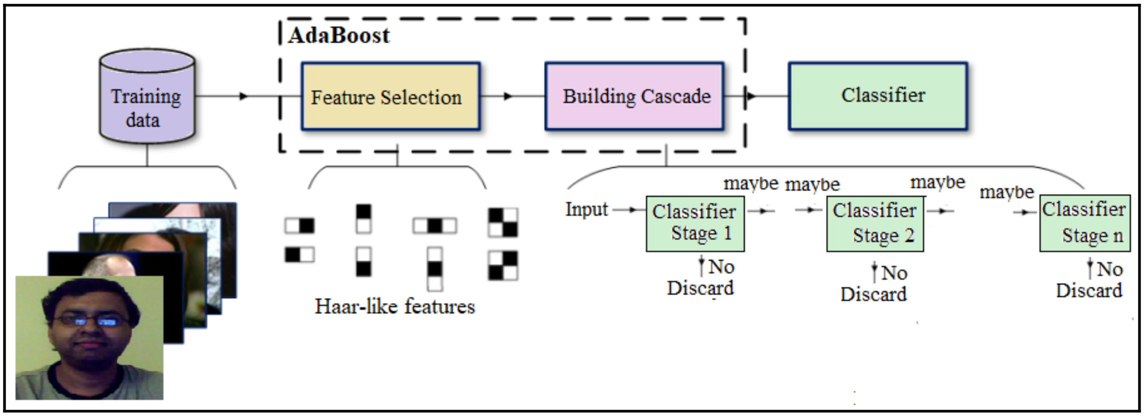

Success
test image
corresponding nearest neighbor image
Failure
test image
corresponding nearest neighbor image

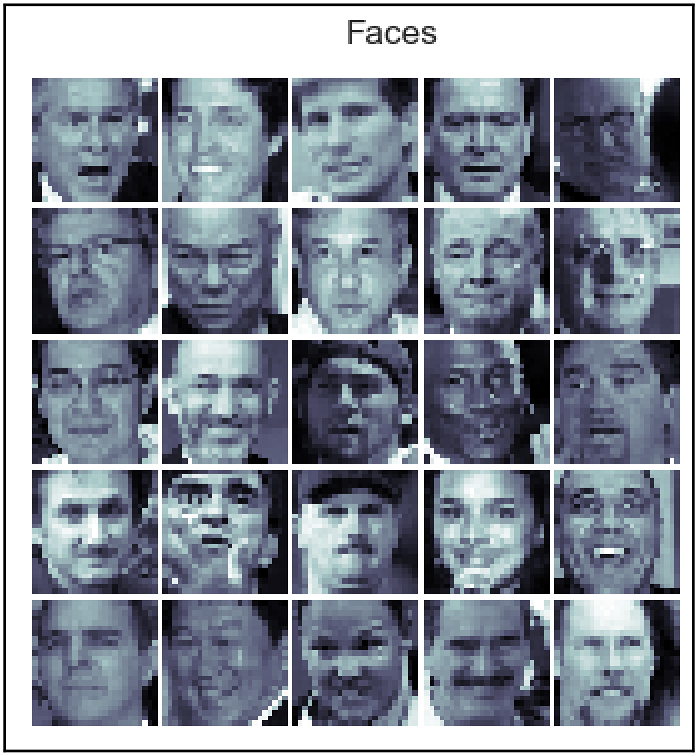


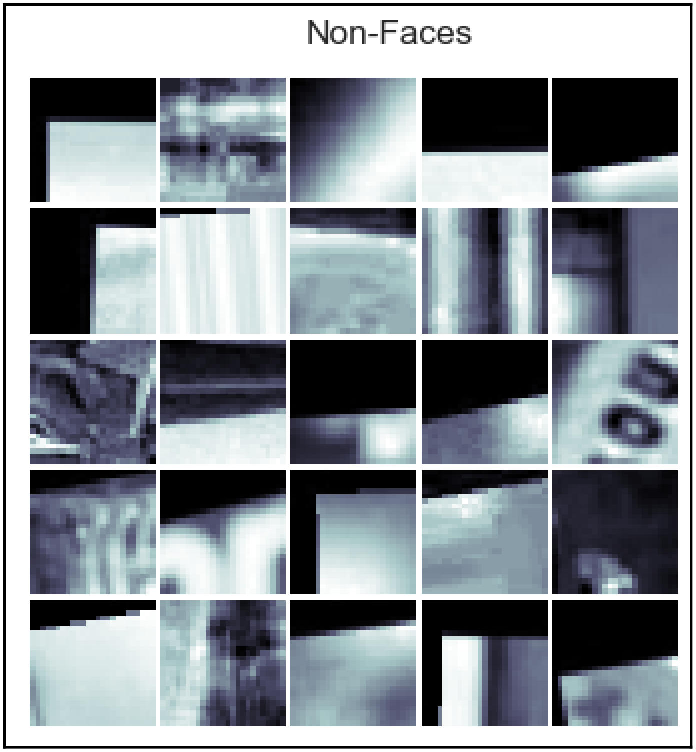

0
1
2
3
4
5
6
7
8
9
973 0 1 2 0 1 1 0 2 0
0 1128 2 1 0 1 1 1 1 0
7 1 1008 1 1 0 4 6 4 0
0 0 3 987 0 5 0 5 7 3
1 0 4 0 966 0 2 0 0 9
2 0 0 9 1 872 3 1 2 2
5 2 2 0 2 5 940 0 2 0
0 6 9 1 1 0 0 1002 1 8
4 0 2 4 3 2 1 4 951 3
2 4 0 4 9 4 0 4 3 979
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

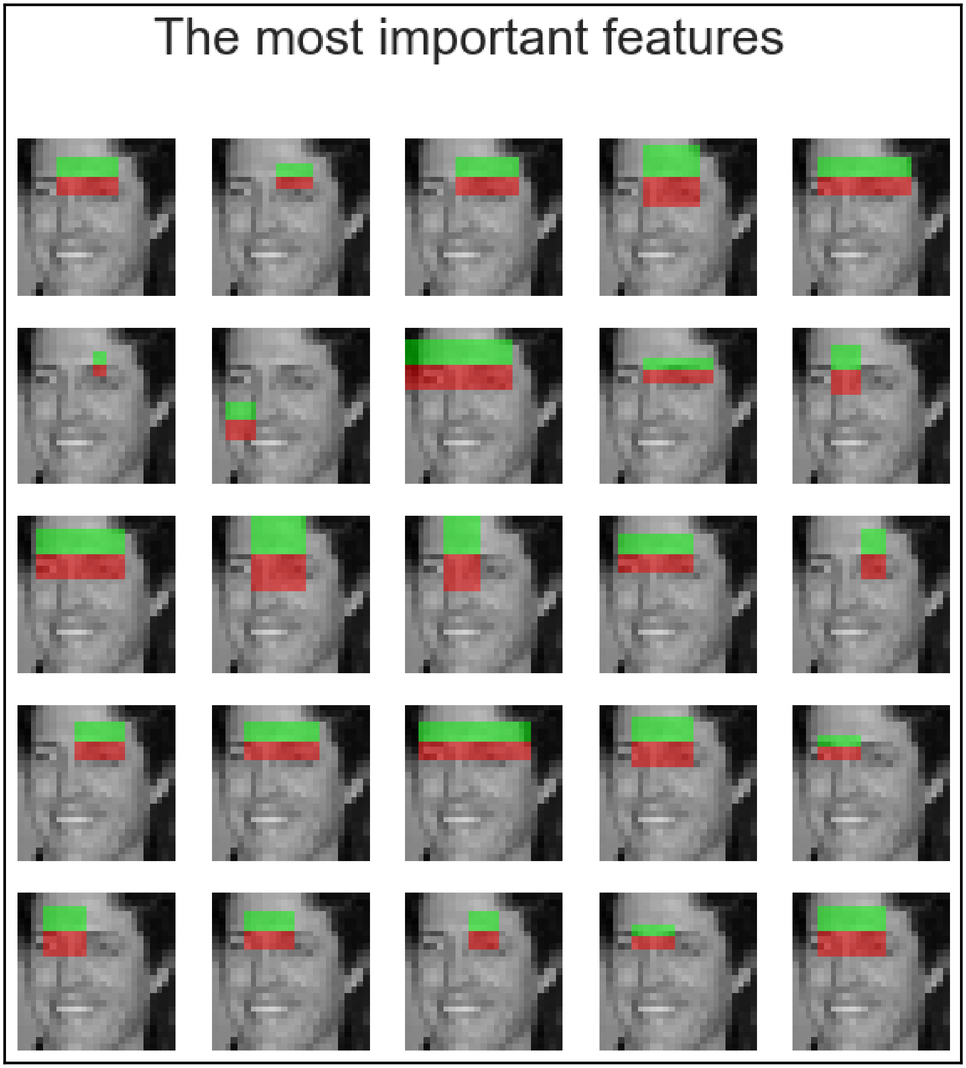

predicted: 7, actual: 2

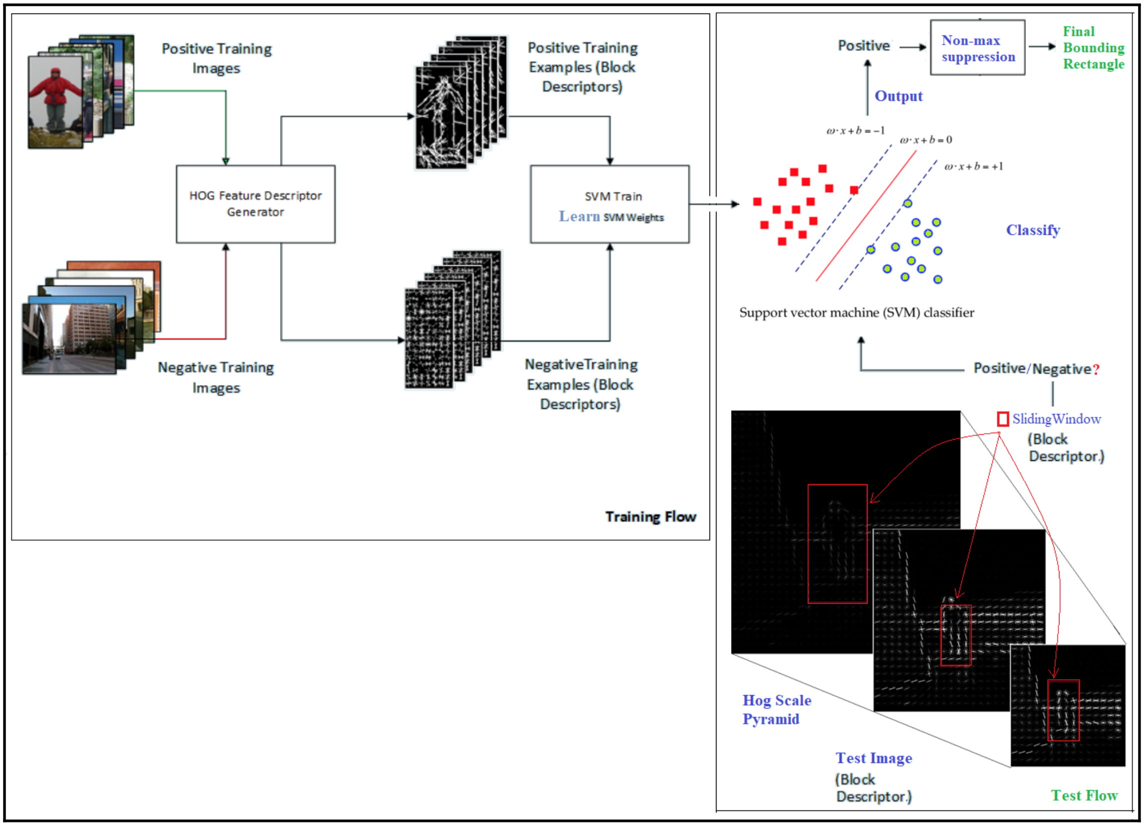

Haar Cascade Classifier
Input
Classifier Stage 1
No
Discard
maybe
Classifier Stage 2
No
Discard
maybe
maybe
Classifier Stage n
No
Discard


AdaBoost
Training data
Feature Selection
Building Cascade
Classifier
Haar-like features
Input
maybe
maybe
maybe
maybe
Classifier Stage 1
Classifier Stage 2
Classifier Stage n
No
Discard
No
Discard
No
Discard

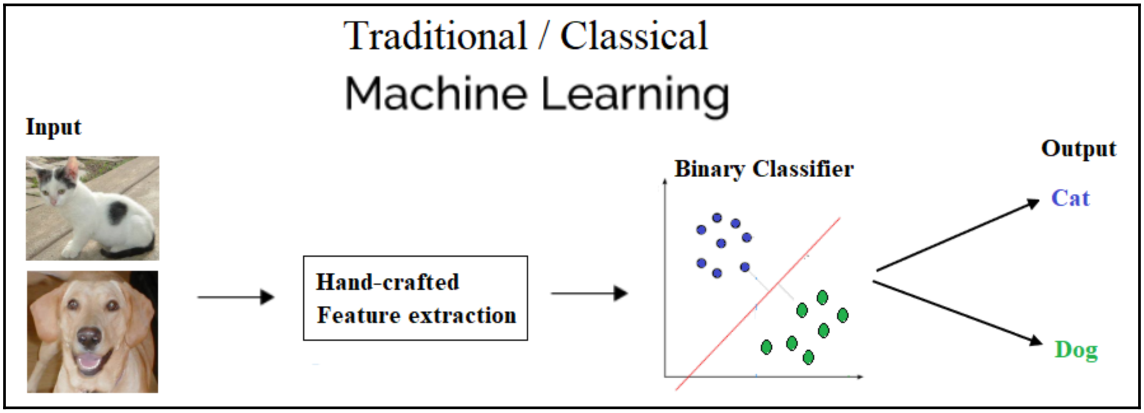

Faces

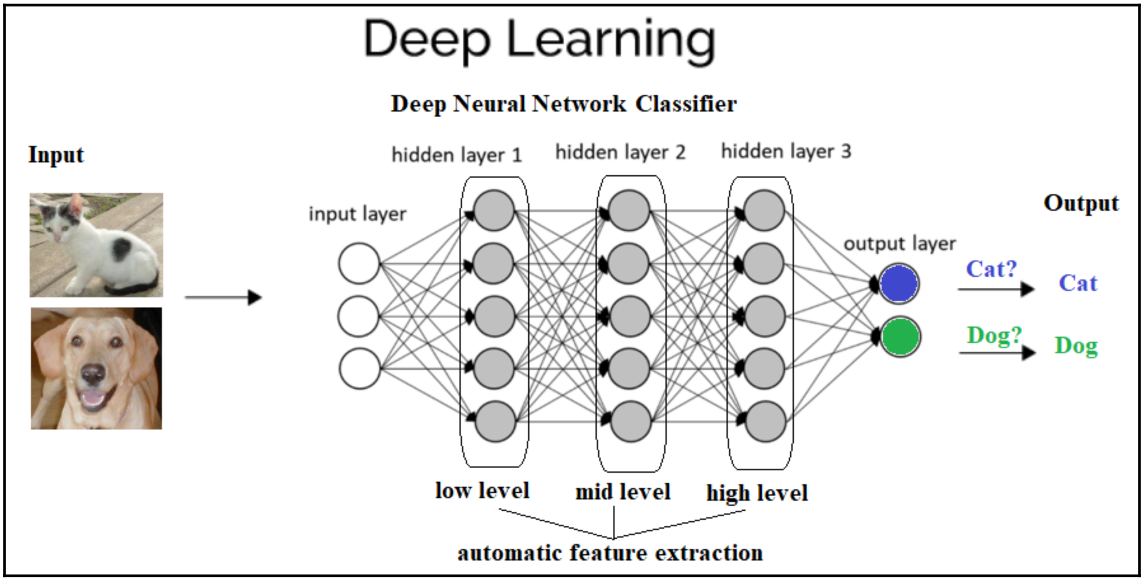

Non-Faces

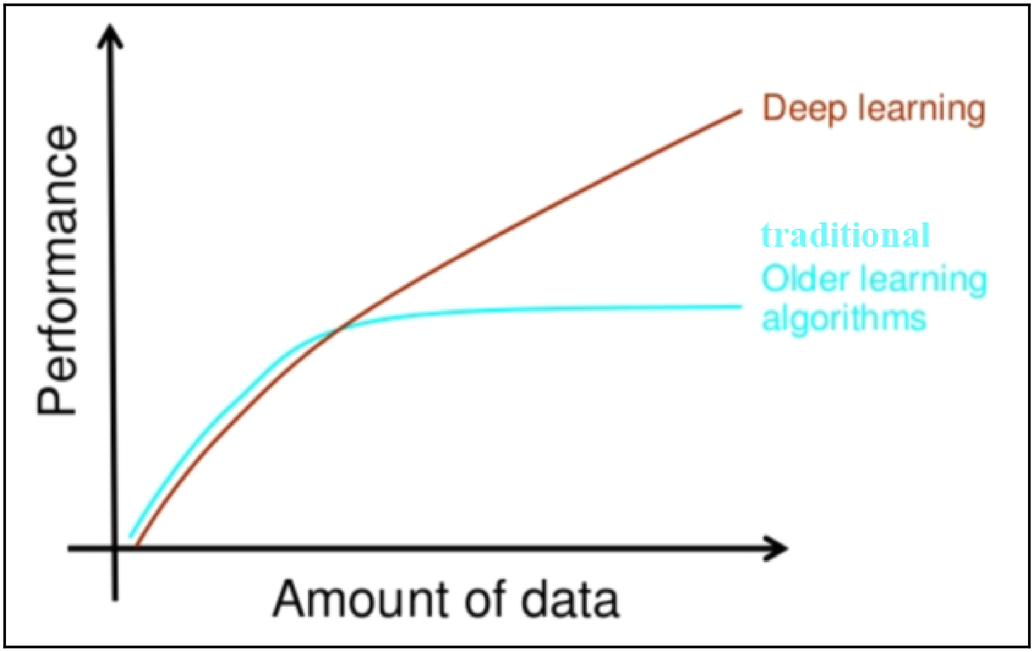

The most important features


Positive Training Images
Positive Training Examples (Block Descriptors)
HOG Feature Descriptor Generator
SVM Train
Learn SVM Weights
Negative Training Images
NegativeTraining Examples (Block Descriptors)
Training Flow
Positive
Non-max suppression
Final Bounding Rectangle
Output
Classify
Support vector machine (SVM) classifier
Positive/Negative?
SlidingWindow (Block Descriptor)
Hog Scale Pyramid
Test Image (Block Descriptor)
Test Flow

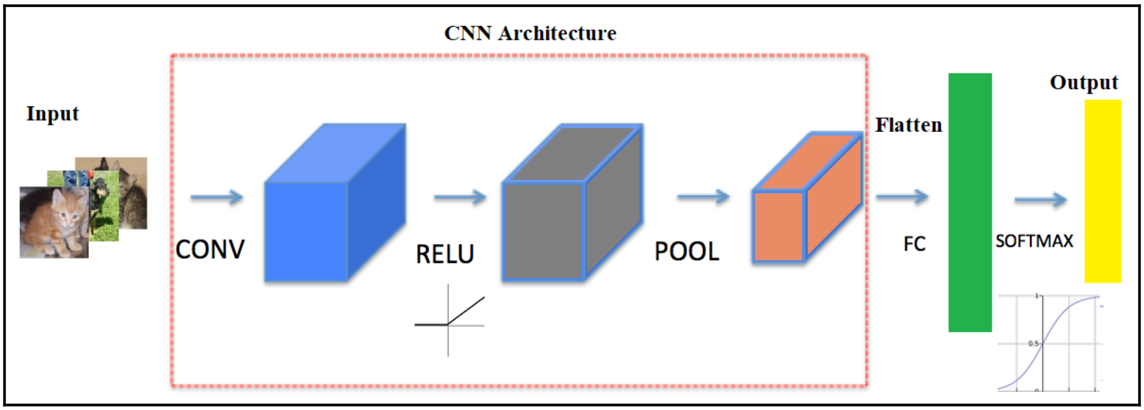
Chapter 10: Deep Learning in Image Processing - Image Classification



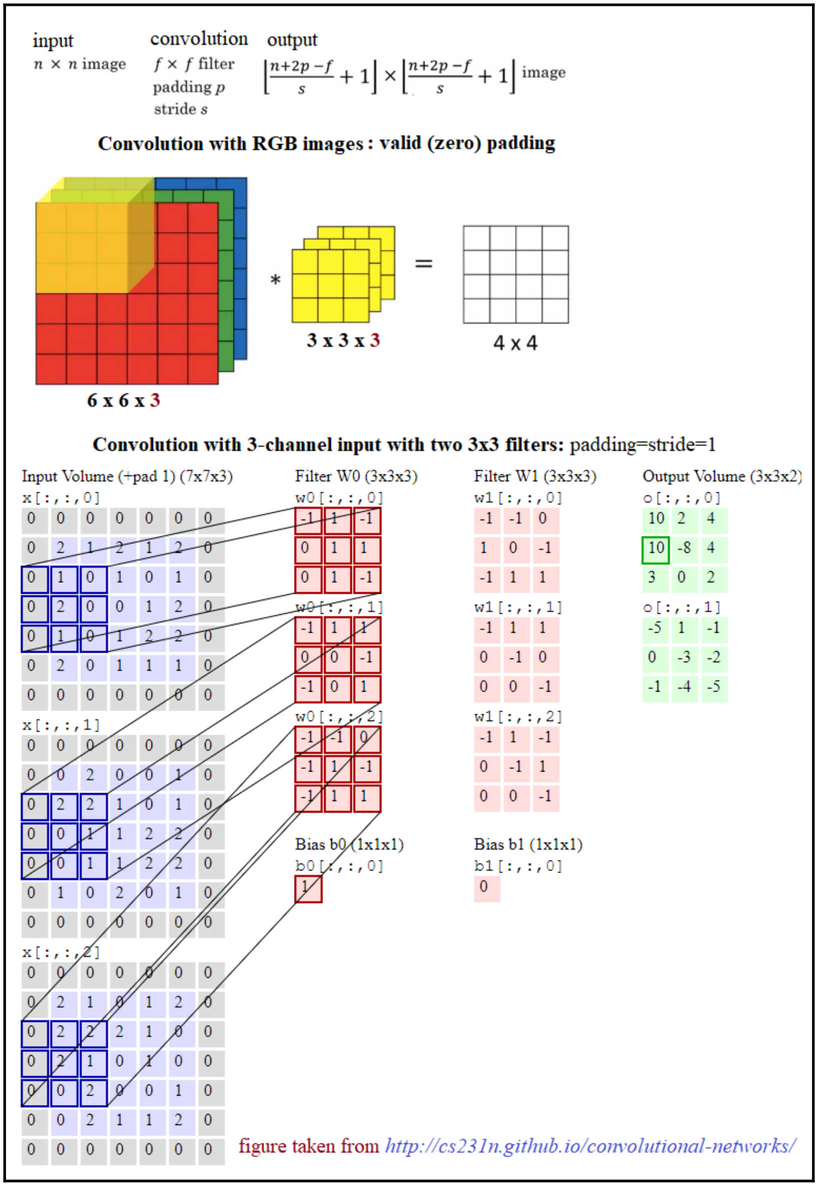


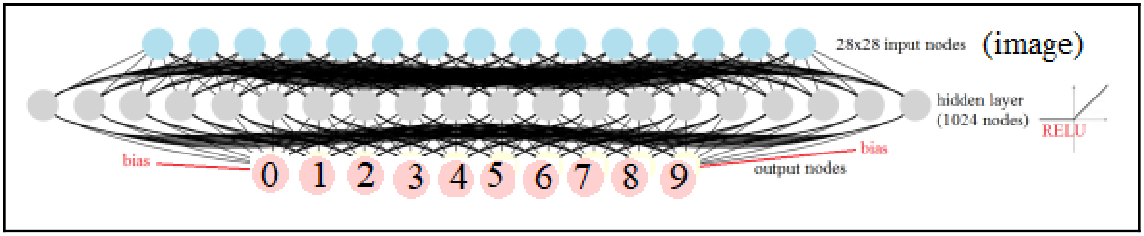


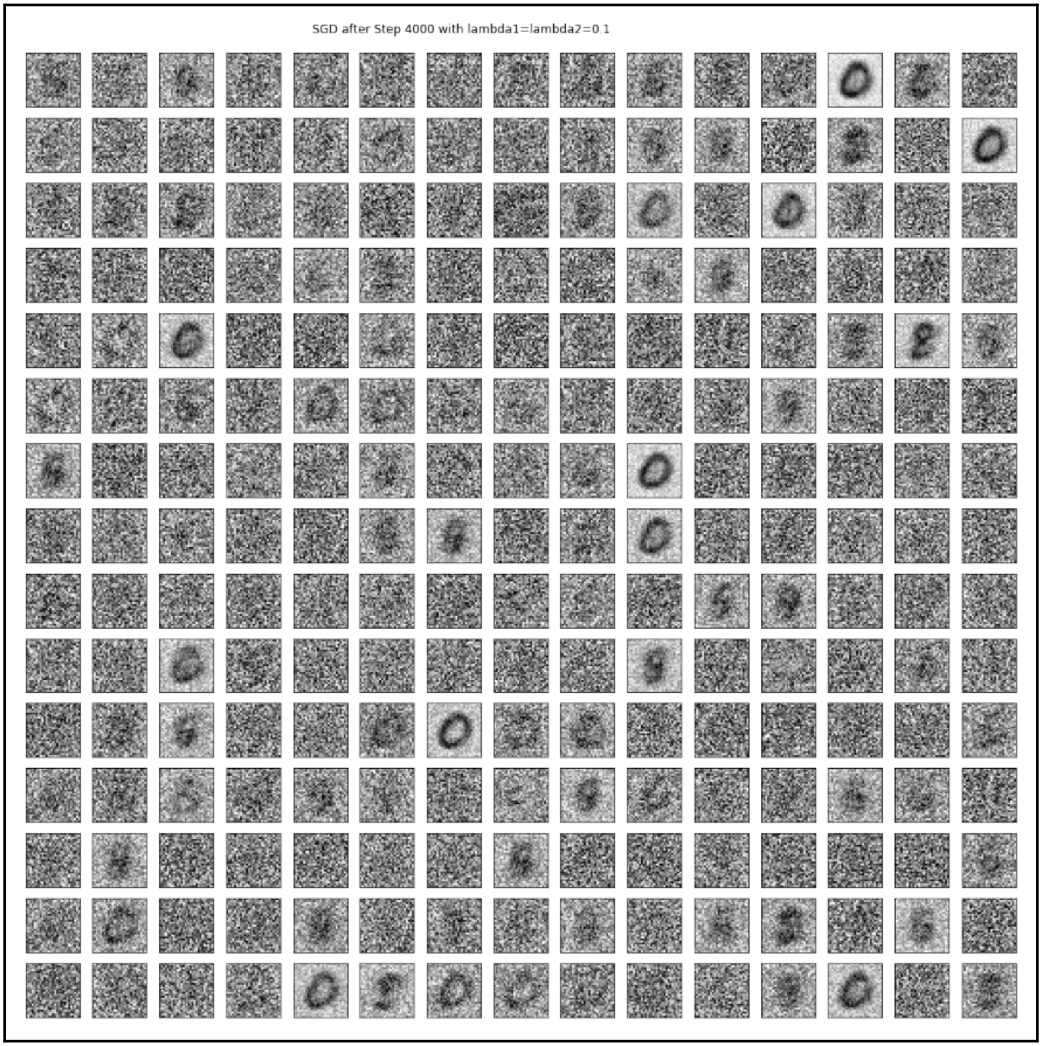
Performance
Deep learning
traditional
Older learning
algorithms
Amount of data

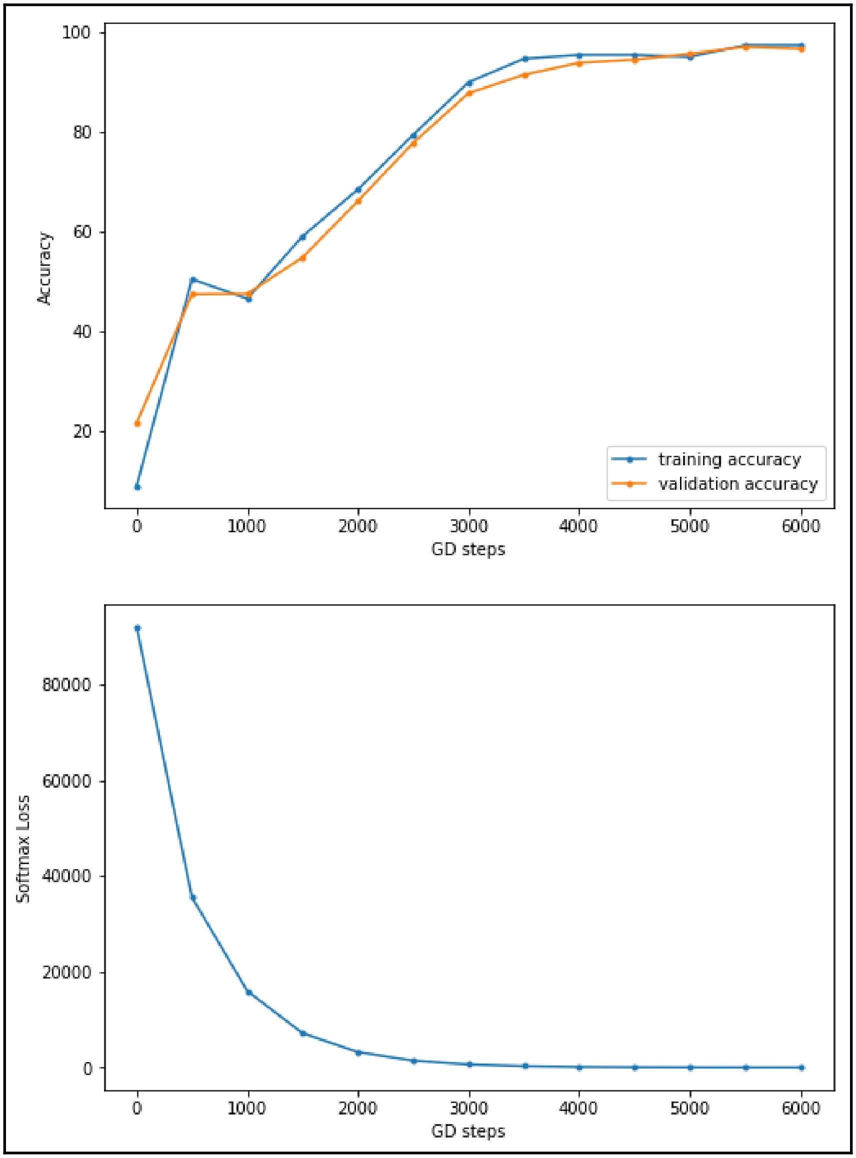


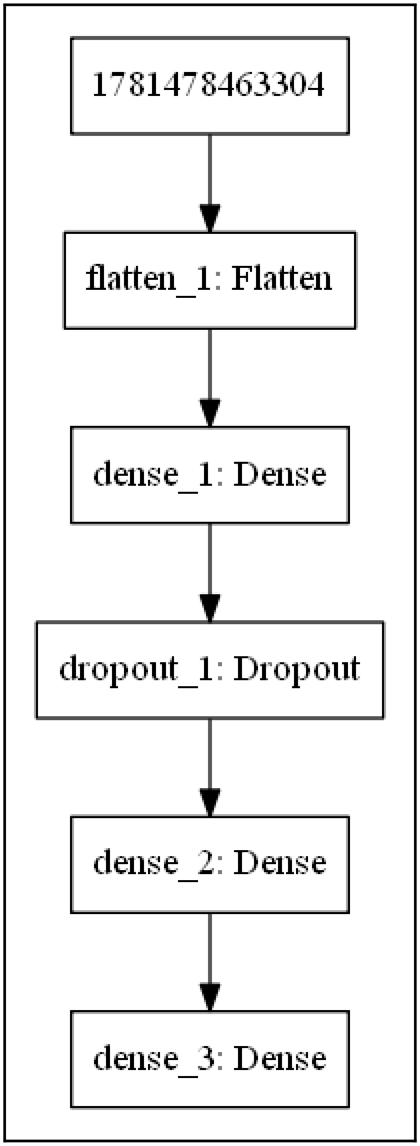
CNN Architecture
Input
CONV
RELU
POOL
Flatten
FC
SOFTMAX
Output

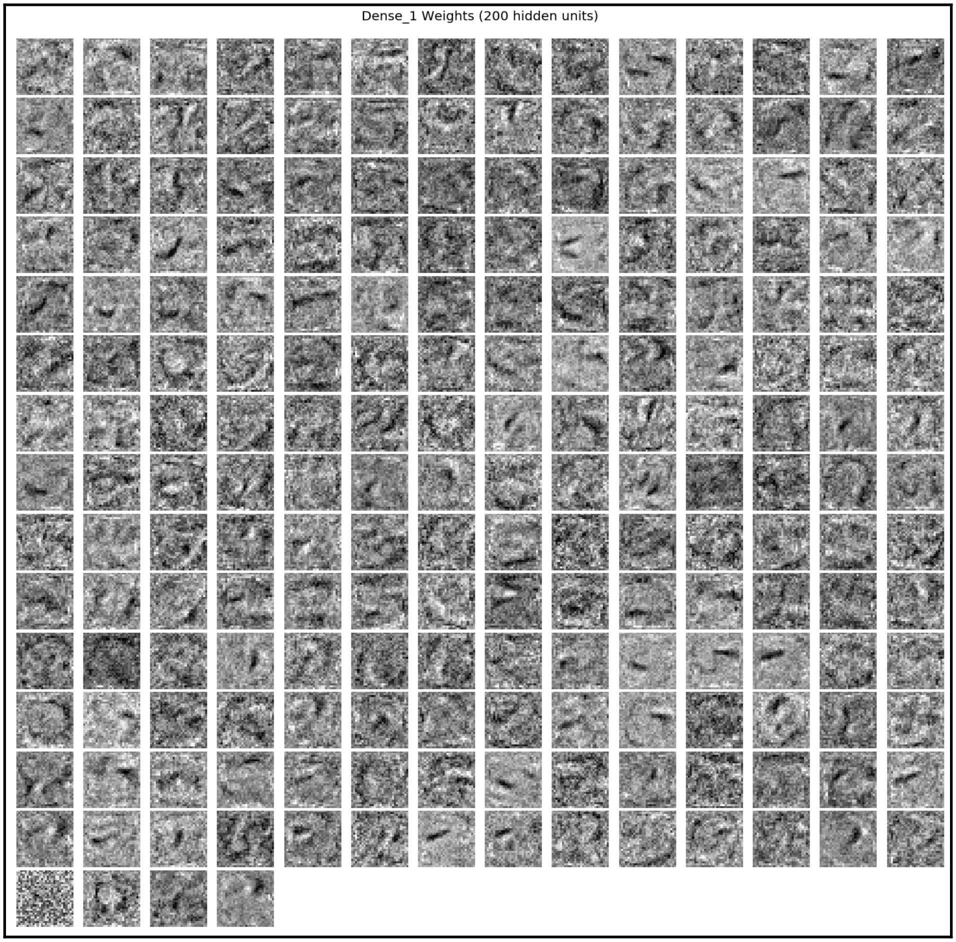

input
n × n image
convolution
f × f filter
padding p
stride s
output
⌊(n+2p−f)/s + 1⌋ × ⌊(n+2p−f)/s + 1⌋ image
Convolution with RGB images : valid (zero) padding
6 x 6 x 3
*
3 x 3 x 3
=
4 x 4
Convolution with 3-channel input with two 3x3 filters: padding=stride=1
Input Volume (+pad 1) (7x7x3)
Filter W0 (3x3x3)
Filter W1 (3x3x3)
Output Volume (3x3x2)
Bias b0 (1x1x1)
Bias b1 (1x1x1)
figure taken from http://cs231n.github.io/convolutional-networks/

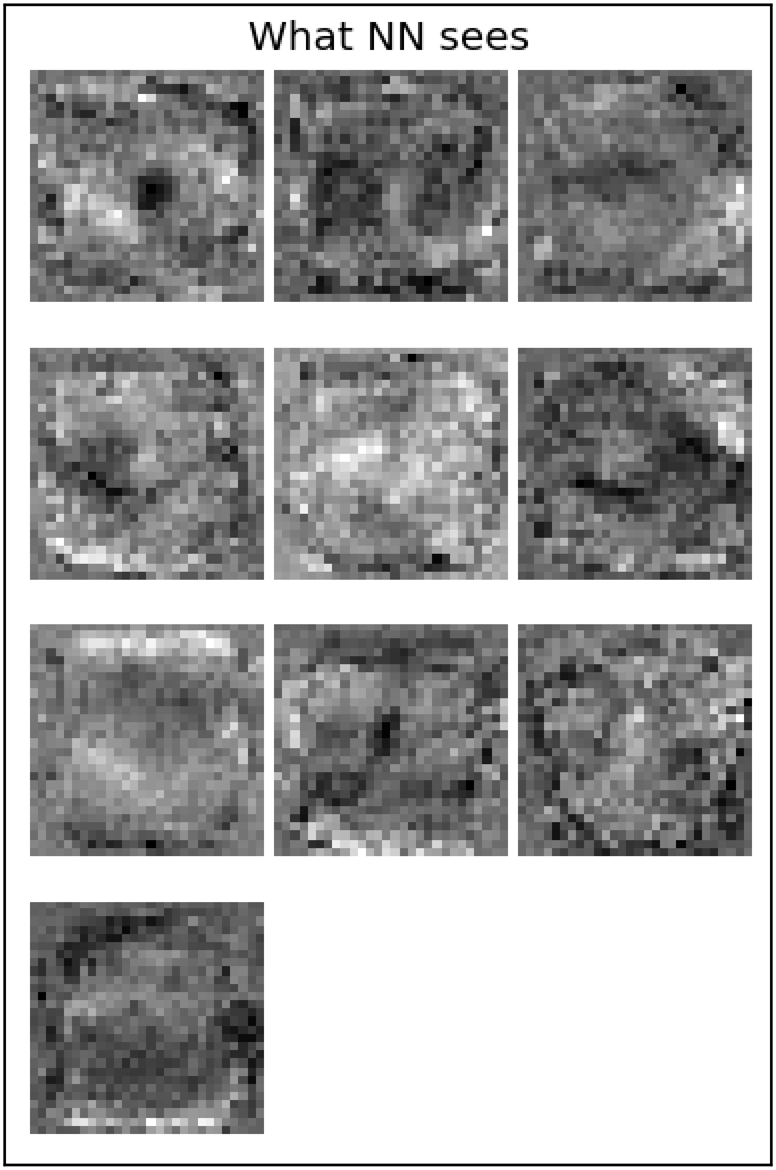

28x28 input nodes
(image)
hidden layer
(1024 nodes)
RELU
bias
bias
0 1 2 3 4 5 6 7 8 9
output nodes

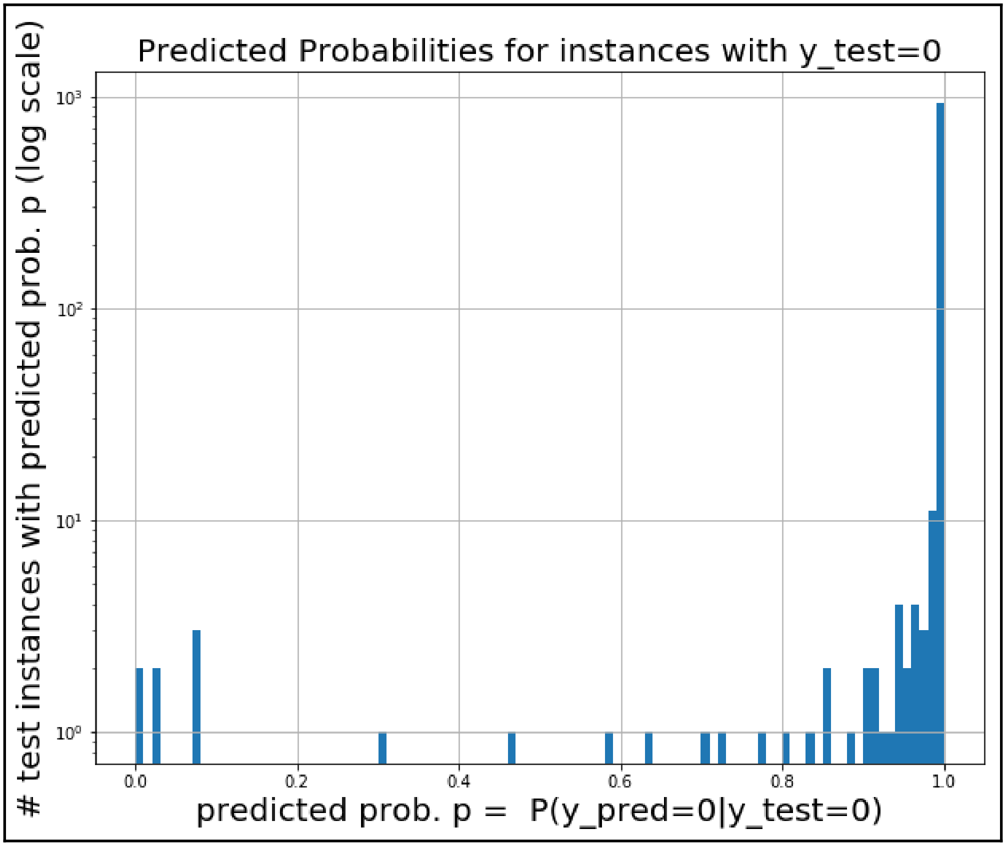

SGD after Step 4000 with lambda1=lambda2=0.1

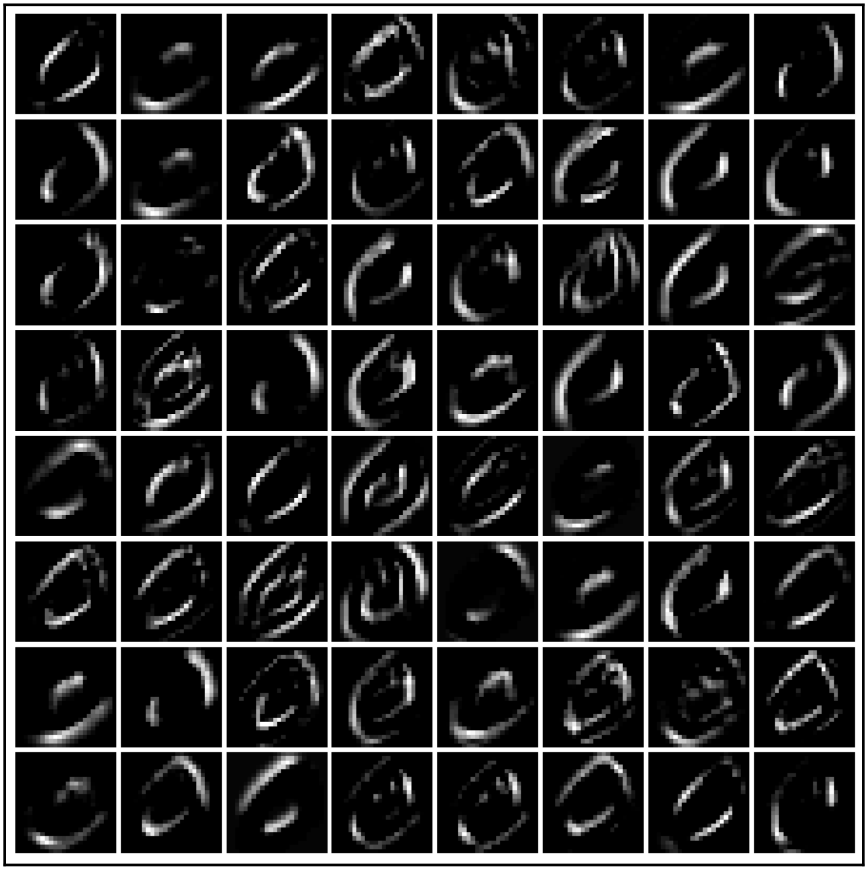

100
80
60
40
20
Accuracy
0
1000
2000
3000
4000
5000
6000
GD steps
training accuracy
validation accuracy
80000
60000
40000
20000
0
Softmax Loss
0
1000
2000
3000
4000
5000
6000
GD steps

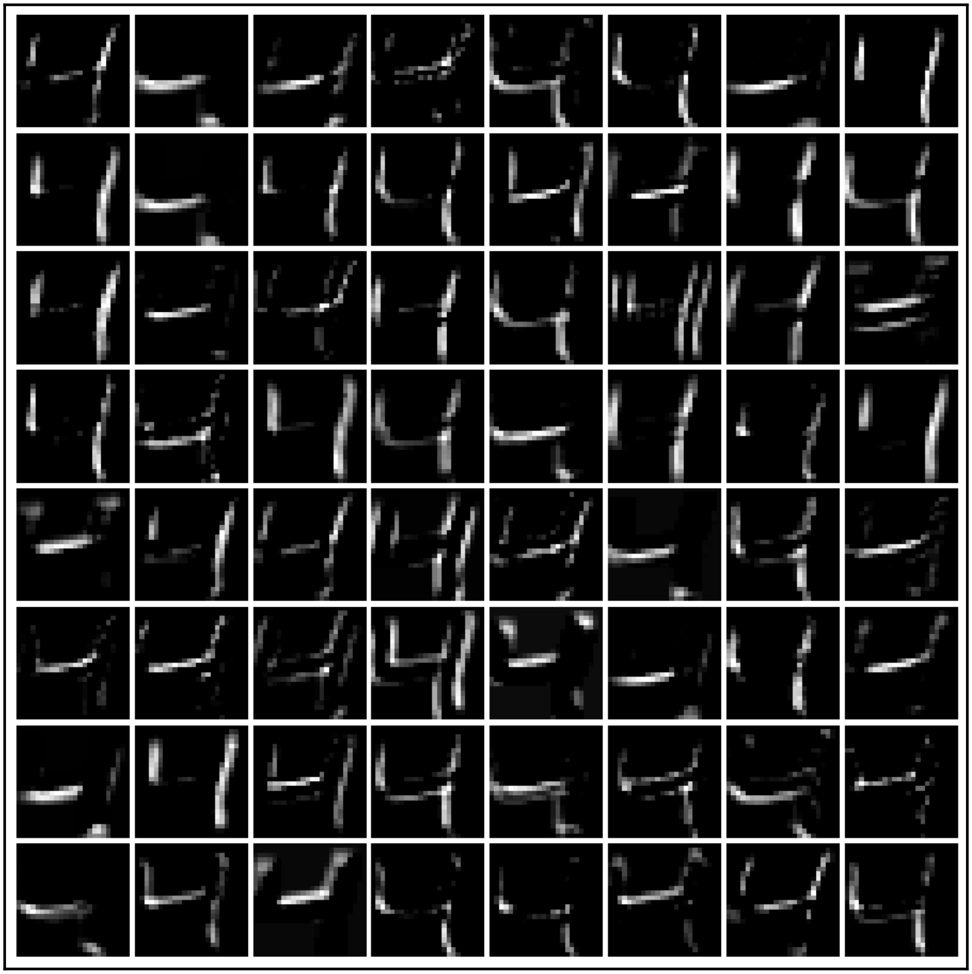

1781478463304
flatten_1: Flatten
dense_1: Dense
dropout_1: Dropout
dense_2: Dense
dense_3: Dense

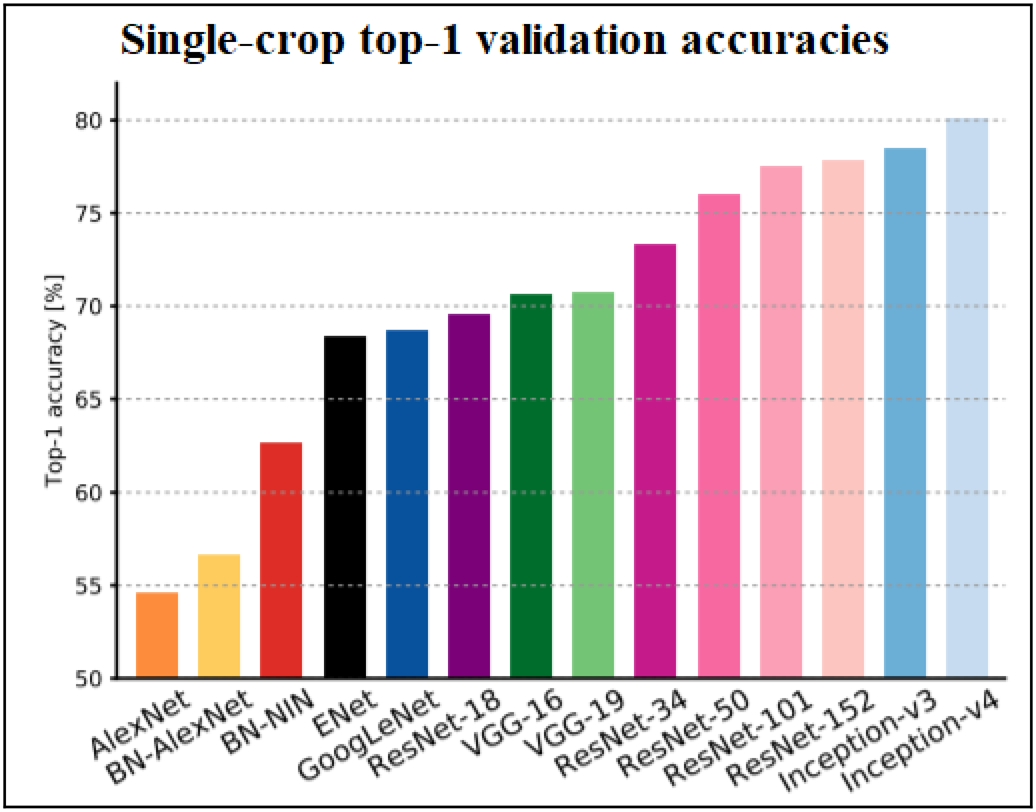

Dense_1 Weights (200 hidden units)

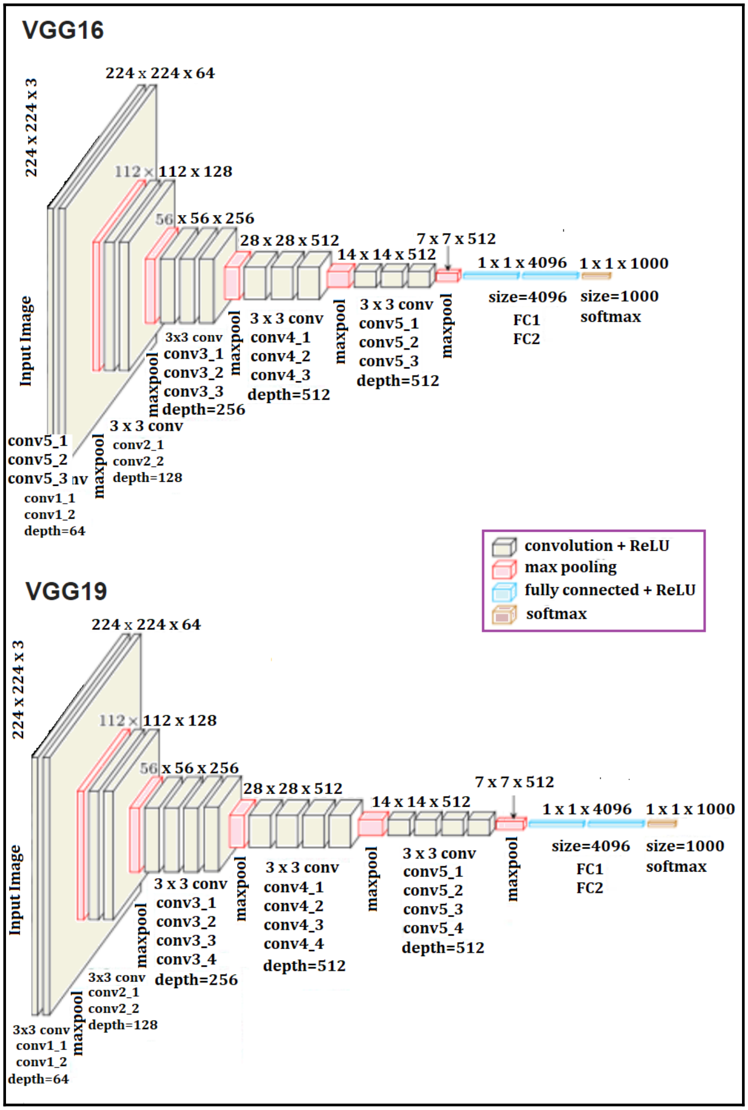

What NN sees

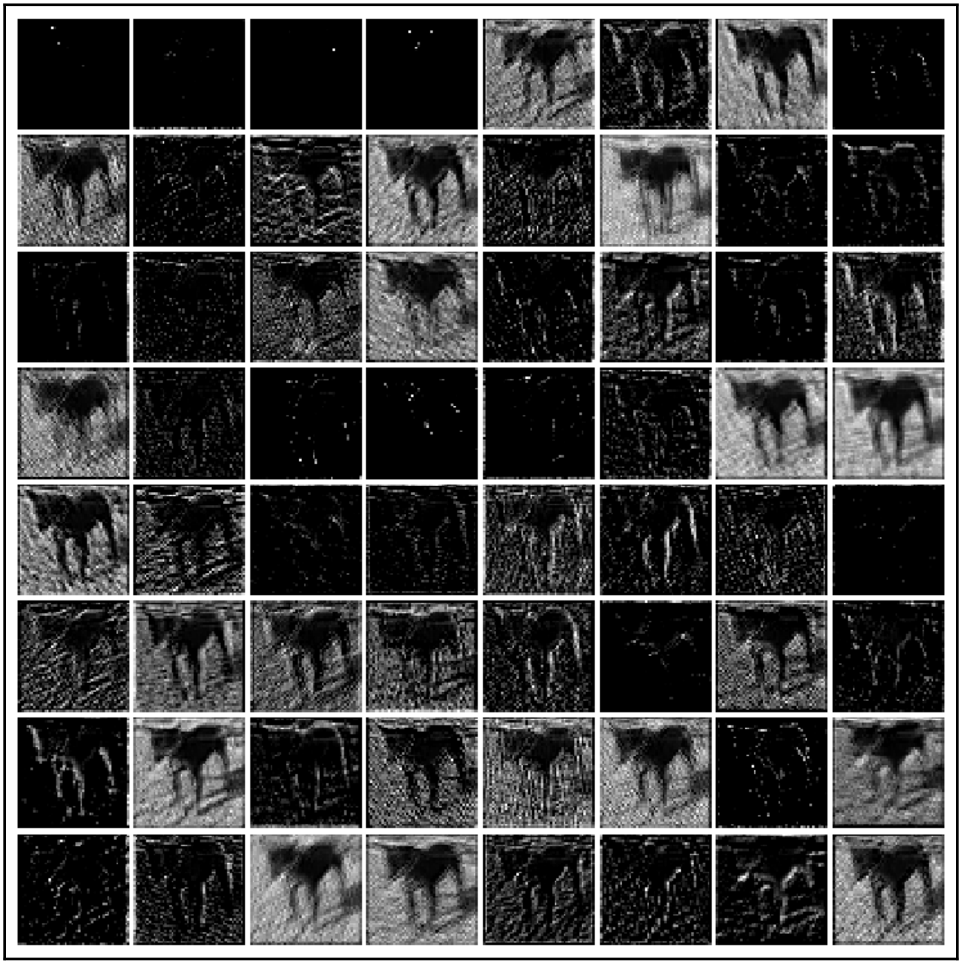

Predicted Probabilities for instances with y_test=0
test instances with predicted prob. p (log scale)
10^3
10^2
10^1
10^0
0.0
0.2
0.4
0.6
0.8
1.0
predicted prob. p = P(y_pred=0|y_test=0)

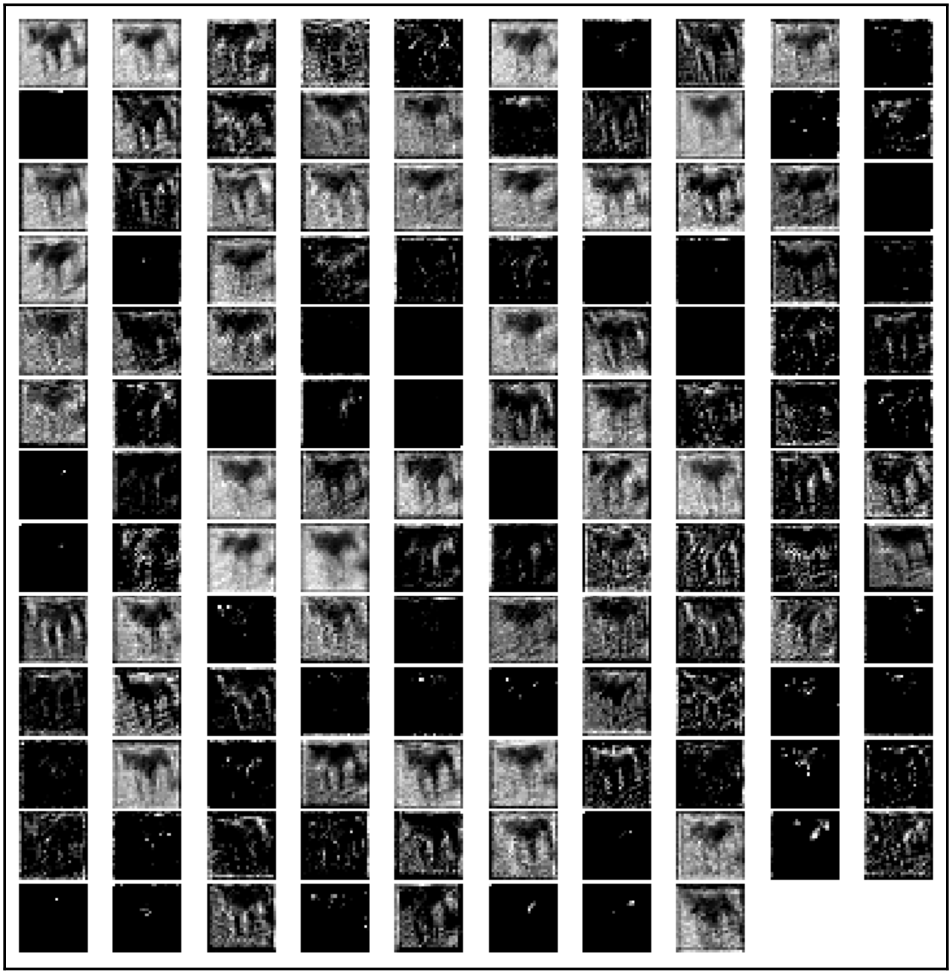




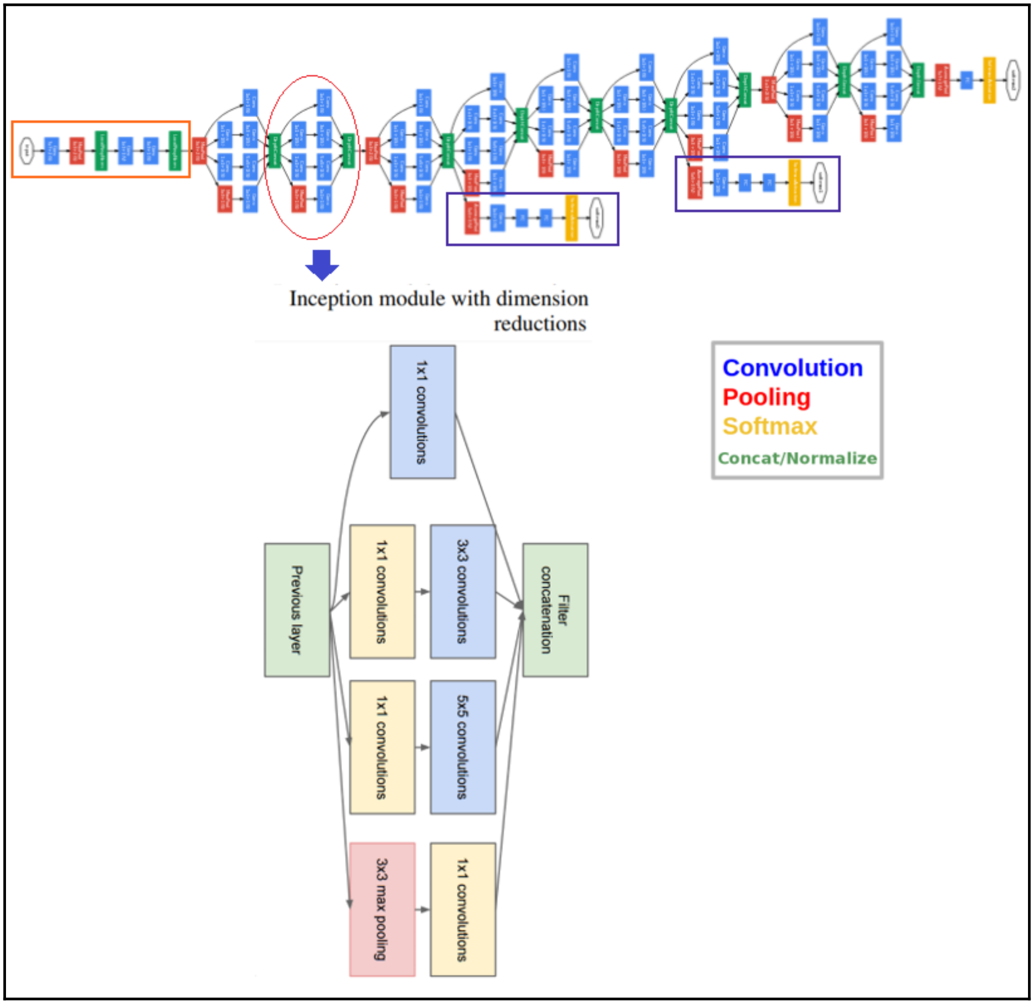

Single-crop top-1 validation accuracies
Top-1 accuracy [%]
80
75
70
65
60
55
50
AlexNet
BN-AlexNet
BN-NIN
ENet
GoogLeNet
ResNet-18
VGG-16
VGG-19
ResNet-34
ResNet-50
ResNet-101
ResNet-152
Inception-v3
Inception-v4

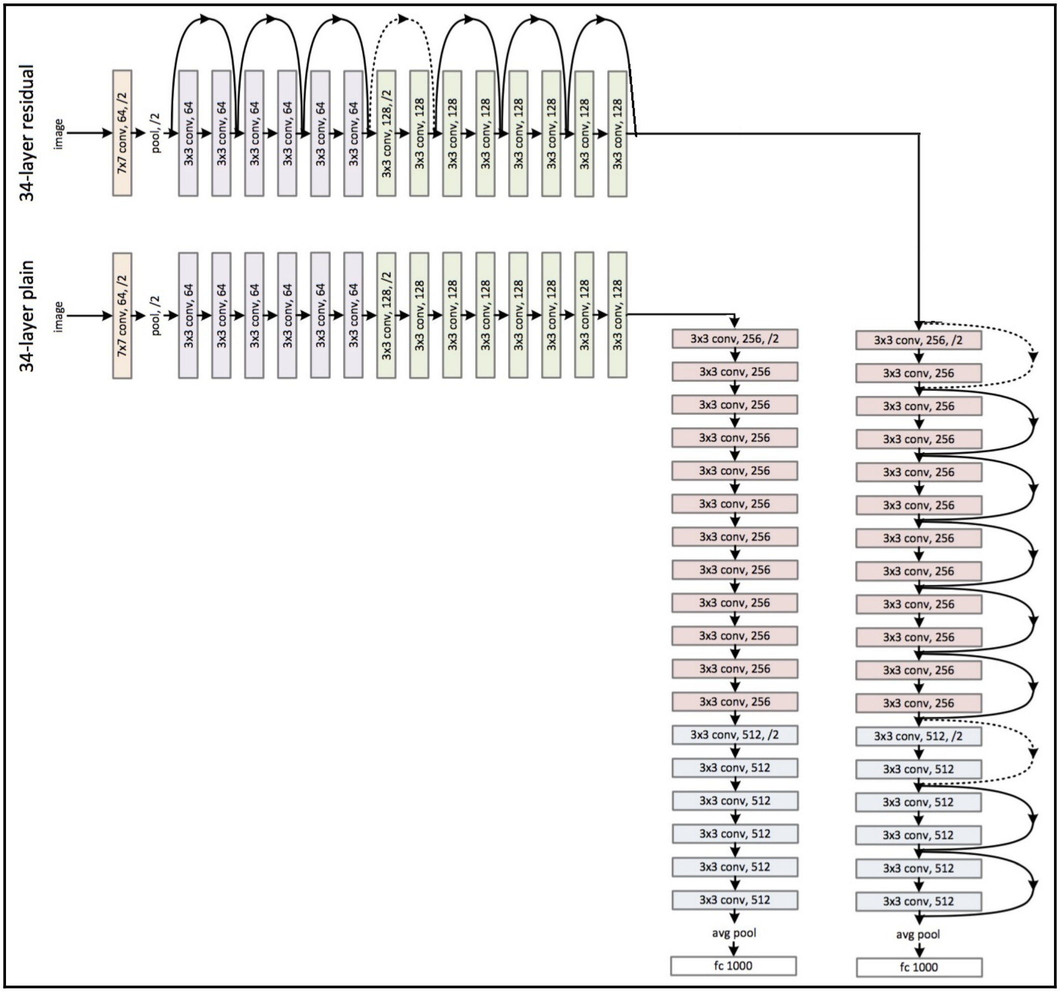

VGG16
224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
Input Image
conv5_1
conv5_2
conv5_3
conv1_1
conv1_2
depth=64
maxpool
3 x 3 conv
conv2_1
conv2_2
depth=128
maxpool
3x3 conv
conv3_1
conv3_2
conv3_3
depth=256
maxpool
3 x 3 conv
conv4_1
conv4_2
conv4_3
depth=512
maxpool
3 x 3 conv
conv5_1
conv5_2
conv5_3
depth=512
maxpool
size=4096
FC1
FC2
size=1000
softmax
convolution + ReLU
max pooling
fully connected + ReLU
softmax
VGG19
224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
Input Image
3x3 conv
conv1_1
conv1_2
depth=64
maxpool
3x3 conv
conv2_1
conv2_2
depth=128
maxpool
3 x 3 conv
conv3_1
conv3_2
conv3_3
conv3_4
depth=256
maxpool
3 x 3 conv
conv4_1
conv4_2
conv4_3
conv4_4
depth=512
maxpool
3 x 3 conv
conv5_1
conv5_2
conv5_3
conv5_4
depth=512
maxpool
size=4096
FC1
FC2
size=1000
softmax



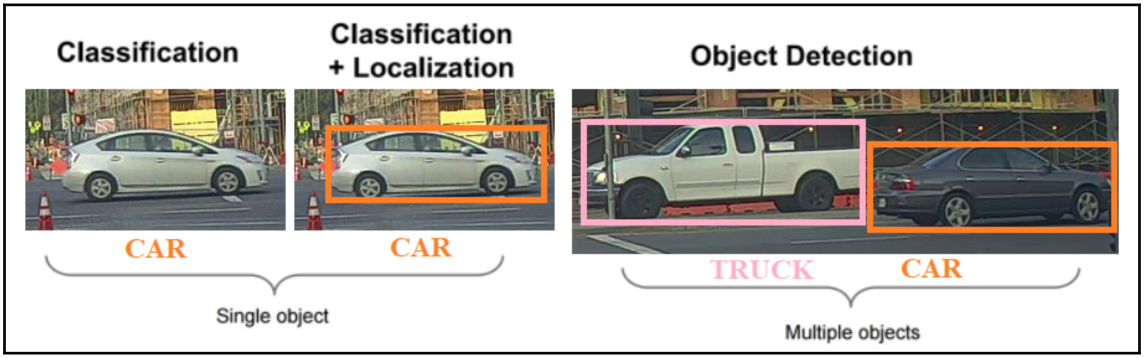


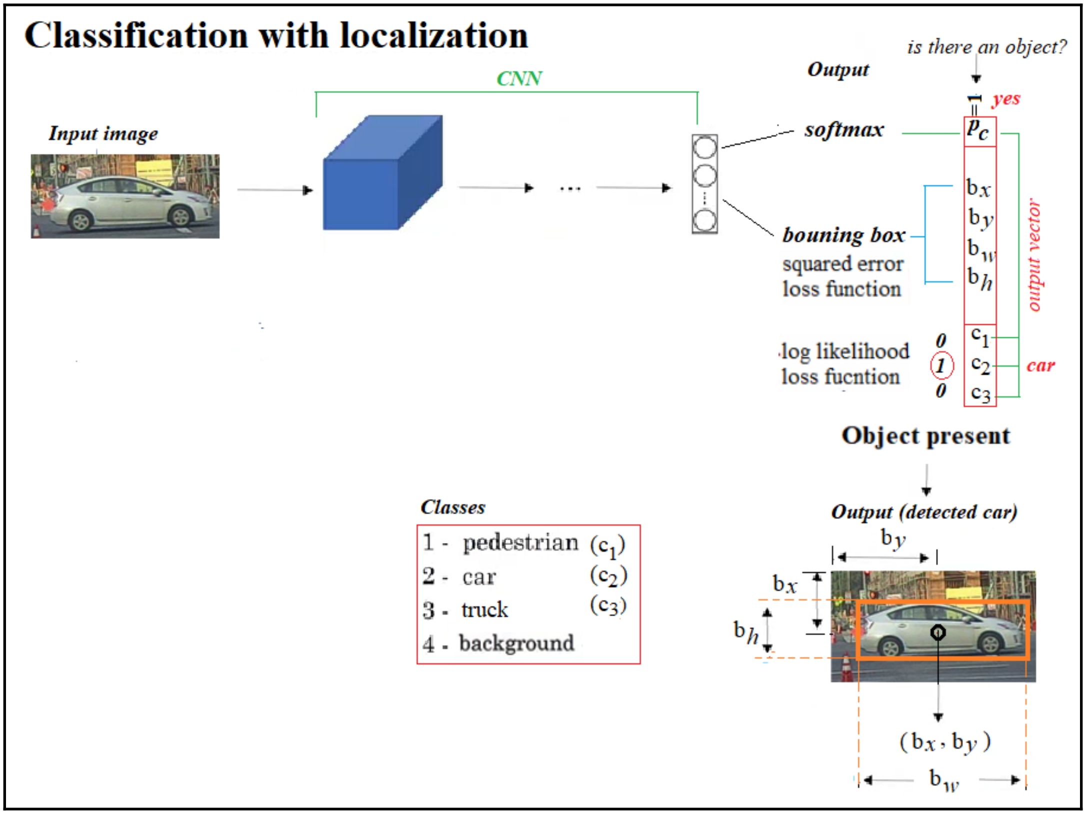

cat, prob=0.89
dog, prob=1.00
dog, prob=0.93
dog, prob=0.76
dog, prob=0.83
dog, prob=0.64
dog, prob=0.96
cat, prob=0.71
dog, prob=0.53
dog, prob=0.62
cat, prob=0.61
cat, prob=0.93
dog, prob=1.00
dog, prob=1.00
cat, prob=0.77
dog, prob=1.00
cat, prob=0.99
dog, prob=0.97
dog, prob=0.94
dog, prob=0.92
cat, prob=0.98
cat, prob=0.53
dog, prob=0.84
dog, prob=0.87
cat, prob=0.64
cat, prob=0.95
cat, prob=0.92
cat, prob=0.60
dog, prob=0.71
cat, prob=0.89
dog, prob=0.58
dog, prob=0.84
cat, prob=0.77
dog, prob=0.59
cat, prob=0.98
cat, prob=0.97
cat, prob=0.91
dog, prob=0.92
dog, prob=0.50
dog, prob=0.64
dog, prob=0.94
dog, prob=0.96
cat, prob=0.96
cat, prob=0.96
dog, prob=0.61
cat, prob=0.88
cat, prob=0.92
dog, prob=0.77
dog, prob=0.88
cat, prob=0.89
cat, prob=0.96
cat, prob=0.55
dog, prob=0.50
dog, prob=0.82
dog, prob=0.99
cat, prob=0.97
dog, prob=0.82
dog, prob=0.80
cat, prob=0.65
cat, prob=0.84
dog, prob=0.68
dog, prob=1.00
dog, prob=0.89
dog, prob=0.96
dog, prob=0.55
dog, prob=0.95
cat, prob=0.92
cat, prob=0.73
dog, prob=0.62
dog, prob=0.70
dog, prob=0.57
cat, prob=0.91
cat, prob=0.75
cat, prob=0.69
dog, prob=1.00
cat, prob=0.52
dog, prob=0.73
dog, prob=0.71
cat, prob=0.52
cat, prob=0.84
cat, prob=0.81
cat, prob=0.81
dog, prob=0.72
cat, prob=0.99
cat, prob=0.59
cat, prob=0.90
cat, prob=0.70
cat, prob=0.93
cat, prob=0.56
cat, prob=0.88
dog, prob=0.99
dog, prob=0.99
cat, prob=0.75
dog, prob=0.99
cat, prob=0.77
cat, prob=0.62
cat, prob=0.57
cat, prob=0.77
dog, prob=0.66
dog, prob=0.81


Inception module with dimension reductions
Convolution
Pooling
Softmax
Concat/Normalize
Previous layer
1x1 convolutions
3x3 convolutions
1x1 convolutions
5x5 convolutions
3x3 max pooling
1x1 convolutions
Filter concatenation


34-layer residual
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

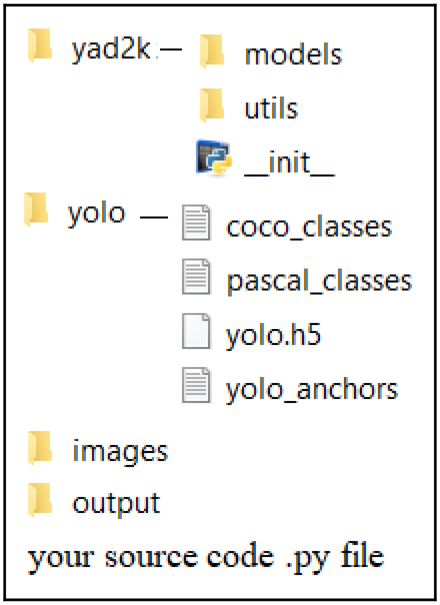
Chapter 11: Deep Learning in Image Processing - Object Detection, and more

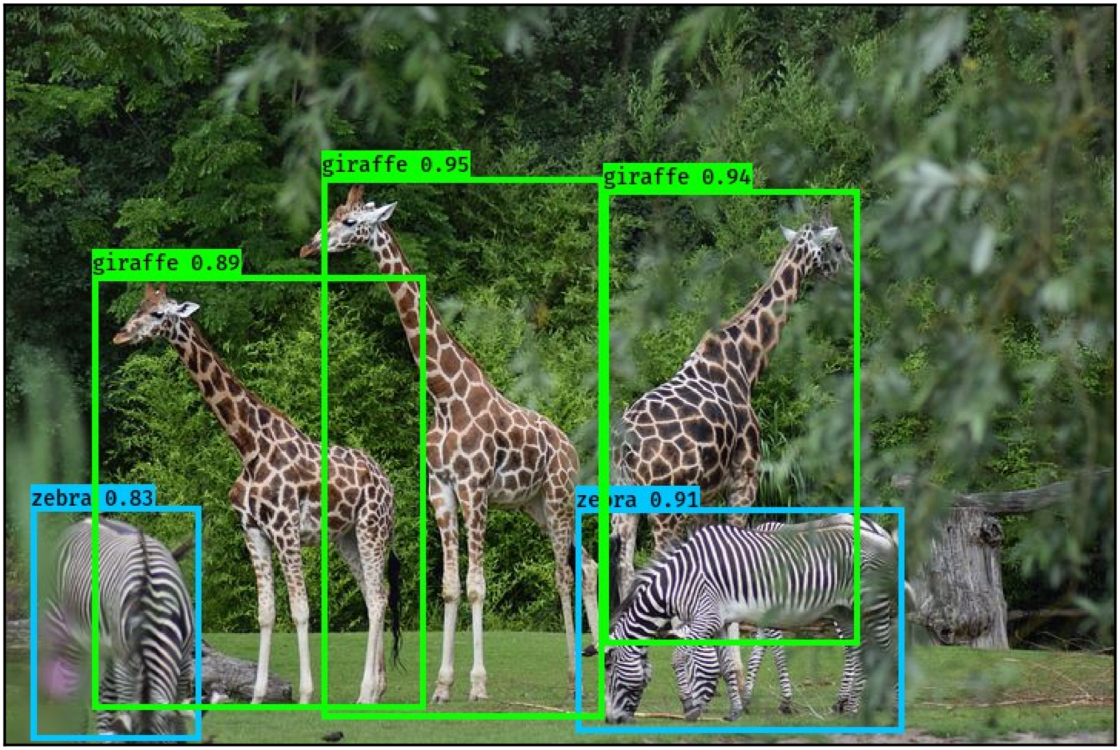




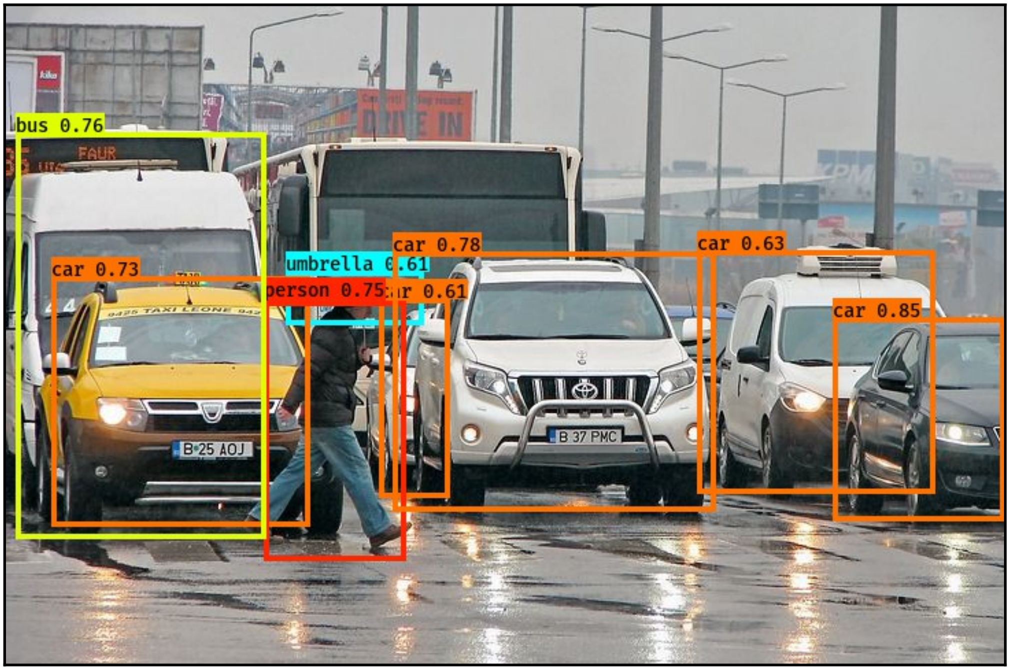
Performance on the COCO Dataset

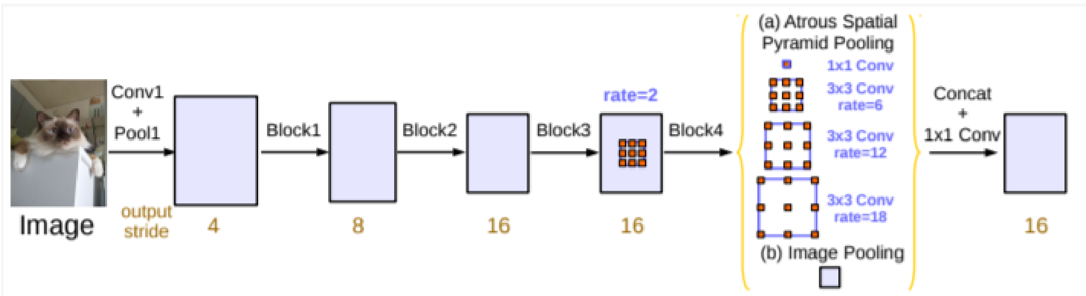
Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights




giraffe 0.95
giraffe 0.94
giraffe 0.89
zebra 0.83
zebra 0.91


FAUR
TAXI
B 25 AOJ
B 37 PMC





Parallel modules with atrous convolution (ASPP), augmented with image-level features.

Taken from *https://arxiv.org/pdf/1706.05587.pdf*

Deeplab V3+ architecture

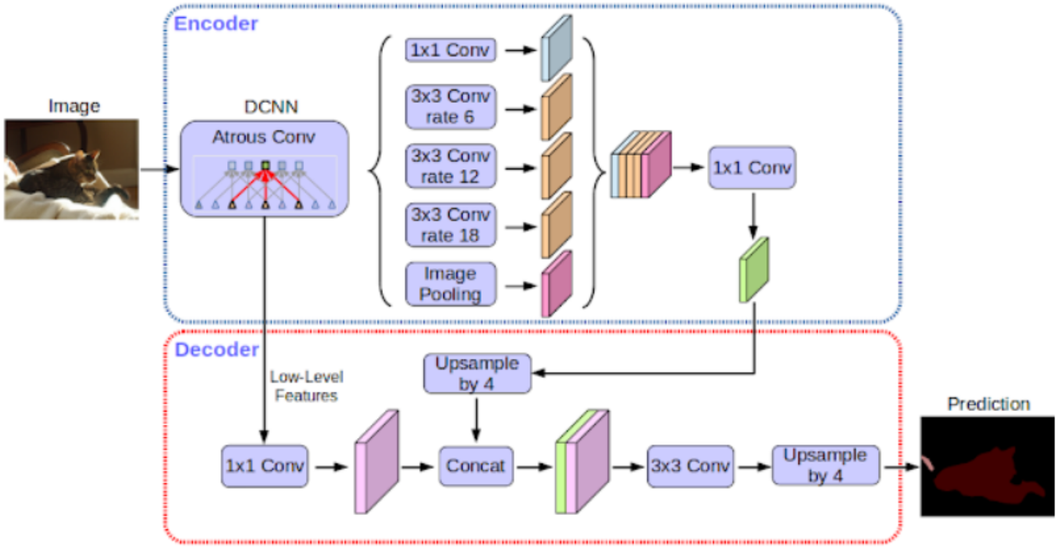
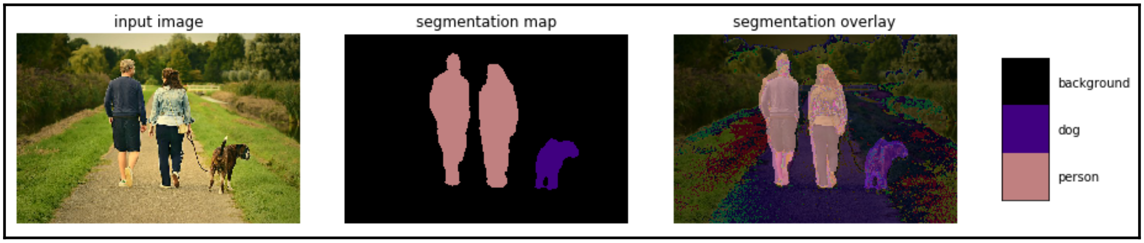
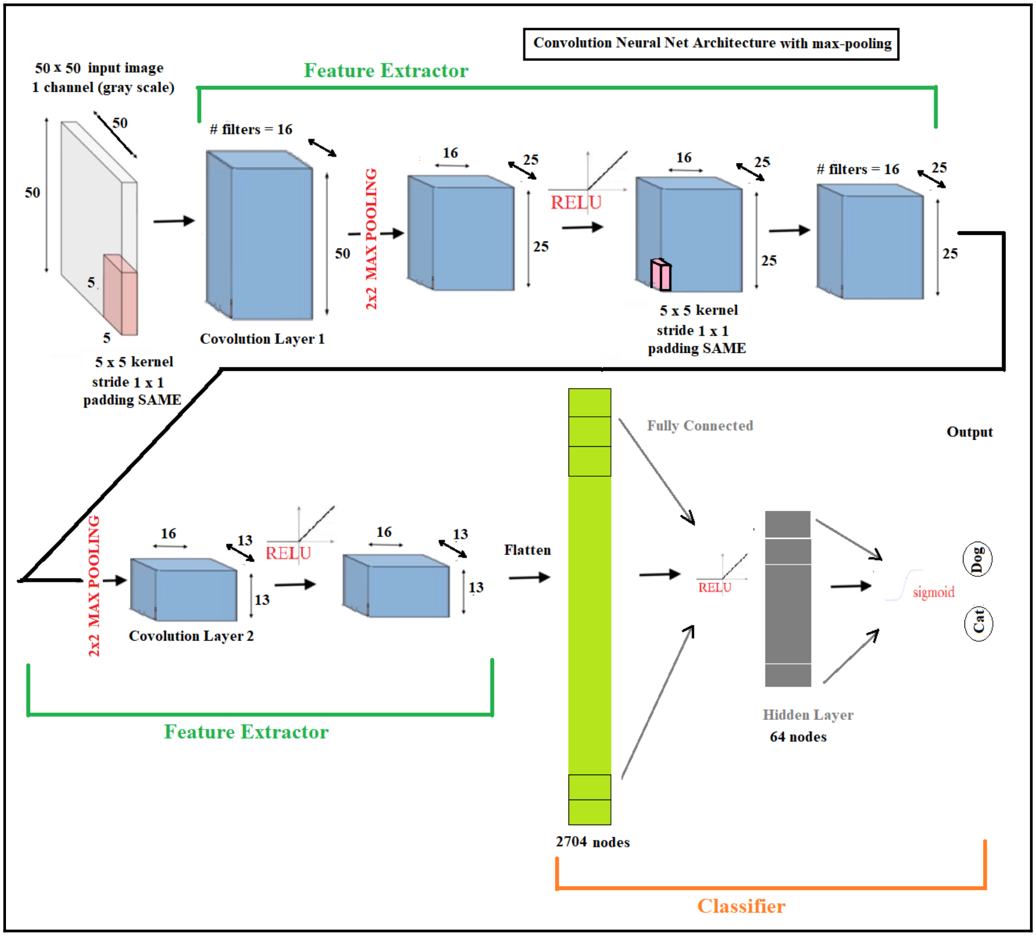


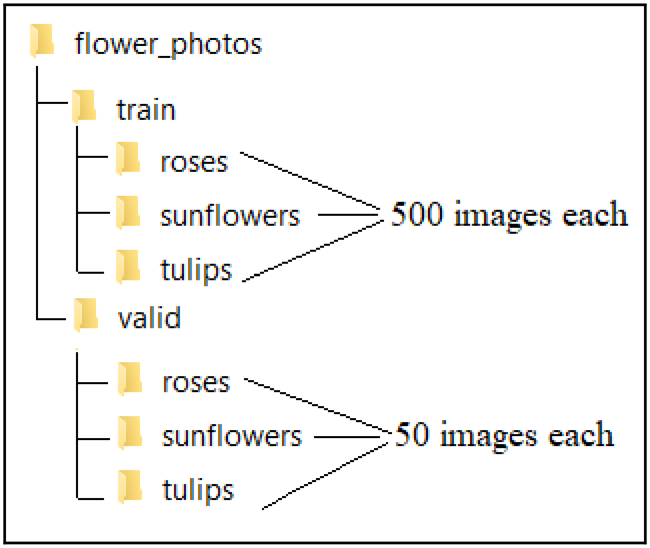
image taken from *https://ai.googleblog.com/2018/03/semantic-image-segmentation-*

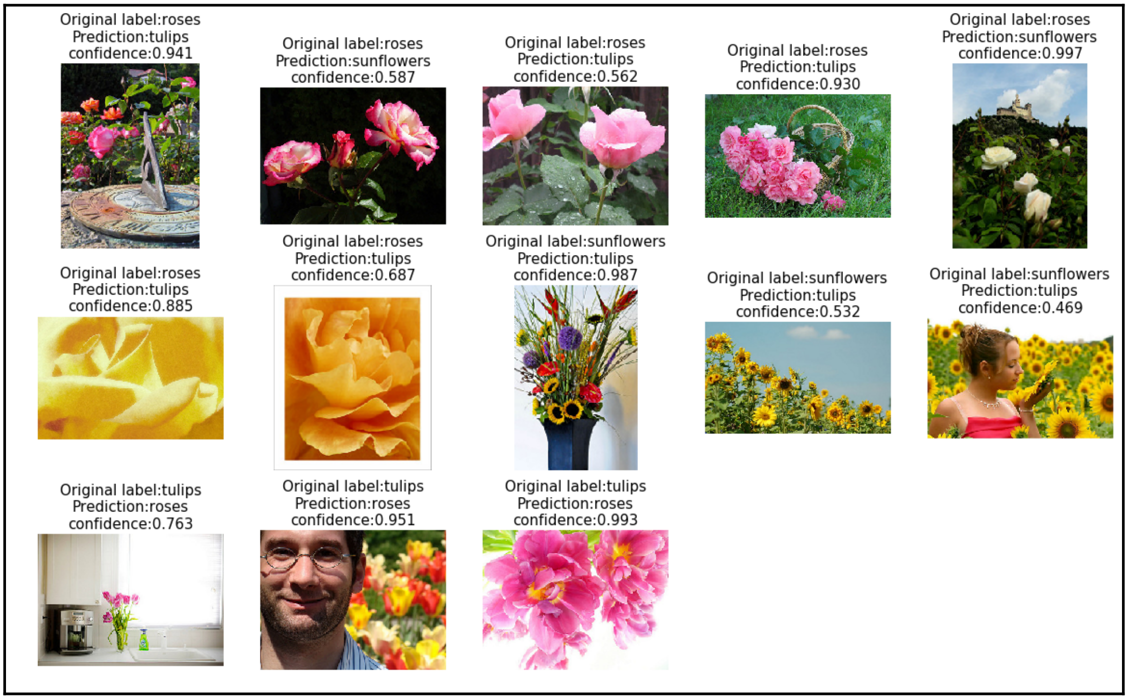

14
12
10
8
6
4
2
0


input image
segmentation map
segmentation overlay
background
dog
person

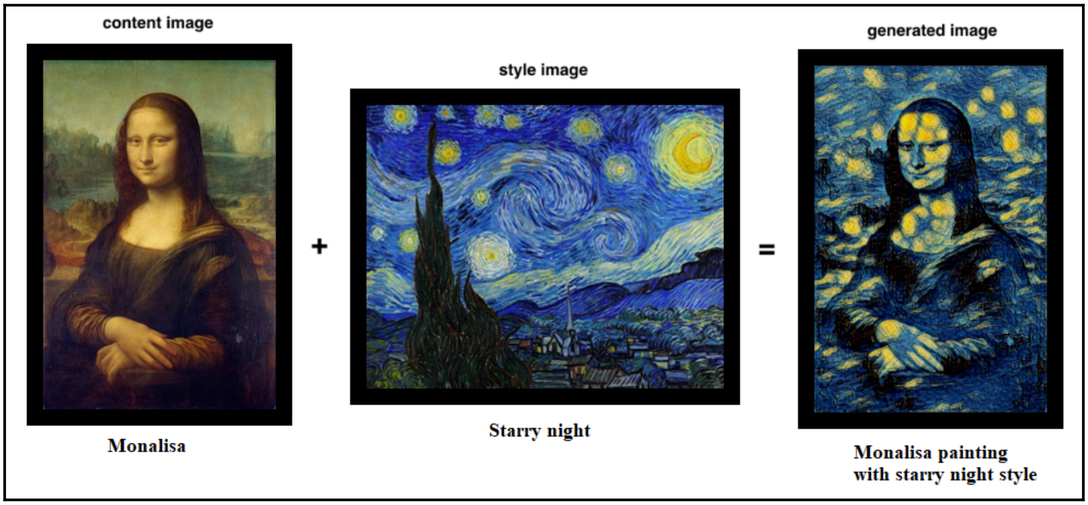

VGG16: flamingo, 0.83
VGG16: daisy, 0.43
VGG16: artichoke, 0.33


Convolution Neural Net Architecture with max-pooling
50 x 50 input image
1 channel (gray scale)
Feature Extractor
50
50
5
5
5 x 5 kernel
stride 1 x 1
padding SAME
filters = 16
50
Covolution Layer 1
2x2 MAX POOLING
16
25
25
RELU
16
25
25
5 x 5 kernel
stride 1 x 1
padding SAME
filters = 16
25
25
2x2 MAX POOLING
16
13
13
Covolution Layer 2
RELU
16
13
13
Feature Extractor
Flatten
2704 nodes
Fully Connected
RELU
Hidden Layer
64 nodes
sigmoid
Output
Dog
Cat
Classifier


flower_photos
train
roses
sunflowers
tulips
500 images each
valid
roses
sunflowers
tulips
50 images each


Original label:roses
Prediction:tulips
confidence:0.941
Original label:roses
Prediction:sunflowers
confidence:0.587
Original label:roses
Prediction:tulips
confidence:0.562
Original label:roses
Prediction:tulips
confidence:0.930
Original label:roses
Prediction:sunflowers
confidence:0.997
Original label:roses
Prediction:tulips
confidence:0.885
Original label:roses
Prediction:tulips
confidence:0.687
Original label:sunflowers
Prediction:tulips
confidence:0.987
Original label:sunflowers
Prediction:tulips
confidence:0.532
Original label:sunflowers
Prediction:tulips
confidence:0.469
Original label:tulips
Prediction:roses
confidence:0.763
Original label:tulips
Prediction:roses
confidence:0.951
Original label:tulips
Prediction:roses
confidence:0.993


VGG16 Transfer Learning: roses, 1.00
VGG16 Transfer Learning: sunflowers, 1.0
VGG16 Transfer Learning: tulips, 1.00


content image
style image
generated image
+
=
Monalisa
Starry night
Monalisa painting
with starry night style