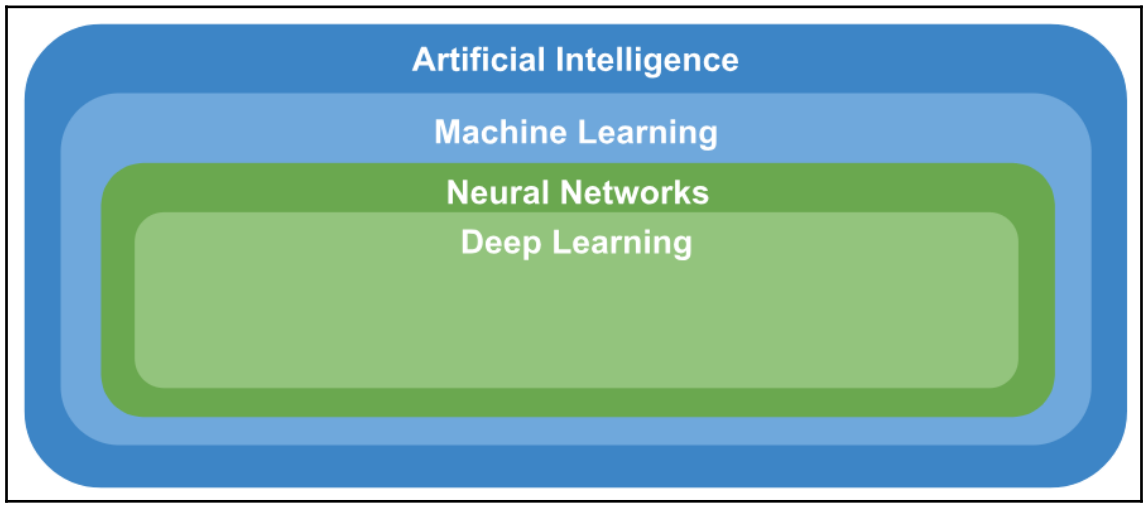
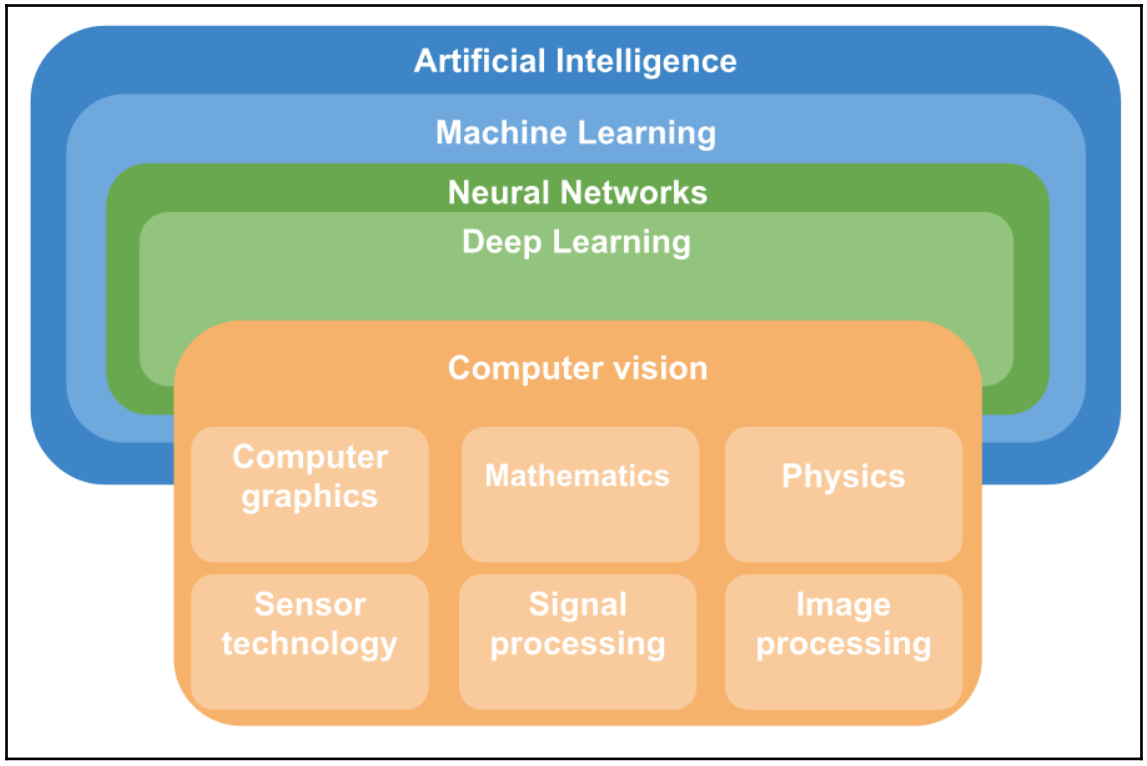
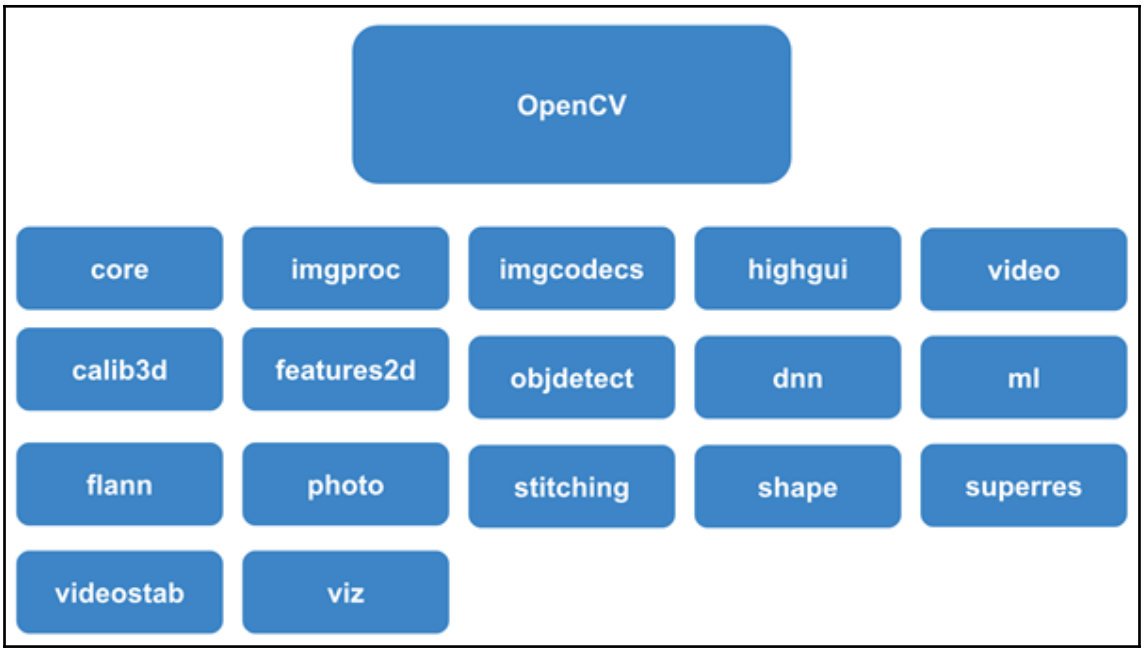
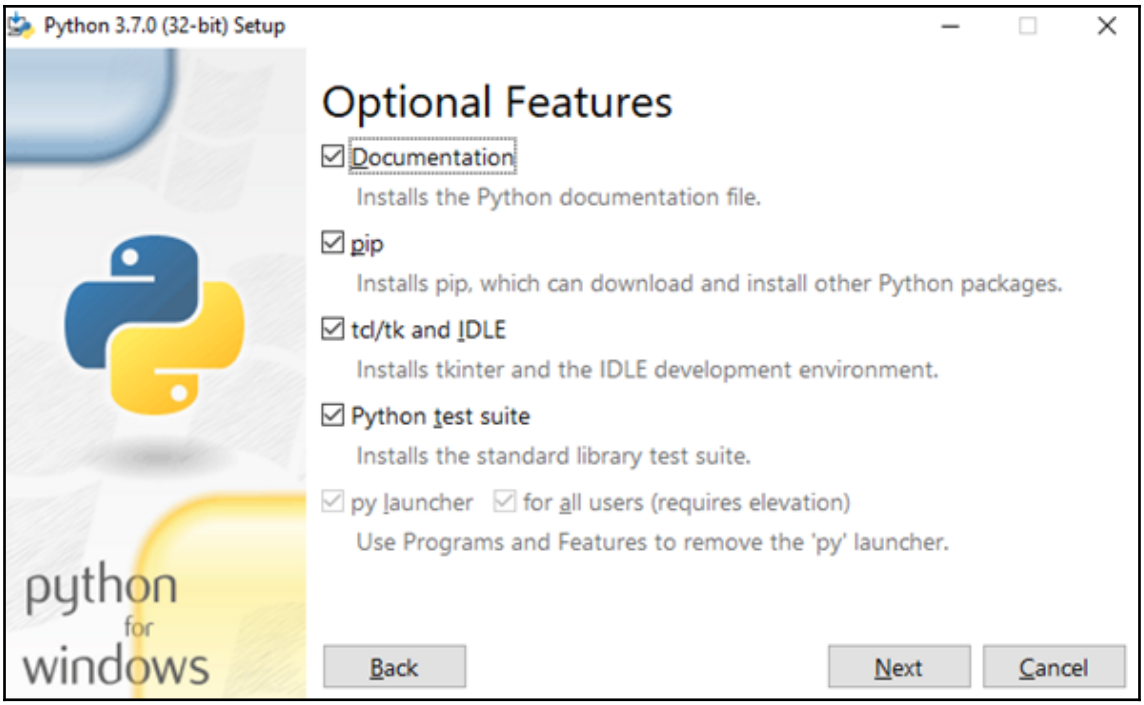
Chapter 1: Setting Up OpenCV

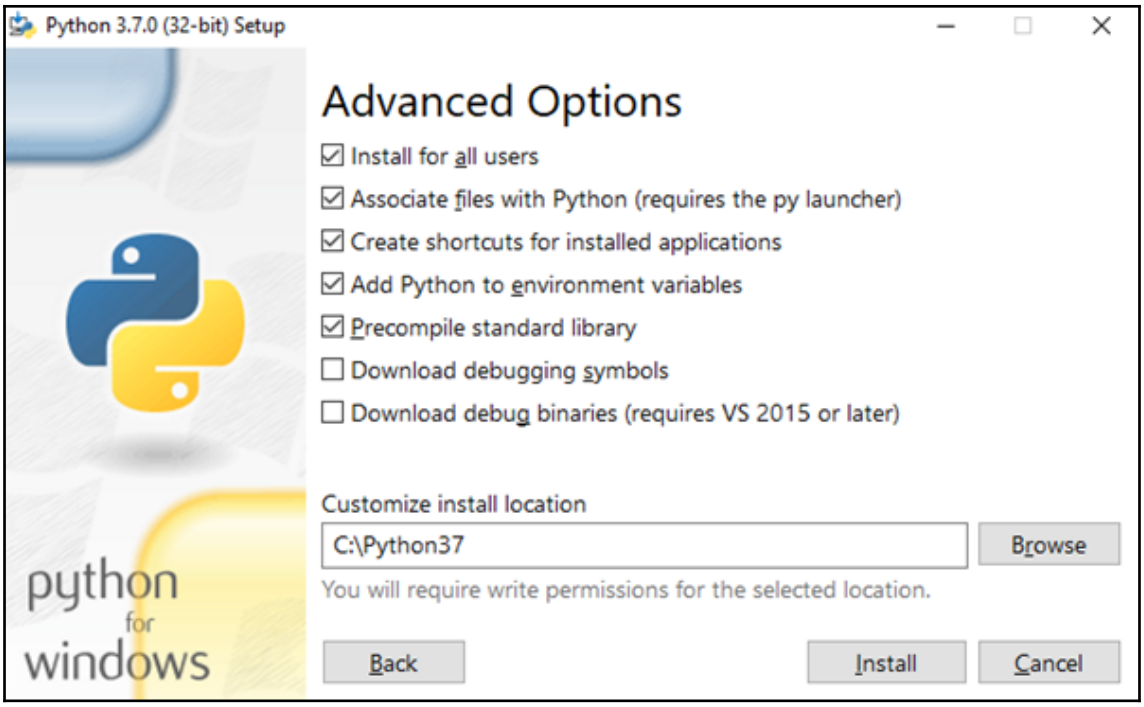


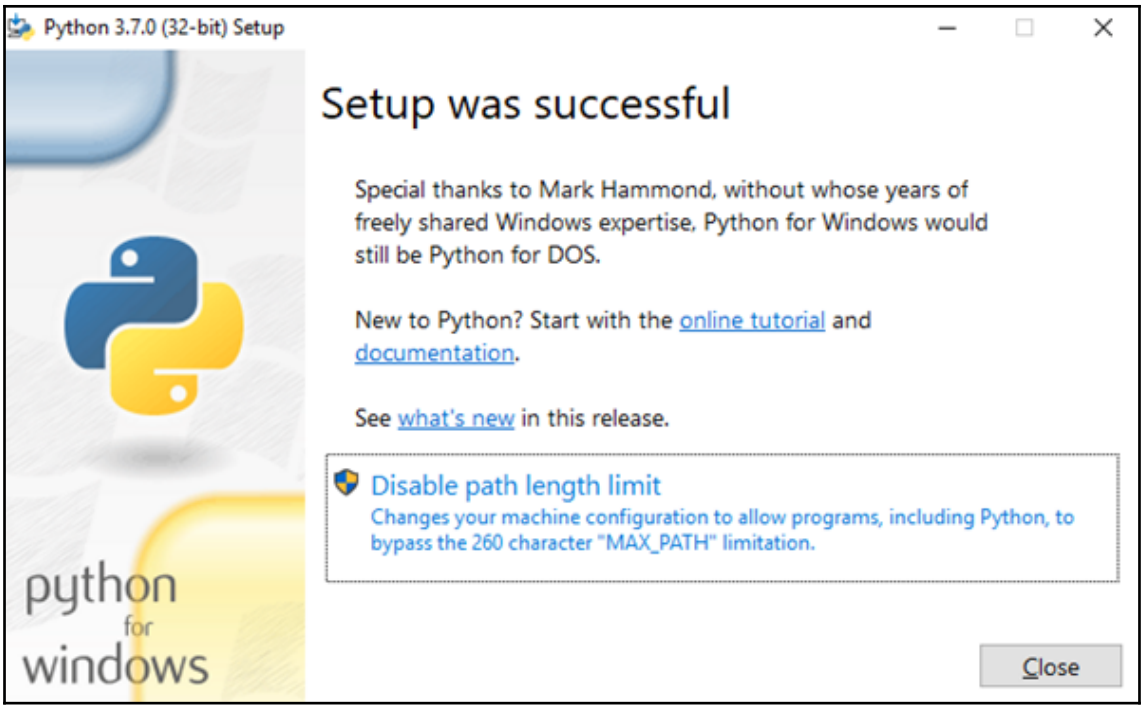


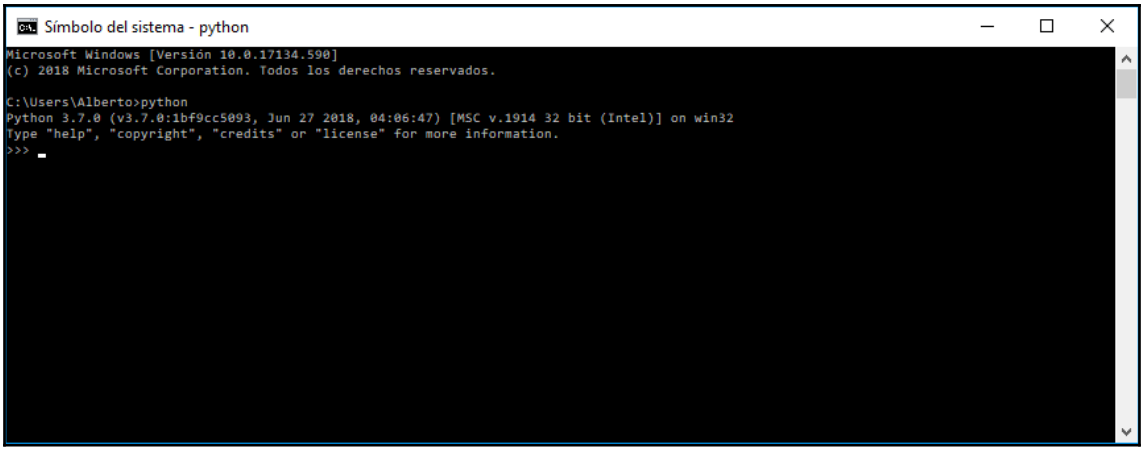

OpenCV
core
imgproc
imgcodecs
highgui
video
calib3d
features2d
objdetect
dnn
ml
flann
photo
stitching
shape
superres
videostab
viz


Python 3.7.0 (32-bit) Setup
Install Python 3.7.0 (32-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
Install Now
C:\Users\Alberto\AppData\Local\Programs\Python\Python37-32
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python
for
windows
Install launcher for all users (recommended)
Add Python 3.7 to PATH
Cancel

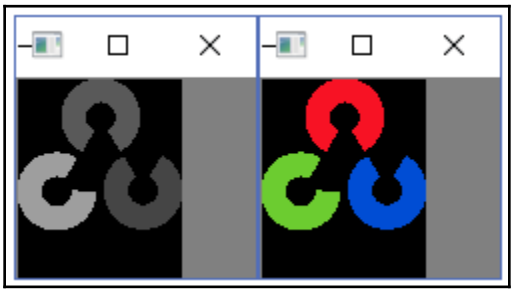

Python 3.7.0 (32-bit) Setup
Optional Features
Documentation
Installs the Python documentation file.
pip
Installs pip, which can download and install other Python packages.
tcl/tk and IDLE
Installs tkinter and the IDLE development environment.
Python test suite
Installs the standard library test suite.
py launcher
for all users (requires elevation)
Use Programs and Features to remove the 'py' launcher.
python
for
windows
Back
Next
Cancel

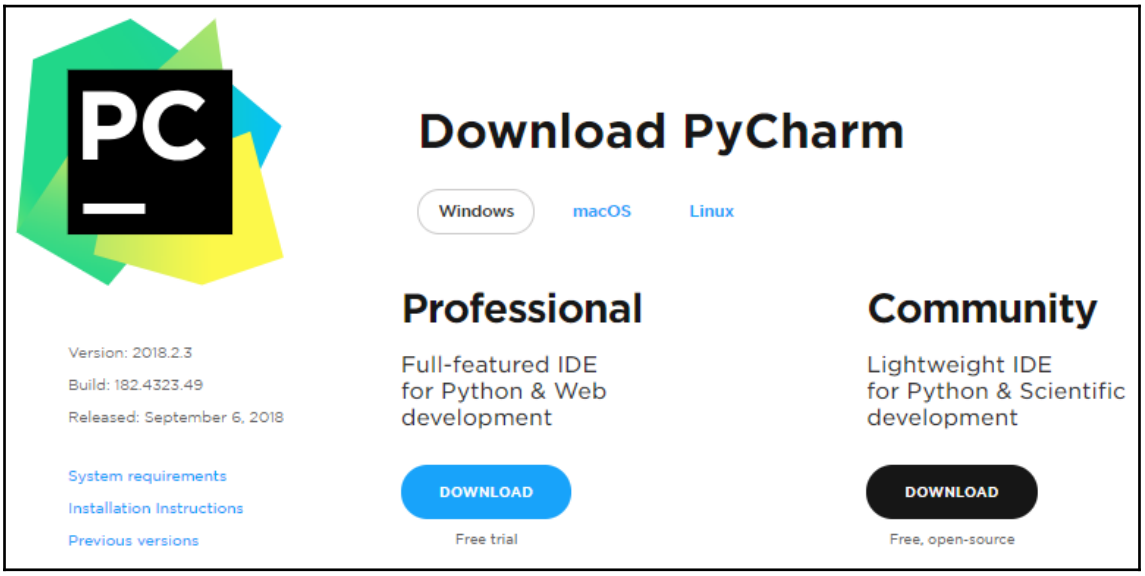

Python 3.7.0 (32-bit) Setup
Advanced Options
Install for all users
Associate files with Python (requires the py launcher)
Create shortcuts for installed applications
Add Python to environment variables
Precompile standard library
Download debugging symbols
Download debug binaries (requires VS 2015 or later)
Customize install location
C:\Python37
Browse
You will require write permissions for the selected location.
python
for
windows
Back
Install
Cancel

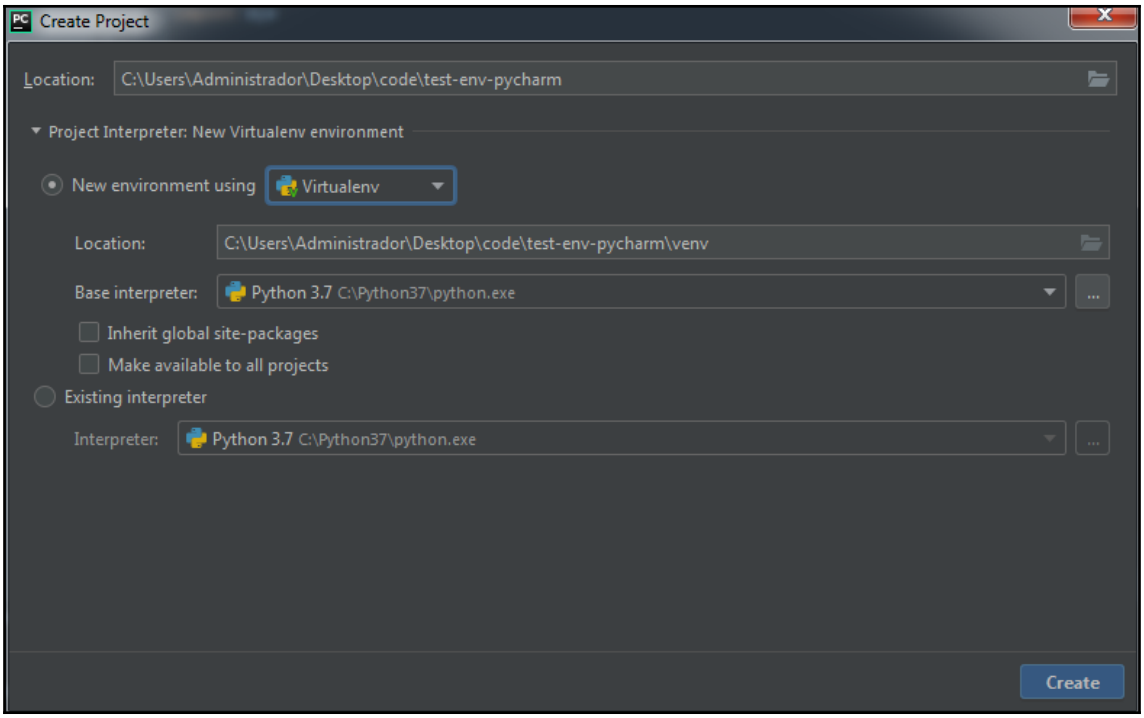

Python 3.7.0 (32-bit) Setup
Setup was successful
Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.
New to Python? Start with the online tutorial and documentation.
See what's new in this release.
Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.
python for windows
Close

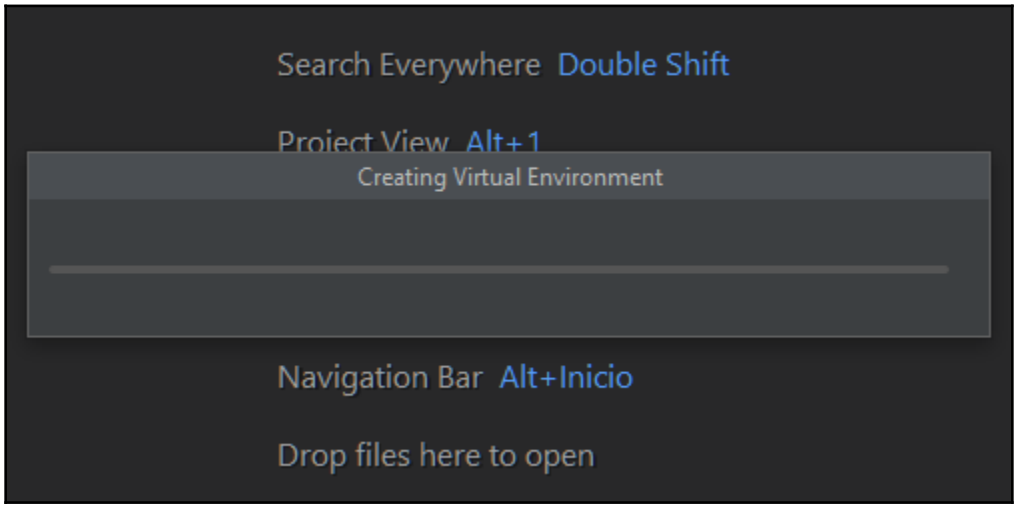

Símbolo del sistema - python
Microsoft Windows [Versión 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.
C:\Users\Alberto>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

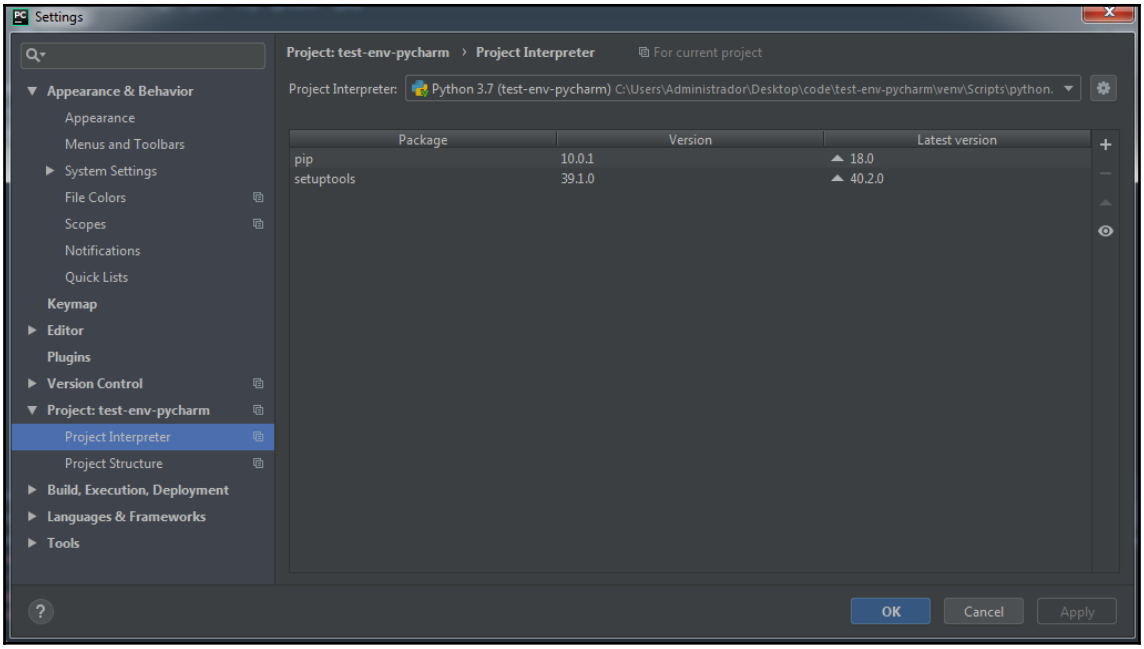

OpenCV
logo.png
test_opencv_inst
allation.py

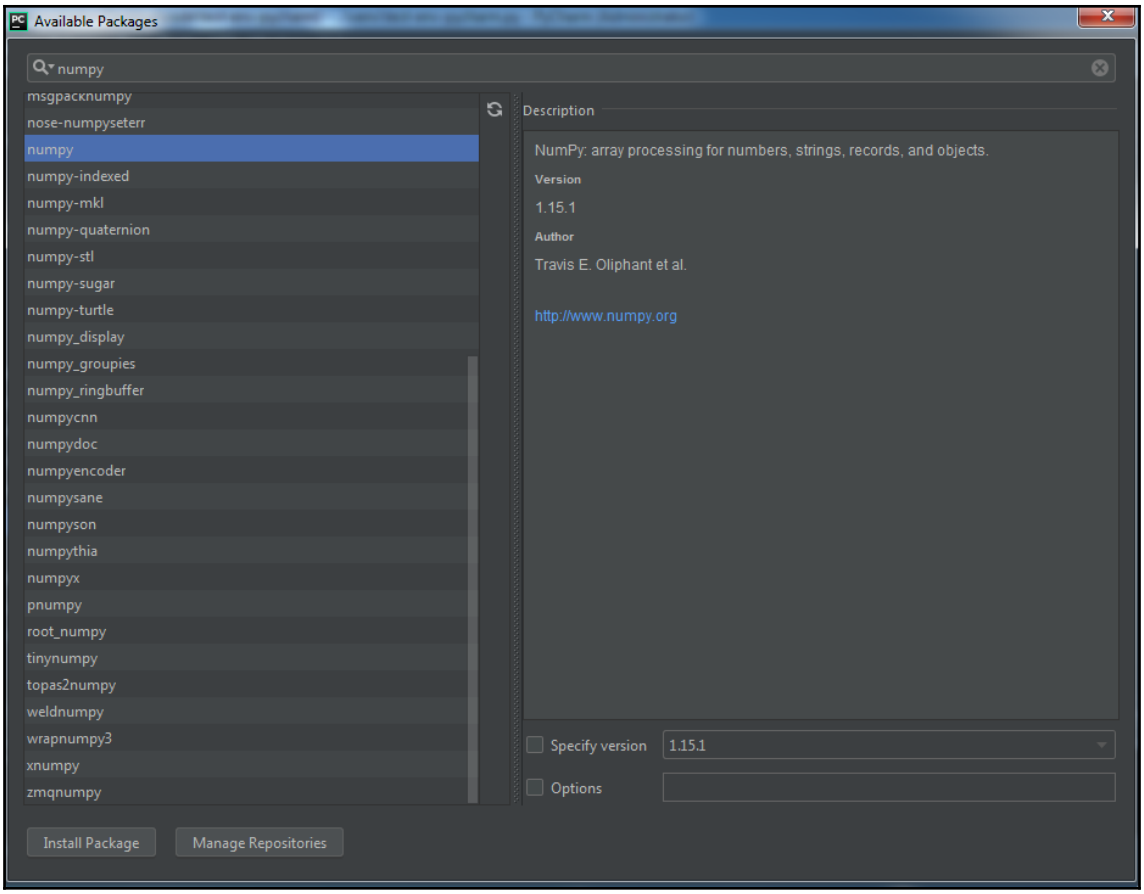


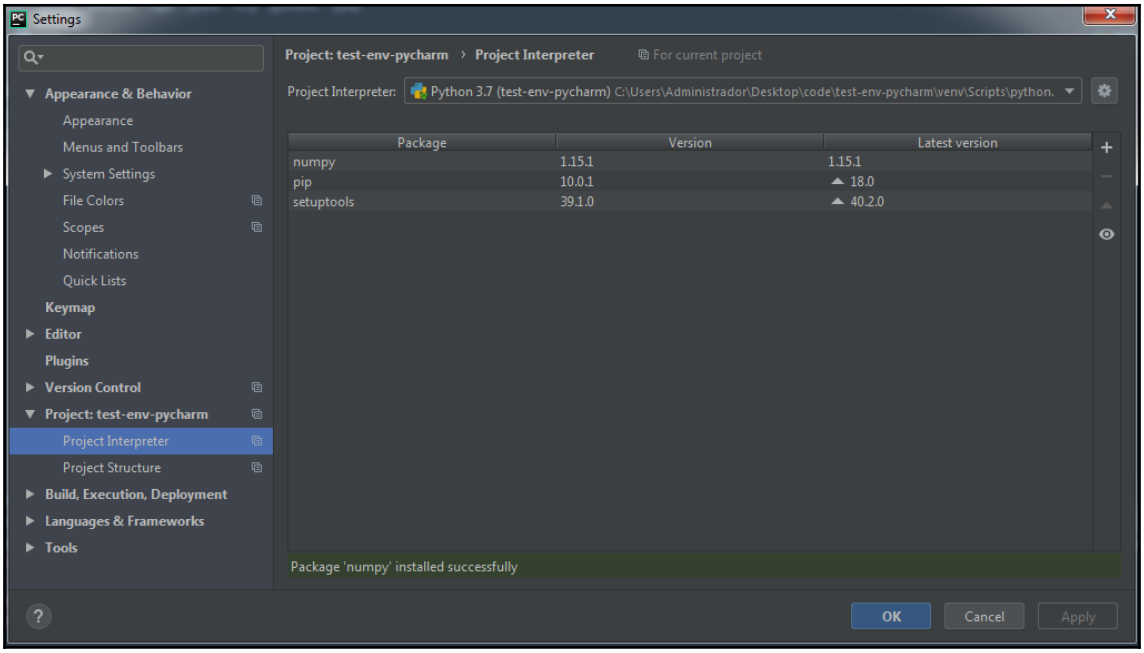

Download PyCharm
Windows
macOS
Linux
Professional
Full-featured IDE
for Python & Web
development
DOWNLOAD
Free trial
Community
Lightweight IDE
for Python & Scientific
development
DOWNLOAD
Free, open-source
Version: 2018.2.3
Build: 182.4323.49
Released: September 6, 2018
System requirements
Installation Instructions
Previous versions

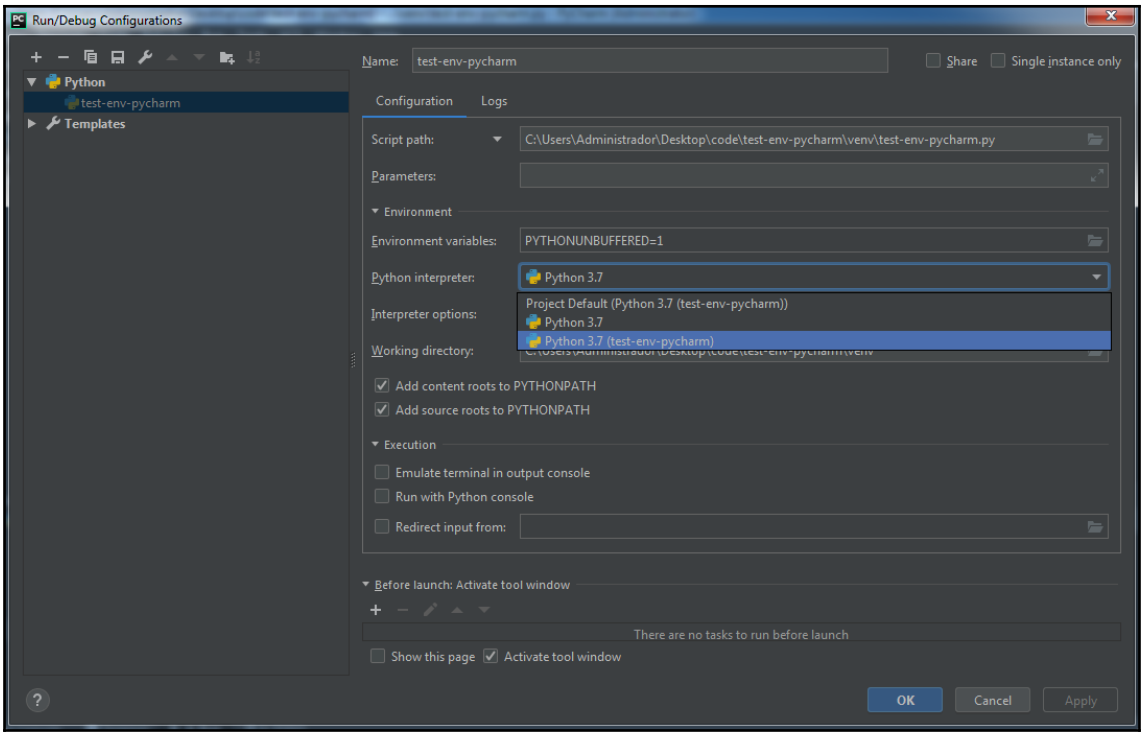

Create Project
Location:
C:\Users\Administrador\Desktop\code\test-env-pycharm
Project Interpreter: New Virtualenv environment
New environment using
Virtualenv
Location:
C:\Users\Administrador\Desktop\code\test-env-pycharm\venv
Base interpreter:
Python 3.7 C:\Python37\python.exe
Inherit global site-packages
Make available to all projects
Existing interpreter
Interpreter:
Python 3.7 C:\Python37\python.exe
Create

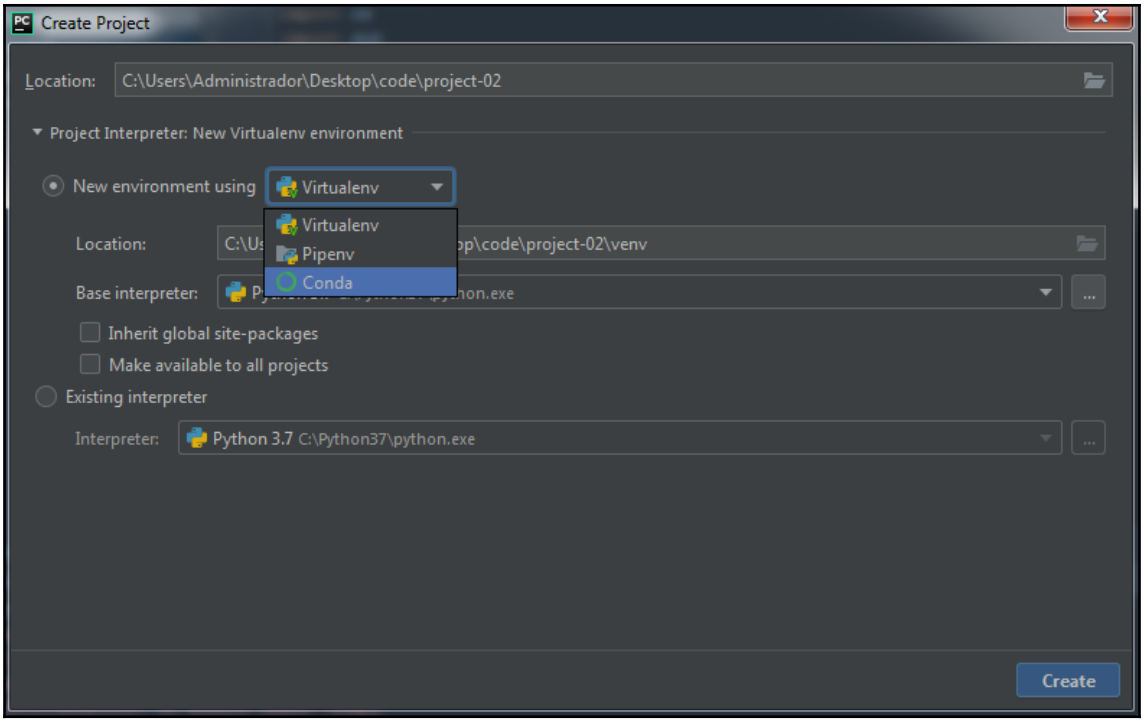

Search Everywhere Double Shift
Project View Alt+1
Creating Virtual Environment
Navigation Bar Alt+Inicio
Drop files here to open

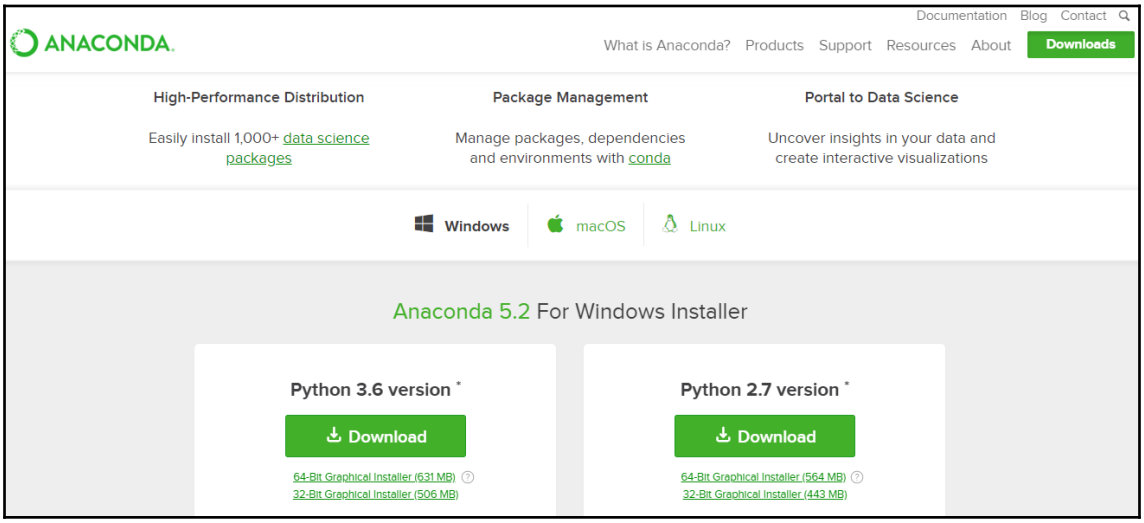

Settings
Project: test-env-pycharm › Project Interpreter
For current project
Project Interpreter: Python 3.7 (test-env-pycharm) C:\Users\Administrador\Desktop\code\test-env-pycharm\venv\Scripts\python.
Package
Version
Latest version
pip
10.0.1
18.0
setuptools
39.1.0
40.2.0
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Keymap
Editor
Plugins
Version Control
Project: test-env-pycharm
Project Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
OK
Cancel
Apply

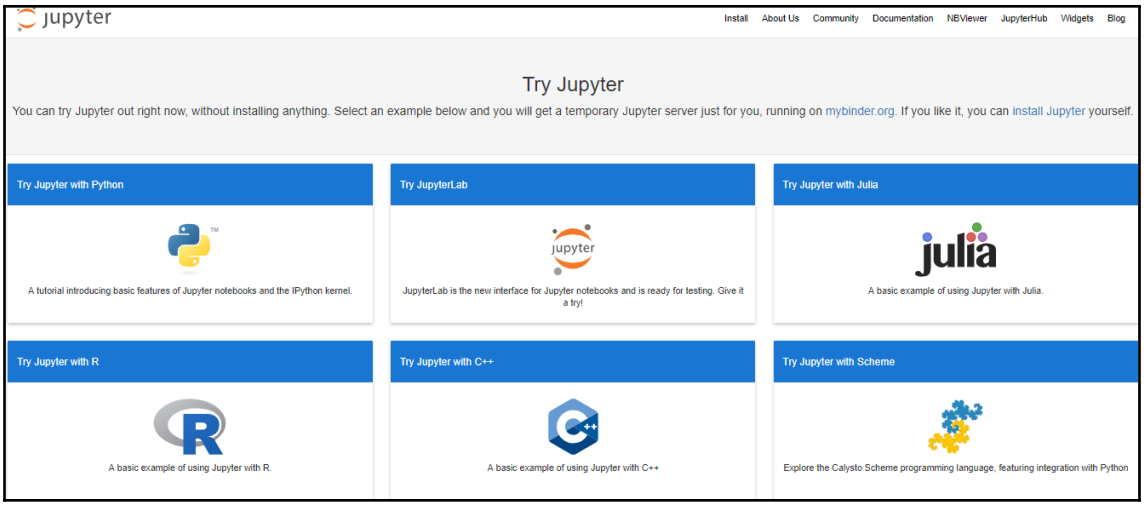

Available Packages
numpy
msgpacknumpy
nose-numpyseterr
numpy
numpy-indexed
numpy-mkl
numpy-quaternion
numpy-stl
numpy-sugar
numpy-turtle
numpy_display
numpy_groupies
numpy_ringbuffer
numpycnn
numpydoc
numpyencoder
numpysane
numpyson
numpythia
numpyx
pnumpy
root_numpy
tinynumpy
topas2numpy
weldnumpy
wrapnumpy3
xnumpy
zmqnumpy
Description
NumPy: array processing for numbers, strings, records, and objects.
Version
1.15.1
Author
Travis E. Oliphant et al.
http://www.numpy.org
Specify version
1.15.1
Options
Install Package
Manage Repositories


Settings
Project: test-env-pycharm › Project Interpreter
For current project
Project Interpreter: Python 3.7 (test-env-pycharm) C:\Users\Administrador\Desktop\code\test-env-pycharm\venv\Scripts\python.
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Keymap
Editor
Plugins
Version Control
Project: test-env-pycharm
Project Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools
Package
Version
Latest version
numpy
1.15.1
1.15.1
pip
10.0.1
18.0
setuptools
39.1.0
40.2.0
Package 'numpy' installed successfully
OK
Cancel
Apply

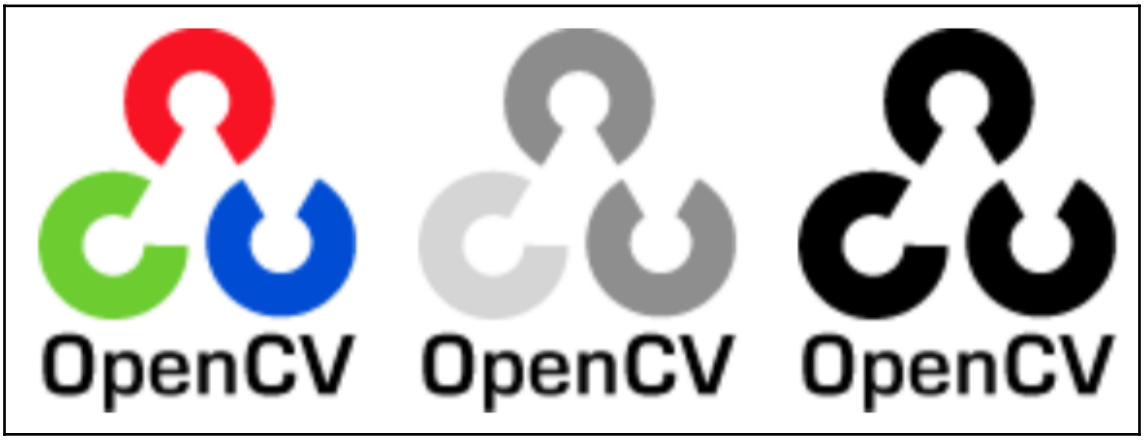

Run/Debug Configurations
Python
test-env-pycharm
Templates
Name: test-env-pycharm
Share
Single instance only
Configuration
Logs
Script path:
C:\Users\Administrador\Desktop\code\test-env-pycharm\venv\test-env-pycharm.py
Parameters:
Environment
Environment variables:
PYTHONUNBUFFERED=1
Python interpreter:
Python 3.7
Project Default (Python 3.7 (test-env-pycharm))
Python 3.7
Python 3.7 (test-env-pycharm)
Interpreter options:
Working directory:
Add content roots to PYTHONPATH
Add source roots to PYTHONPATH
Execution
Emulate terminal in output console
Run with Python console
Redirect input from:
Before launch: Activate tool window
There are no tasks to run before launch
Show this page
Activate tool window
OK
Cancel
Apply

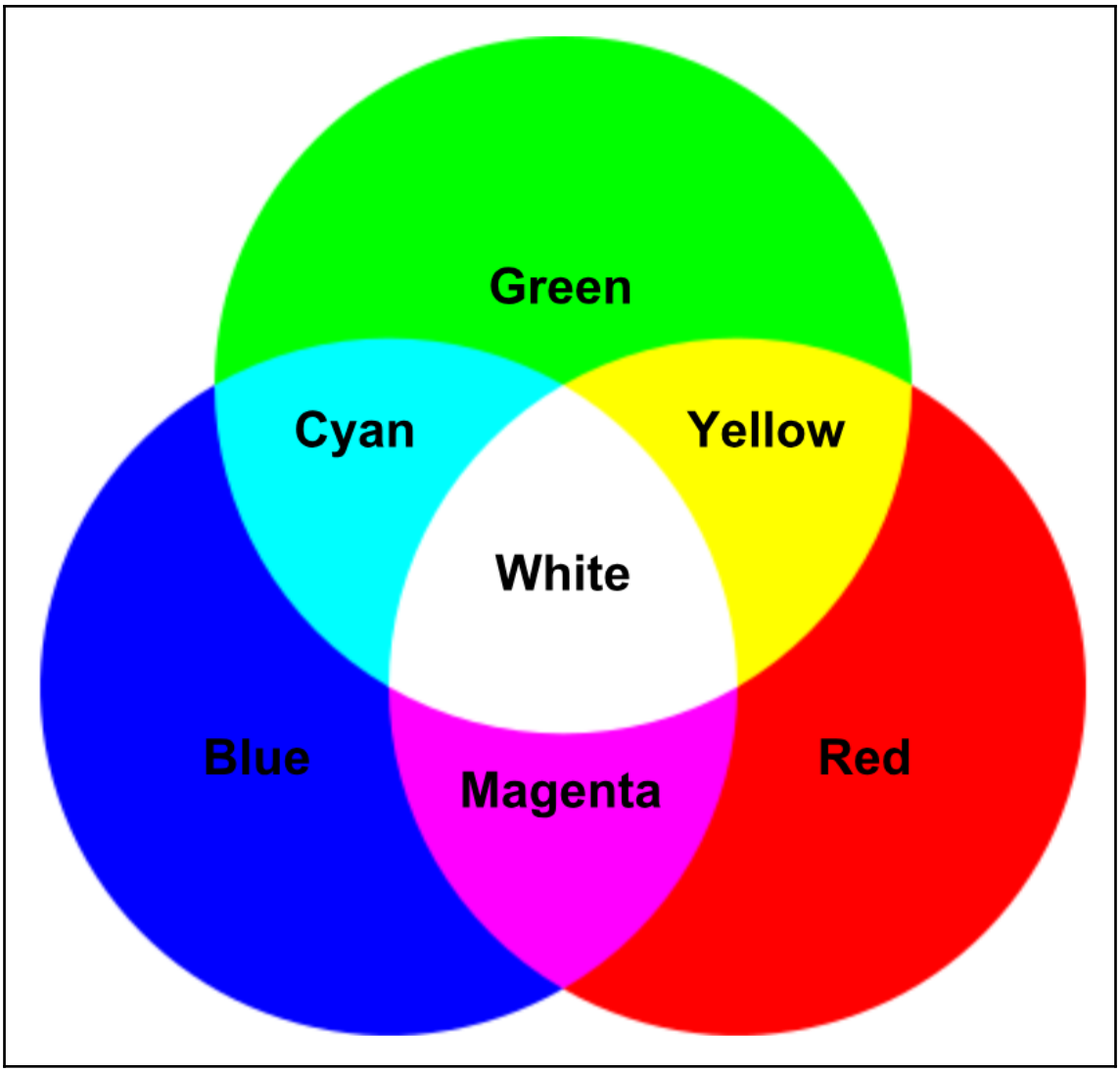

Create Project
Location:
C:\Users\Administrador\Desktop\code\project-02
Project Interpreter: New Virtualenv environment
New environment using
Virtualenv
Virtualenv
Pipenv
Conda
Location:
p\code\project-02\venv
Base interpreter:
Inherit global site-packages
Make available to all projects
Existing interpreter
Interpreter:
Python 3.7 C:\Python37\python.exe
Create

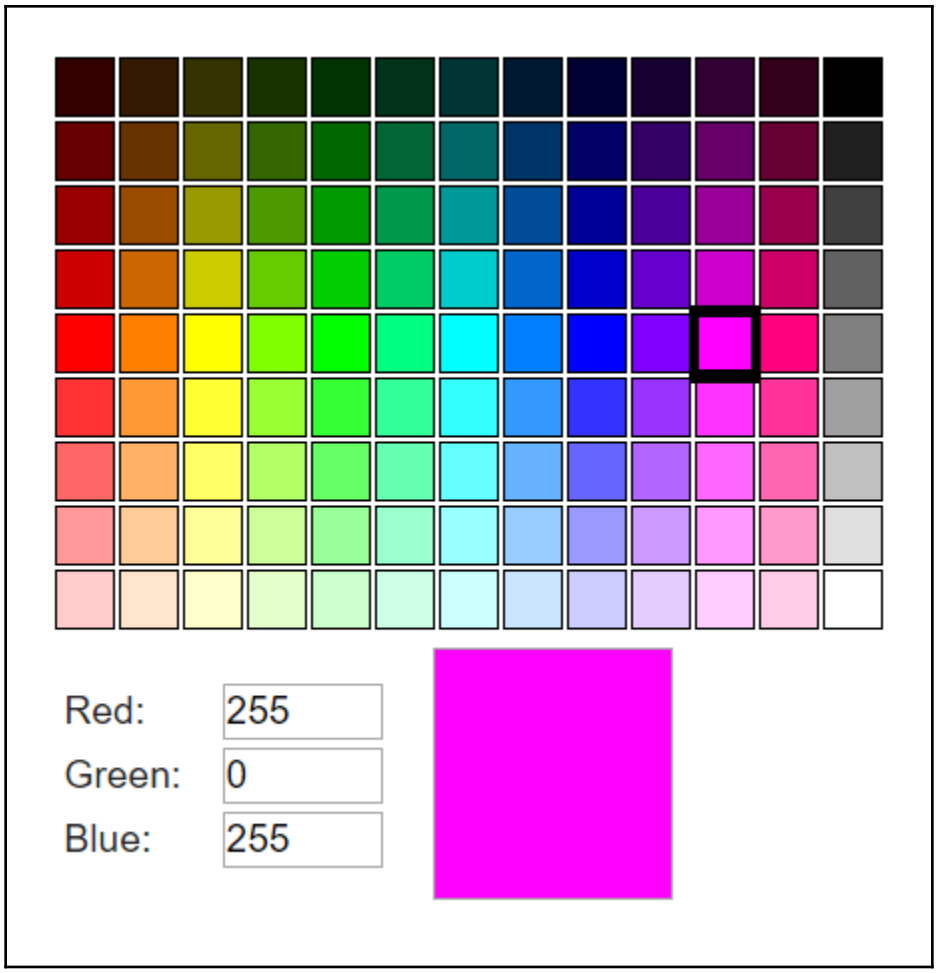

ANACONDA.
Documentation
Blog
Contact
What is Anaconda?
Products
Support
Resources
About
Downloads
High-Performance Distribution
Easily install 1,000+ data science packages
Package Management
Manage packages, dependencies and environments with conda
Portal to Data Science
Uncover insights in your data and create interactive visualizations
Windows
macOS
Linux
Anaconda 5.2 For Windows Installer
Python 3.6 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (506 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (564 MB)
32-Bit Graphical Installer (443 MB)

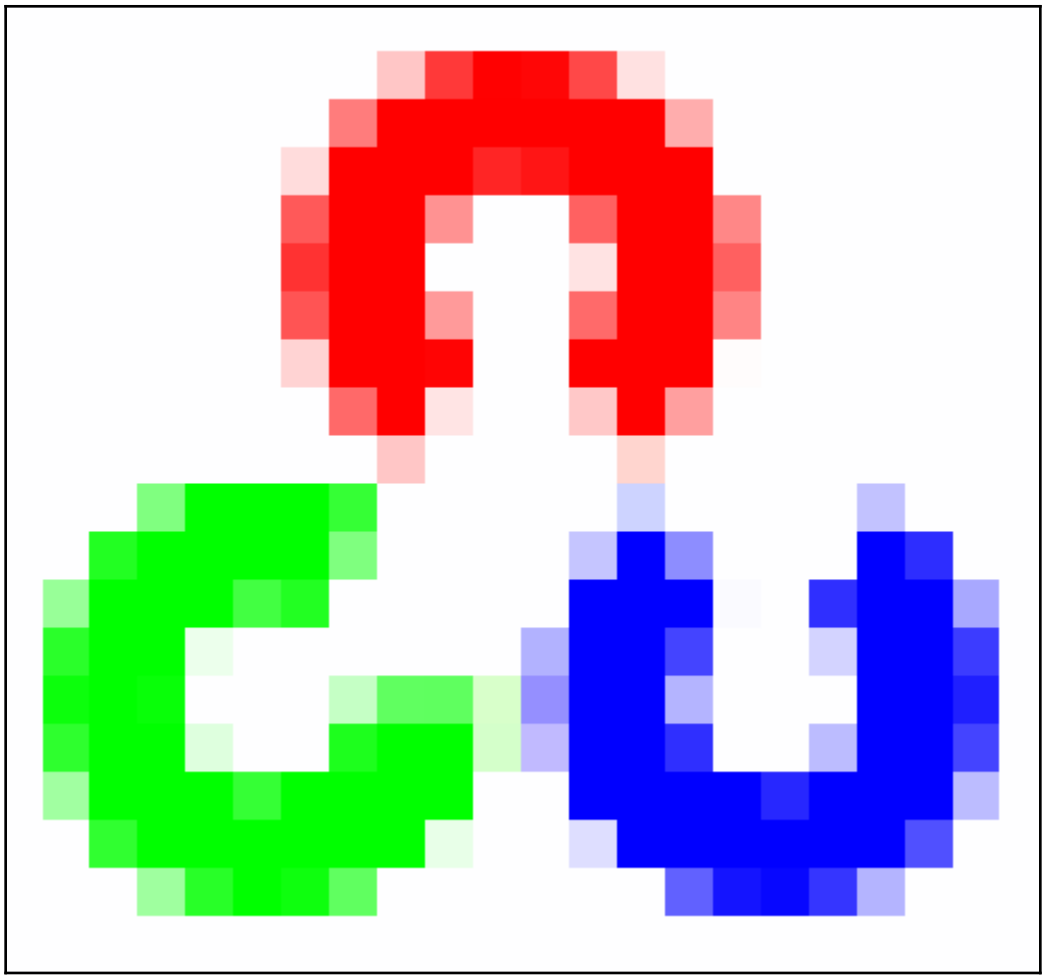

jupyter
Install
About Us
Community
Documentation
NBViewer
JupyterHub
Widgets
Blog
Try Jupyter
You can try Jupyter out right now, without installing anything. Select an example below and you will get a temporary Jupyter server just for you, running on mybinder.org. If you like it, you can install Jupyter yourself.
Try Jupyter with Python
A tutorial introducing basic features of Jupyter notebooks and the IPython kernel.
Try JupyterLab
JupyterLab is the new interface for Jupyter notebooks and is ready for testing. Give it a try!
Try Jupyter with Julia
A basic example of using Jupyter with Julia.
Try Jupyter with R
A basic example of using Jupyter with R.
Try Jupyter with C++
A basic example of using Jupyter with C++
Try Jupyter with Scheme
Explore the Calysto Scheme programming language, featuring integration with Python

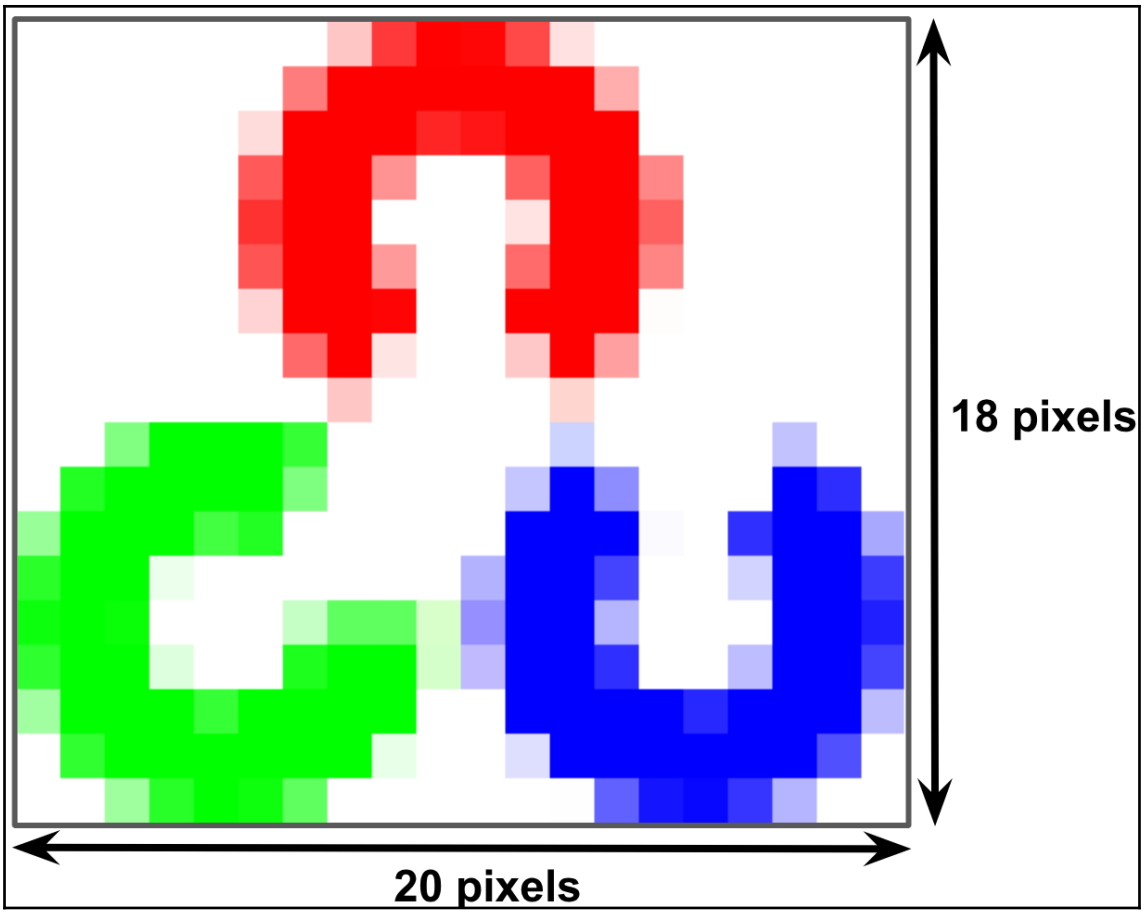
Chapter 2: Image Basics in OpenCV

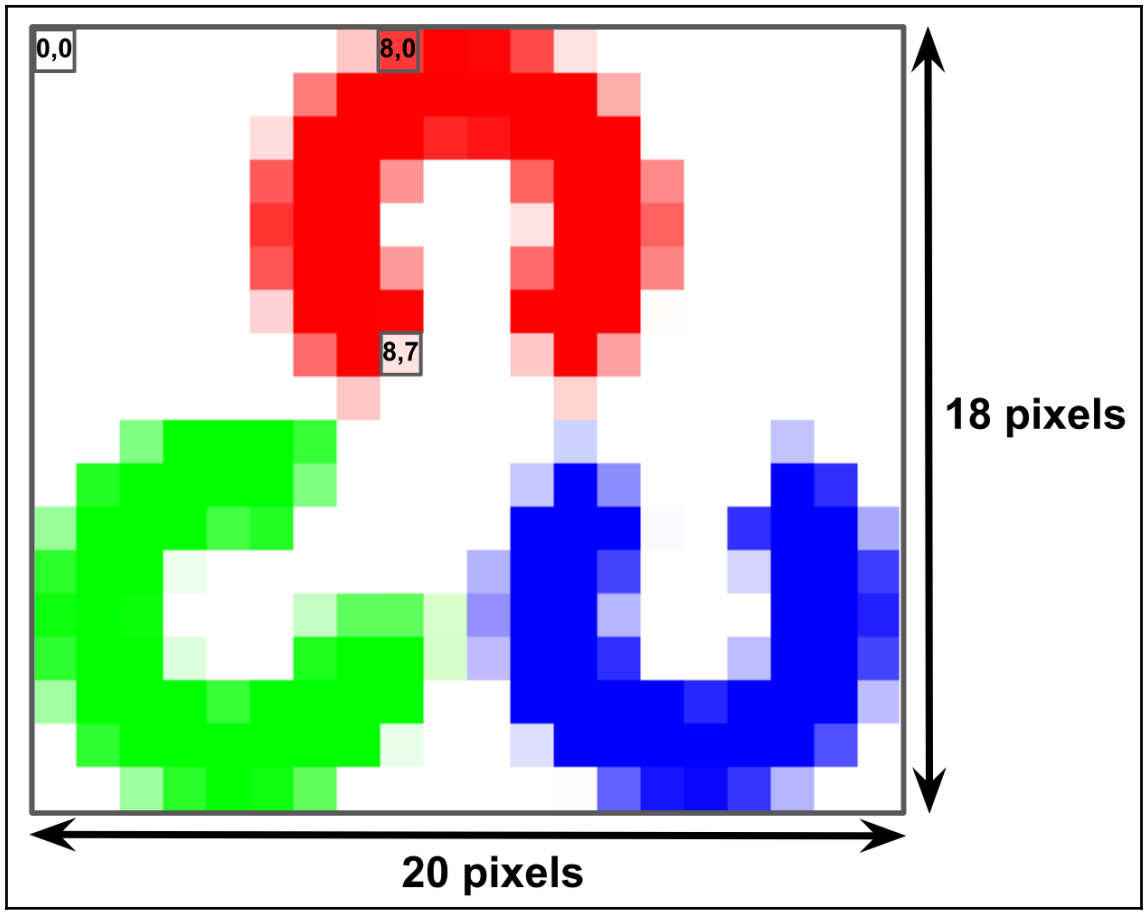


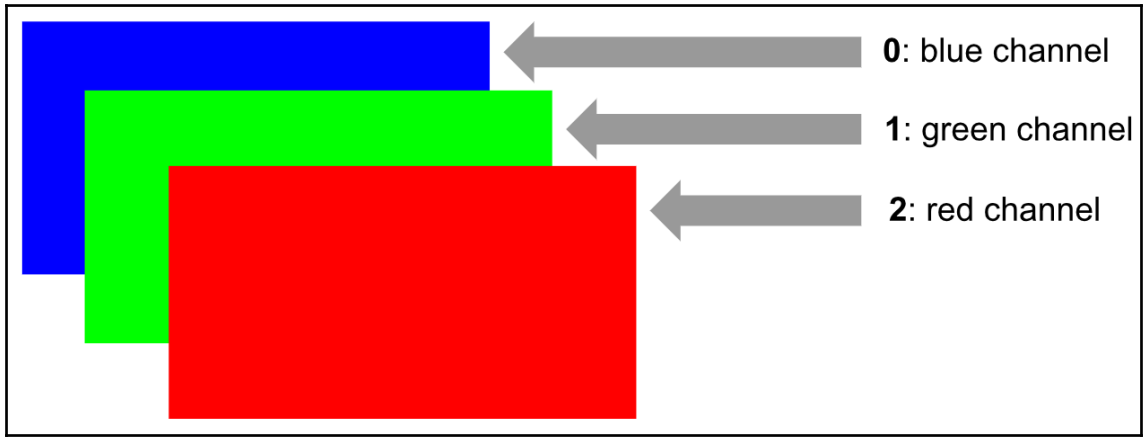

Green
Cyan
Yellow
White
Blue
Magenta
Red

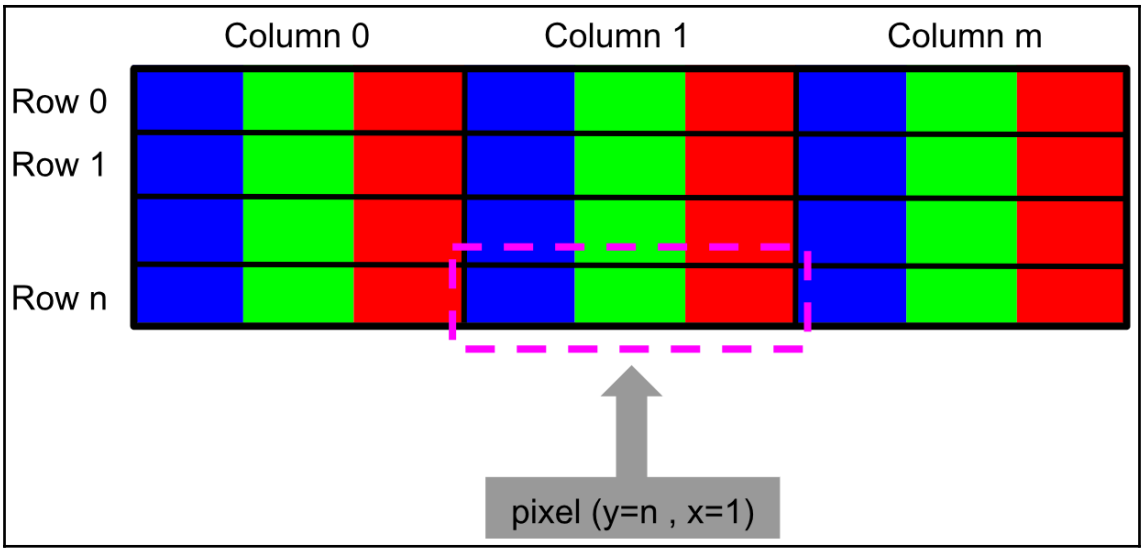

Red:
255
Green:
0
Blue:
255




18 pixels
20 pixels


0,0
8,0
8,7
18 pixels
20 pixels


0: blue channel
1: green channel
2: red channel


Column 0
Column 1
Column m
Row 0
Row 1
Row n
pixel (y=n , x=1)

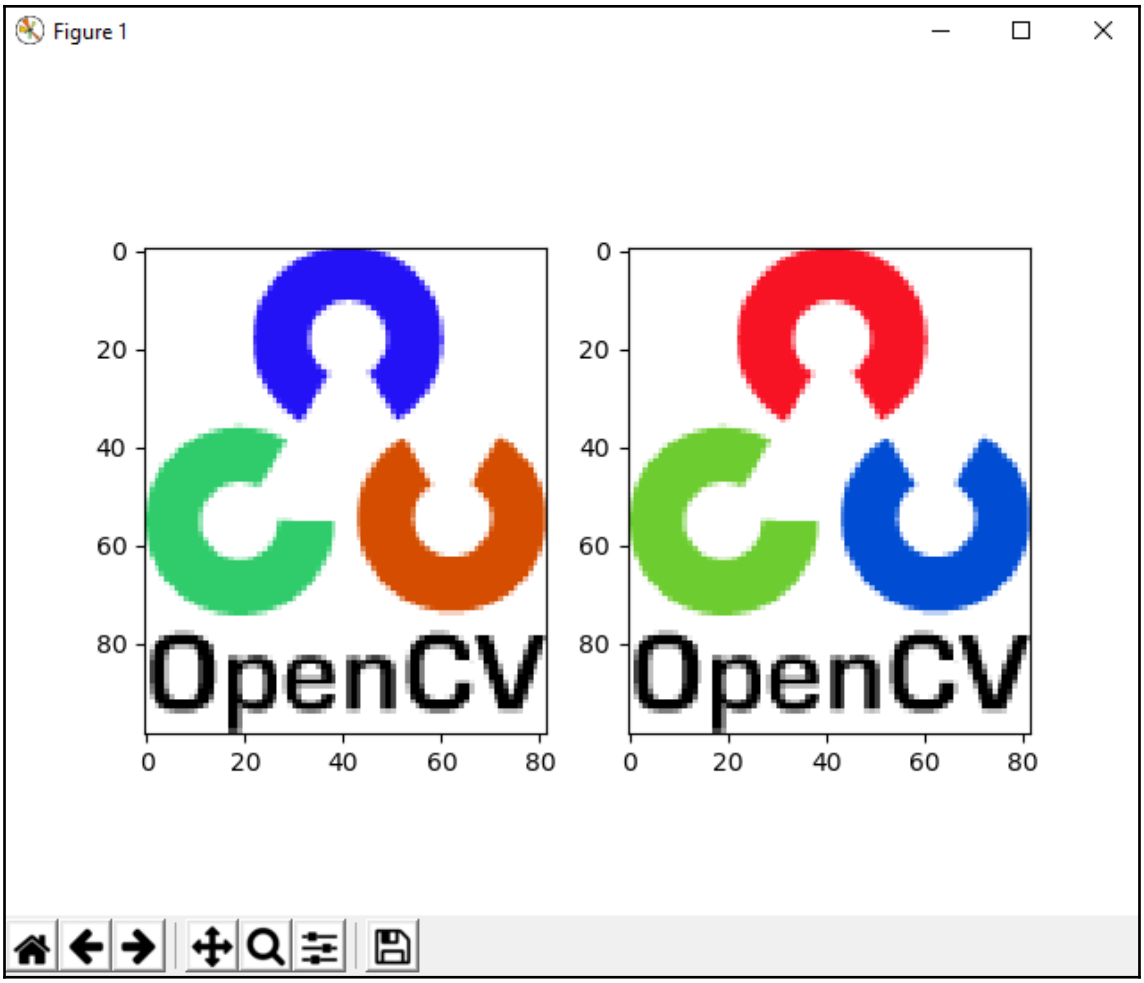
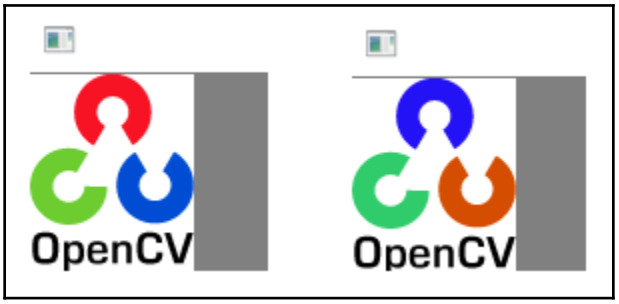
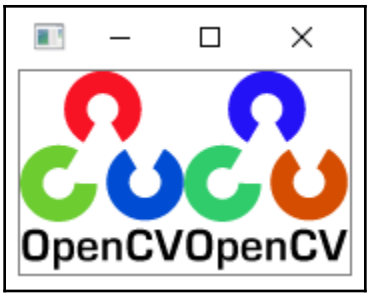
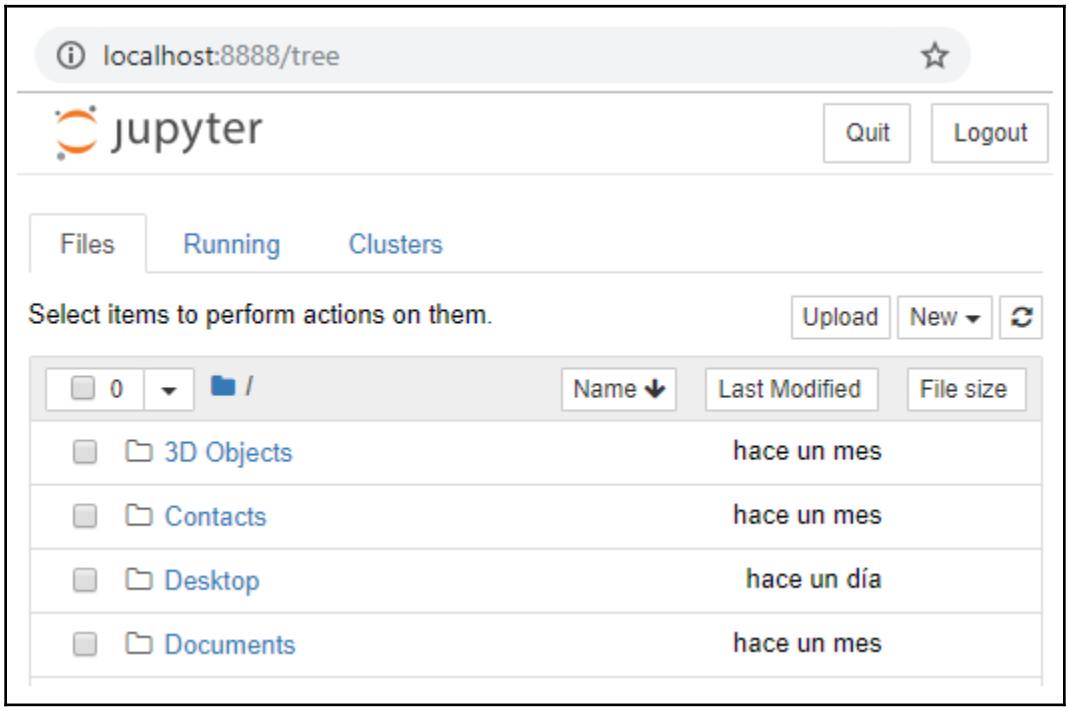
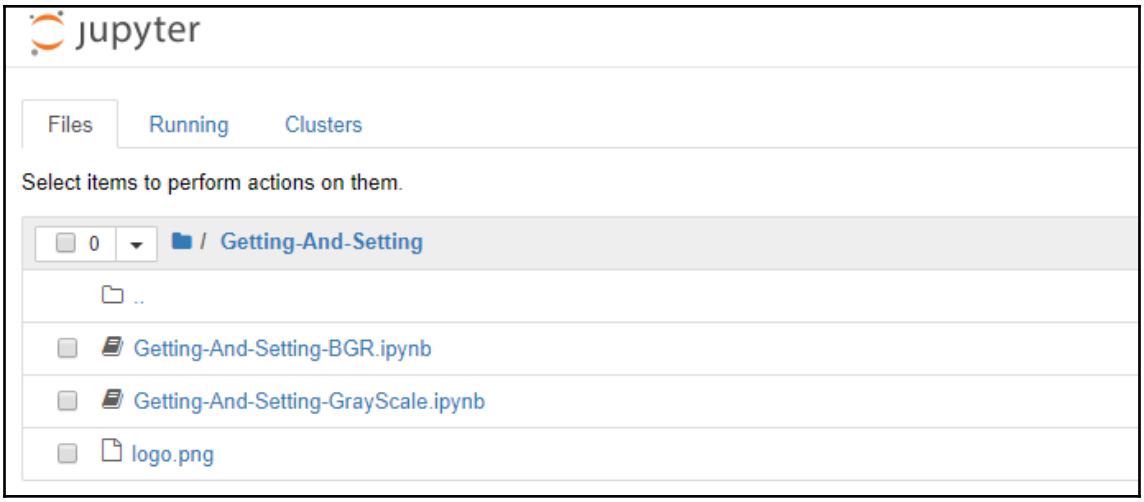


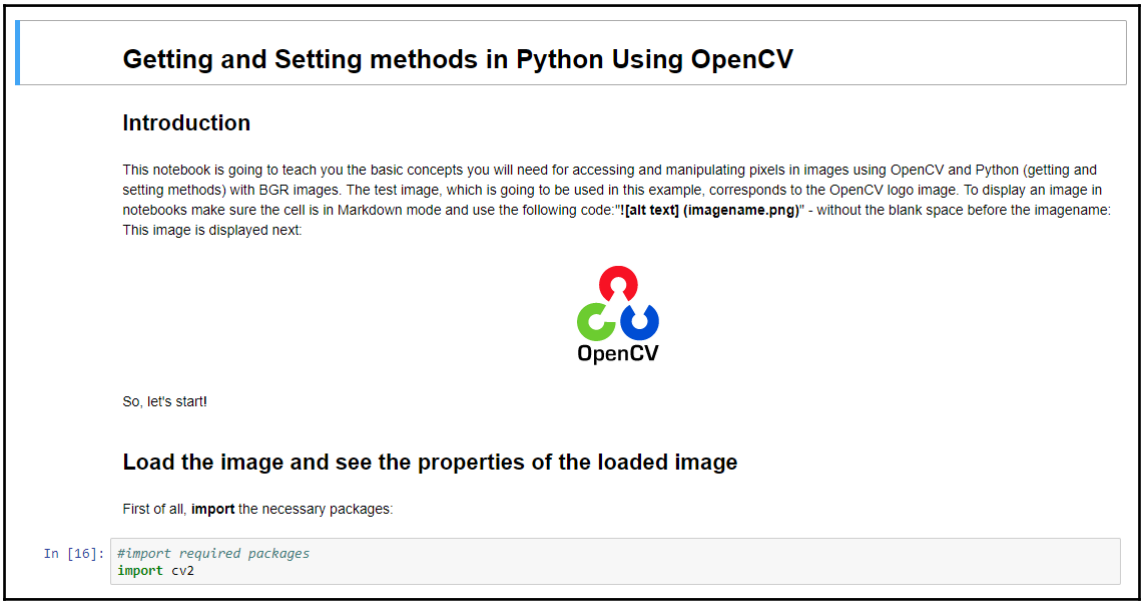
Figure 1
0
20
40
60
80
OpenCV
0
20
40
60
80
0
20
40
60
80
OpenCV
0
20
40
60
80


OpenCV
OpenCV

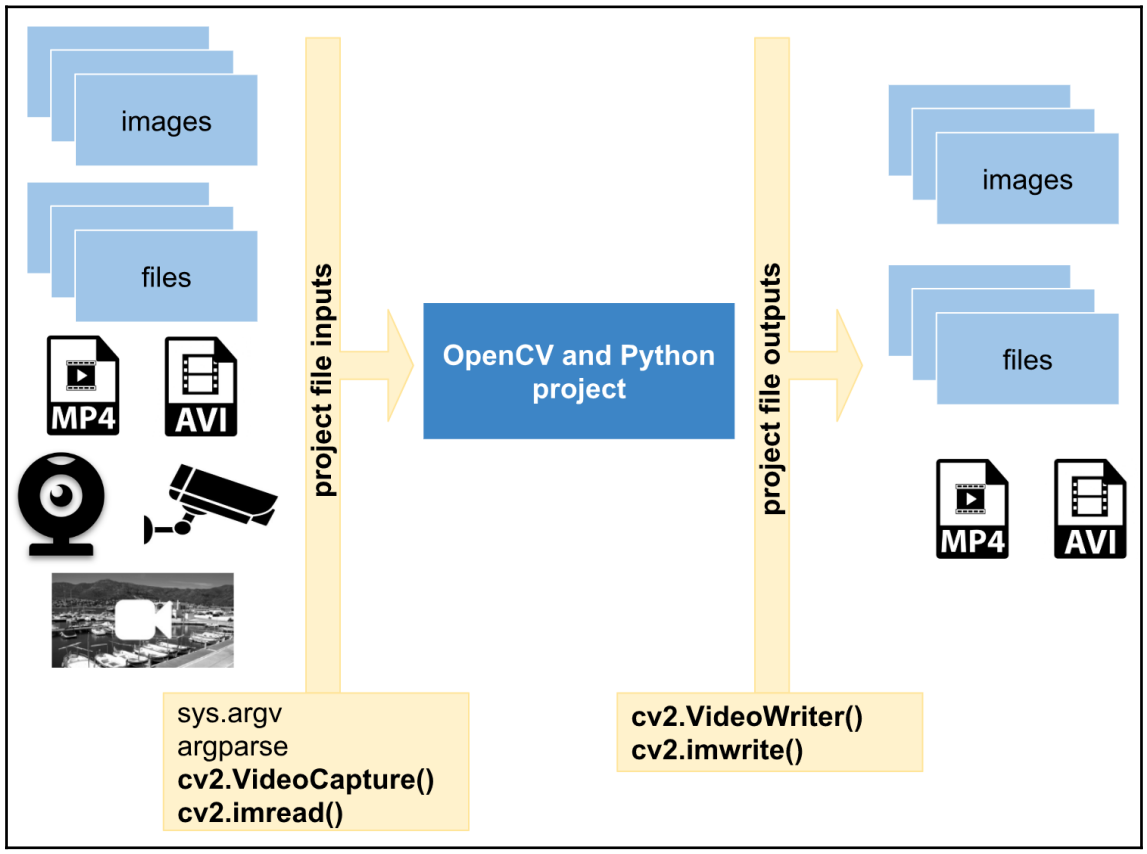

OpenCVOpenCV

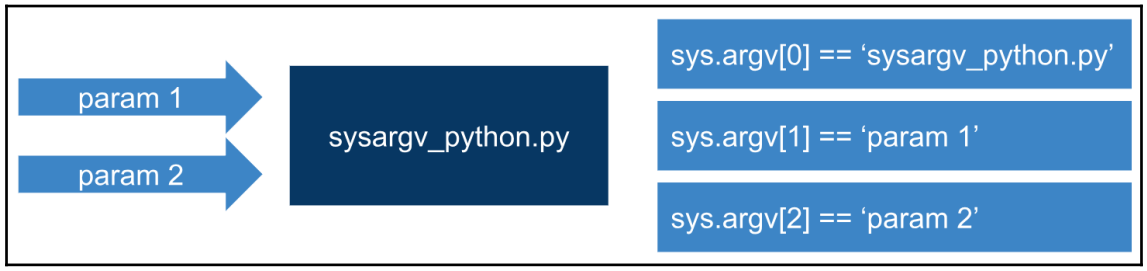

localhost:8888/tree
jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/
Name
Last Modified
File size
3D Objects
hace un mes
Contacts
hace un mes
Desktop
hace un día
Documents
hace un mes

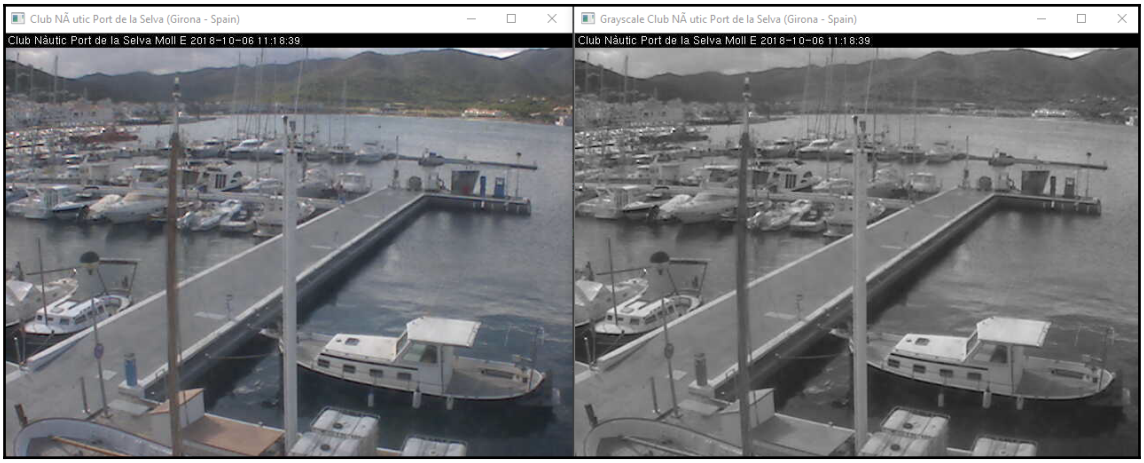

jupyter
Files
Running
Clusters
Select items to perform actions on them.
0
/ Getting-And-Setting
..
Getting-And-Setting-BGR.ipynb
Getting-And-Setting-GrayScale.ipynb
logo.png

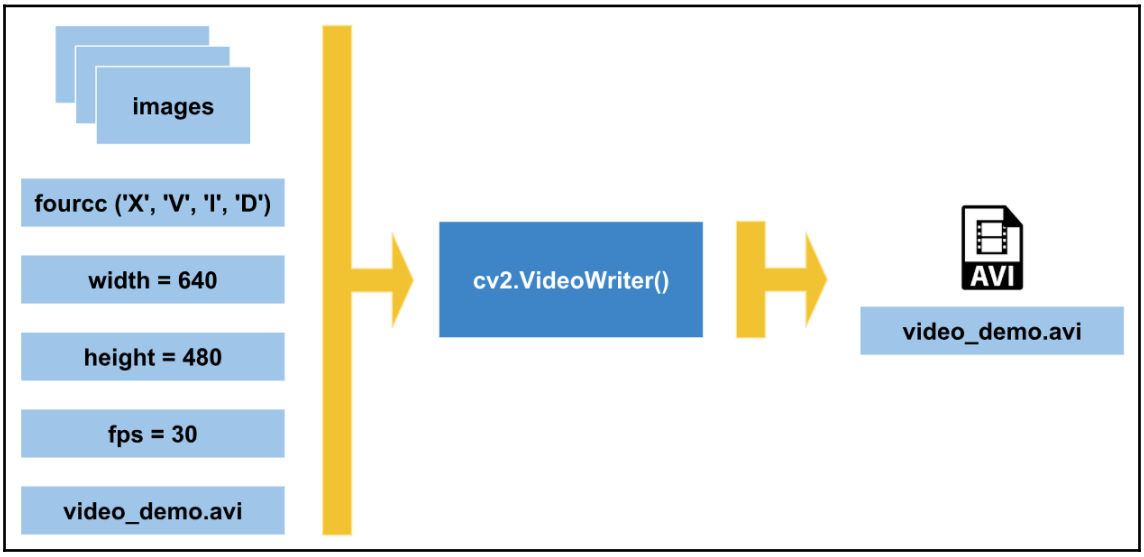

Getting and Setting methods in Python Using OpenCV
Introduction
This notebook is going to teach you the basic concepts you will need for accessing and manipulating pixels in images using OpenCV and Python (getting and setting methods) with BGR images. The test image, which is going to be used in this example, corresponds to the OpenCV logo image. To display an image in notebooks make sure the cell is in Markdown mode and use the following code:"![alt text] (imagename.png)" - without the blank space before the imagename: This image is displayed next:
OpenCV
So, let's start!
Load the image and see the properties of the loaded image
First of all, import the necessary packages:
In [16]:
#import required packages
import cv2

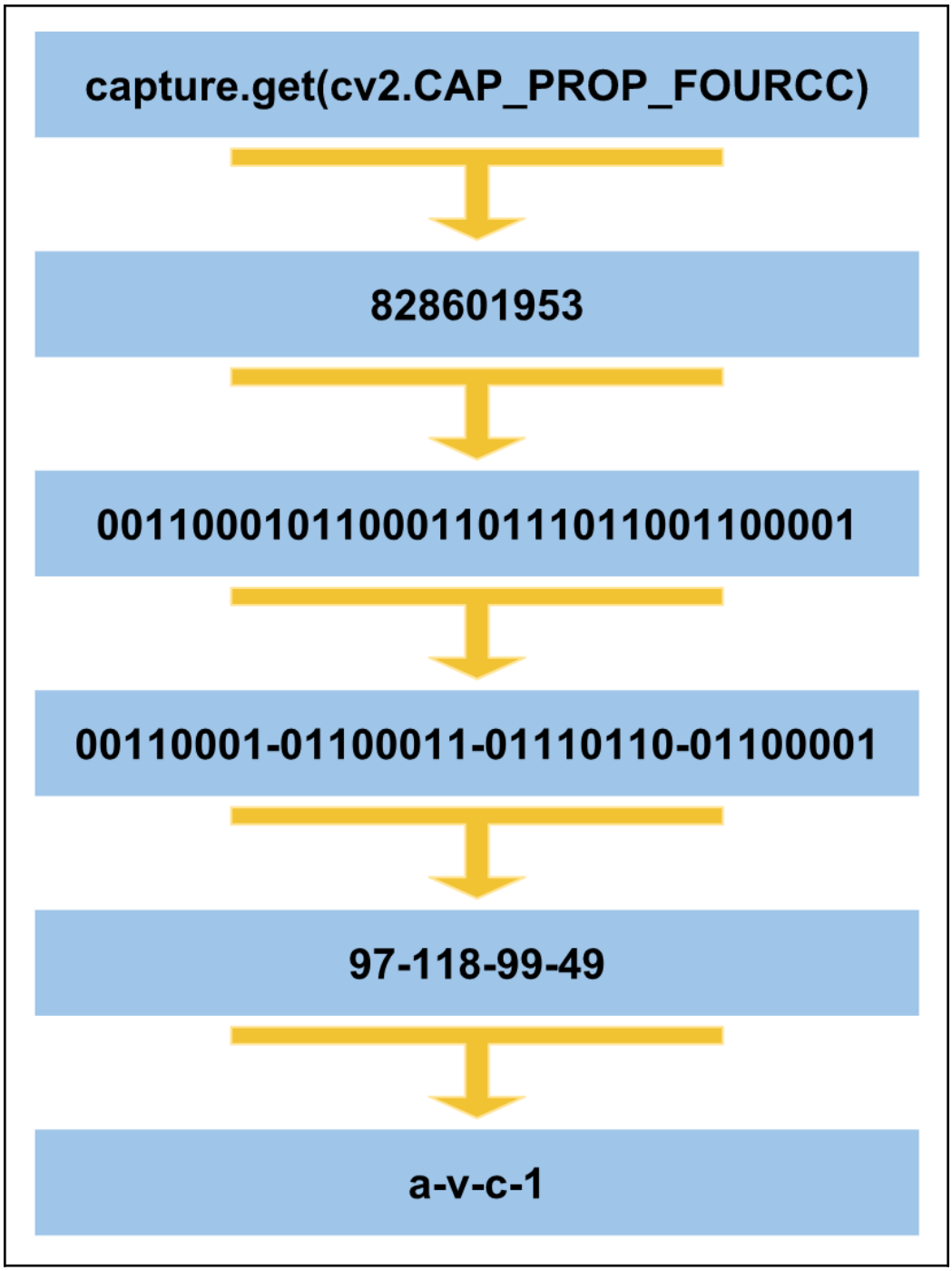
Chapter 3: Handling Files and Images



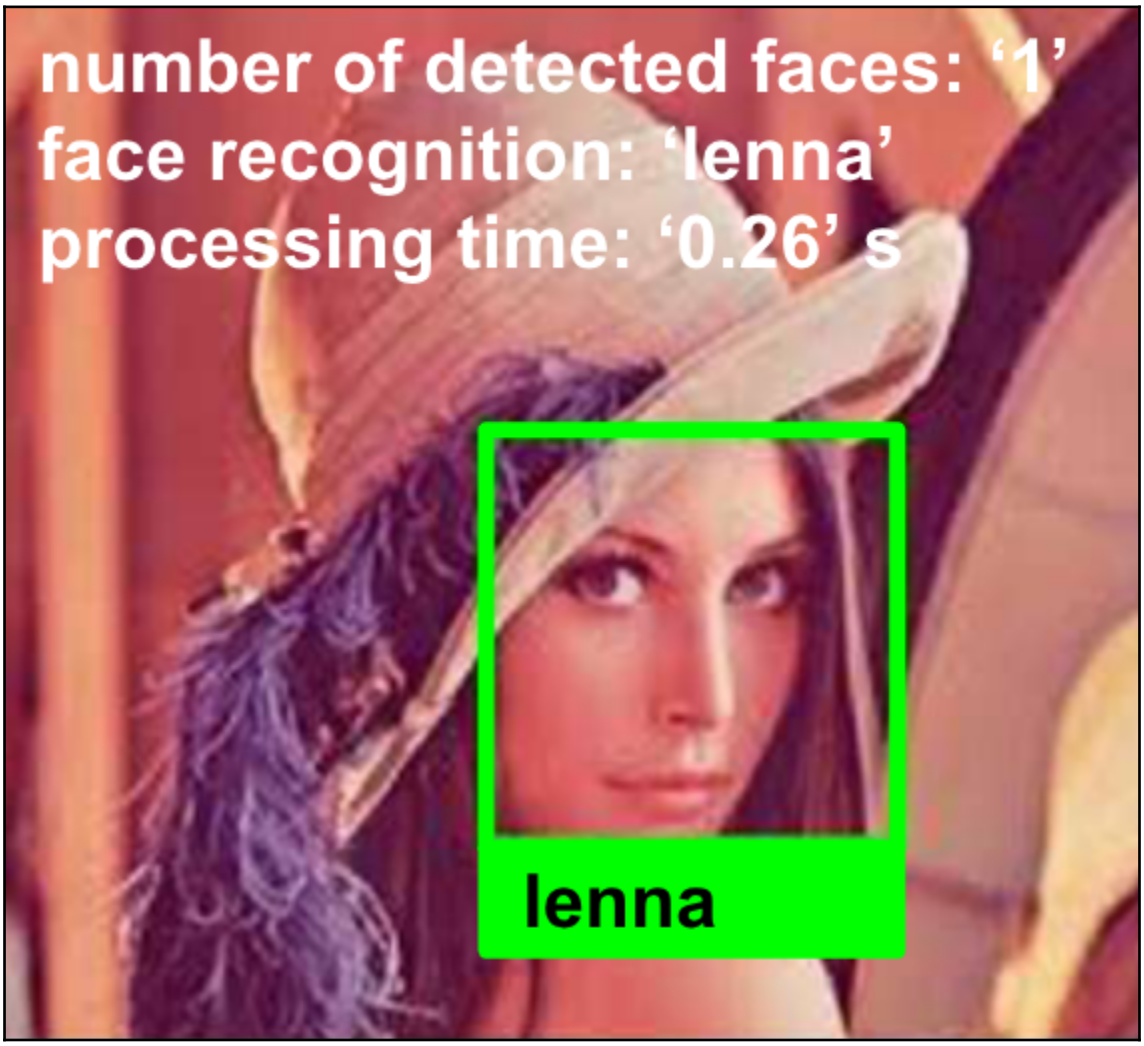


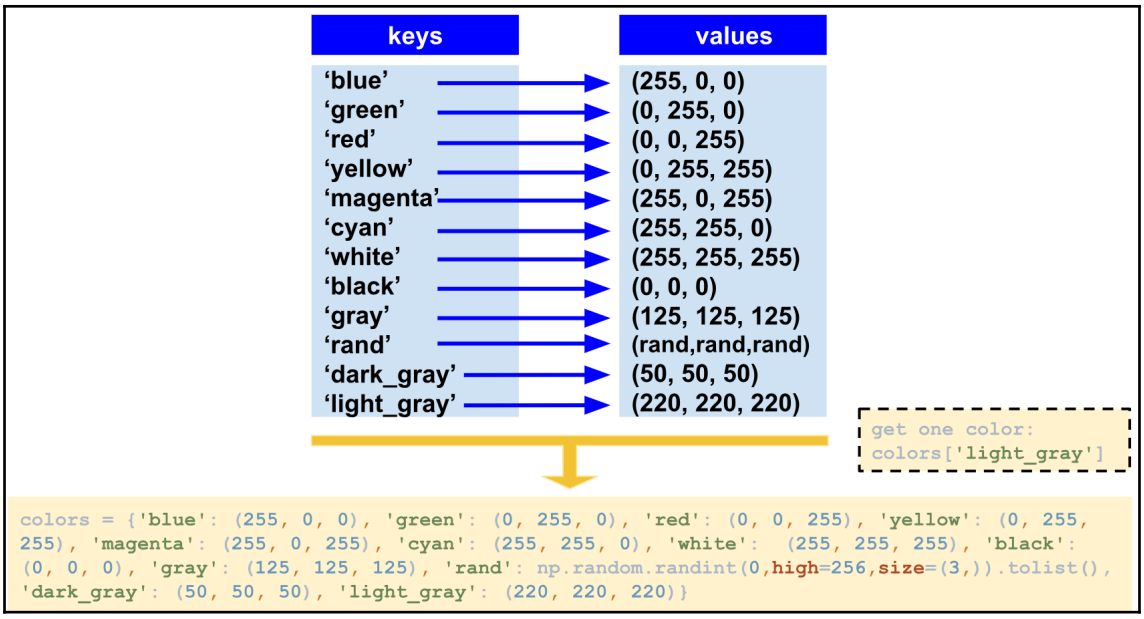

Club NÃ utic Port de la Selva (Girona - Spain)
Club Nàutic Port de la Selva Moll E 2018-10-06 11:18:39
Grayscale Club NÃ utic Port de la Selva (Girona - Spain)
Club Nàutic Port de la Selva Moll E 2018-10-06 11:18:39


images
fourcc ('X', 'V', 'I', 'D')
width = 640
height = 480
fps = 30
video_demo.avi
cv2.VideoWriter()
AVI
video_demo.avi


capture.get(cv2.CAP_PROP_FOURCC)
828601953
00110000101100011011011001100001
00110001-01100011-01110110-01100001
97-118-99-49
a-v-c-1

Chapter 4: Constructing Basic Shapes in OpenCV




keys
values
'blue'
'green'
'red'
'yellow'
'magenta'
'cyan'
'white'
'black'
'gray'
'rand'
'dark_gray'
'light_gray'
(255, 0, 0)
(0, 255, 0)
(0, 0, 255)
(0, 255, 255)
(255, 0, 255)
(255, 255, 0)
(255, 255, 255)
(0, 0, 0)
(125, 125, 125)
(rand,rand,rand)
(50, 50, 50)
(220, 220, 220)
get one color:
colors['light_gray']
colors = {'blue': (255, 0, 0), 'green': (0, 255, 0), 'red': (0, 0, 255), 'yellow': (0, 255, 255), 'magenta': (255, 0, 255), 'cyan': (255, 255, 0), 'white': (255, 255, 255), 'black': (0, 0, 0), 'gray': (125, 125, 125), 'rand': np.random.randint(0,high=256,size=(3,)).tolist(), 'dark_gray': (50, 50, 50), 'light_gray': (220, 220, 220)}

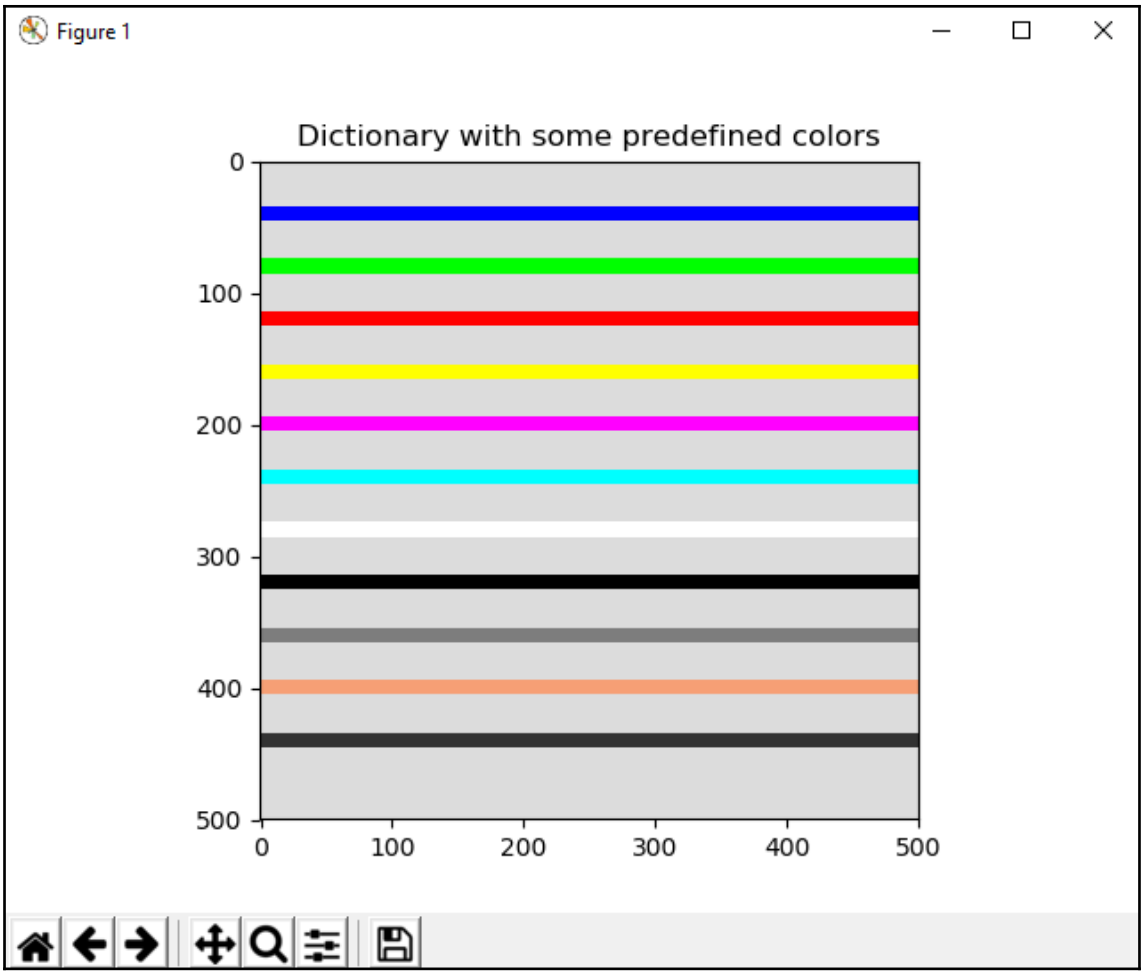


Figure 1
Dictionary with some predefined colors
0
100
200
300
400
500
0
100
200
300
400
500

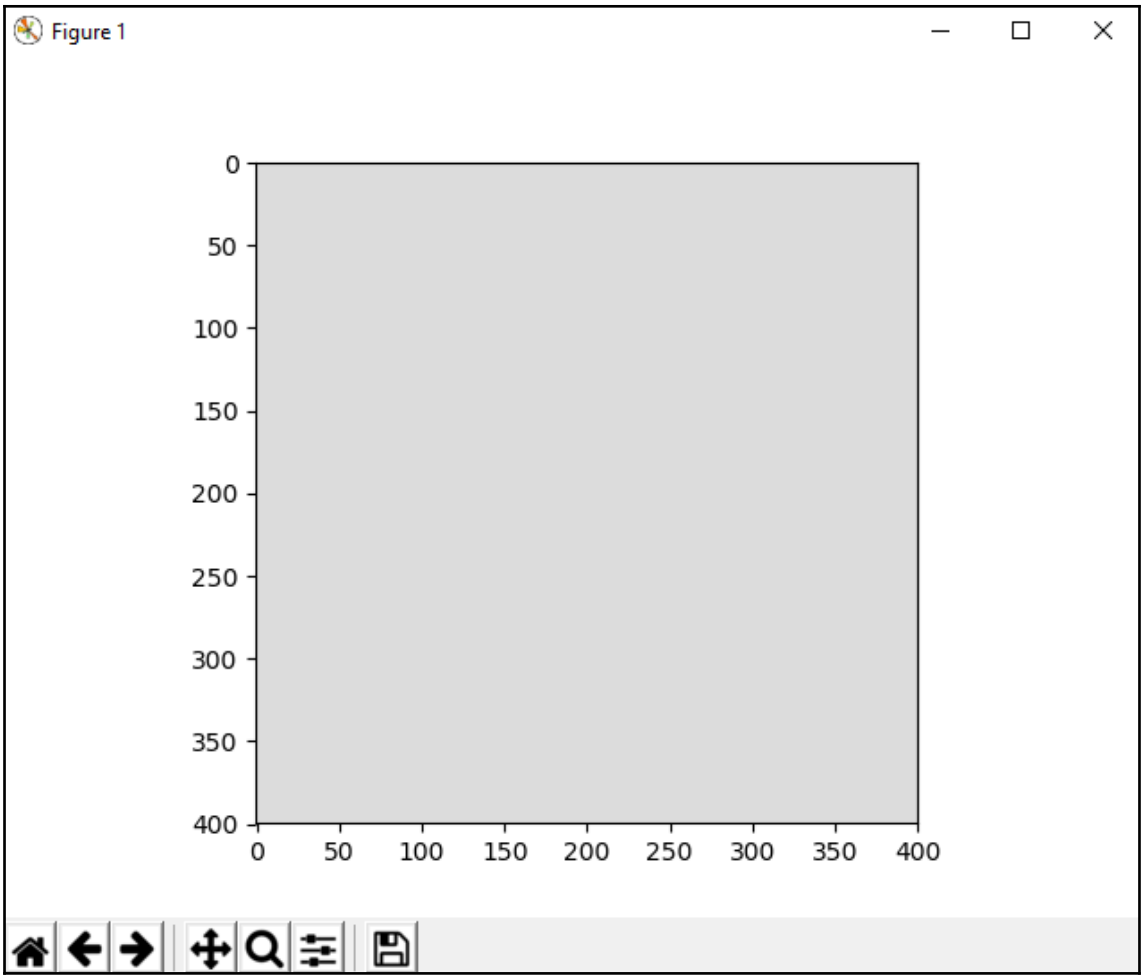


Figure 1
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300
350
400

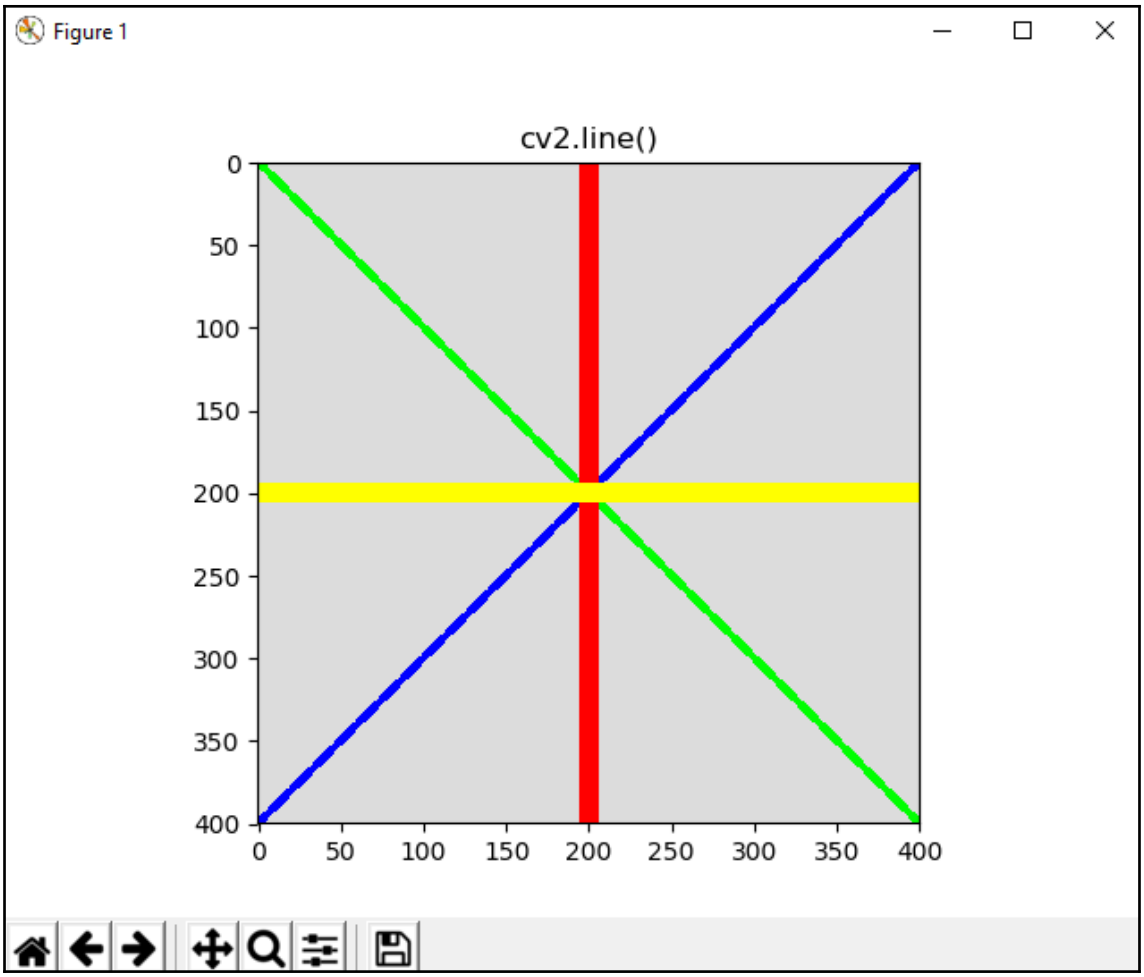


Figure 1
cv2.line()
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300
350
400

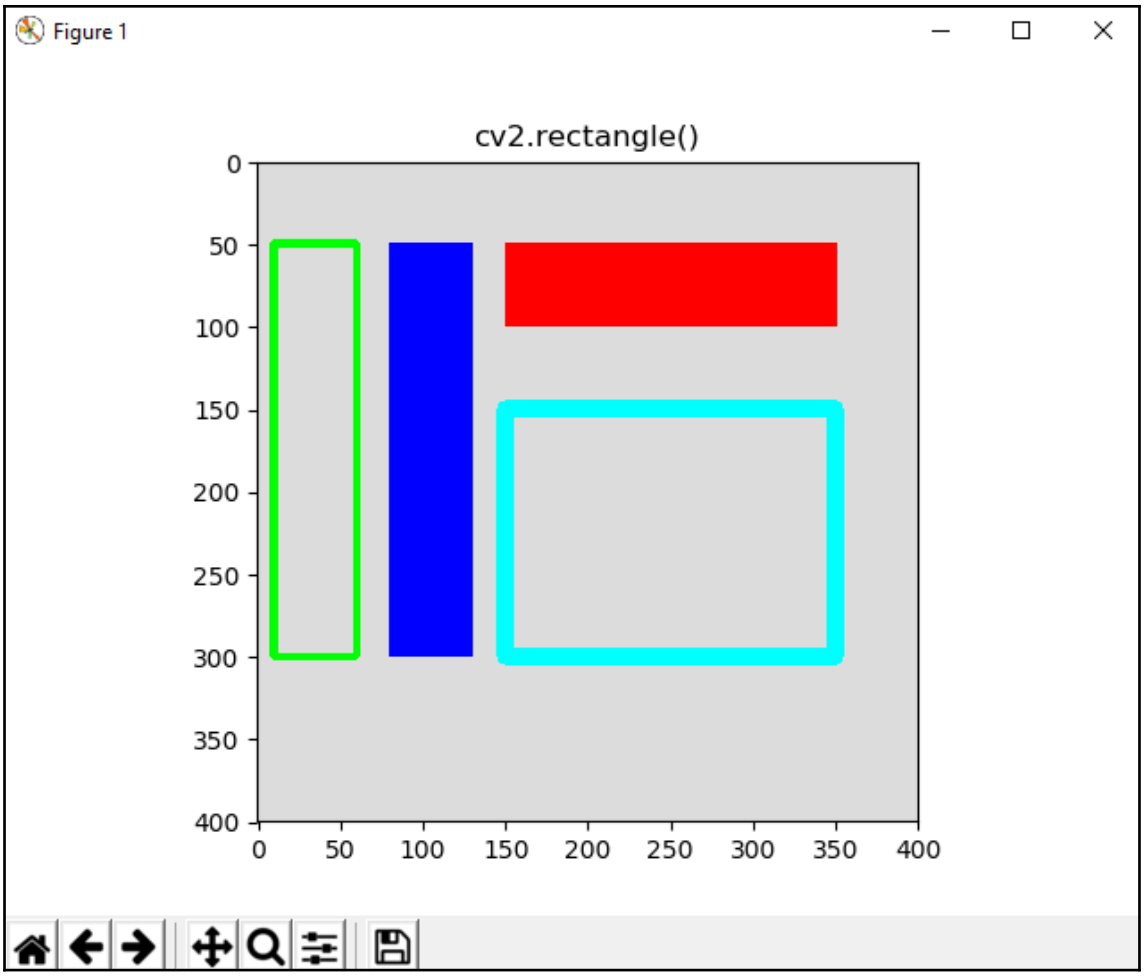


Figure 1
cv2.rectangle()
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300
350
400

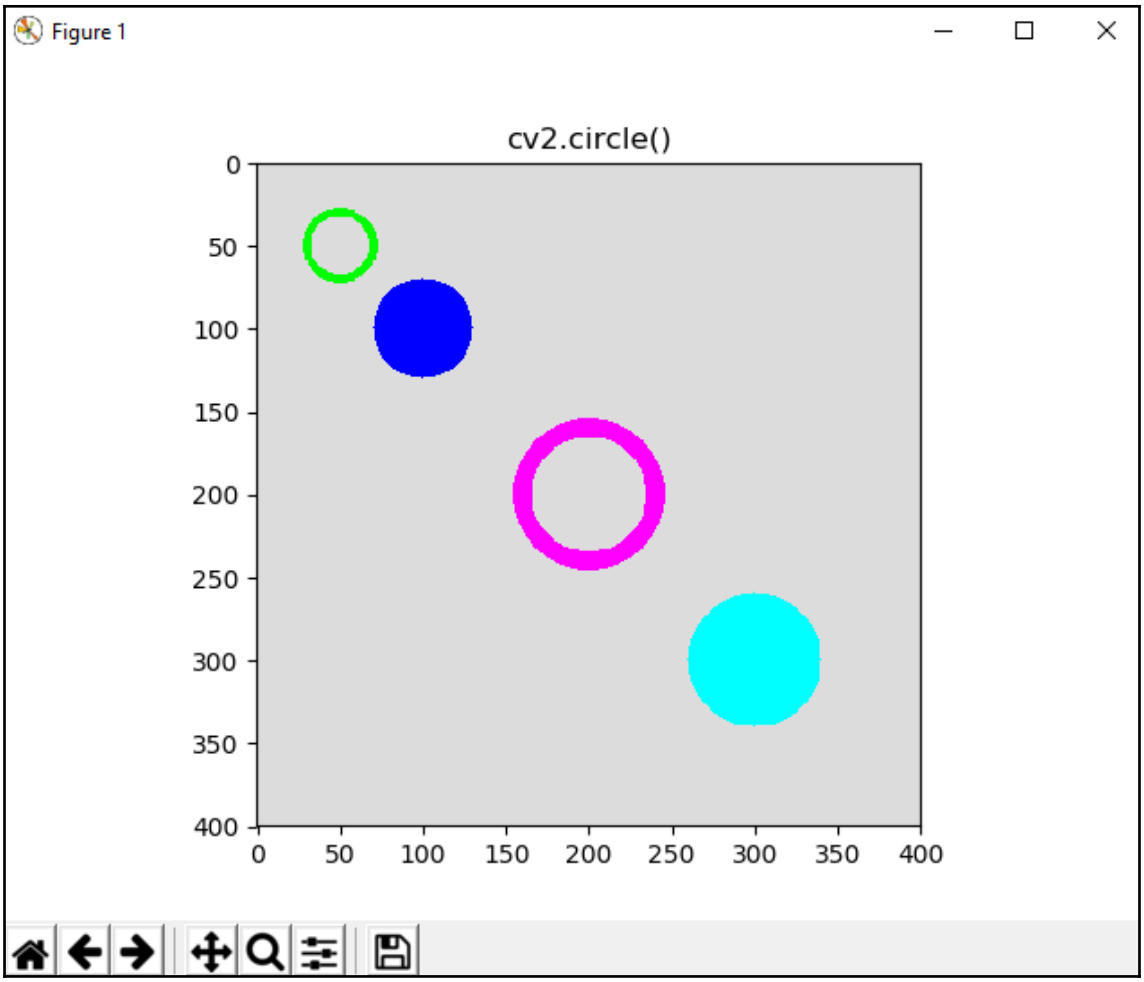


Figure 1
cv2.circle()
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300
350
400

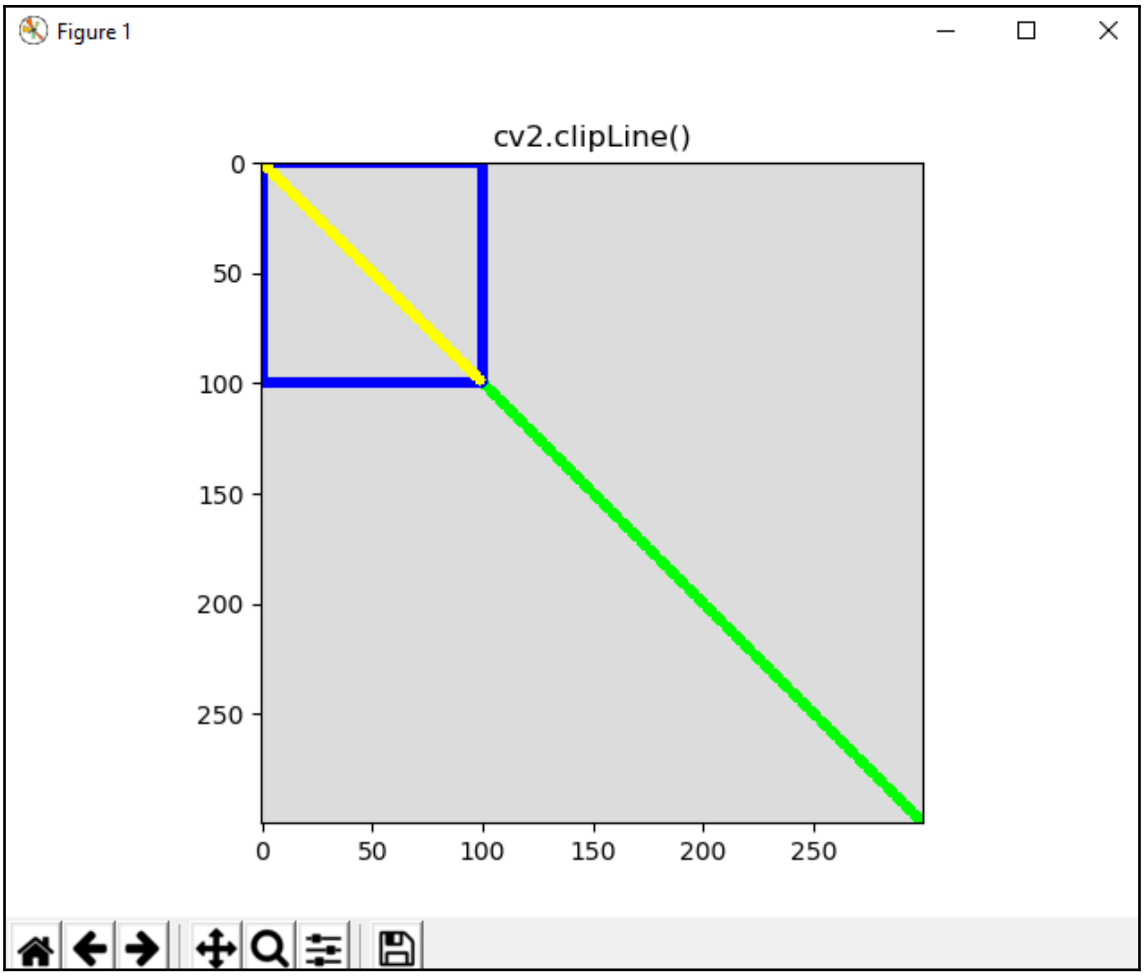


Figure 1
cv2.clipLine()
0
50
100
150
200
250
0
50
100
150
200
250

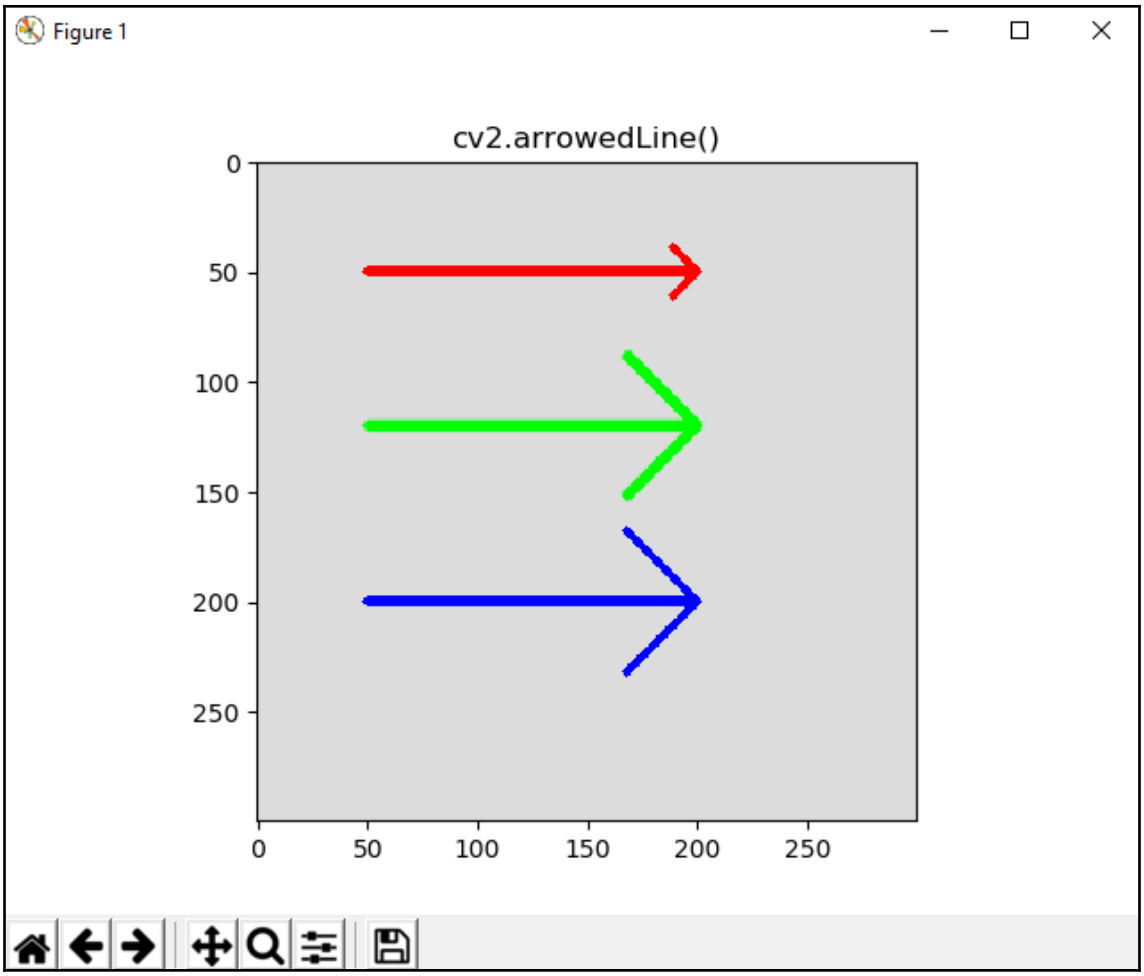


Figure 1
cv2.arrowedLine()
0
50
100
150
200
250
0
50
100
150
200
250

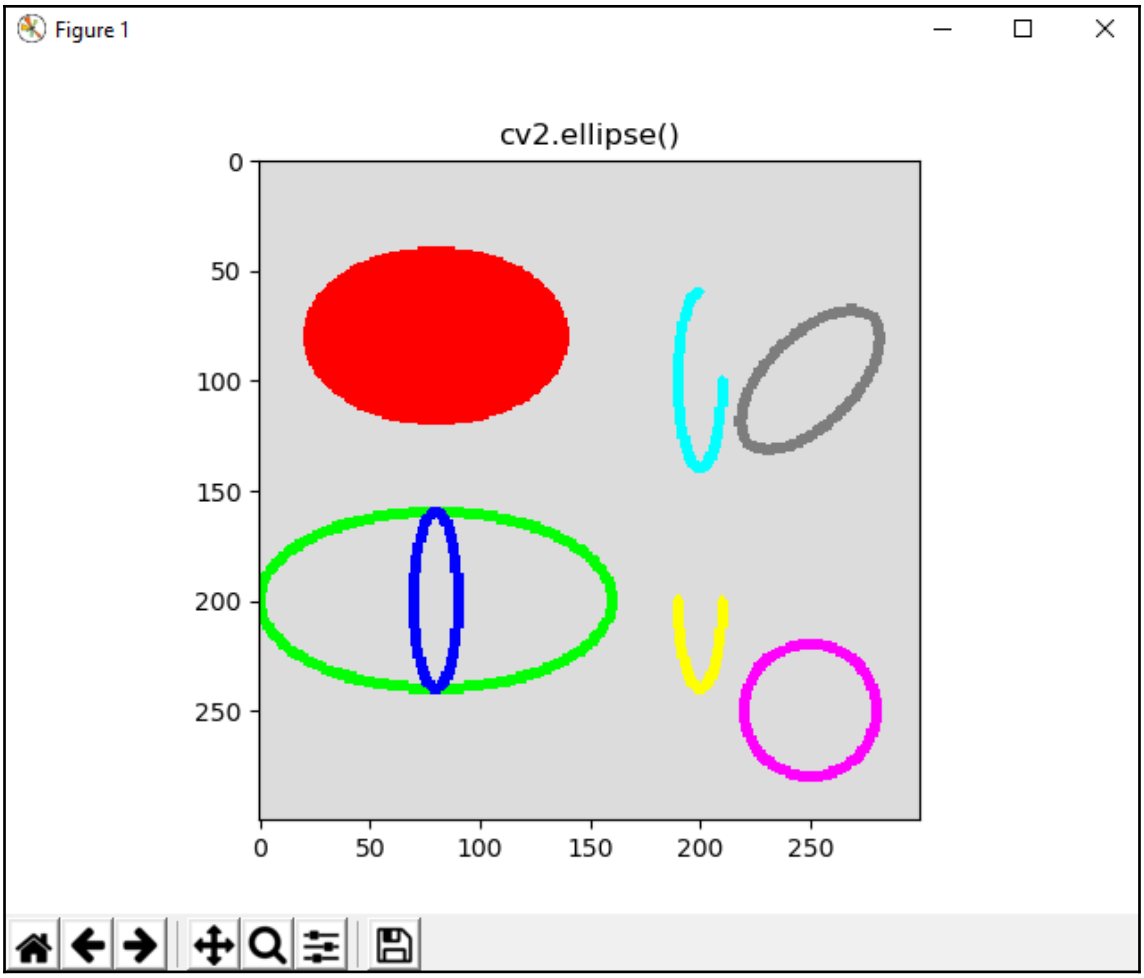


Figure 1
cv2.ellipse()
0
50
100
150
200
250
0
50
100
150
200
250

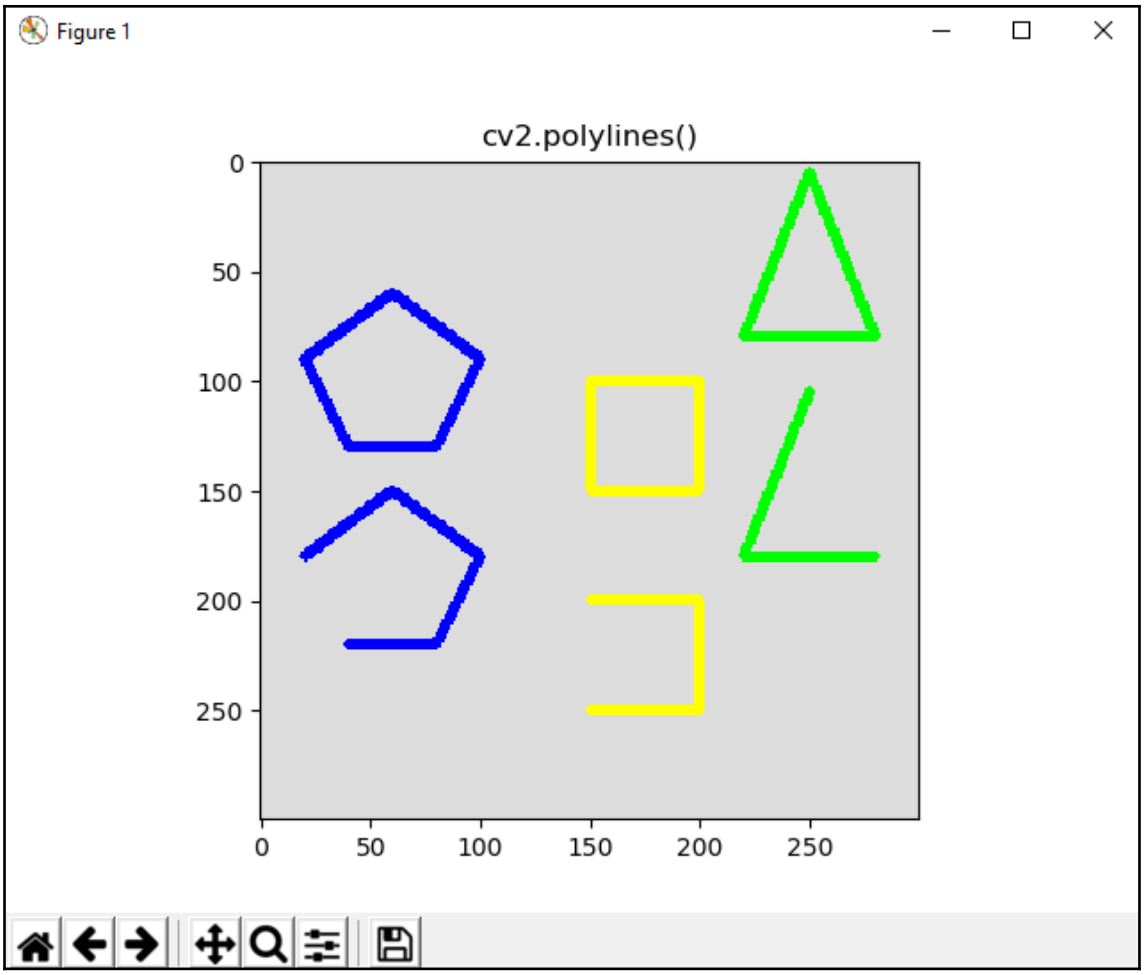


Figure 1
cv2.polylines()
0
50
100
150
200
250
0
50
100
150
200
250

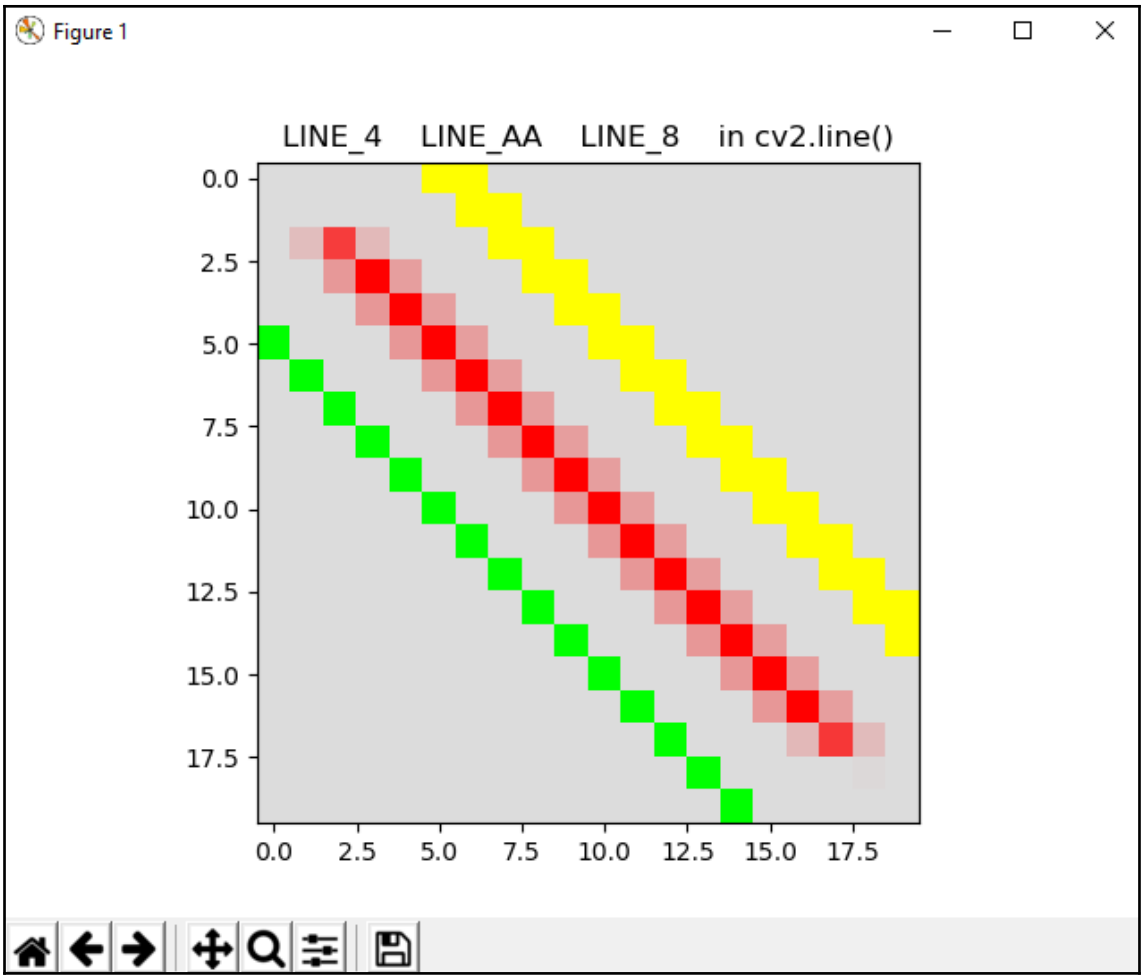


Figure 1
LINE_4 LINE_AA LINE_8 in cv2.line()
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

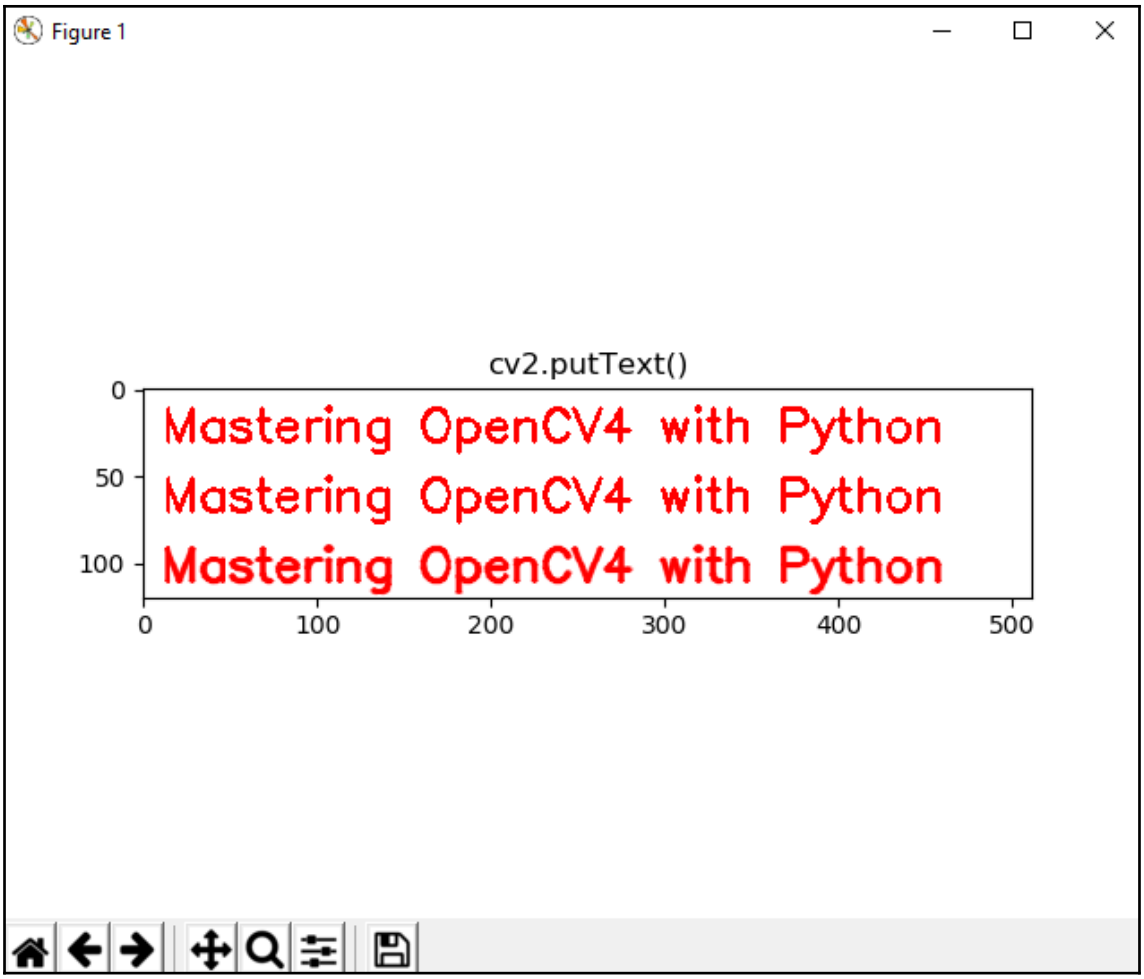


Figure 1
cv2.putText()
Mastering OpenCV4 with Python
Mastering OpenCV4 with Python
Mastering OpenCV4 with Python
0
50
100
0
100
200
300
400
500

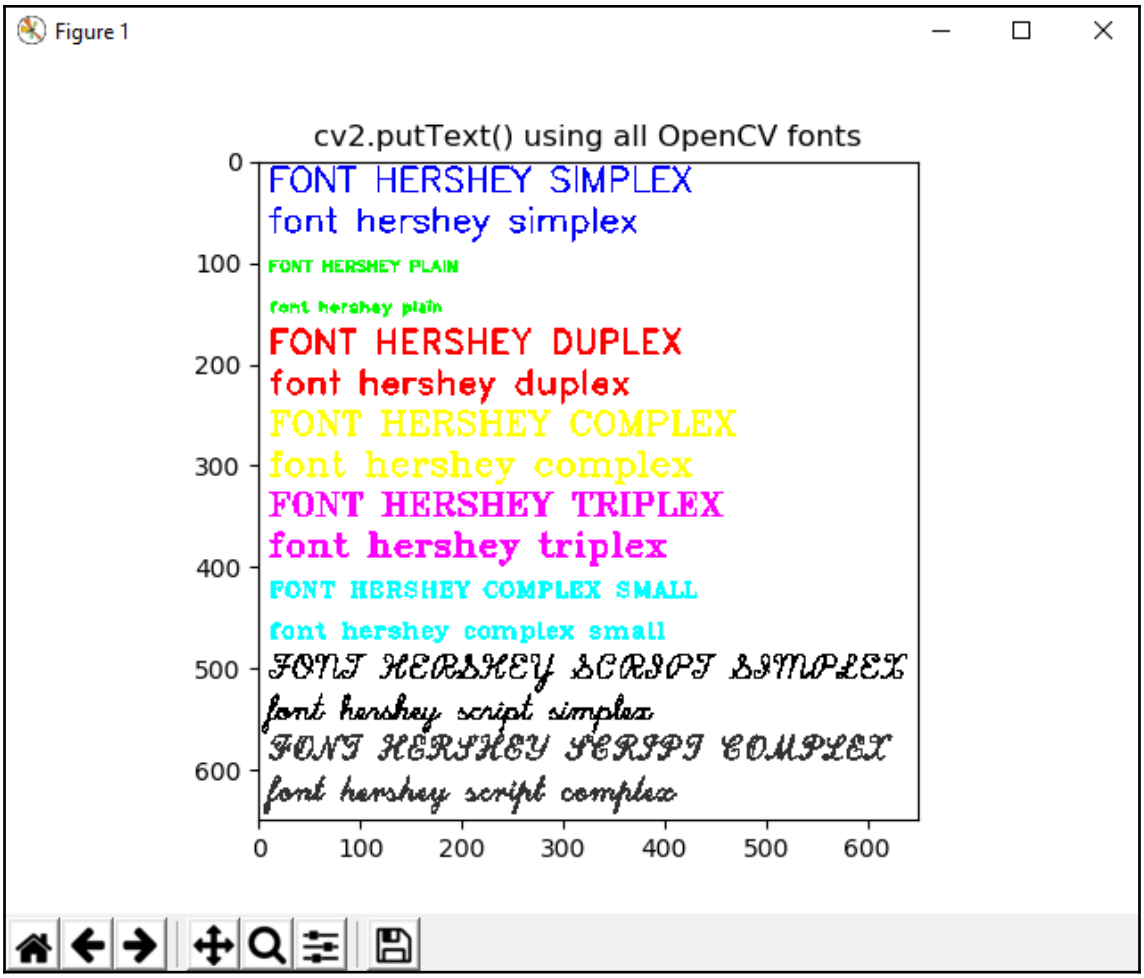


Figure 1
cv2.putText() using all OpenCV fonts
FONT HERSHEY SIMPLEX
font hershey simplex
FONT HERSHEY PLAIN
font hershey plain
FONT HERSHEY DUPLEX
font hershey duplex
FONT HERSHEY COMPLEX
font hershey complex
FONT HERSHEY TRIPLEX
font hershey triplex
FONT HERSHEY COMPLEX SMALL
font hershey complex small
FONT HERSHEY SCRIPT SIMPLEX
font hershey script simplex
FONT HERSHEY SCRIPT COMPLEX
font hershey script complex
0
100
200
300
400
500
600


Figure 1
cv2.getTextSize() + cv2.putText()
abcdefghijklmnopqrstuvwxyz
0
100
200
300
400
0
200
400
600
800
1000
1200

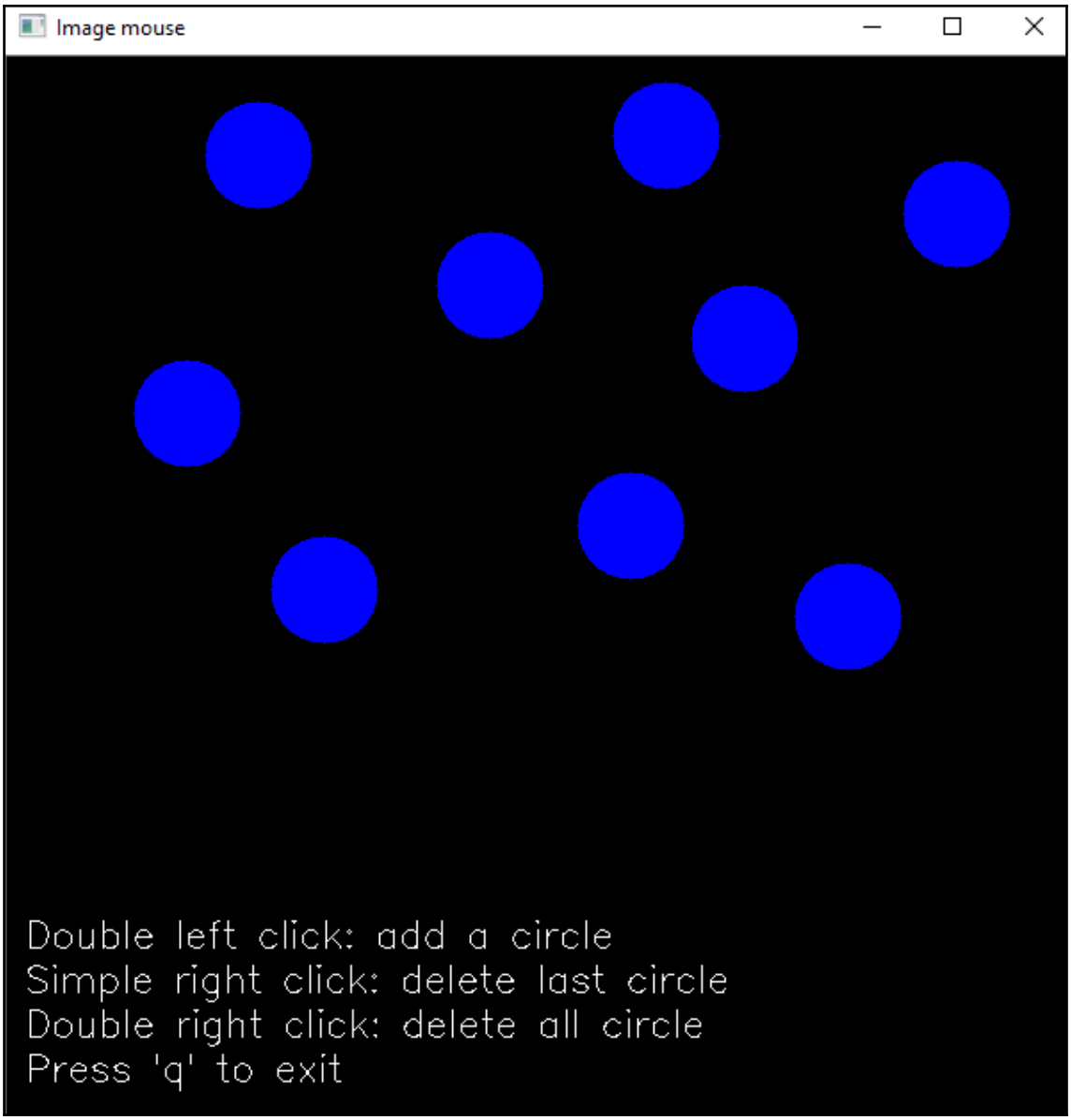
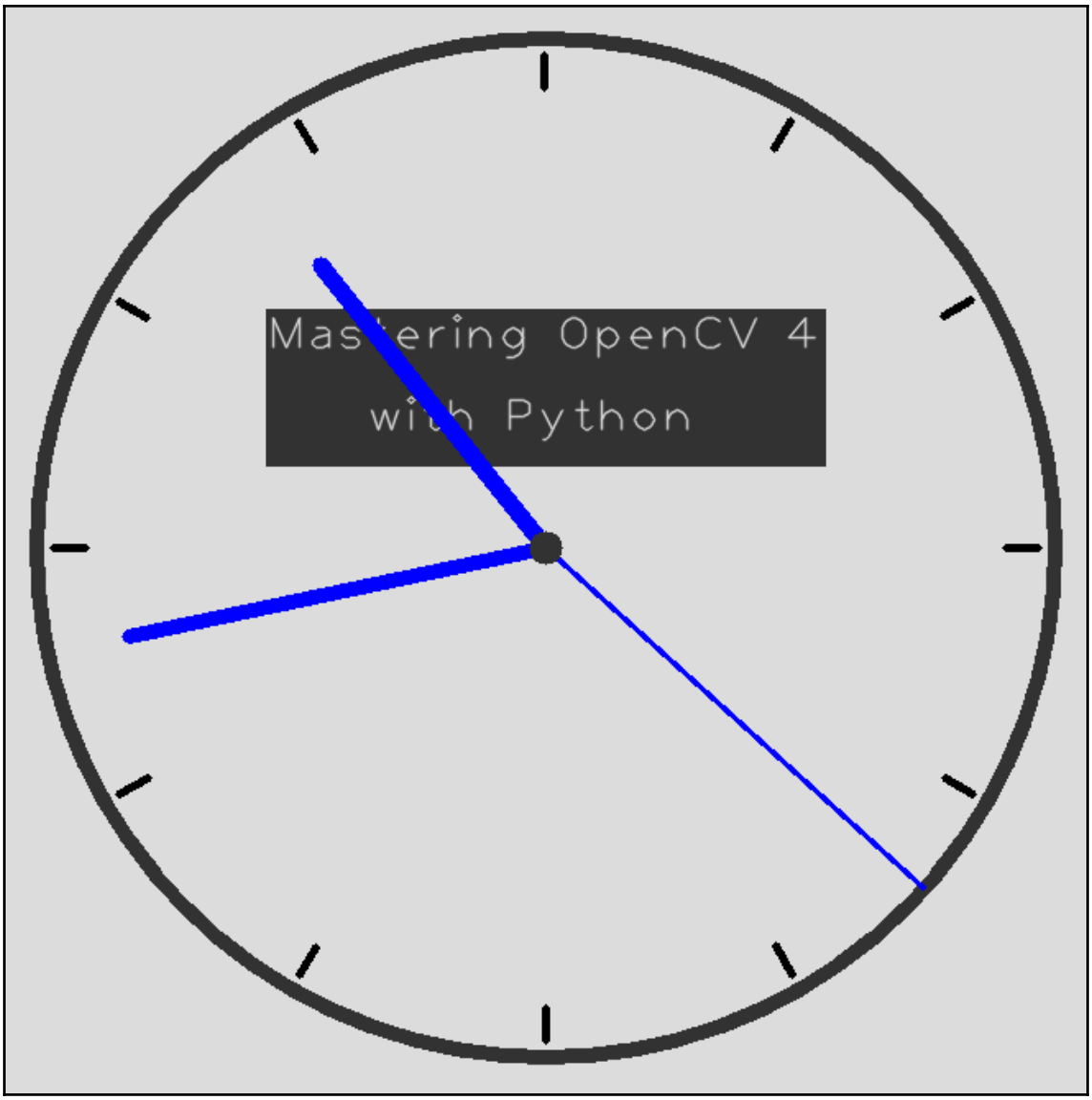
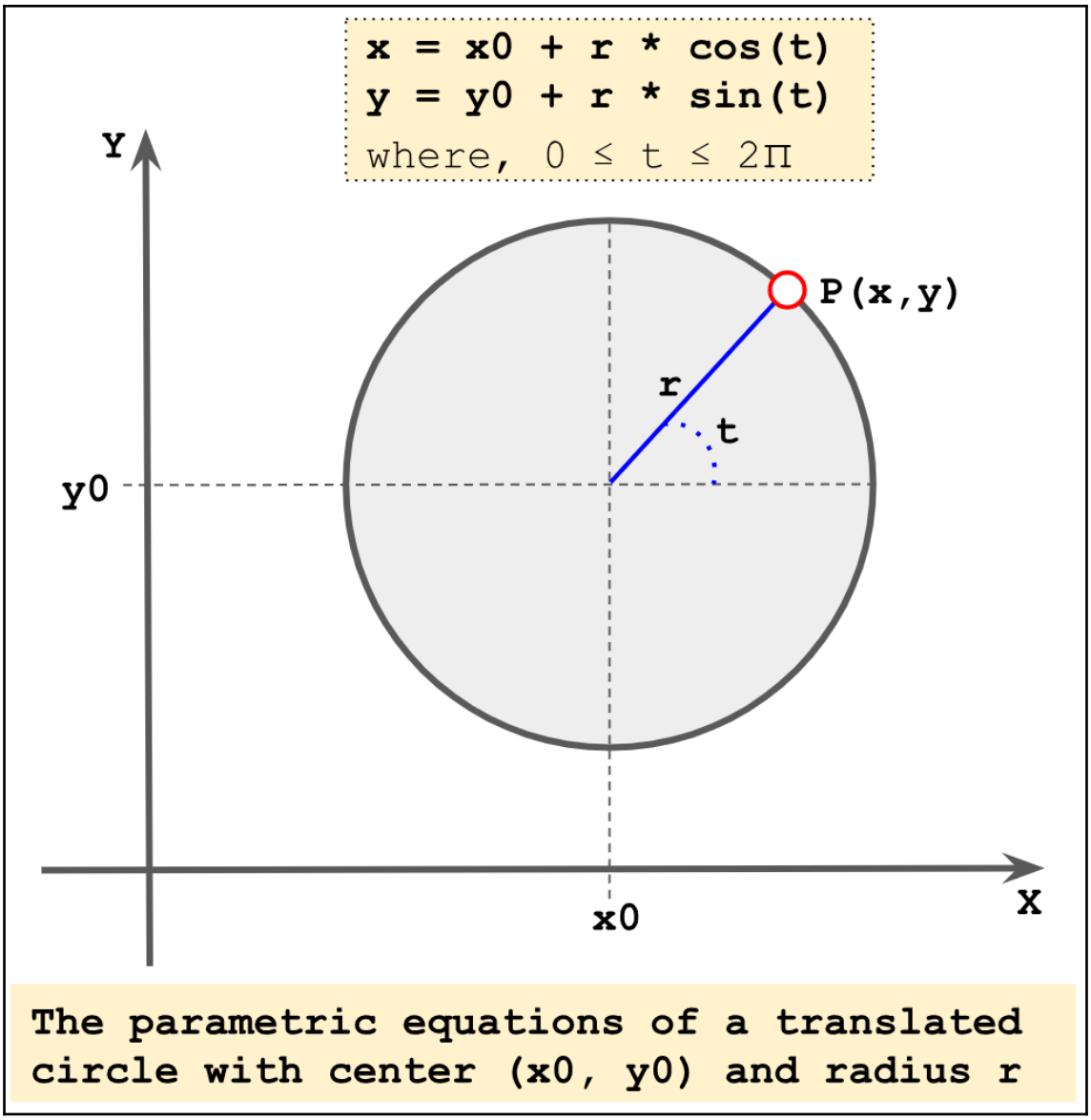


Image mouse
Double left click: add a circle
Simple right click: delete last circle
Double right click: delete all circle
Press 'q' to exit

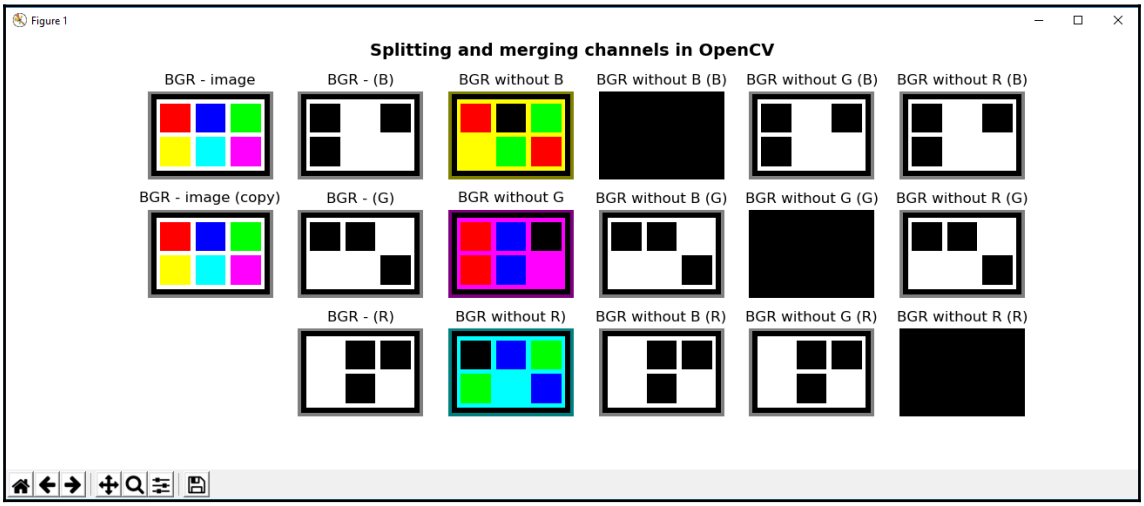

Mastering OpenCV 4
with Python

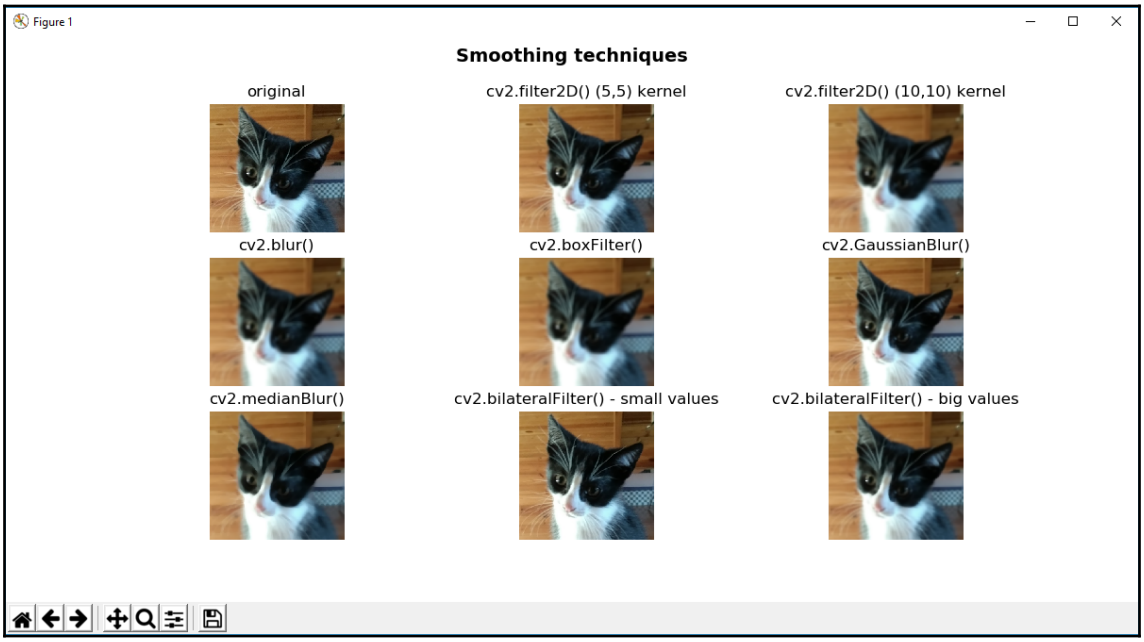


The parametric equations of a translated circle with center (x0, y0) and radius r


Figure 1
very basic meme generator
Hello World
Goodbye World
0
25
50
75
100
125
150
175
200
0
50
100
150
200

Chapter 5: Image Processing Techniques





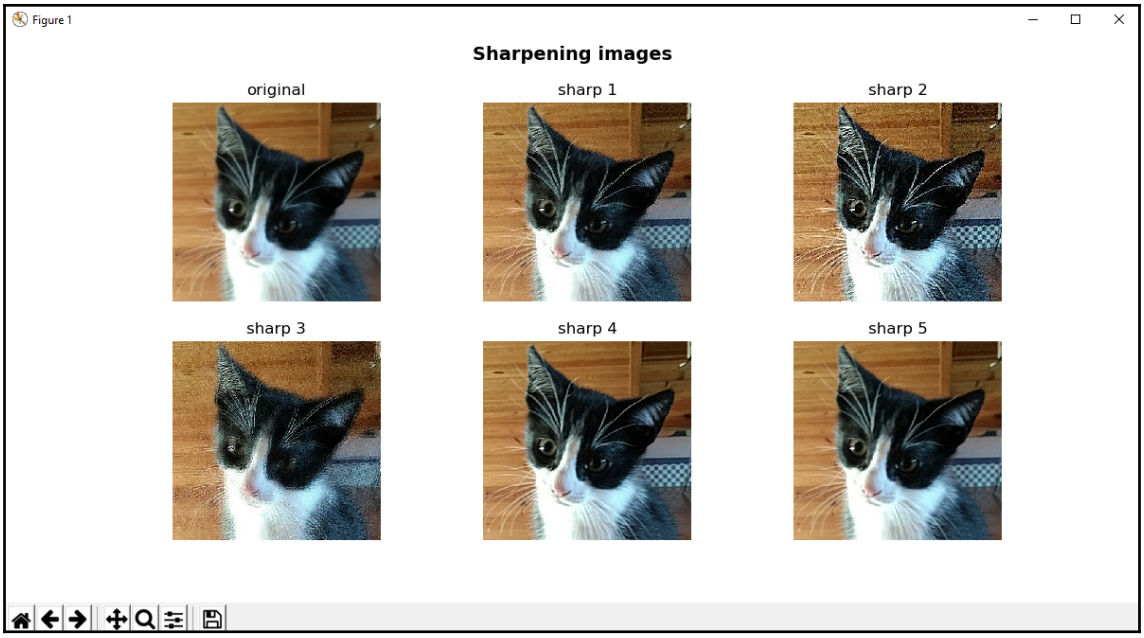


Figure 1
Sharpening images
original
sharp 1
sharp 2
sharp 3
sharp 4
sharp 5

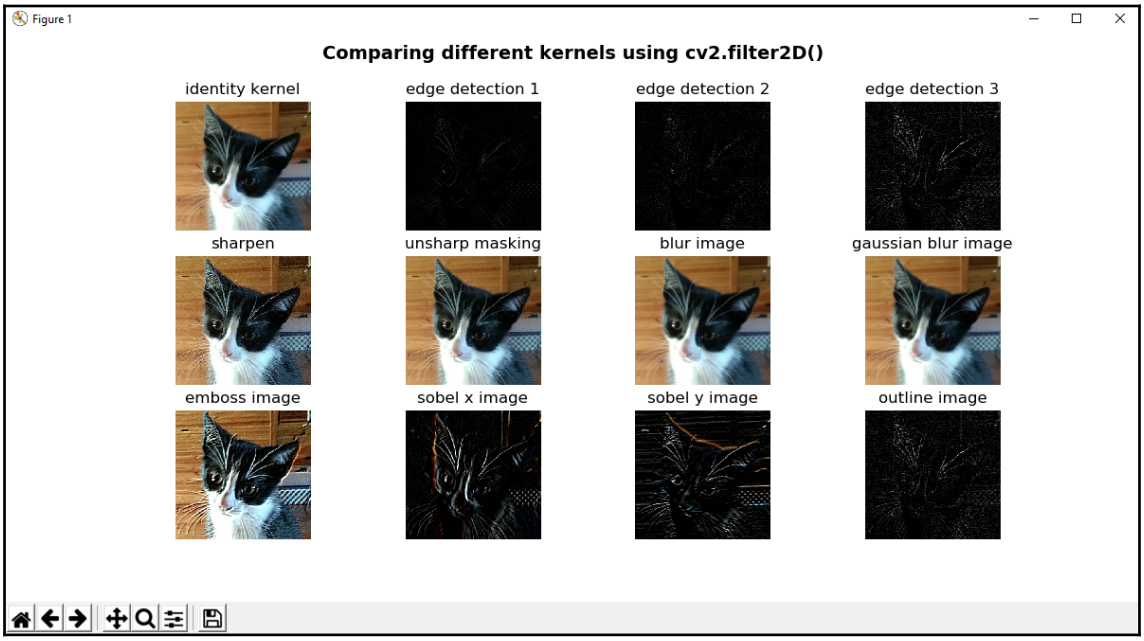


Figure 1
Comparing different kernels using cv2.filter2D()
identity kernel
edge detection 1
edge detection 2
edge detection 3
sharpen
unsharp masking
blur image
gaussian blur image
emboss image
sobel x image
sobel y image
outline image

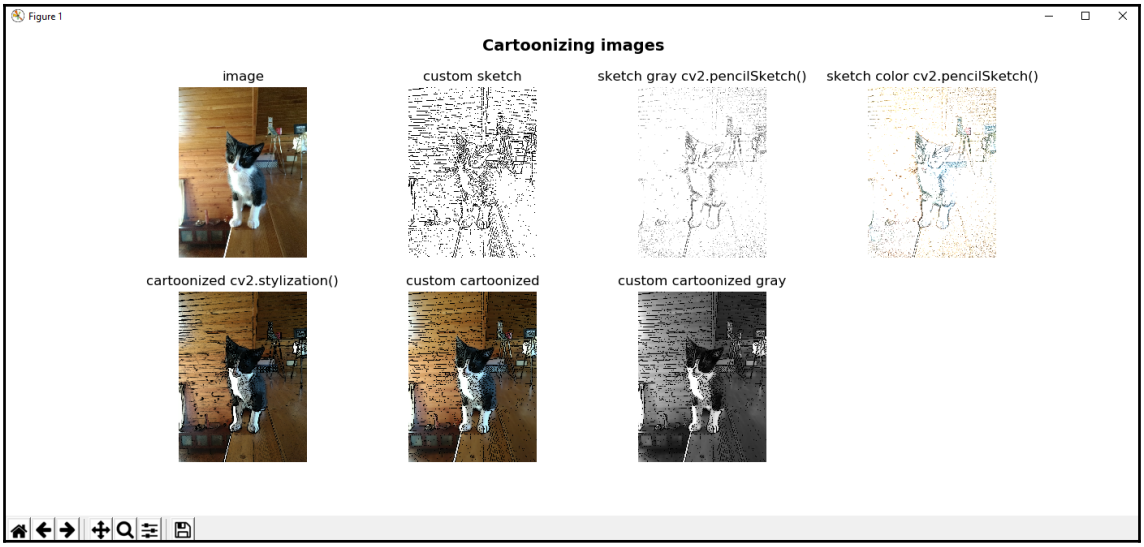


Figure 1
Cartoonizing images
image
custom sketch
sketch gray cv2.pencilSketch()
sketch color cv2.pencilSketch()
cartoonized cv2.stylization()
custom cartoonized
custom cartoonized gray


Figure 1
Arithmetic with images
image
added 60 (image + image)
subtracted 60 (image - images)
added 110 (image + scalar)
subtracted 110 (image - scalar)

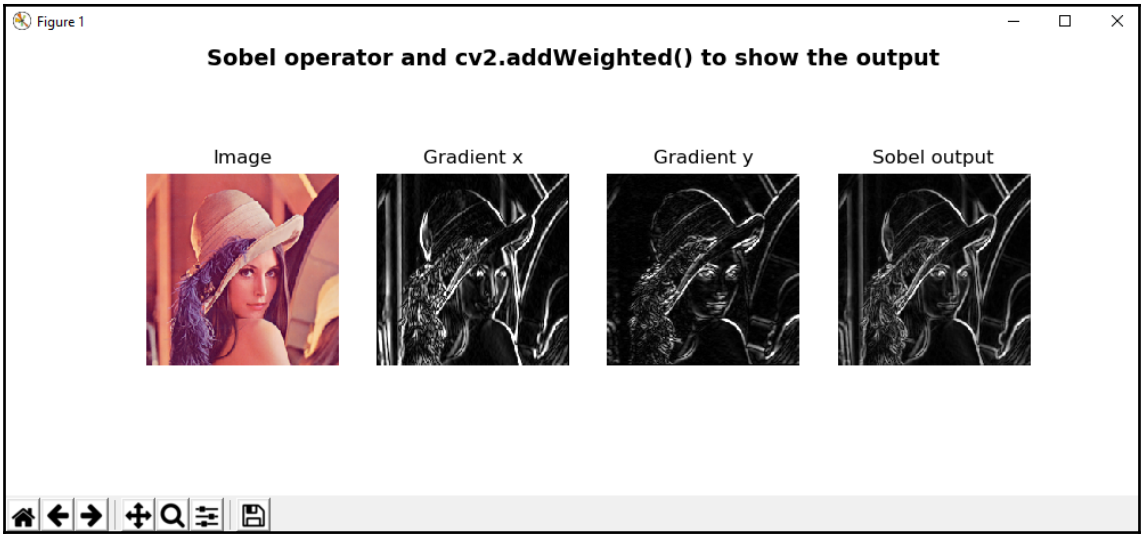


Figure 1
Sobel operator and cv2.addWeighted() to show the output
Image
Gradient x
Gradient y
Sobel output

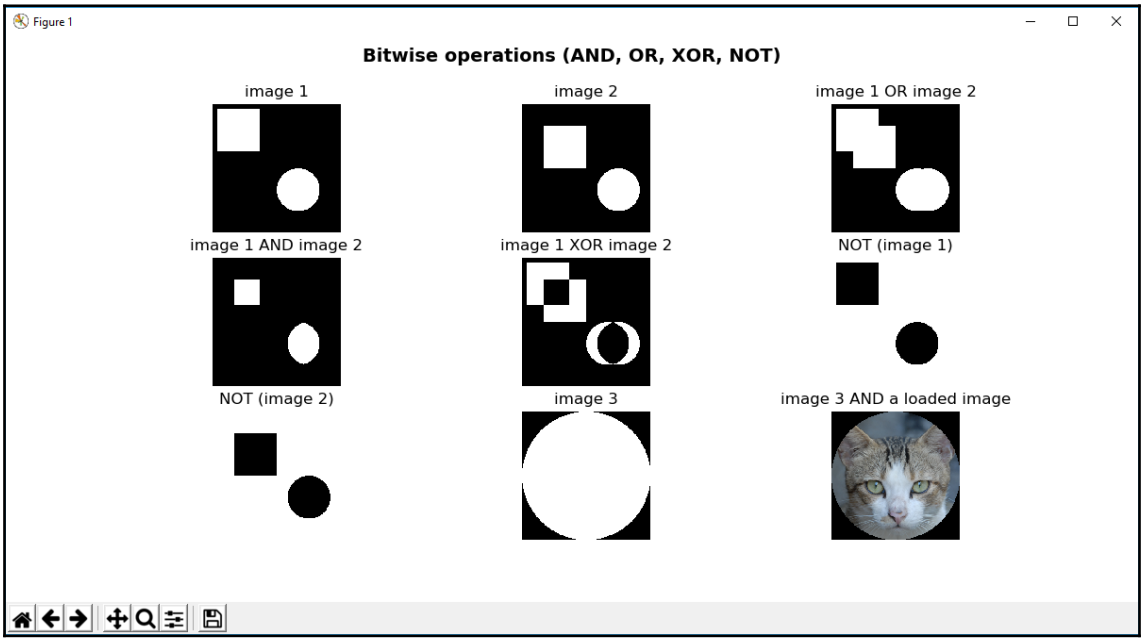


Figure 1
Bitwise operations (AND, OR, XOR, NOT)
image 1
image 2
image 1 OR image 2
image 1 AND image 2
image 1 XOR image 2
NOT (image 1)
NOT (image 2)
image 3
image 3 AND a loaded image

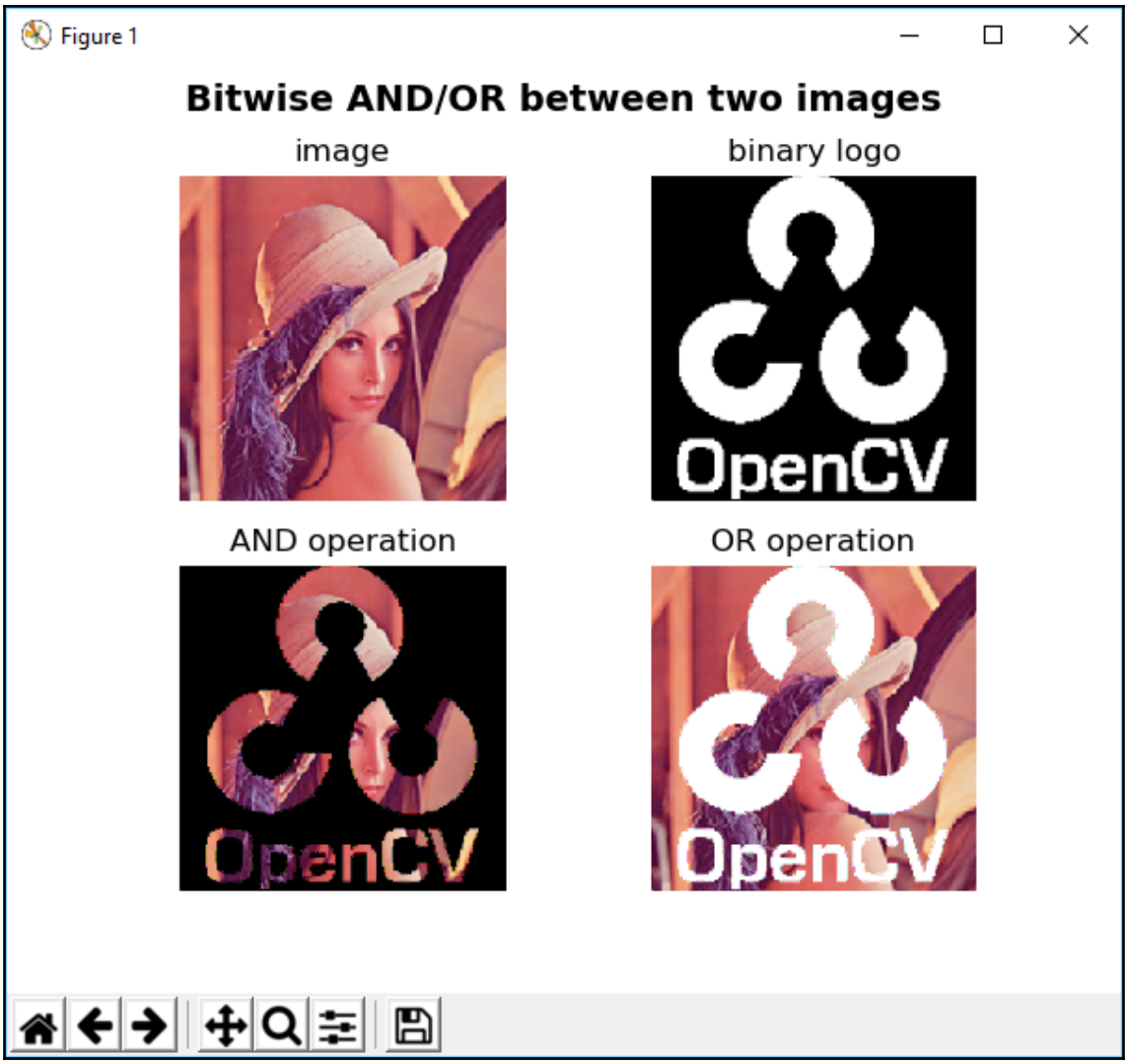


Figure 1
Bitwise AND/OR between two images
image
binary logo
AND operation
OR operation
OpenCV
OpenCV
OpenCV

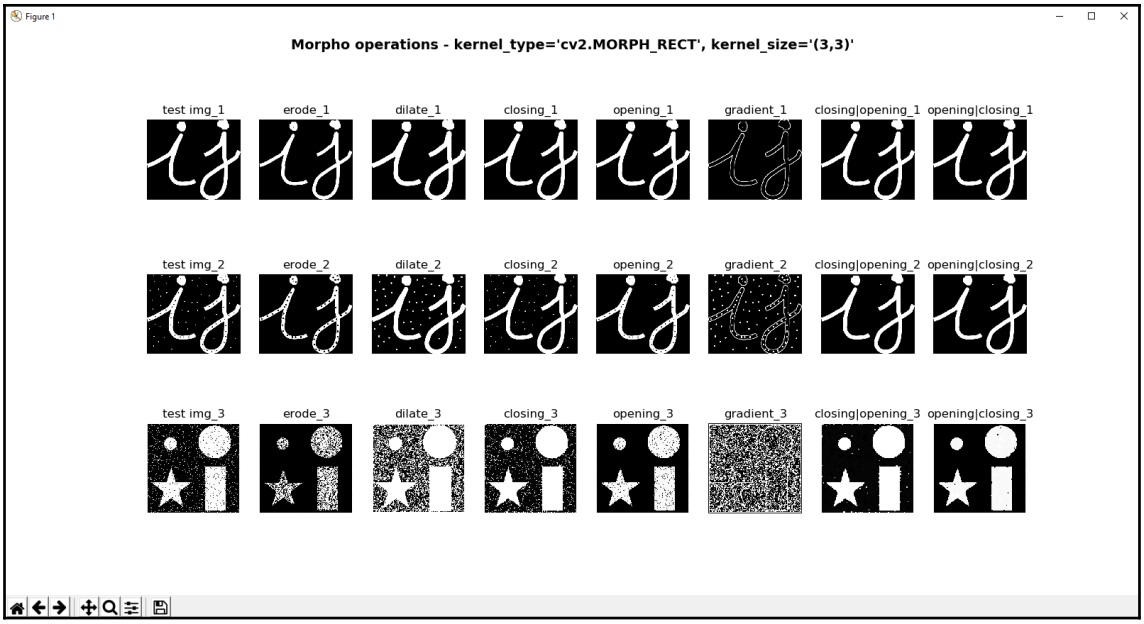


Figure 1
Morpho operations - kernel_type='cv2.MORPH_RECT', kernel_size='(3,3)'
test img_1
erode_1
dilate_1
closing_1
opening_1
gradient_1
closing|opening_1
opening|closing_1
test img_2
erode_2
dilate_2
closing_2
opening_2
gradient_2
closing|opening_2
opening|closing_2
test img_3
erode_3
dilate_3
closing_3
opening_3
gradient_3
closing|opening_3
opening|closing_3

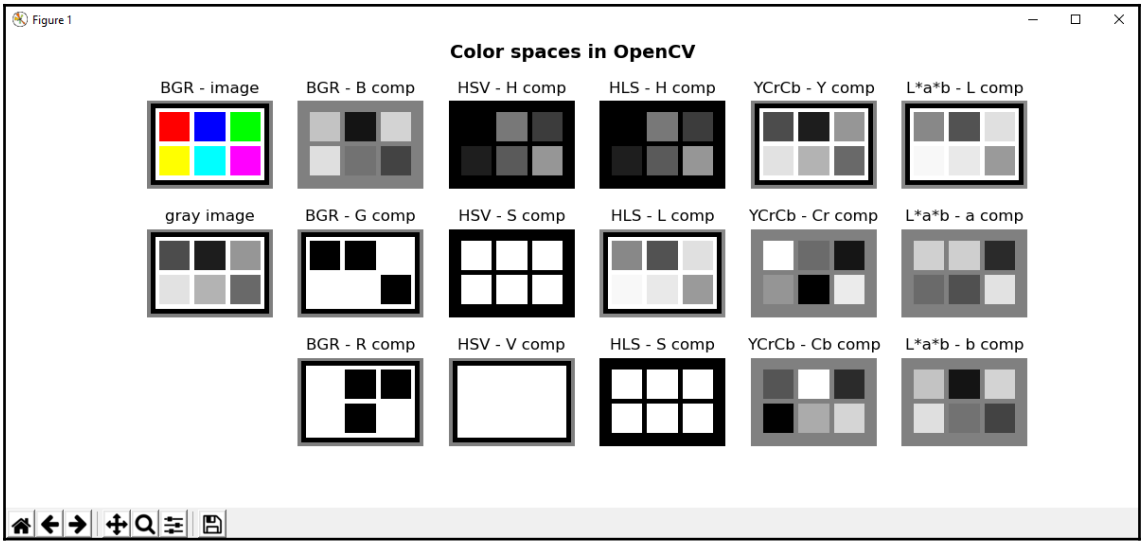


Figure 1
Color spaces in OpenCV
BGR - image
BGR - B comp
HSV - H comp
HLS - H comp
YCrCb - Y comp
L*a*b - L comp
gray image
BGR - G comp
HSV - S comp
HLS - L comp
YCrCb - Cr comp
L*a*b - a comp
BGR - R comp
HSV - V comp
HLS - S comp
YCrCb - Cb comp
L*a*b - b comp

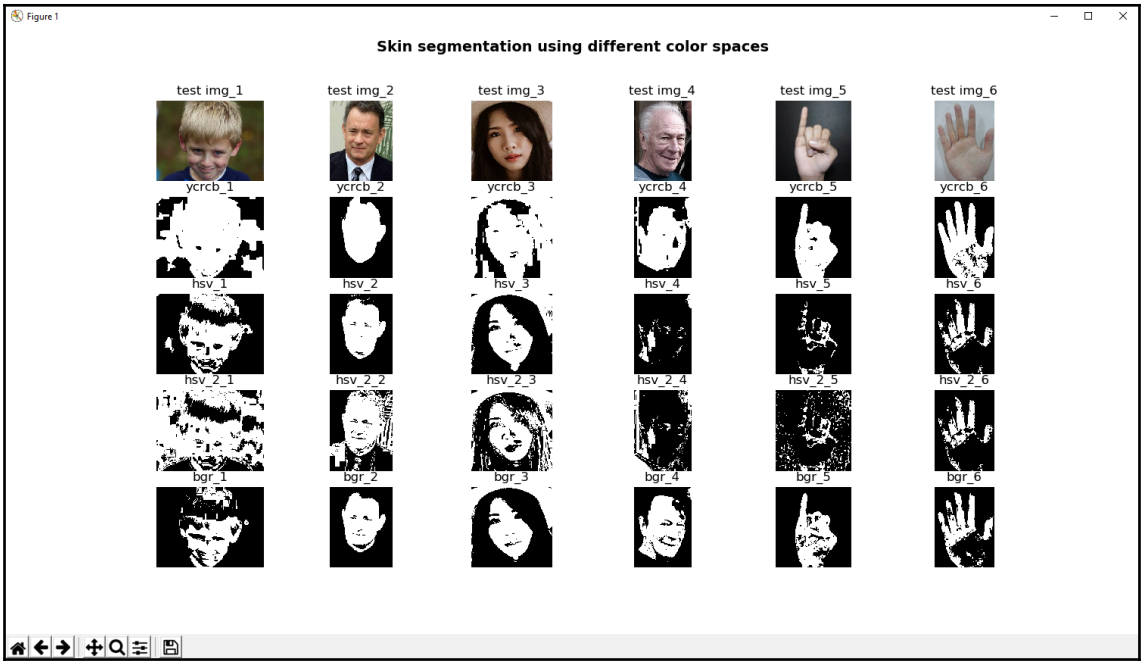


Figure 1
Skin segmentation using different color spaces
test img_1
test img_2
test img_3
test img_4
test img_5
test img_6
ycrcb_1
ycrcb_2
ycrcb_3
ycrcb_4
ycrcb_5
ycrcb_6
hsv_1
hsv_2
hsv_3
hsv_4
hsv_5
hsv_6
hsv_2_1
hsv_2_2
hsv_2_3
hsv_2_4
hsv_2_5
hsv_2_6
bgr_1
bgr_2
bgr_3
bgr_4
bgr_5
bgr_6

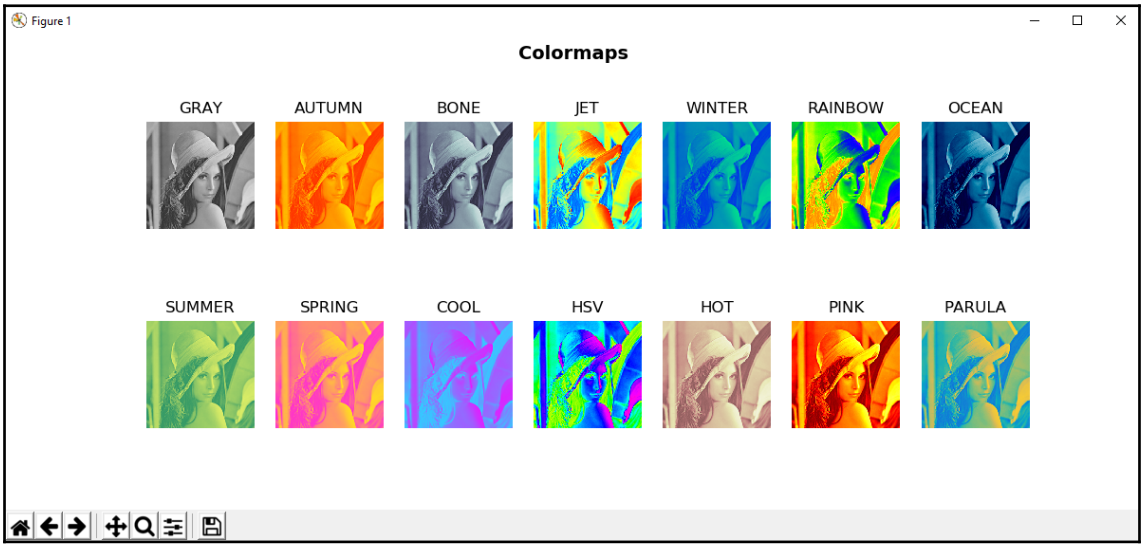


Figure 1
Colormaps
GRAY
AUTUMN
BONE
JET
WINTER
RAINBOW
OCEAN
SUMMER
SPRING
COOL
HSV
HOT
PINK
PARULA

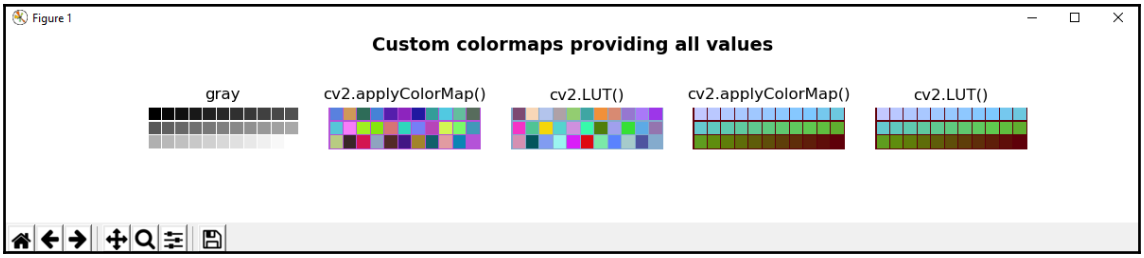
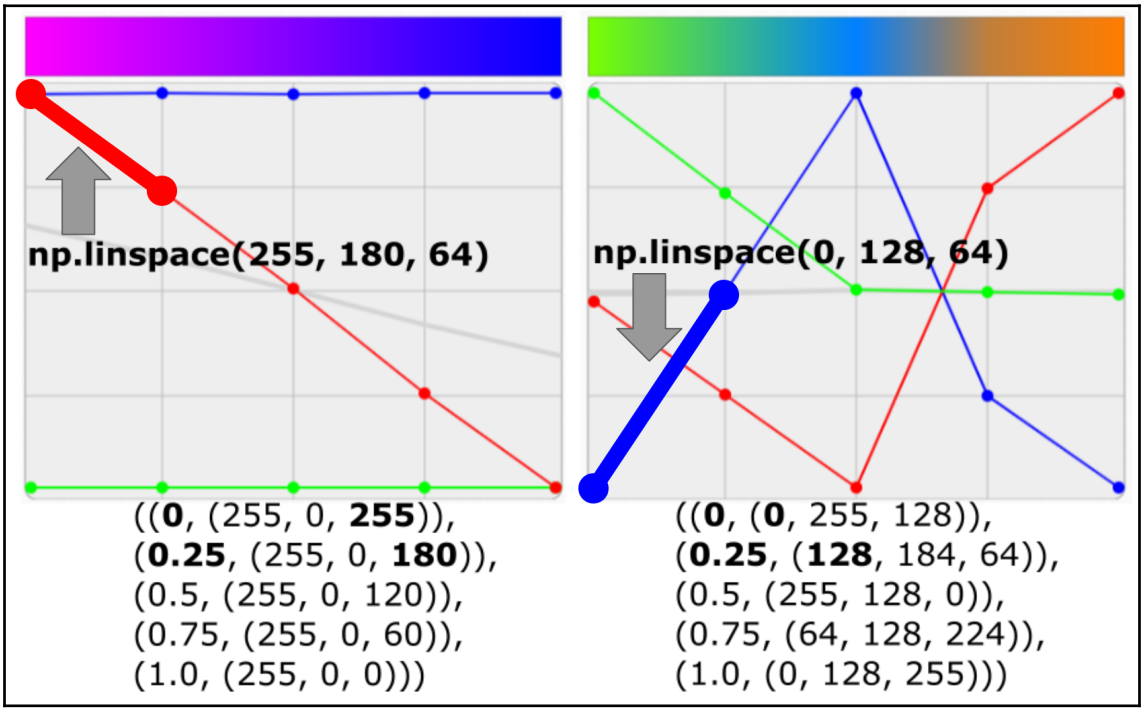


Figure 1
Custom colormaps providing all values
gray
cv2.applyColorMap()
cv2.LUT()
cv2.applyColorMap()
cv2.LUT()


np.linspace(255, 180, 64)
np.linspace(0, 128, 64)
((0, (255, 0, 255)),
(0.25, (255, 0, 180)),
(0.5, (255, 0, 120)),
(0.75, (255, 0, 60)),
(1.0, (255, 0, 0)))
((0, (0, 255, 128)),
(0.25, (128, 184, 64)),
(0.5, (255, 128, 0)),
(0.75, (64, 128, 224)),
(1.0, (0, 128, 255)))

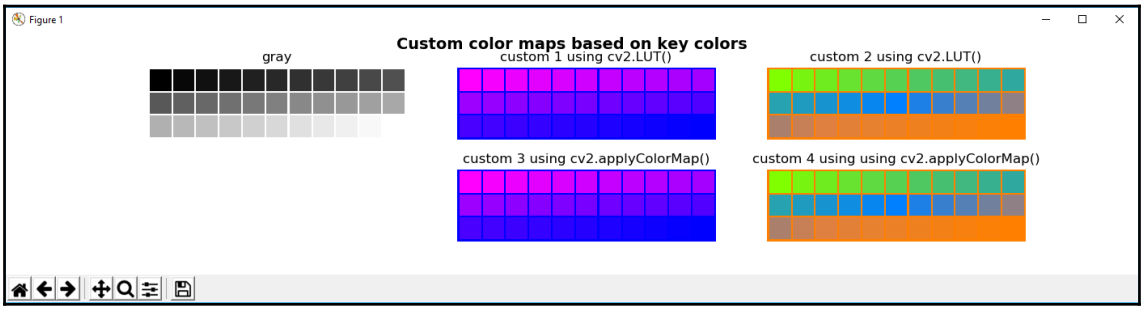


Figure 1
Custom color maps based on key colors
gray
custom 1 using cv2.LUT()
custom 2 using cv2.LUT()
custom 3 using cv2.applyColorMap()
custom 4 using using cv2.applyColorMap()

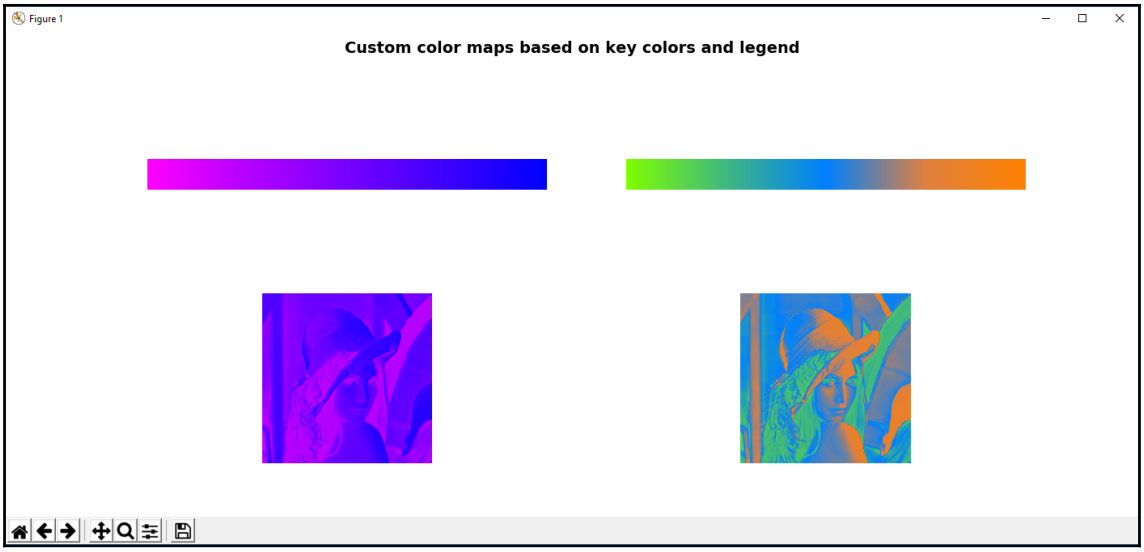
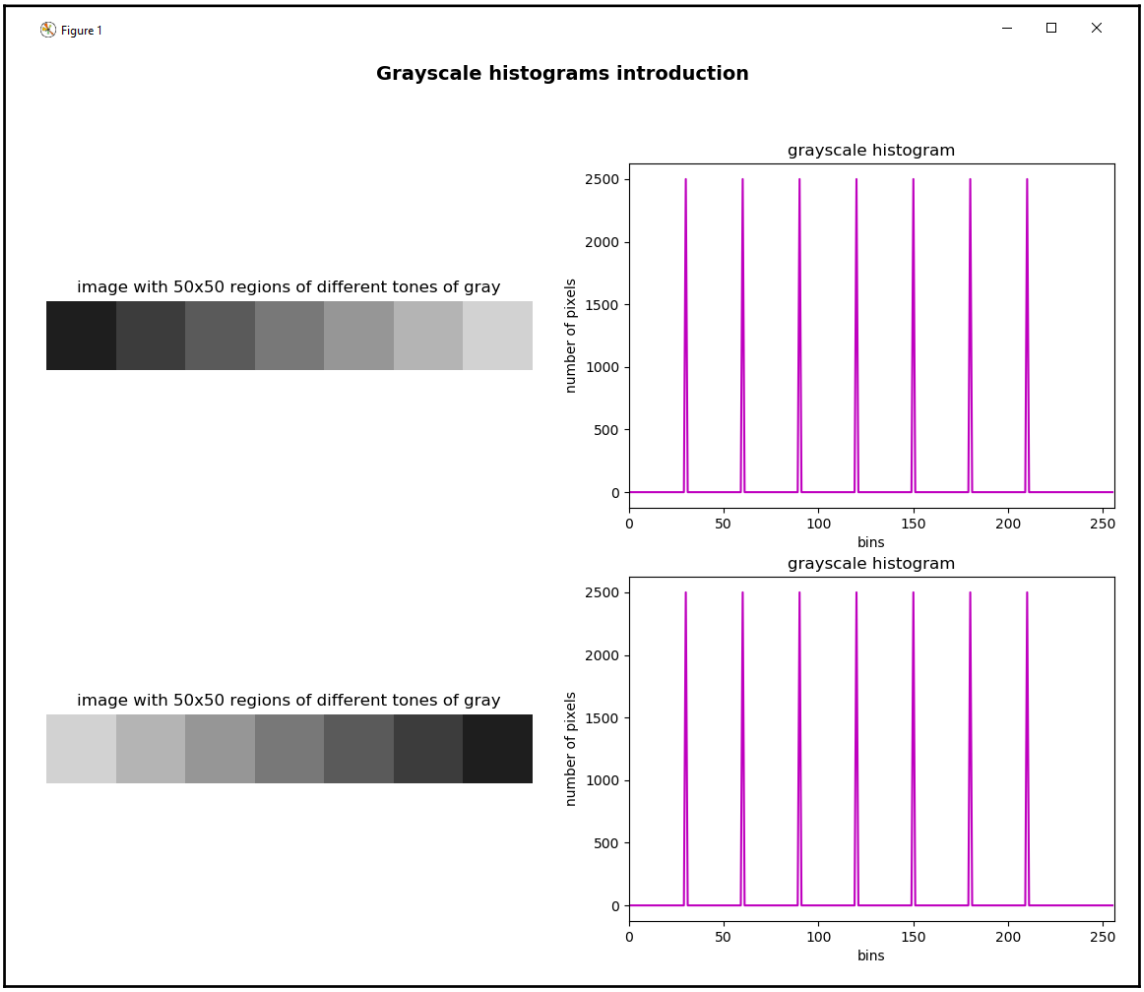


Figure 1
Custom color maps based on key colors and legend

Chapter 6: Constructing and Building Histograms



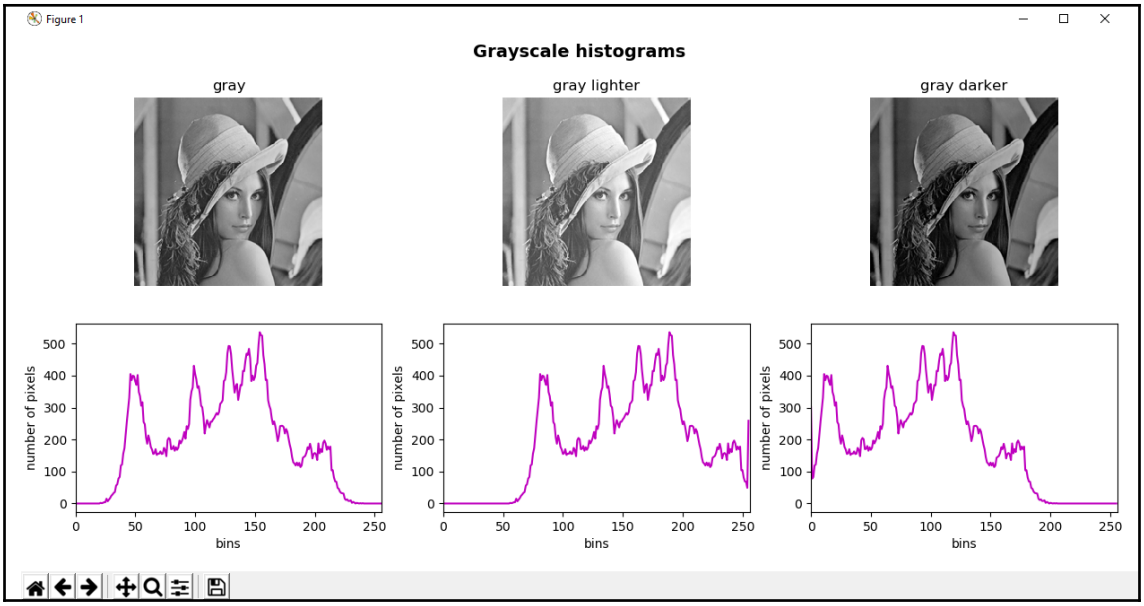


Figure 1
Grayscale histograms
gray
gray lighter
gray darker
number of pixels
bins
0
50
100
150
200
250
100
200
300
400
500

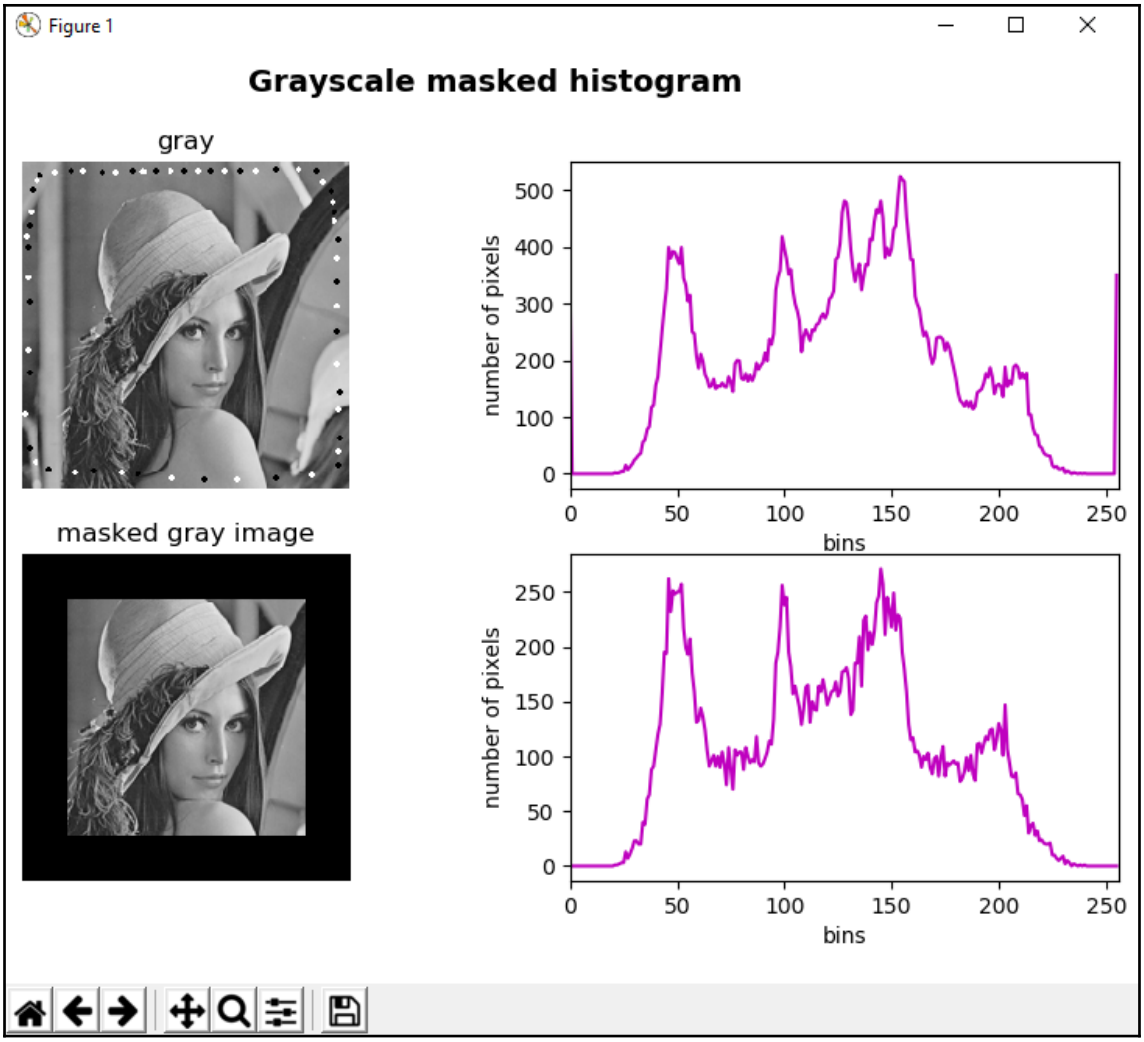


Figure 1
Grayscale masked histogram
gray
masked gray image
number of pixels
bins
number of pixels
bins

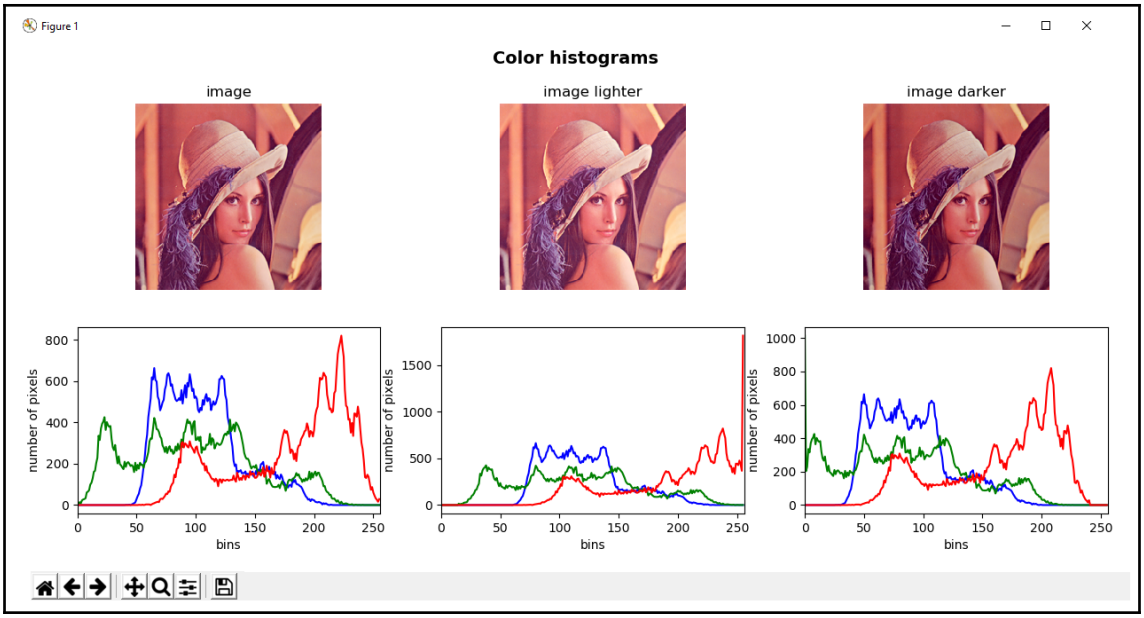


Figure 1
Color histograms
image
image lighter
image darker
number of pixels
bins
0
50
100
150
200
250
0
200
400
600
800
0
500
1000
1500
0
200
400
600
800
1000

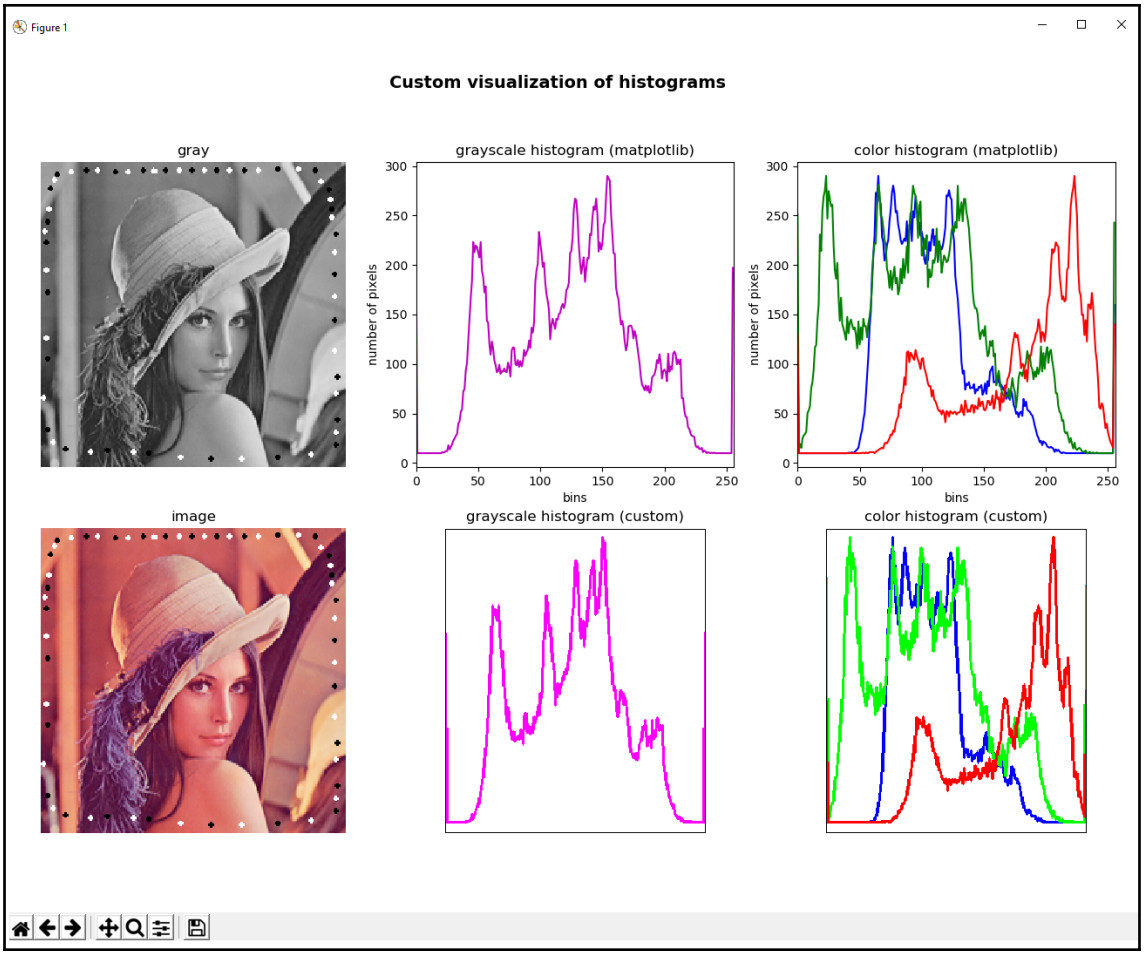


Figure 1
Custom visualization of histograms
gray
grayscale histogram (matplotlib)
color histogram (matplotlib)
number of pixels
bins
image
grayscale histogram (custom)
color histogram (custom)

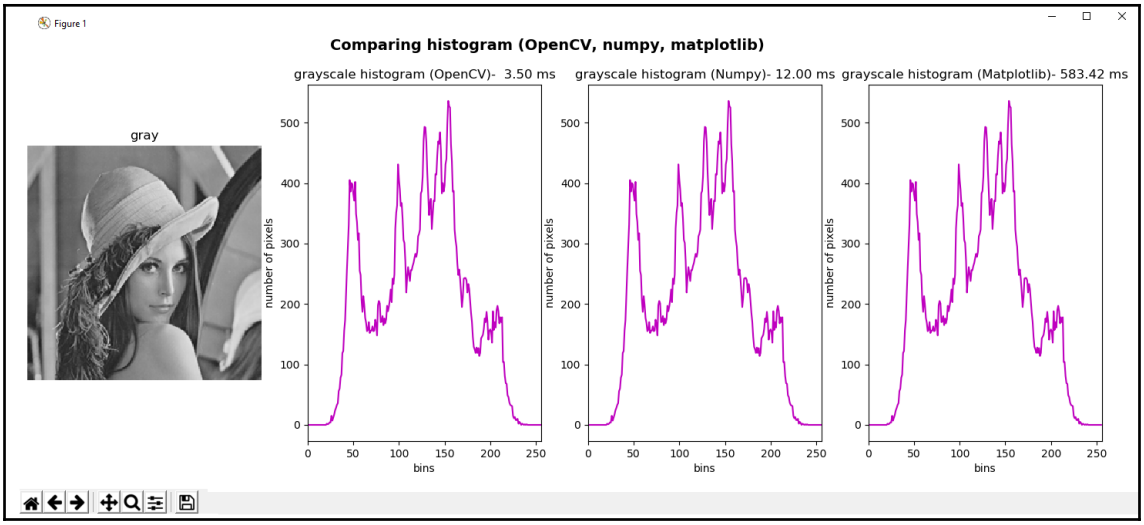


Figure 1
Comparing histogram (OpenCV, numpy, matplotlib)
gray
grayscale histogram (OpenCV)- 3.50 ms
grayscale histogram (Numpy)- 12.00 ms
grayscale histogram (Matplotlib)- 583.42 ms
number of pixels
bins

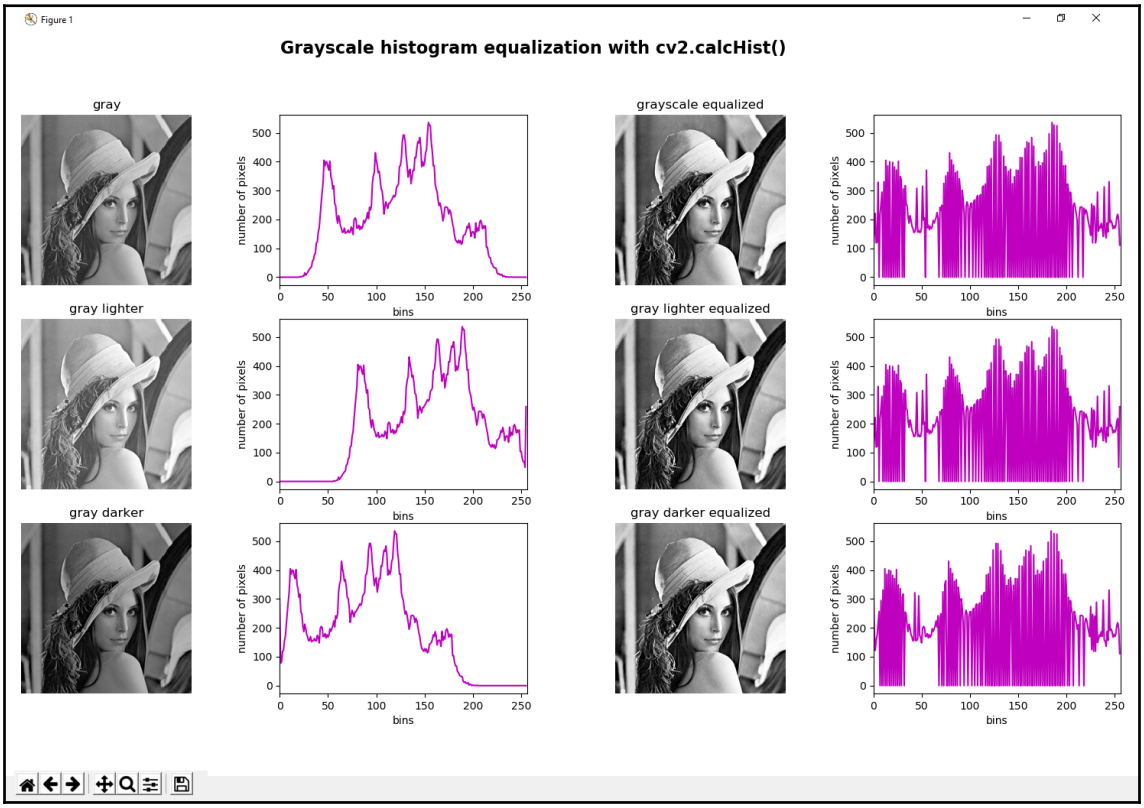


Figure 1
Grayscale histogram equalization with cv2.calcHist()
gray
gray lighter
gray darker
grayscale equalized
gray lighter equalized
gray darker equalized
number of pixels
bins

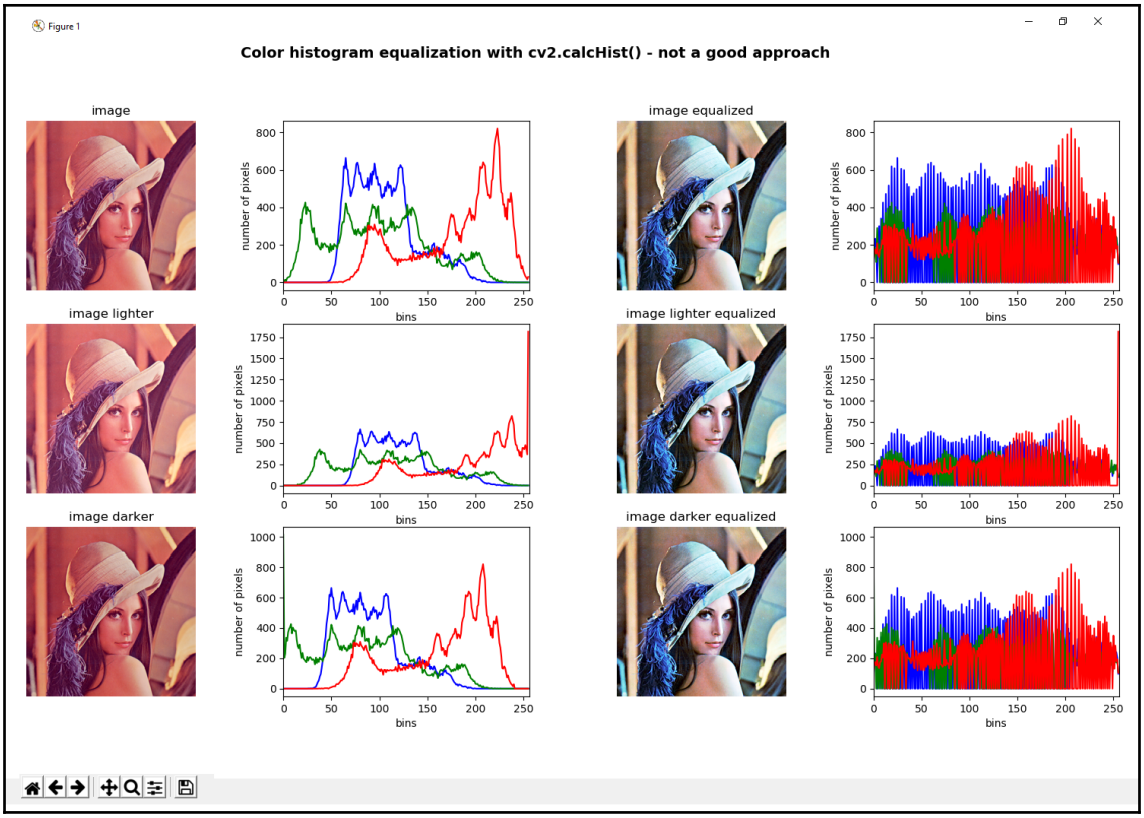


Figure 1
Color histogram equalization with cv2.calcHist() - not a good approach
image
image lighter
image darker
image equalized
image lighter equalized
image darker equalized
number of pixels
bins

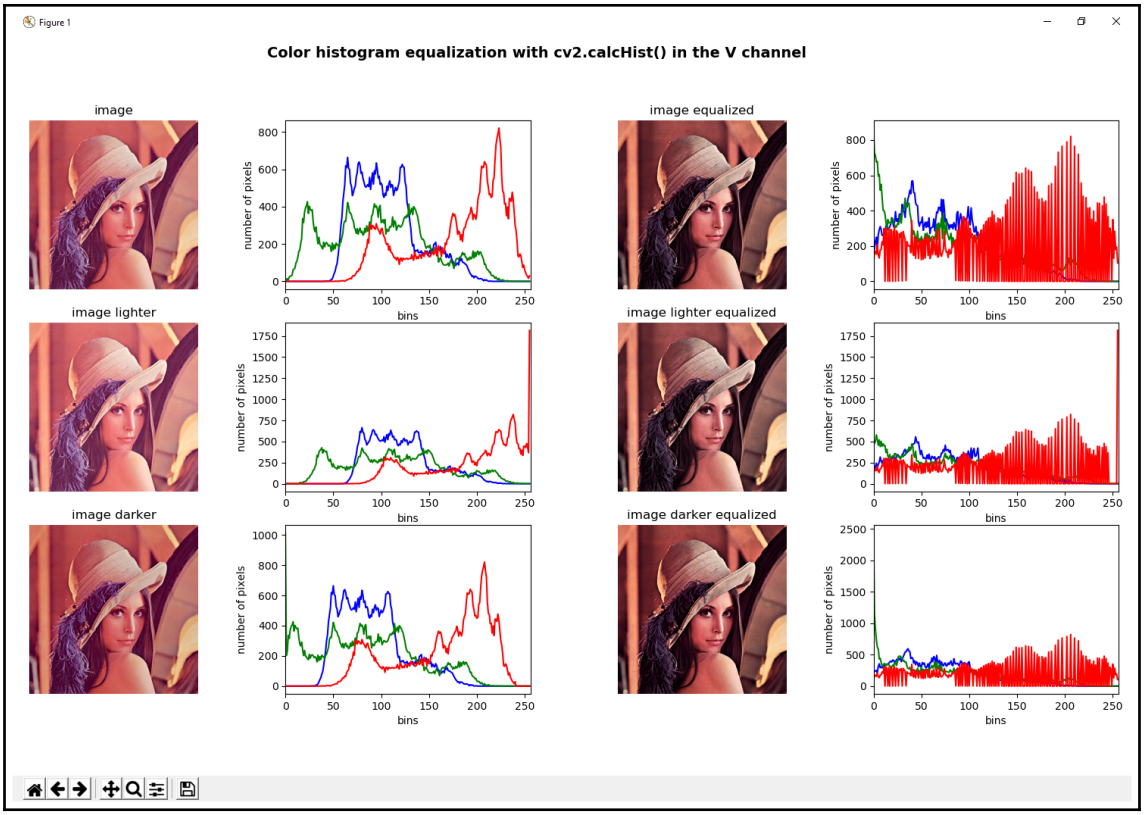


Figure 1
Color histogram equalization with cv2.calcHist() in the V channel
image
image lighter
image darker
image equalized
image lighter equalized
image darker equalized
number of pixels
bins


Figure 1
Histogram equalization using CLAHE
gray
gray CLAHE clipLimit=2.0
gray CLAHE clipLimit=5.0
gray CLAHE clipLimit=10.0
gray CLAHE clipLimit=20.0
color
clahe on each channel (BGR)
clahe on L channel (LAB)
clahe on V channel (HSV)
clahe on Y channel (YUV)

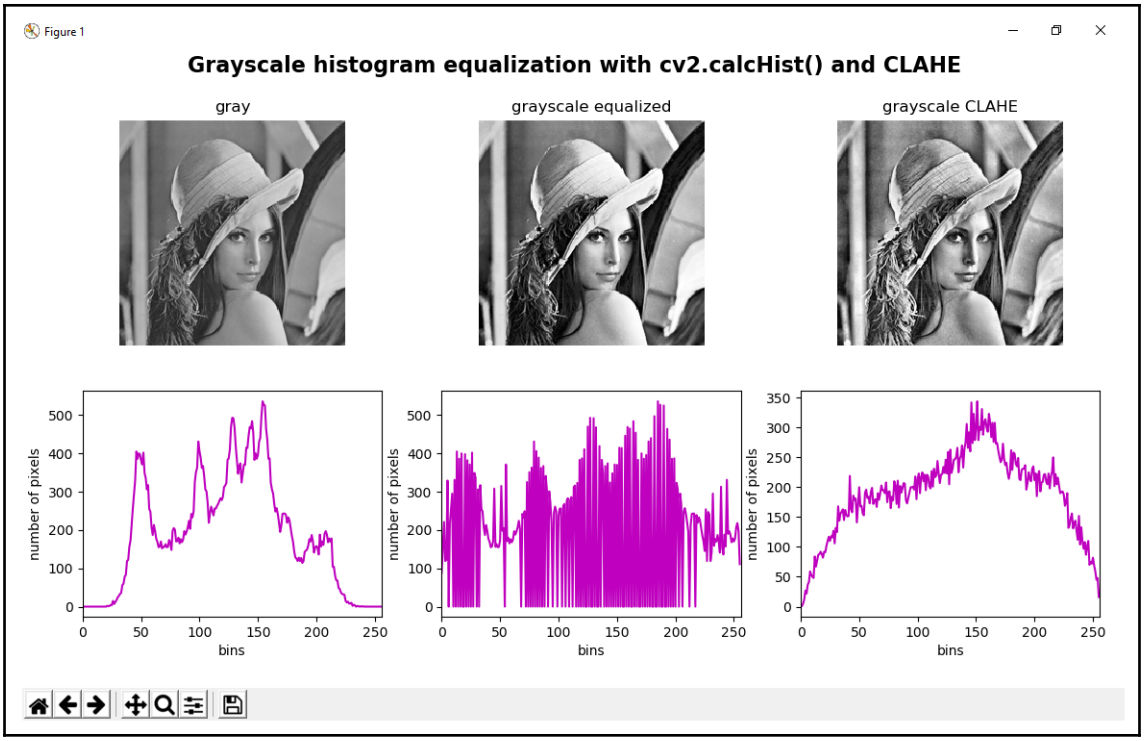


Figure 1
Grayscale histogram equalization with cv2.calcHist() and CLAHE
gray
grayscale equalized
grayscale CLAHE
number of pixels
bins
0
50
100
150
200
250
0
100
200
300
400
500
0
50
100
150
200
250
300
350

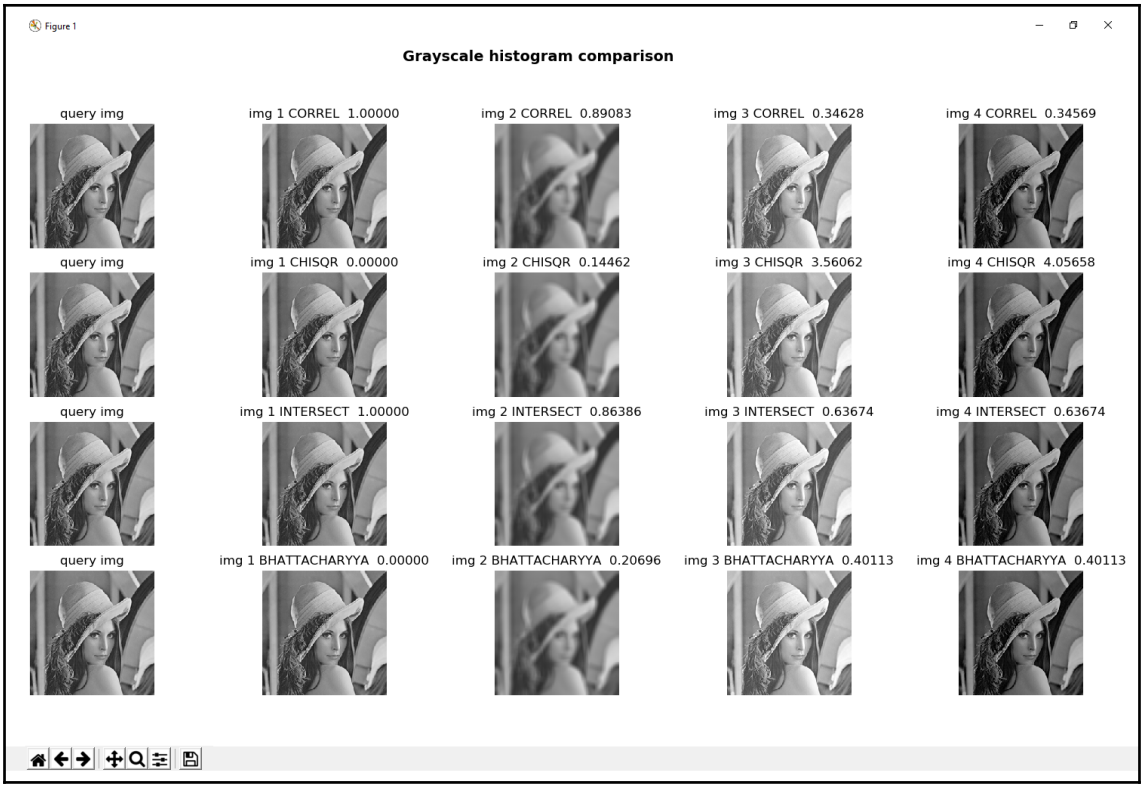


Figure 1
Grayscale histogram comparison
query img
img 1 CORREL 1.00000
img 2 CORREL 0.89083
img 3 CORREL 0.34628
img 4 CORREL 0.34569
query img
img 1 CHISQR 0.00000
img 2 CHISQR 0.14462
img 3 CHISQR 3.56062
img 4 CHISQR 4.05658
query img
img 1 INTERSECT 1.00000
img 2 INTERSECT 0.86386
img 3 INTERSECT 0.63674
img 4 INTERSECT 0.63674
query img
img 1 BHATTACHARYYA 0.00000
img 2 BHATTACHARYYA 0.20696
img 3 BHATTACHARYYA 0.40113
img 4 BHATTACHARYYA 0.40113

Chapter 7: Thresholding Techniques

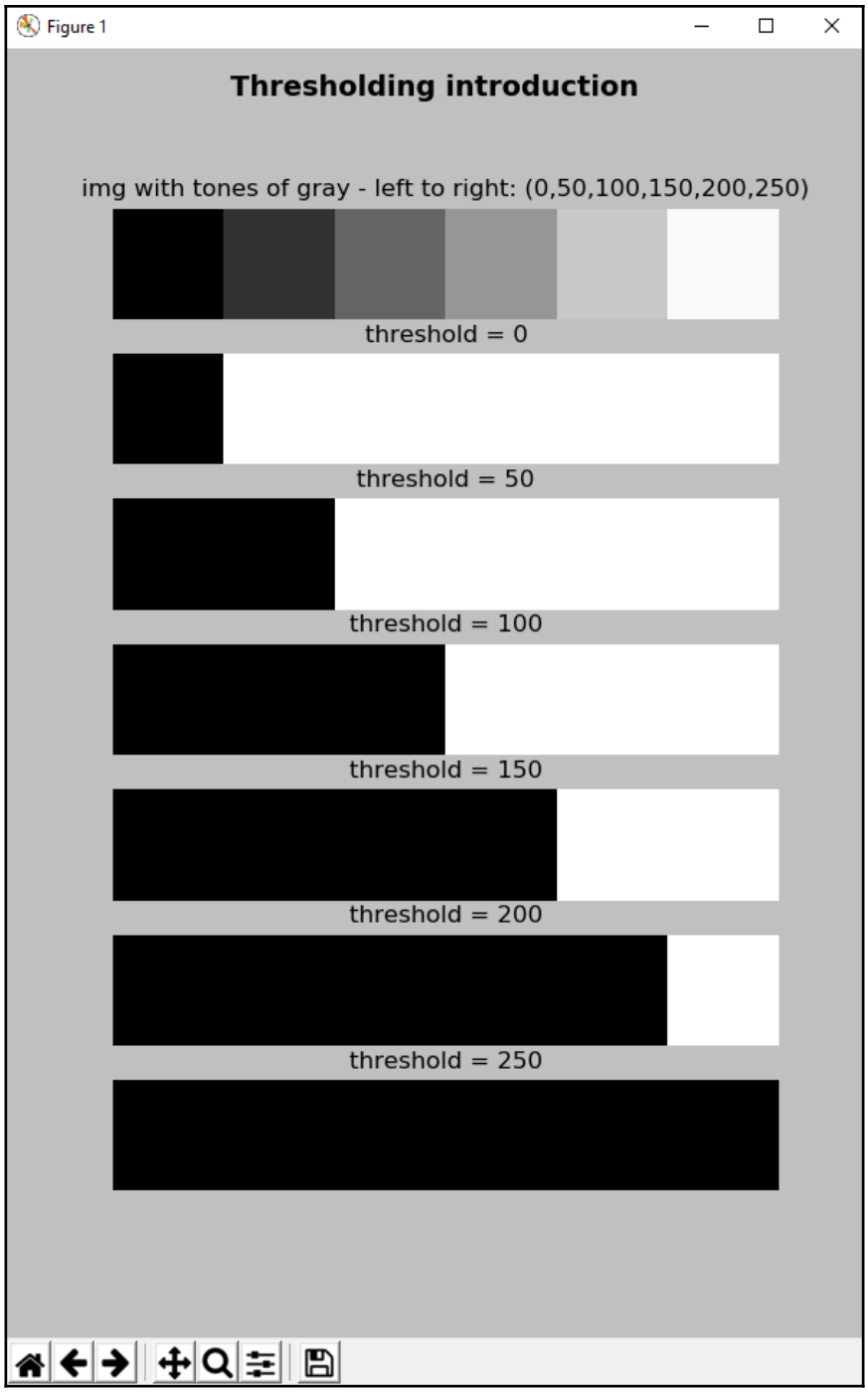


Figure 1
Thresholding introduction
img with tones of gray - left to right: (0,50,100,150,200,250)
threshold = 0
threshold = 50
threshold = 100
threshold = 150
threshold = 200
threshold = 250

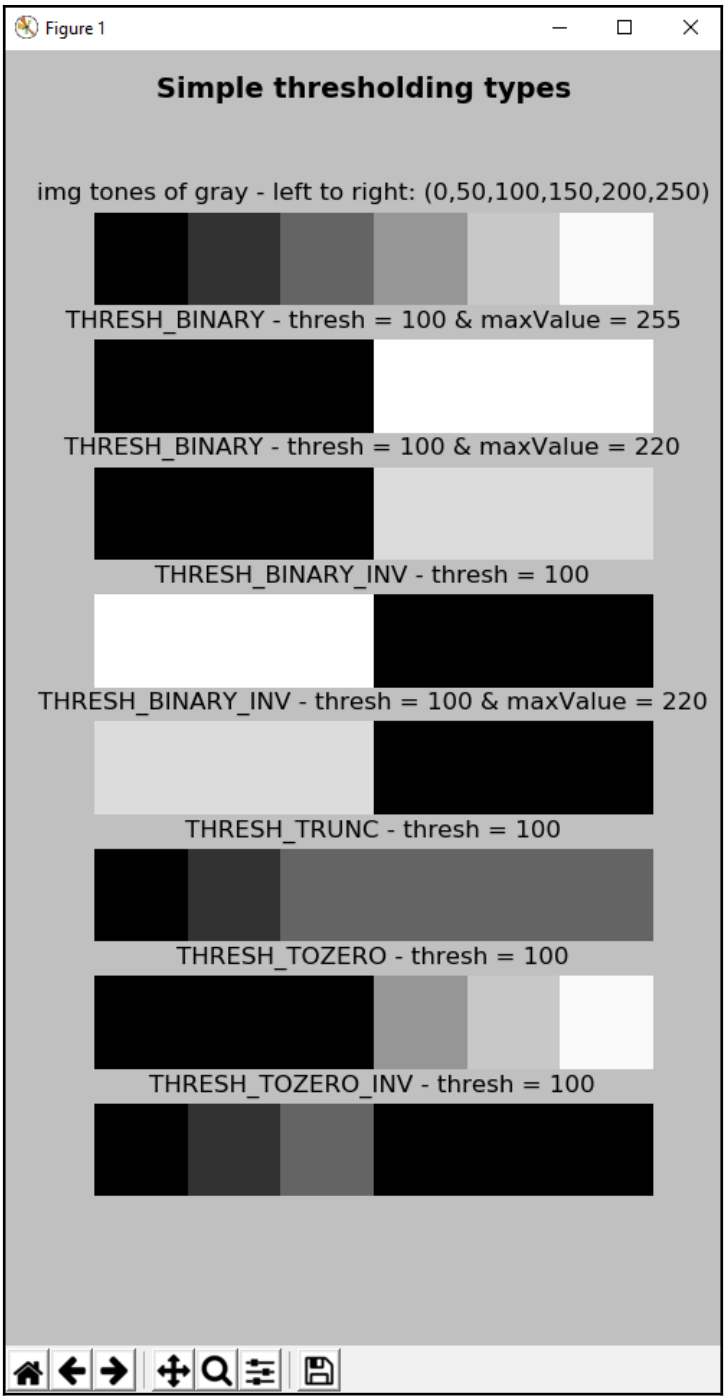


Figure 1
Simple thresholding types
img tones of gray - left to right: (0,50,100,150,200,250)
THRESH_BINARY - thresh = 100 & maxValue = 255
THRESH_BINARY - thresh = 100 & maxValue = 220
THRESH_BINARY_INV - thresh = 100
THRESH_BINARY_INV - thresh = 100 & maxValue = 220
THRESH_TRUNC - thresh = 100
THRESH_TOZERO - thresh = 100
THRESH_TOZERO_INV - thresh = 100

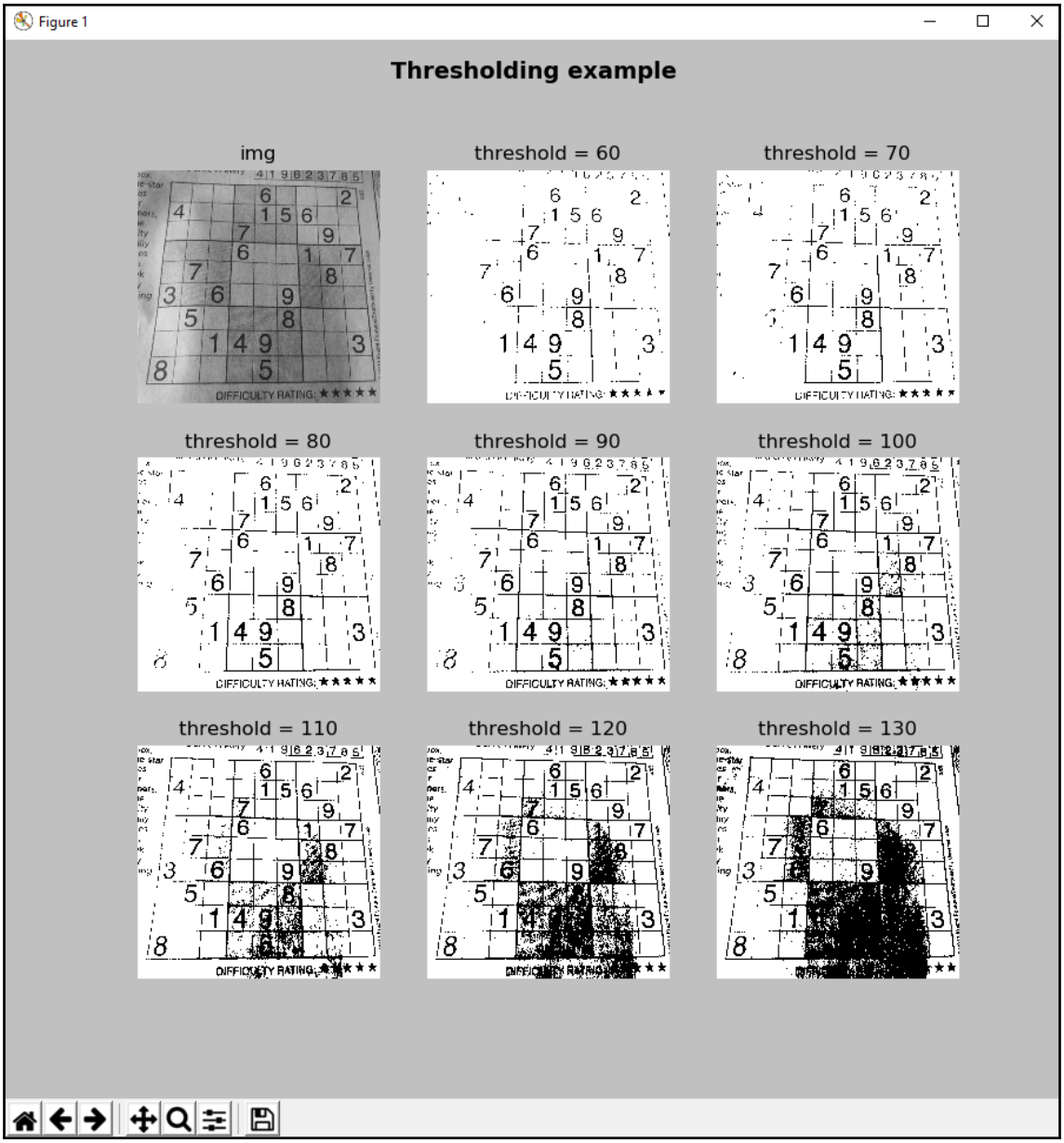


Figure 1
Thresholding example
img
threshold = 60
threshold = 70
threshold = 80
threshold = 90
threshold = 100
threshold = 110
threshold = 120
threshold = 130

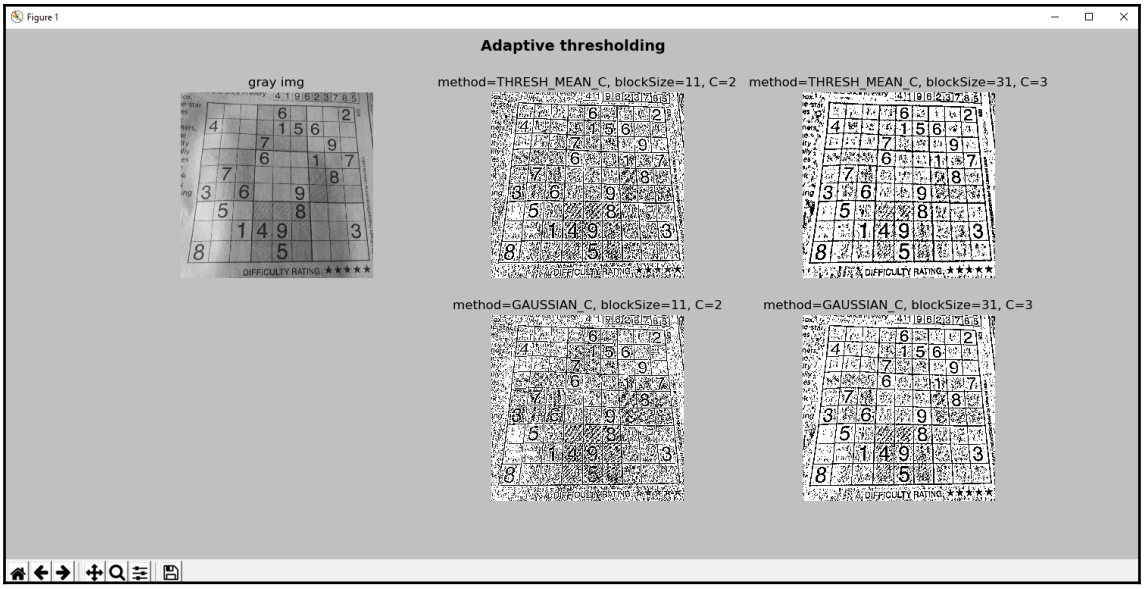


Figure 1
Adaptive thresholding
gray img
method=THRESH_MEAN_C, blockSize=11, C=2
method=THRESH_MEAN_C, blockSize=31, C=3
method=GAUSSIAN_C, blockSize=11, C=2
method=GAUSSIAN_C, blockSize=31, C=3

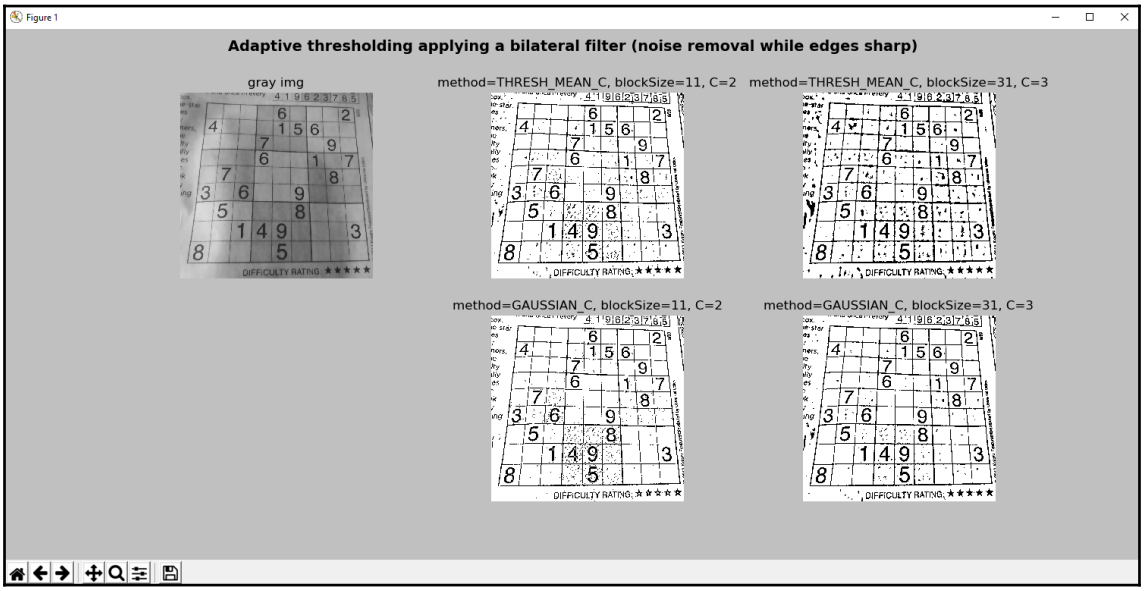


Figure 1
Adaptive thresholding applying a bilateral filter (noise removal while edges sharp)
gray img
method=THRESH_MEAN_C, blockSize=11, C=2
method=THRESH_MEAN_C, blockSize=31, C=3
method=GAUSSIAN_C, blockSize=11, C=2
method=GAUSSIAN_C, blockSize=31, C=3

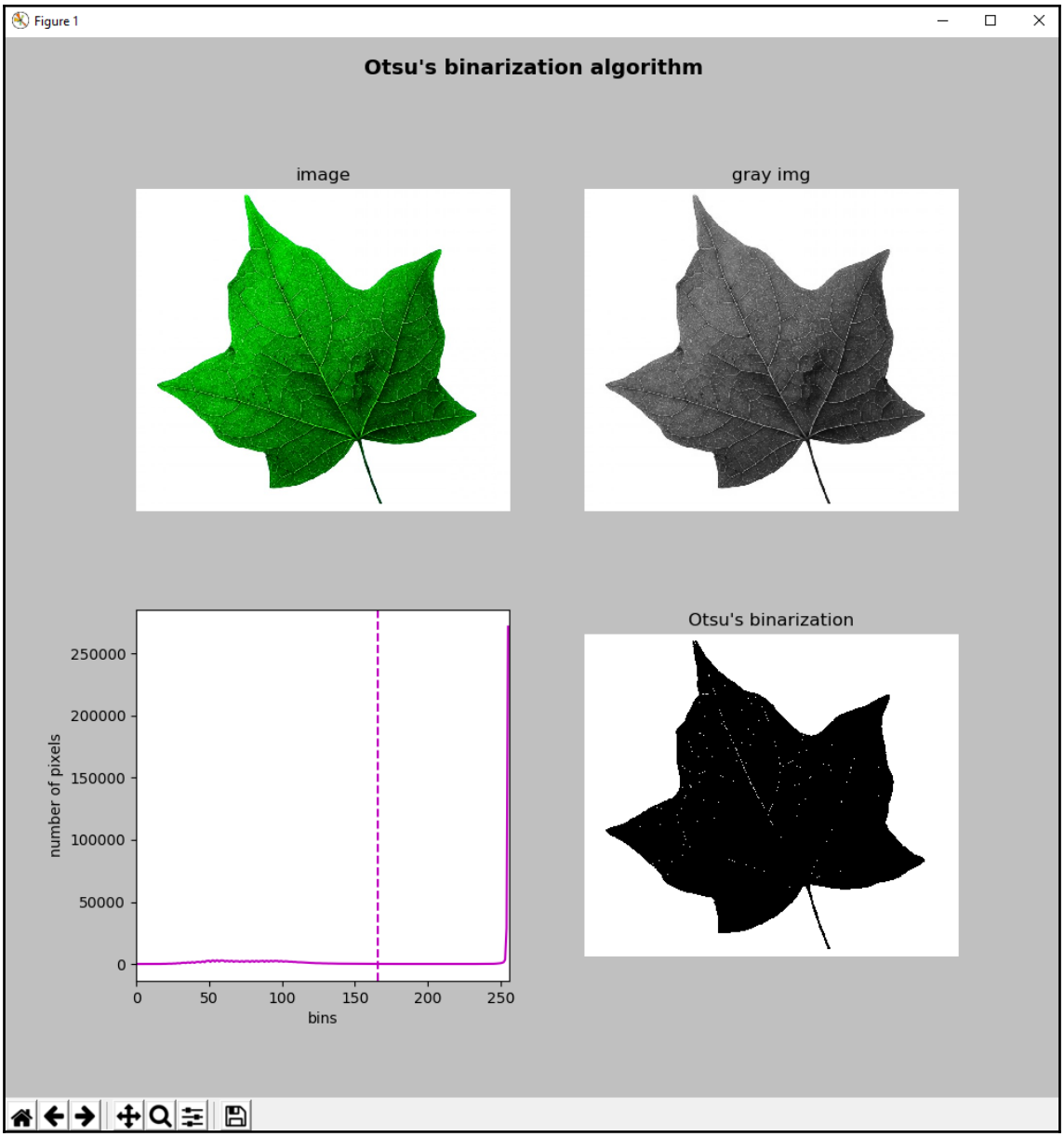


Figure 1
Otsu's binarization algorithm
image
gray img
Otsu's binarization
number of pixels
bins
0
50
100
150
200
250
50000
100000
150000
200000
250000

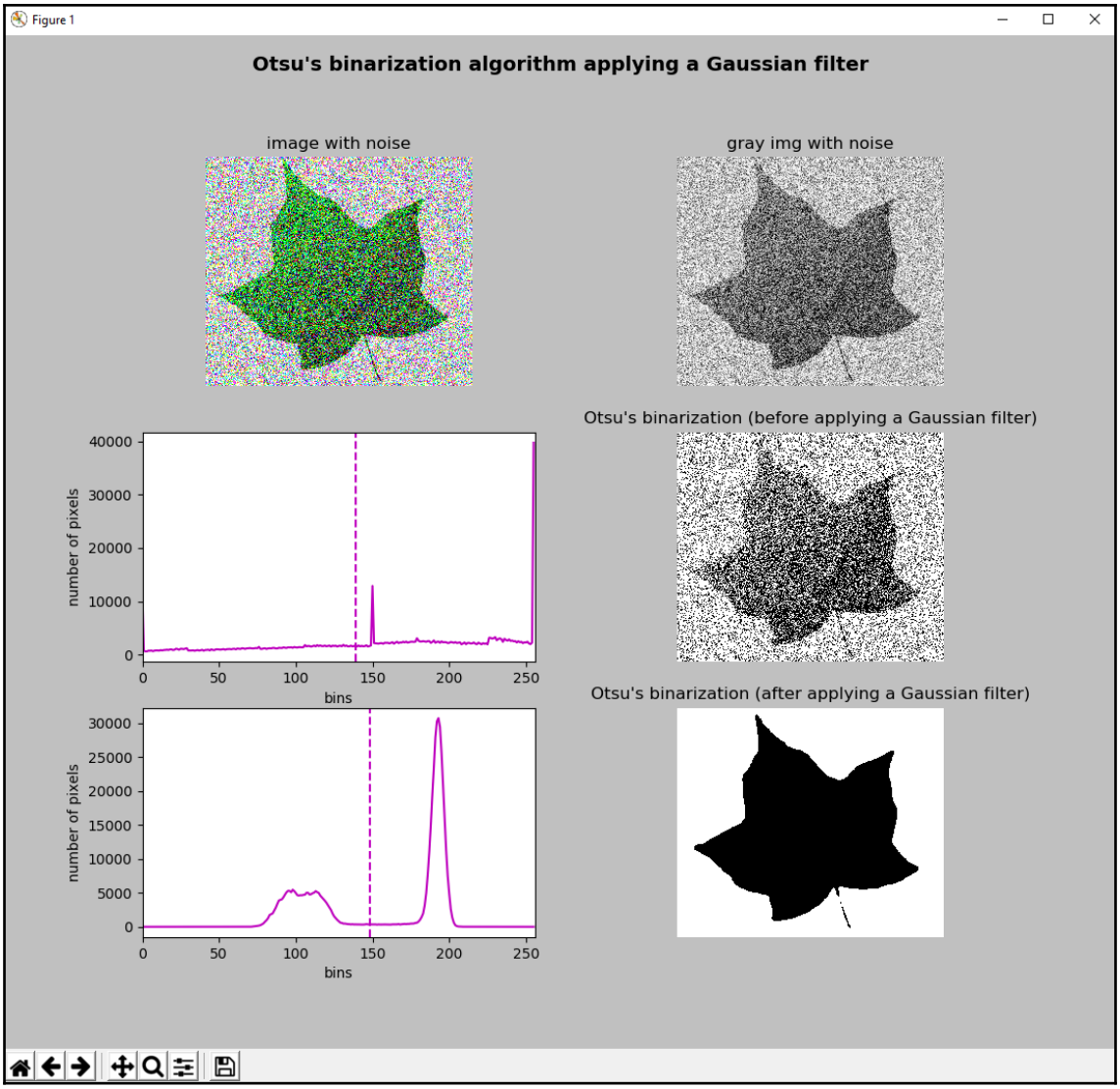


Figure 1
Otsu's binarization algorithm applying a Gaussian filter
image with noise
gray img with noise
Otsu's binarization (before applying a Gaussian filter)
Otsu's binarization (after applying a Gaussian filter)
number of pixels
bins
number of pixels
bins

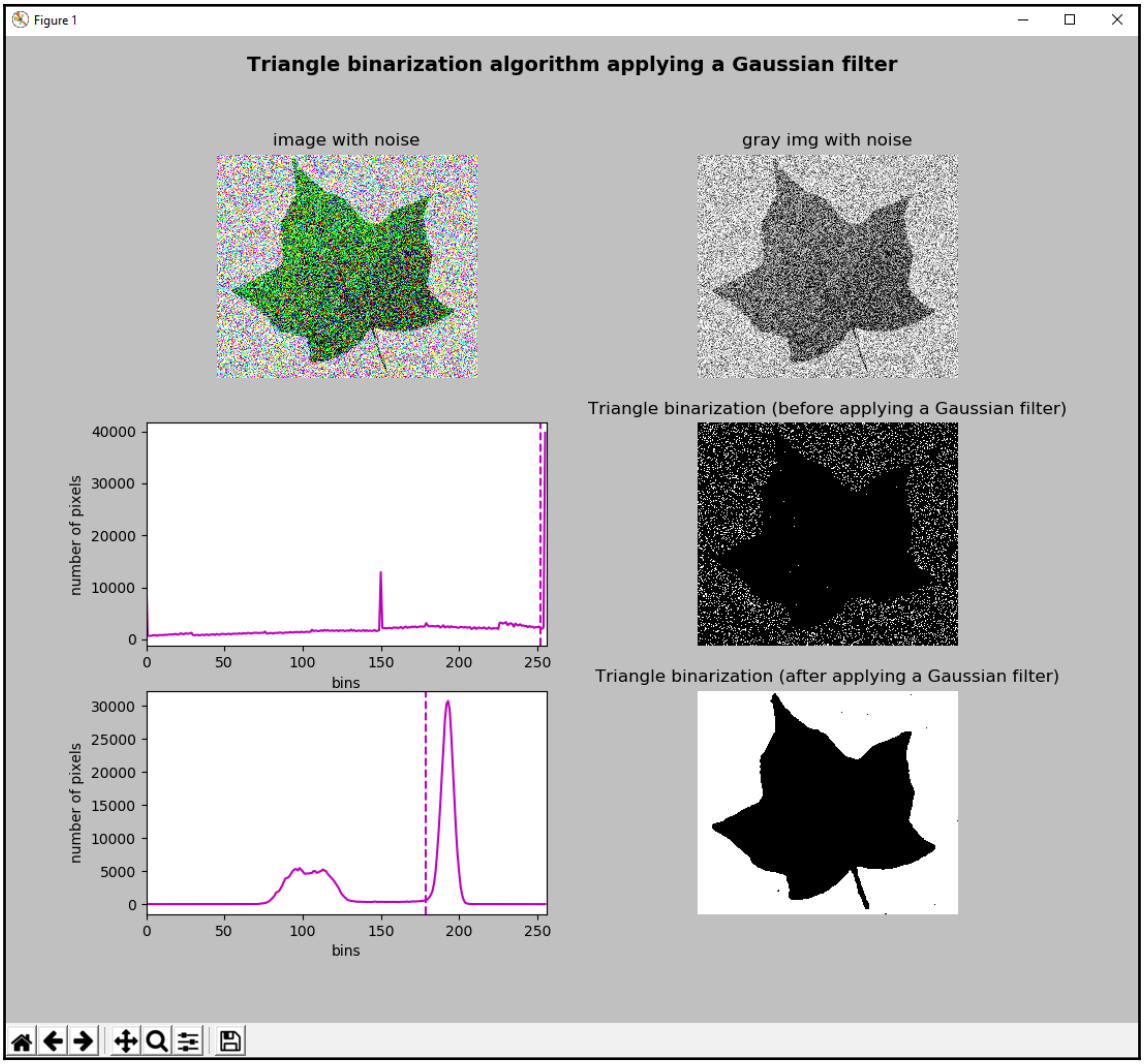


Figure 1
Triangle binarization algorithm applying a Gaussian filter
image with noise
gray img with noise
Triangle binarization (before applying a Gaussian filter)
Triangle binarization (after applying a Gaussian filter)
number of pixels
bins
number of pixels
bins


Figure 1
Thresholding BGR images
image
threshold (150) BGR image
threshold (150) each channel and merge

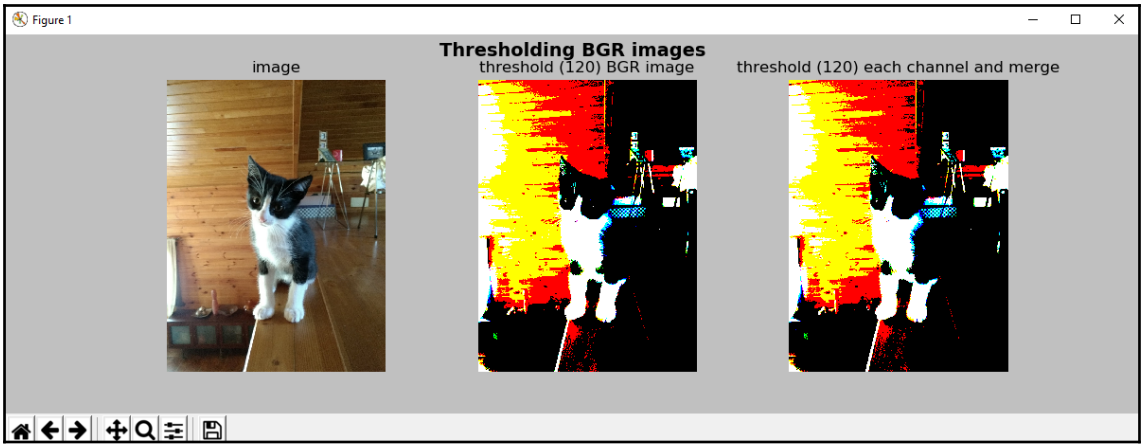


Figure 1
Thresholding BGR images
image
threshold (120) BGR image
threshold (120) each channel and merge

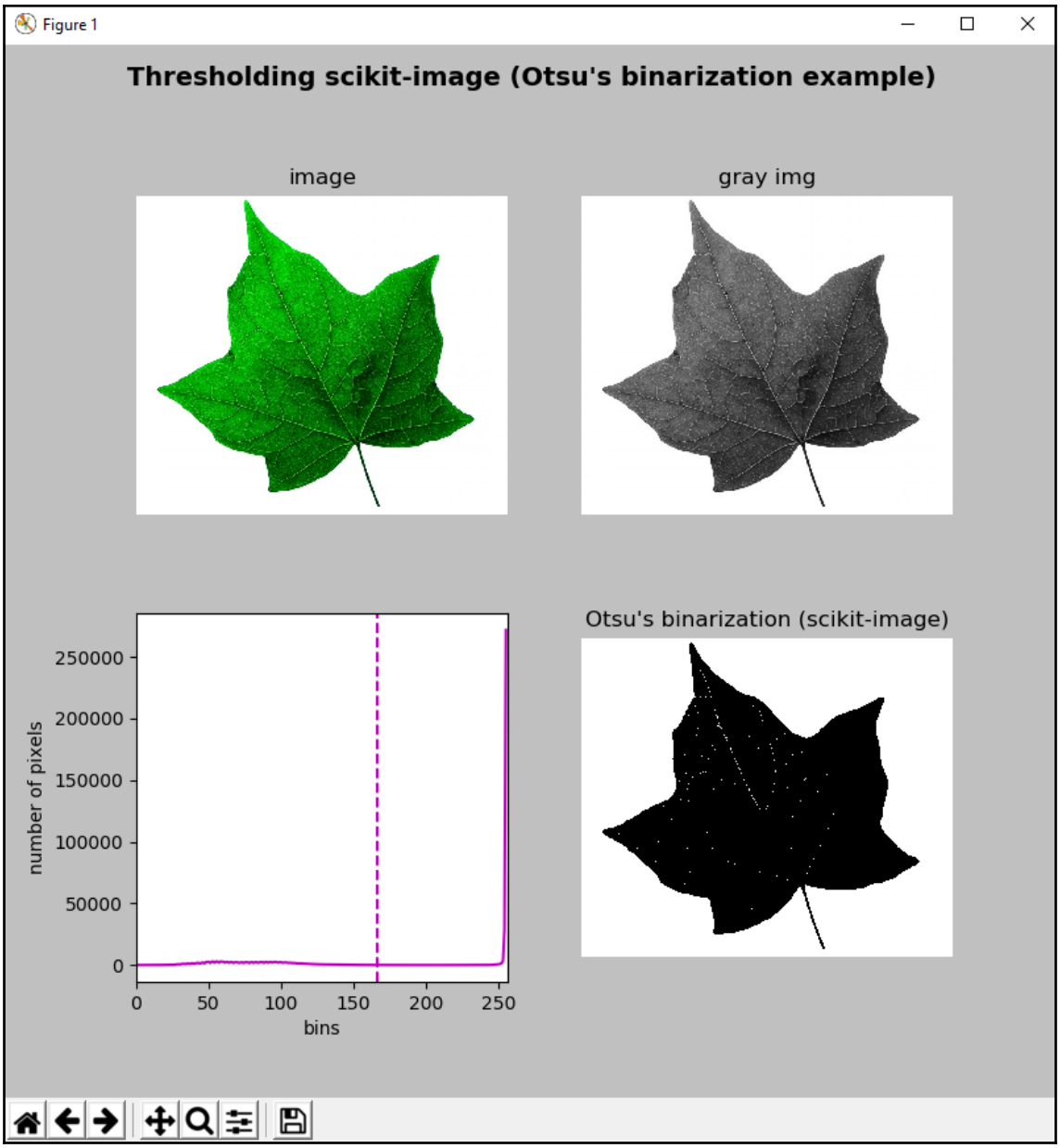


Figure 1
Thresholding scikit-image (Otsu's binarization example)
image
gray img
Otsu's binarization (scikit-image)
number of pixels
250000
200000
150000
100000
50000
0
0
50
100
150
200
250
bins

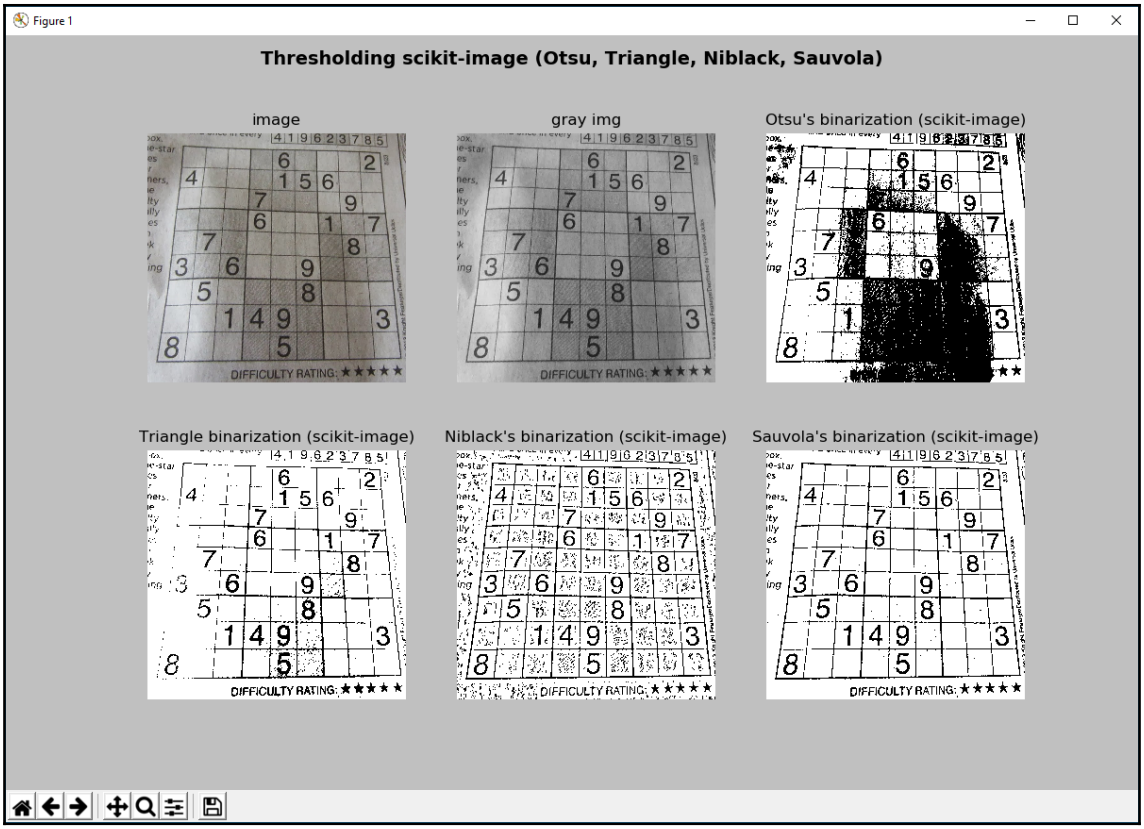
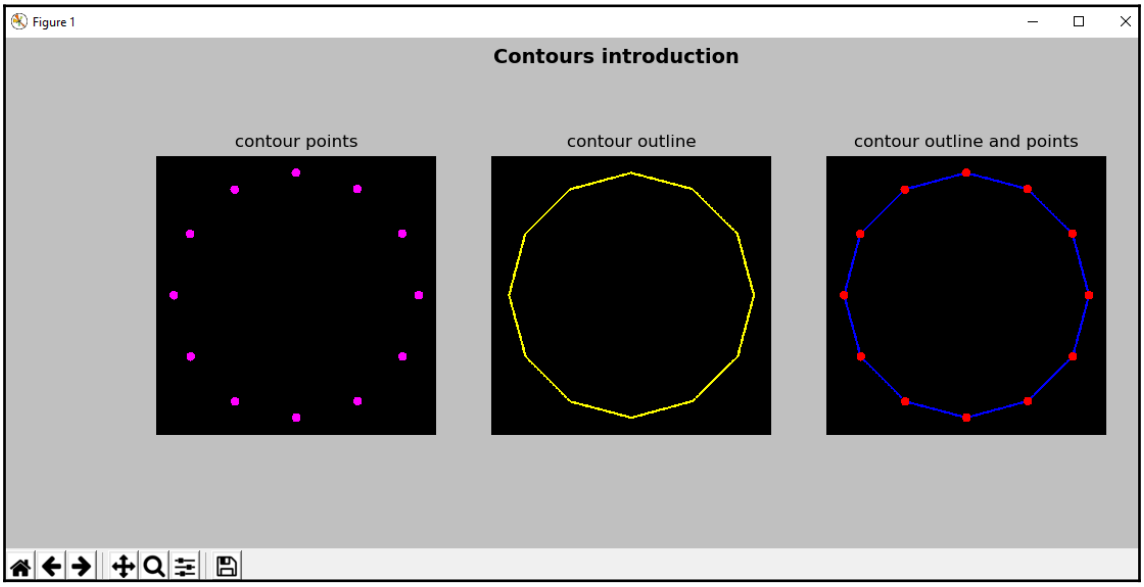
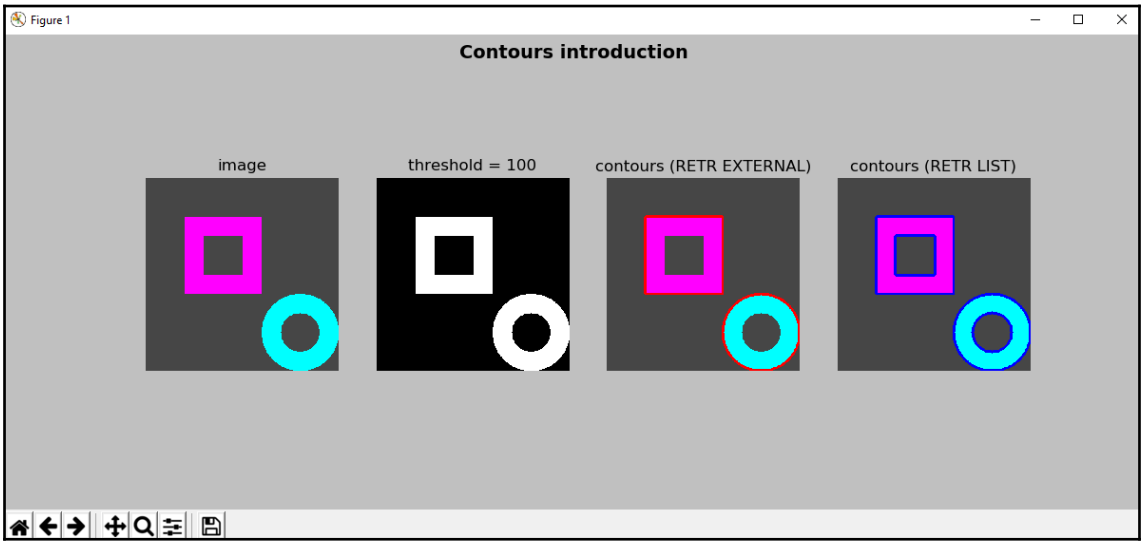


Figure 1
Thresholding scikit-image (Otsu, Triangle, Niblack, Sauvola)
image
gray img
Otsu's binarization (scikit-image)
Triangle binarization (scikit-image)
Niblack's binarization (scikit-image)
Sauvola's binarization (scikit-image)

Chapter 8: Contour Detection, Filtering, and Drawing





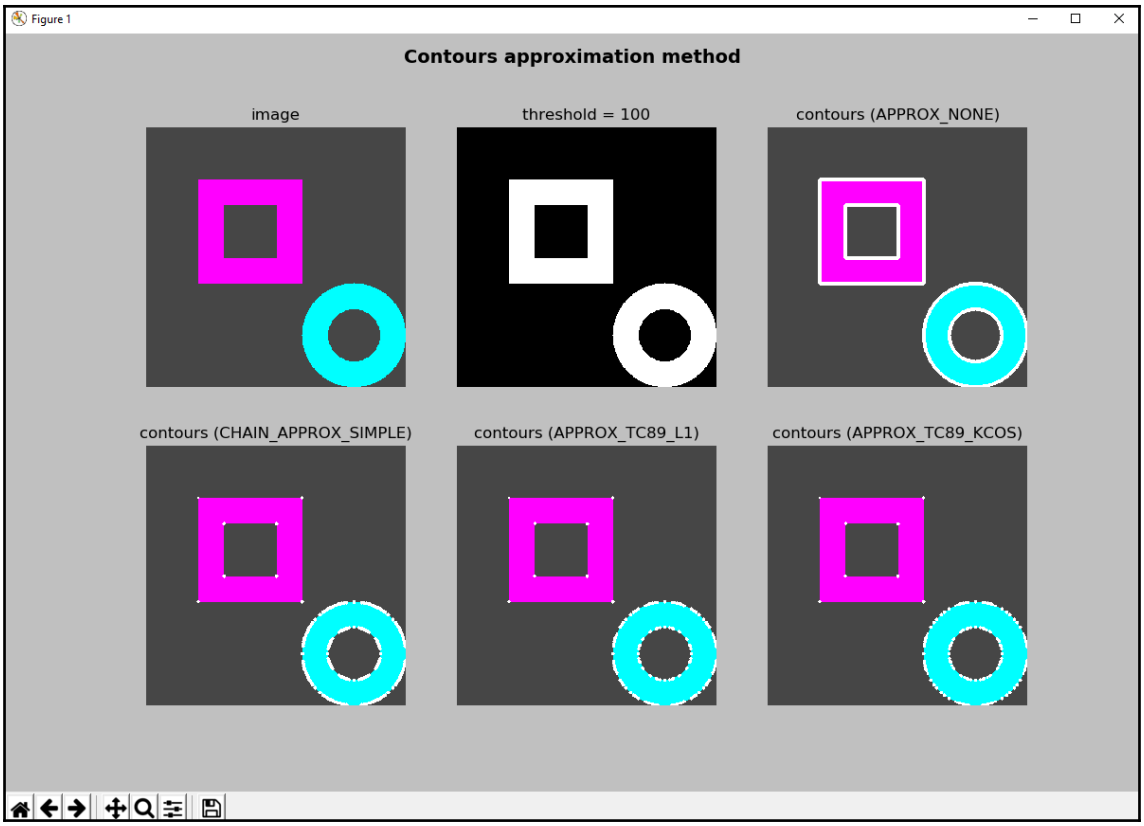


Figure 1
Contours approximation method
image
threshold = 100
contours (APPROX_NONE)
contours (CHAIN_APPROX_SIMPLE)
contours (APPROX_TC89_L1)
contours (APPROX_TC89_KCOS)

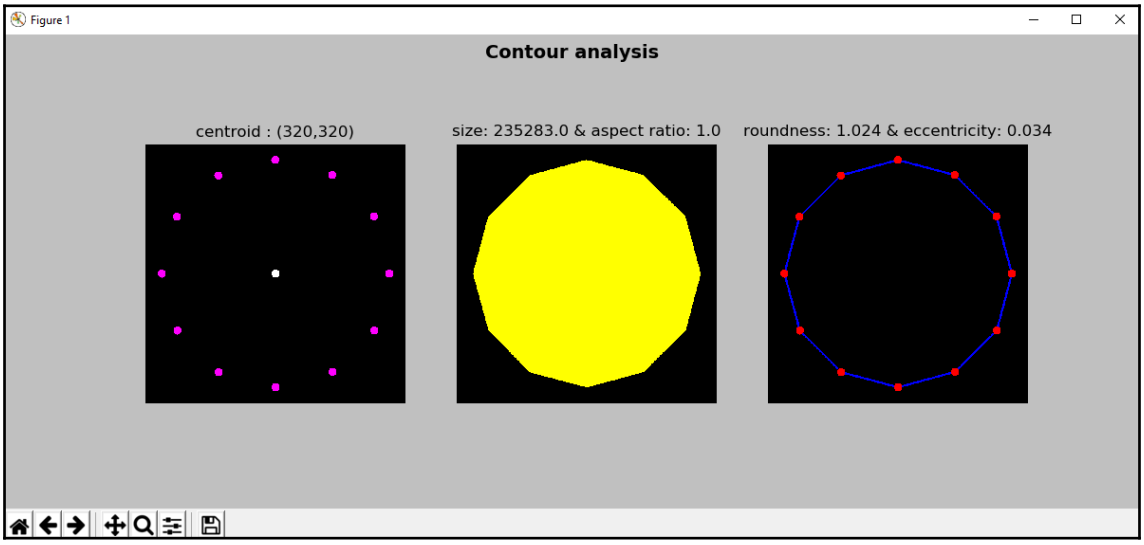


Figure 1
Contour analysis
centroid : (320,320)
size: 235283.0 & aspect ratio: 1.0
roundness: 1.024 & eccentricity: 0.034

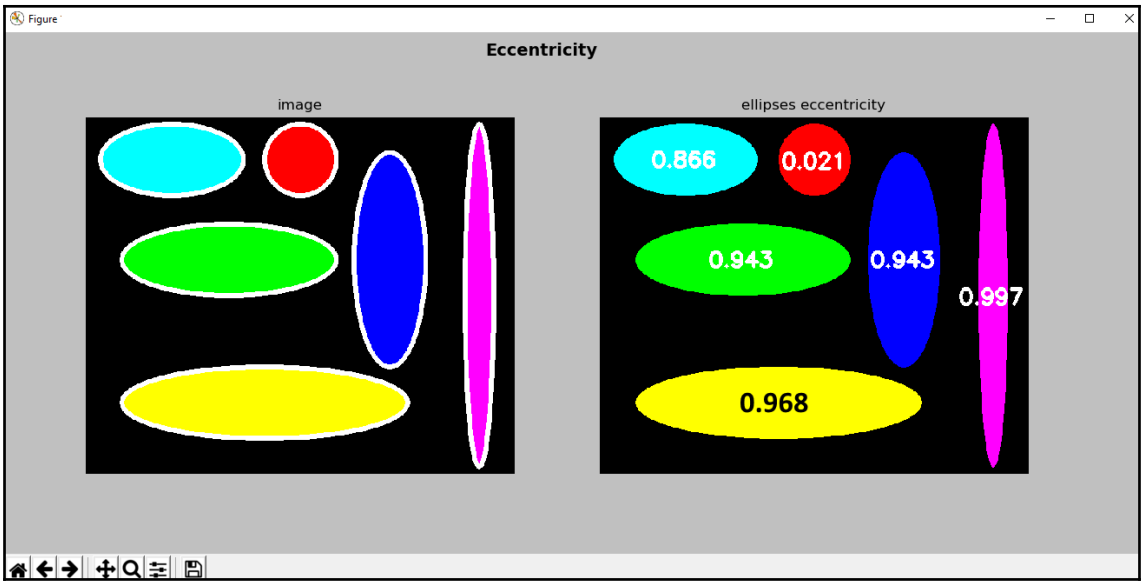


Figure
Eccentricity
image
ellipses eccentricity
0.866
0.021
0.943
0.943
0.997
0.968

$$m_{ji}$$

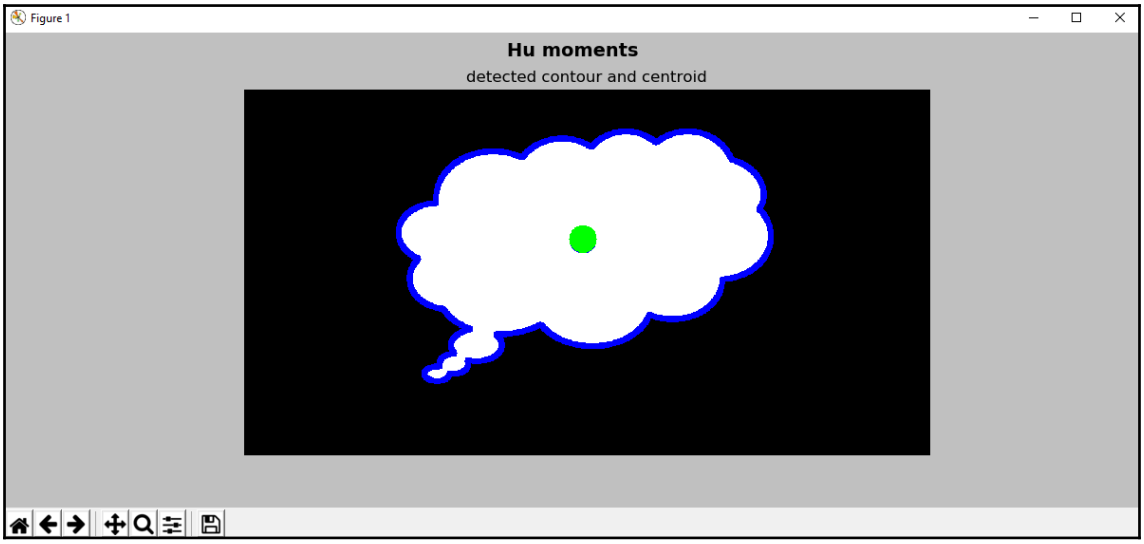


Figure 1
Hu moments
detected contour and centroid

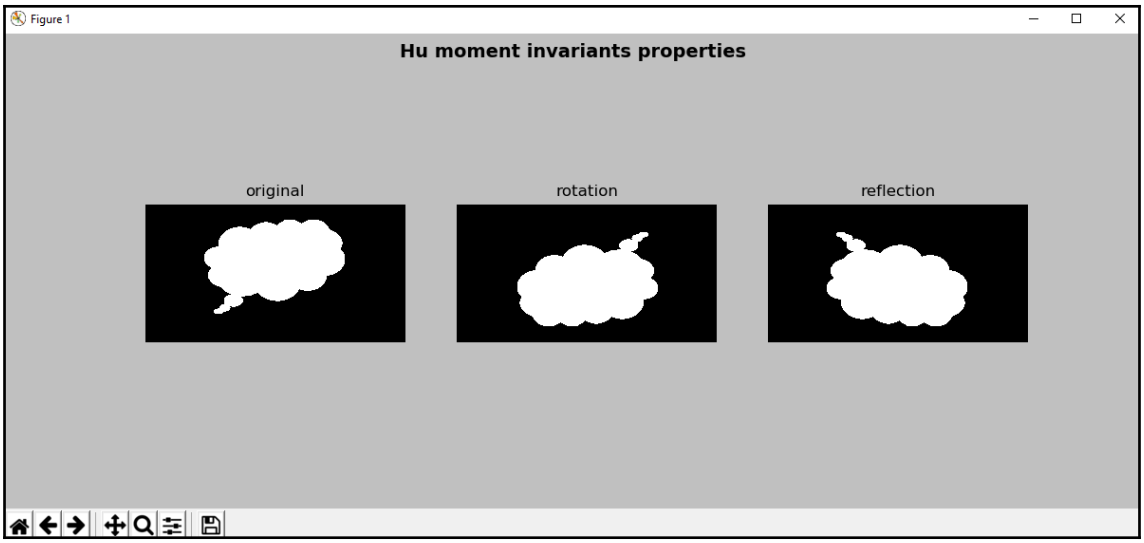


Figure 1
Hu moment invariants properties
original
rotation
reflection

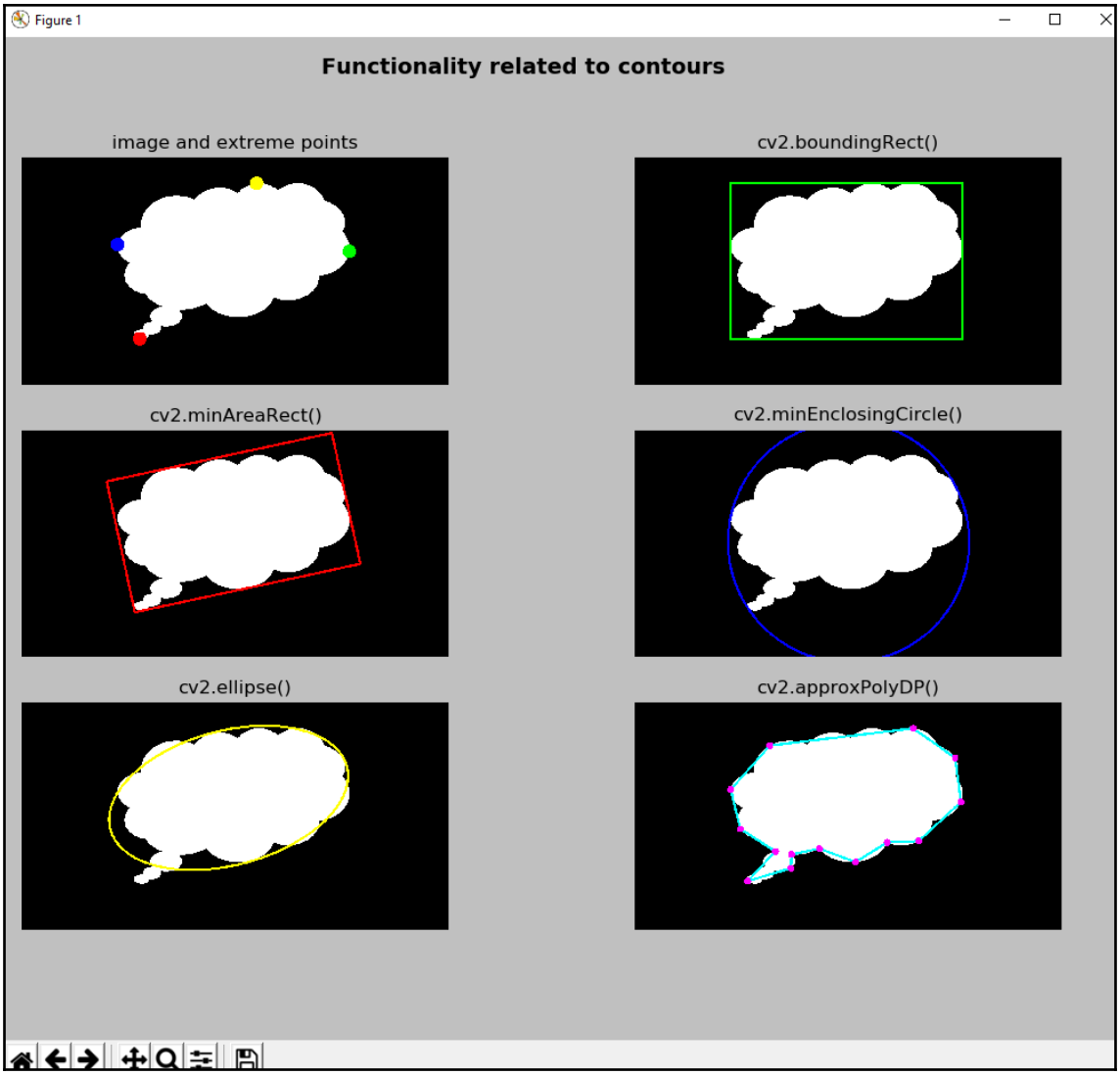


Figure 1
Functionality related to contours
image and extreme points
cv2.boundingRect()
cv2.minAreaRect()
cv2.minEnclosingCircle()
cv2.ellipse()
cv2.approxPolyDP()

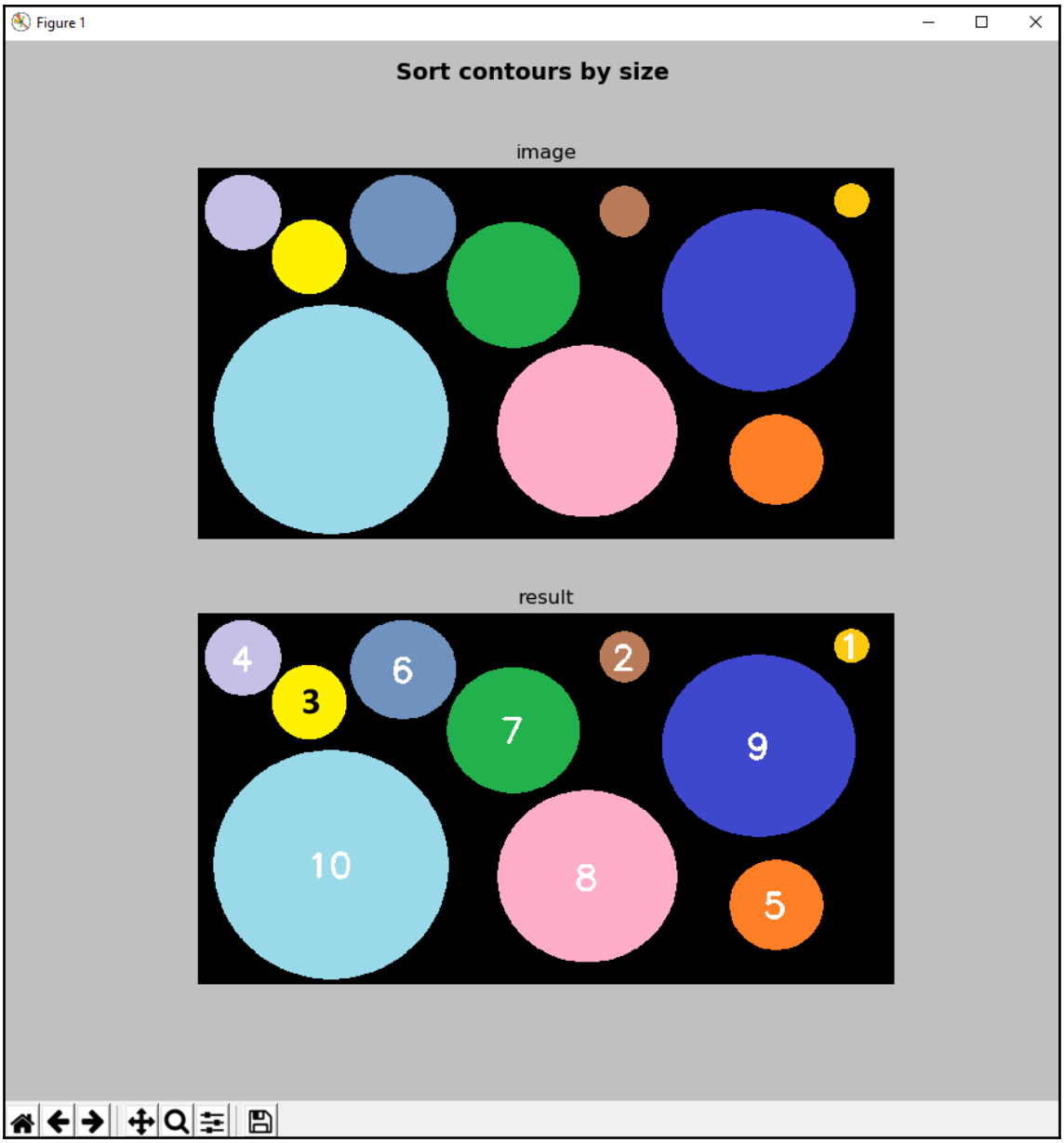


Figure 1
Sort contours by size
image
result
4
3
6
7
2
1
9
10
8
5

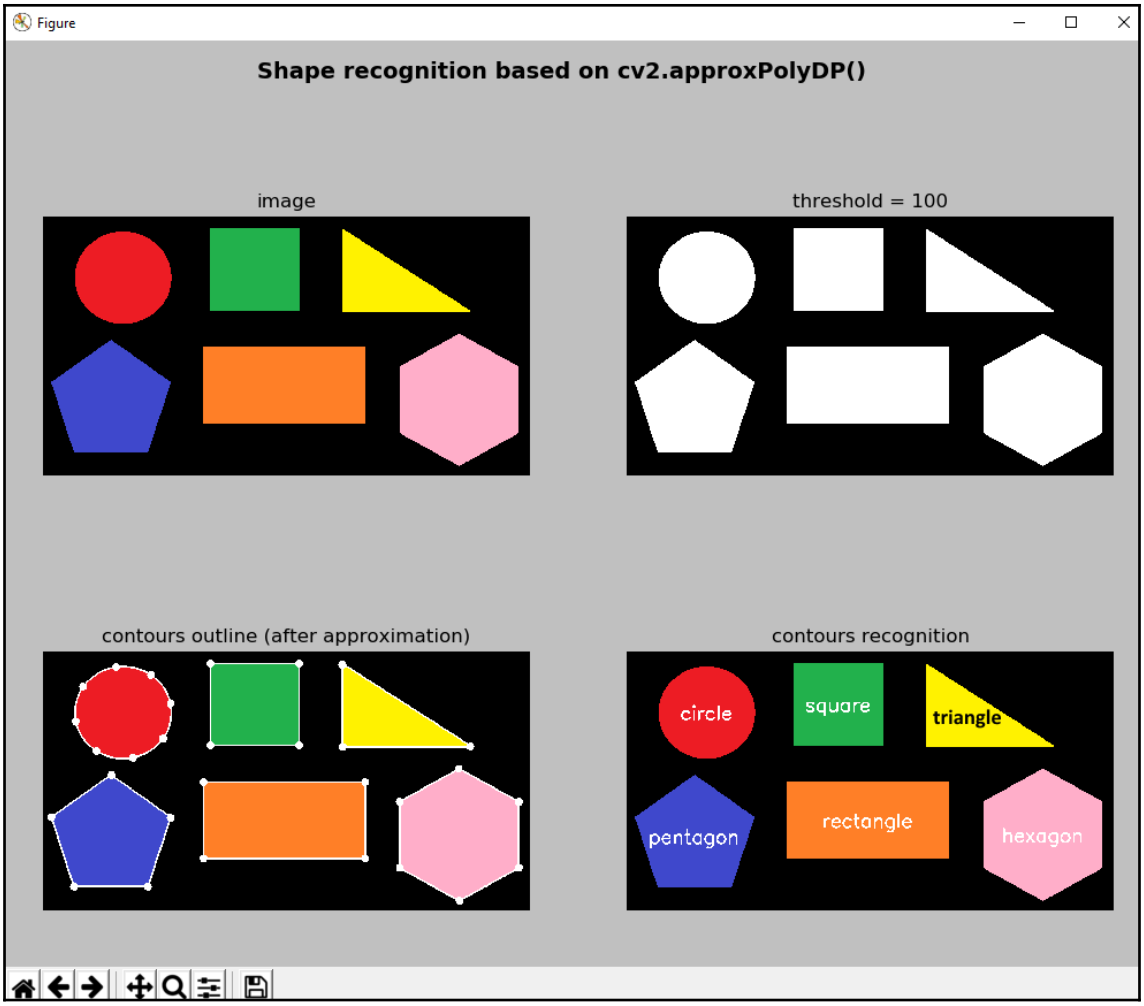


Figure
Shape recognition based on cv2.approxPolyDP()
image
threshold = 100
contours outline (after approximation)
contours recognition
circle
square
triangle
pentagon
rectangle
hexagon

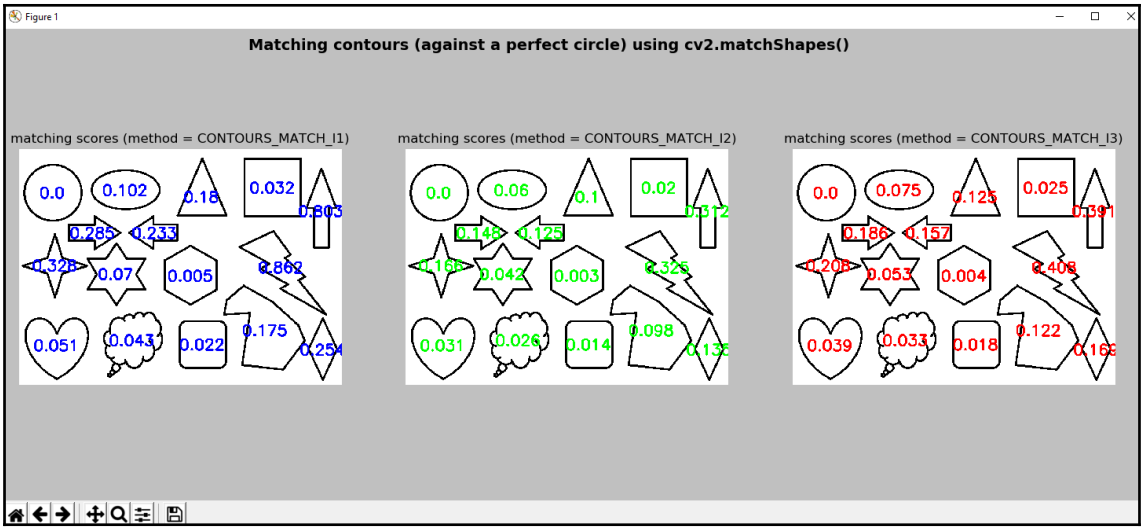
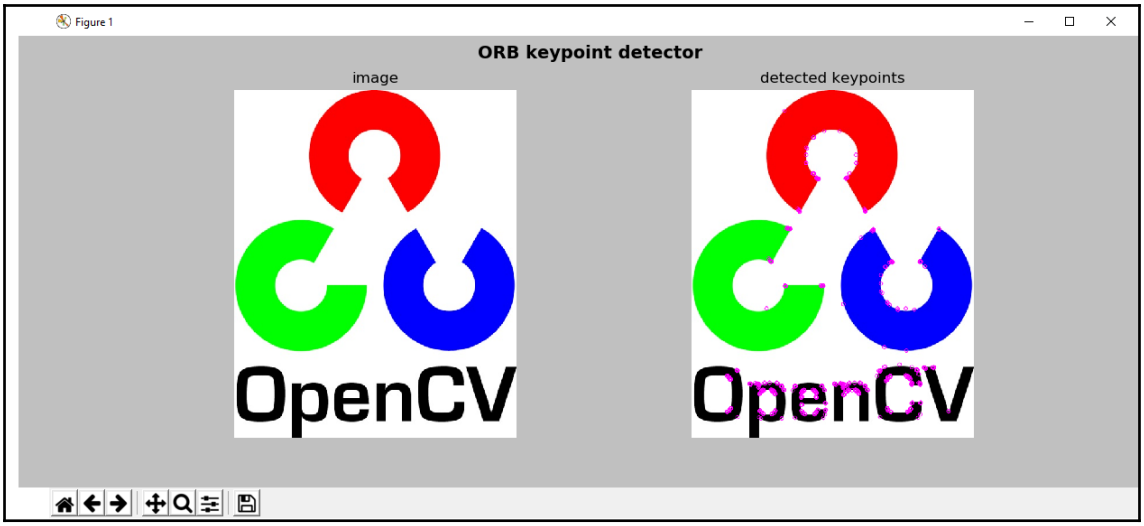


Figure 1
Matching contours (against a perfect circle) using cv2.matchShapes()
matching scores (method = CONTOURS_MATCH_I1)
0.0
0.102
0.18
0.032
0.285
0.233
0.07
0.005
0.051
0.043
0.022
0.175
matching scores (method = CONTOURS_MATCH_I2)
0.0
0.06
0.1
0.02
0.148
0.125
0.042
0.003
0.031
0.026
0.014
0.098
matching scores (method = CONTOURS_MATCH_I3)
0.0
0.075
0.125
0.025
0.186
0.157
0.053
0.004
0.039
0.033
0.018
0.122

Chapter 9: Augmented Reality



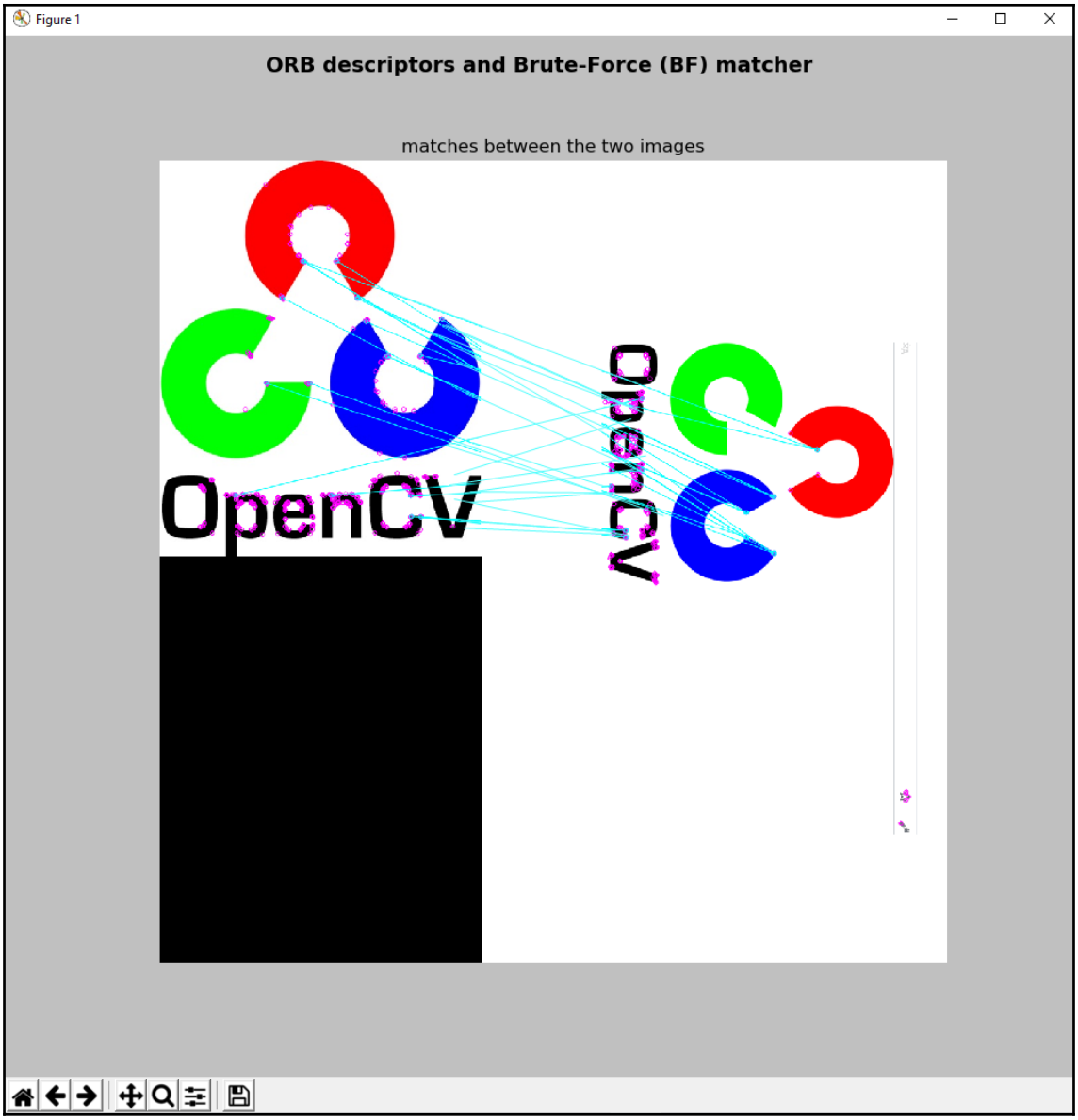


Figure 1
ORB descriptors and Brute-Force (BF) matcher
matches between the two images
OpenCV
OpenCV

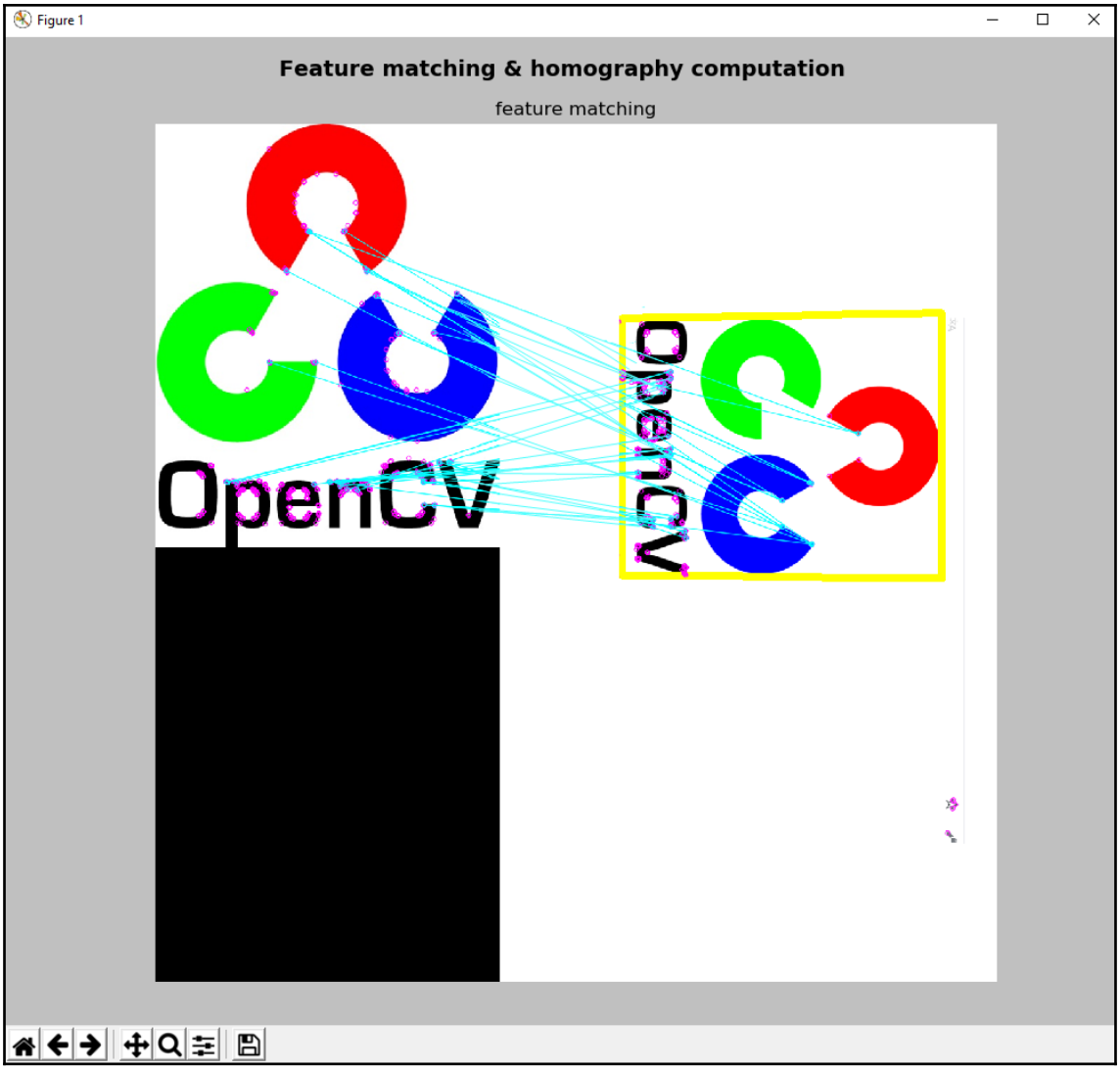


Figure 1
Feature matching & homography computation
feature matching
OpenCV
OpenCV

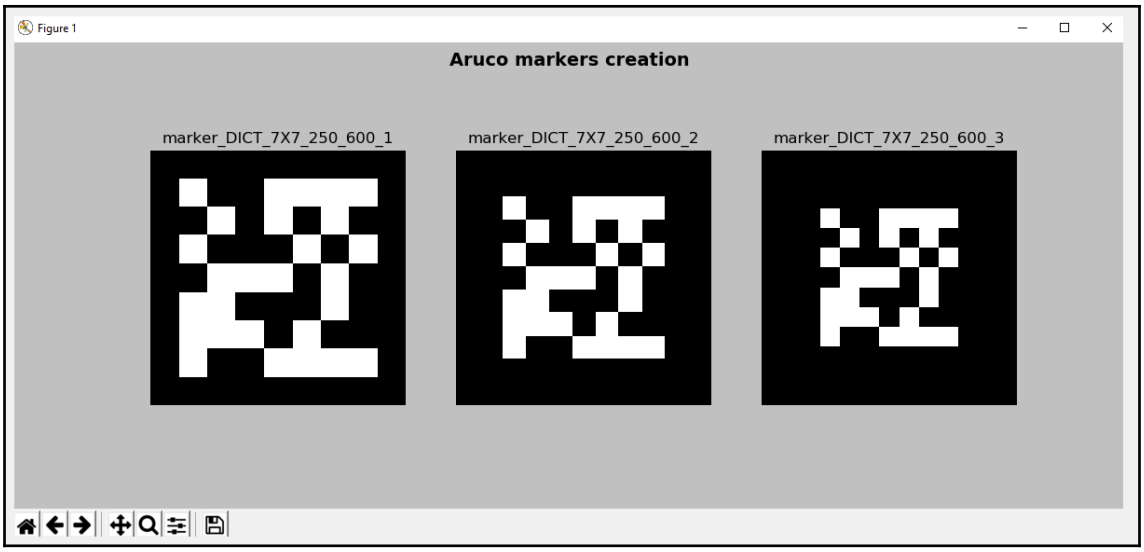
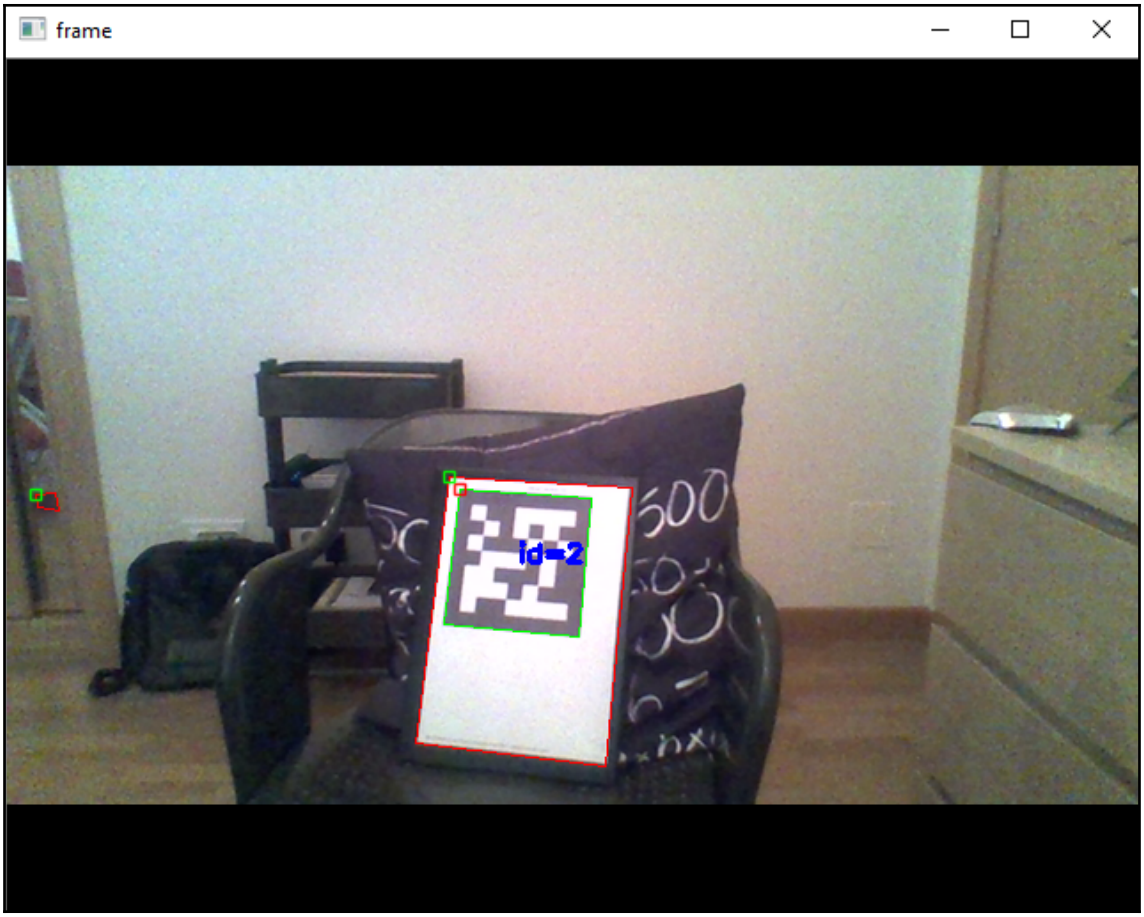
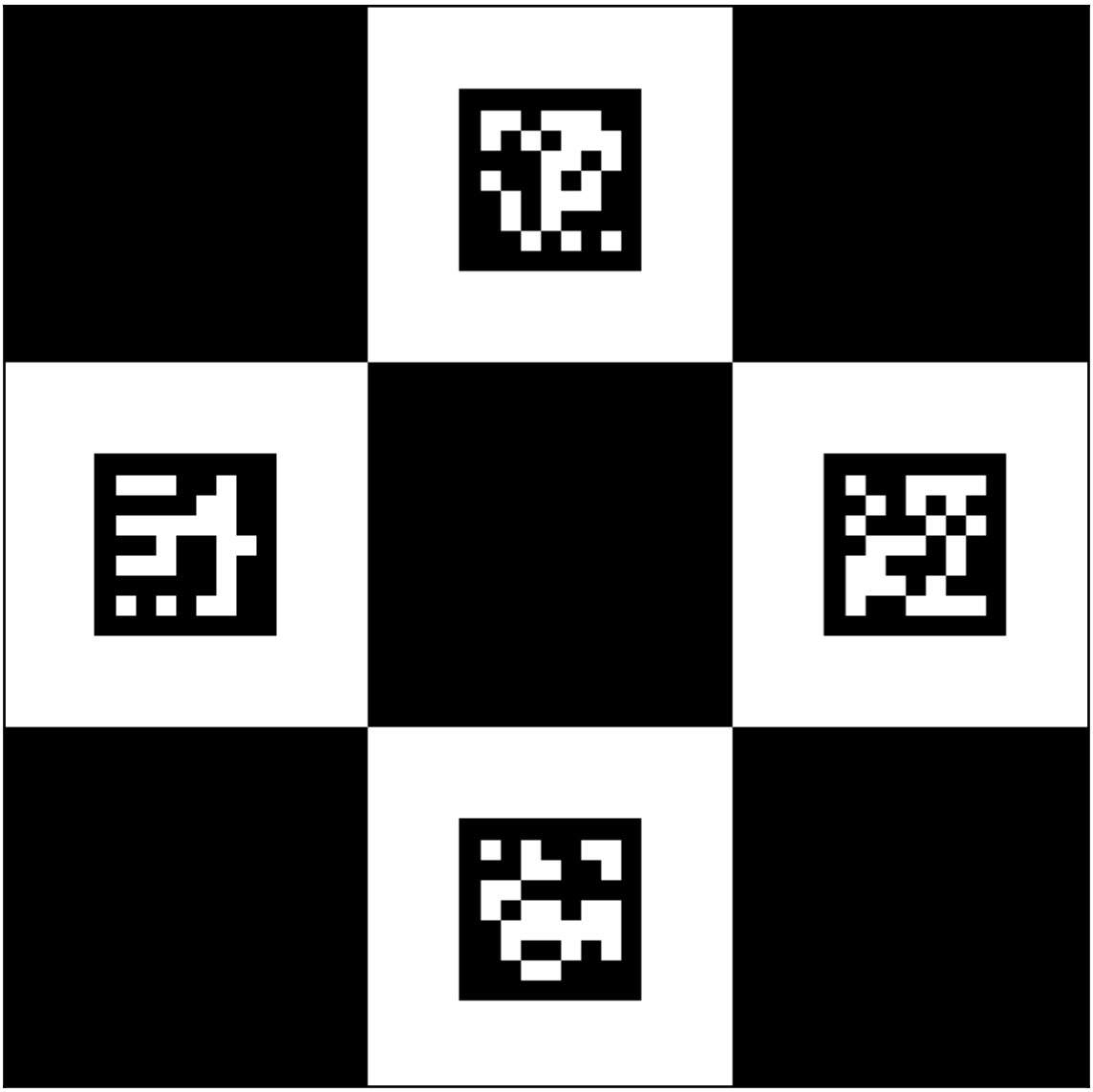
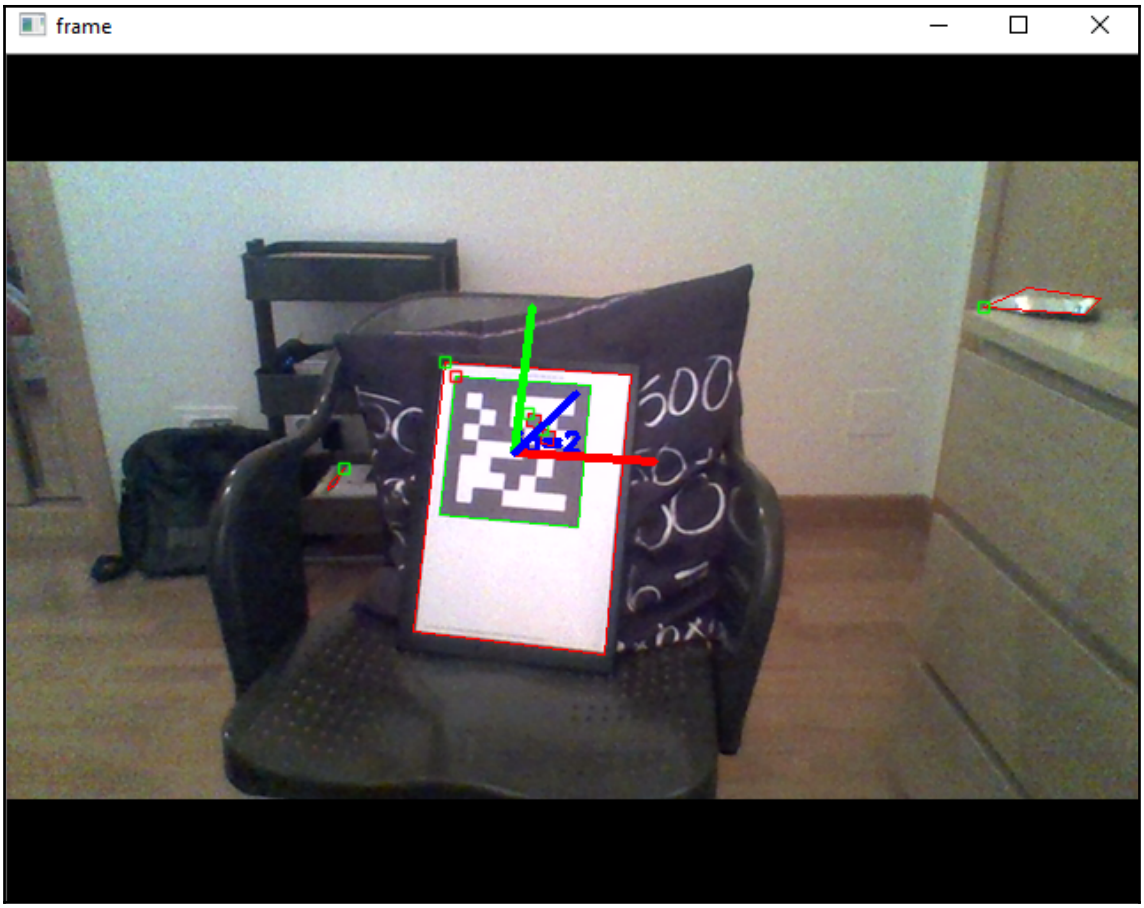
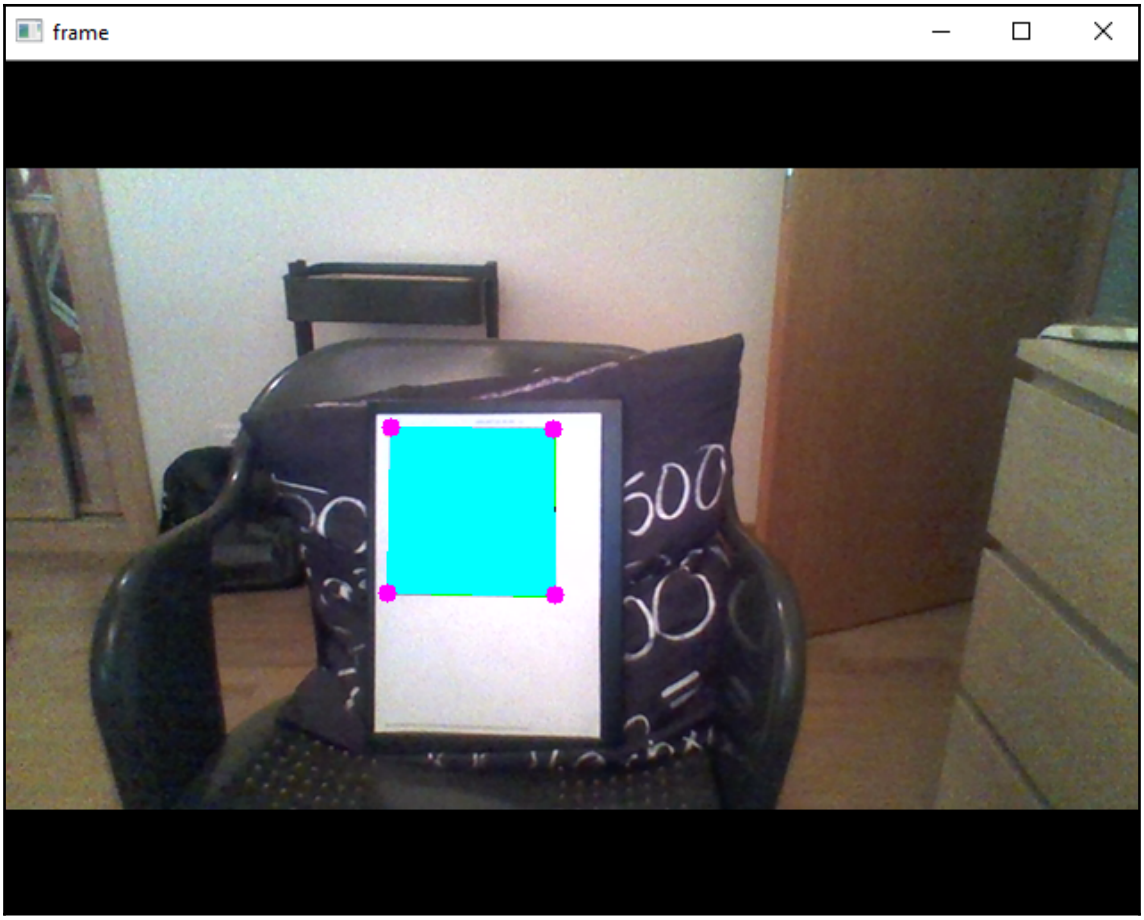


Figure 1
Aruco markers creation
marker_DICT_7X7_250_600_1
marker_DICT_7X7_250_600_2
marker_DICT_7X7_250_600_3

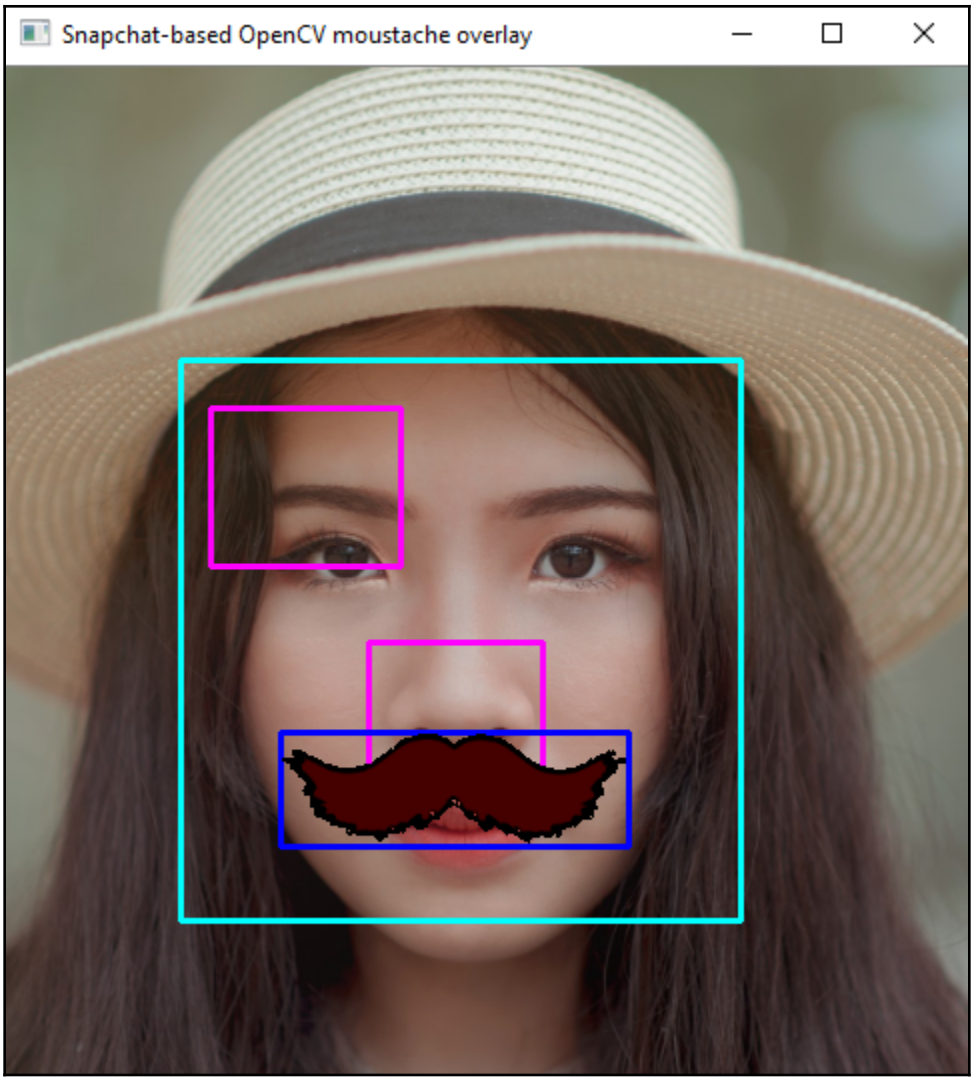

frame
id=2

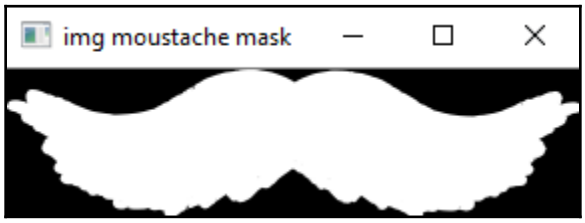



frame

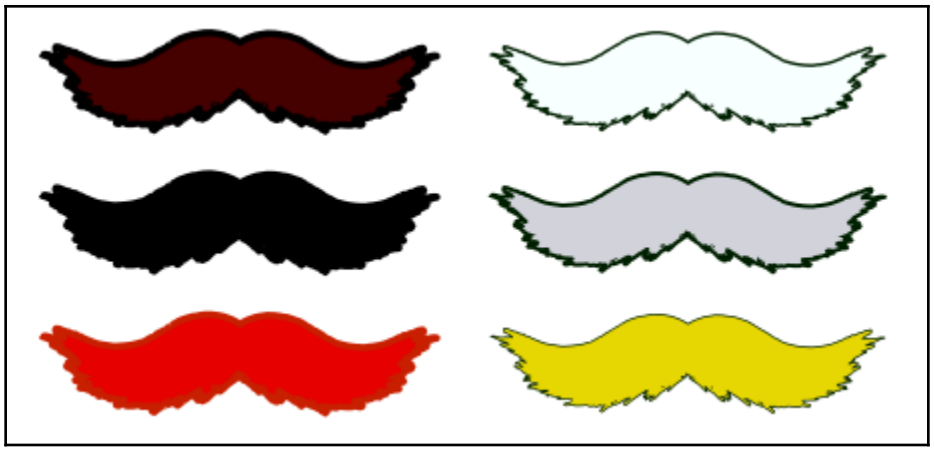

frame

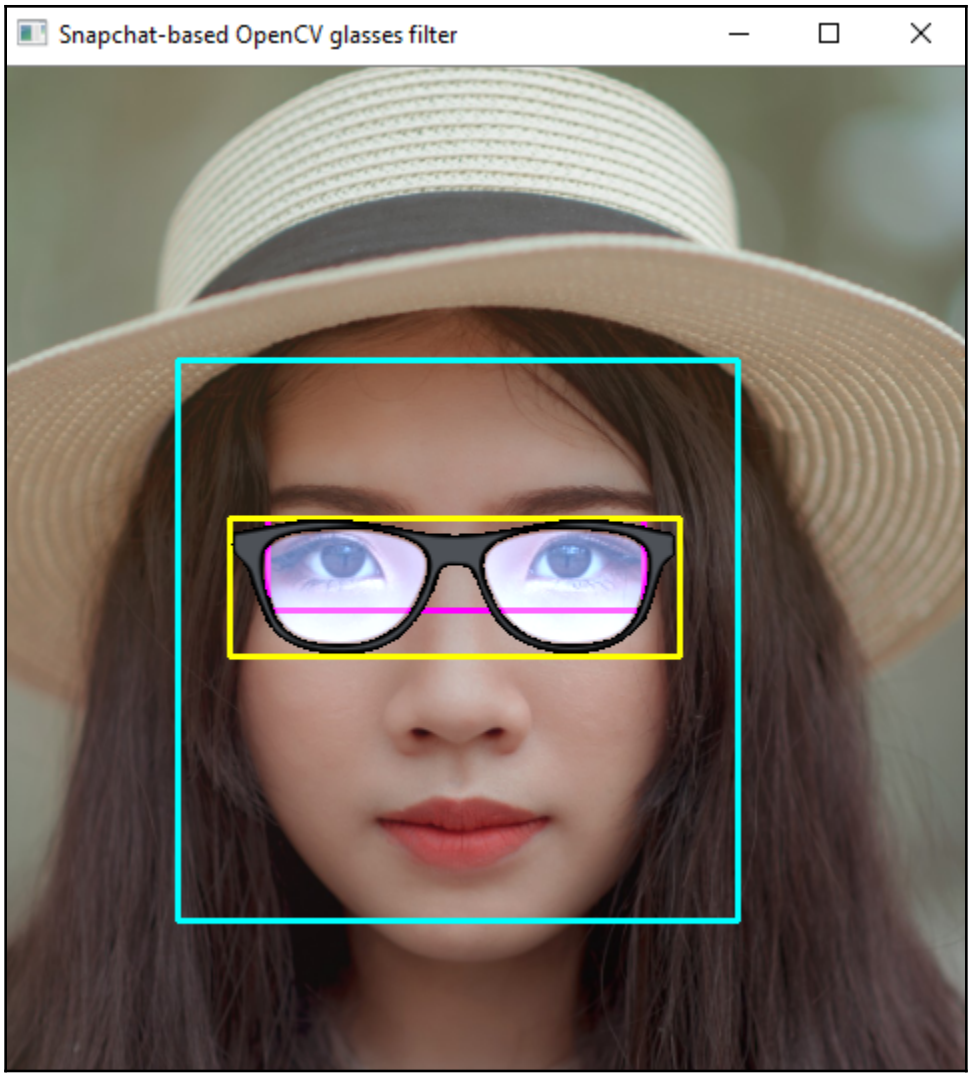

frame


Snapchat-based OpenCV moustache overlay


img moustache mask

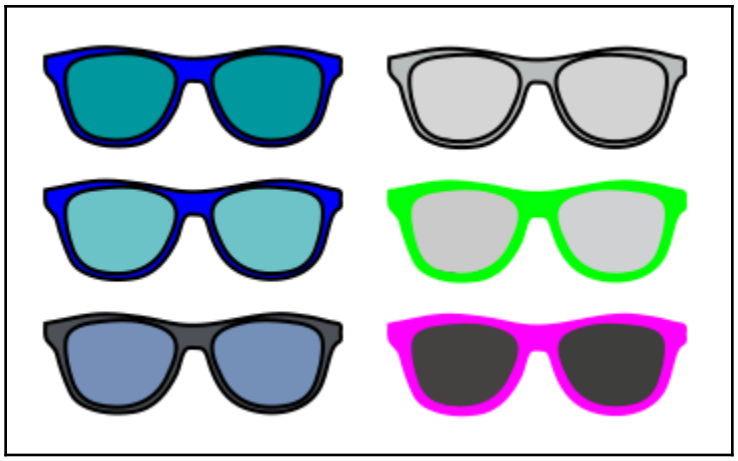

Snapchat-based OpenCV moustache overlay




Snapchat-based OpenCV glasses filter


img gl...


Snapchat-based OpenCV glasses filter



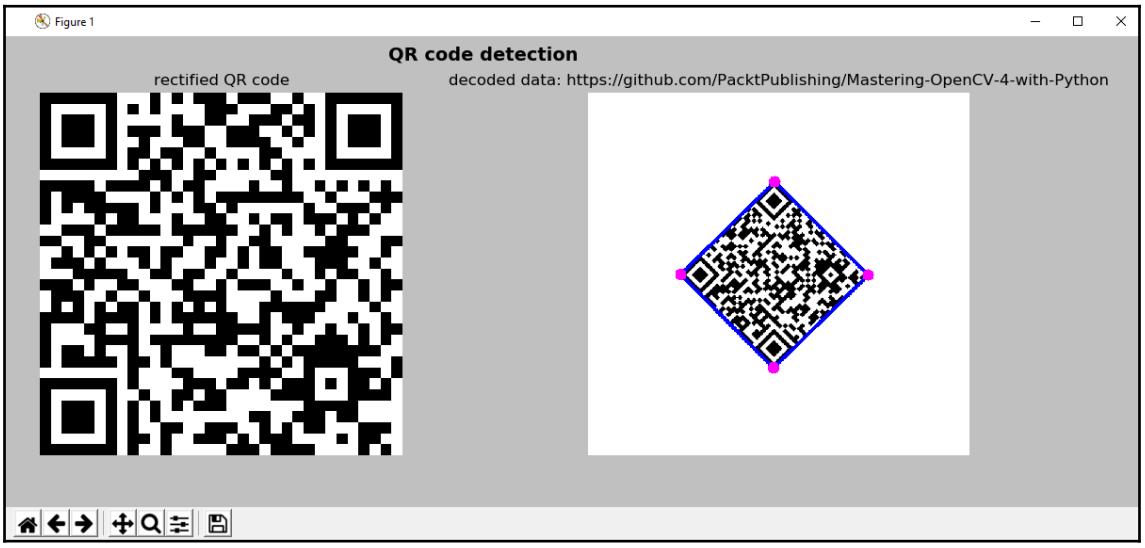
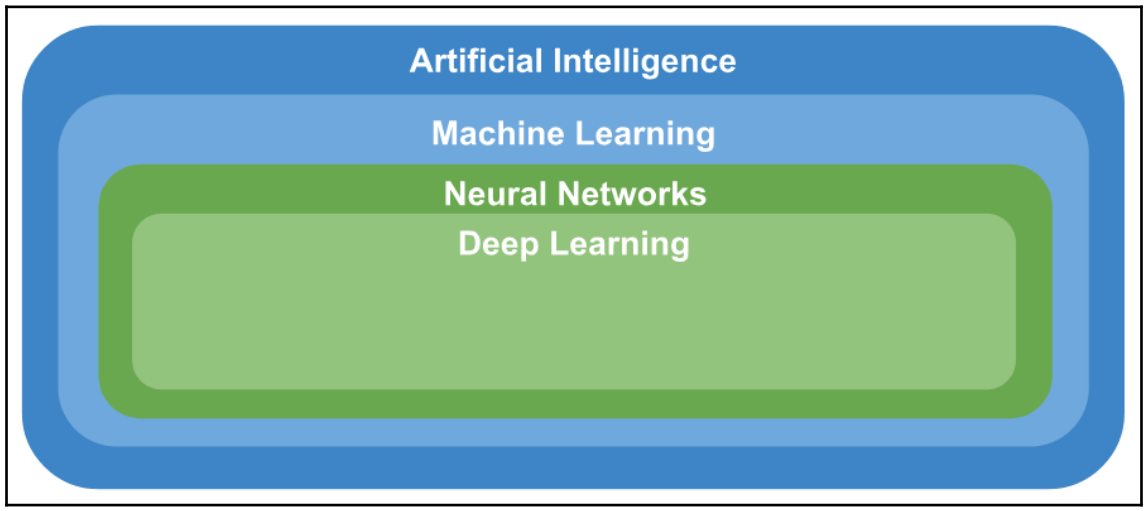
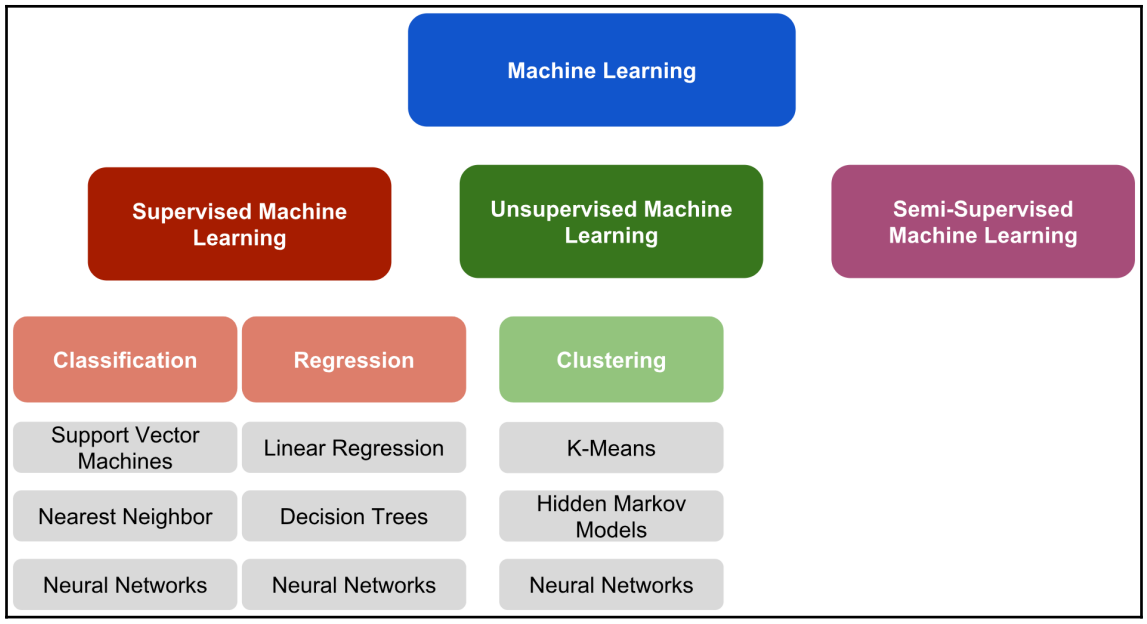
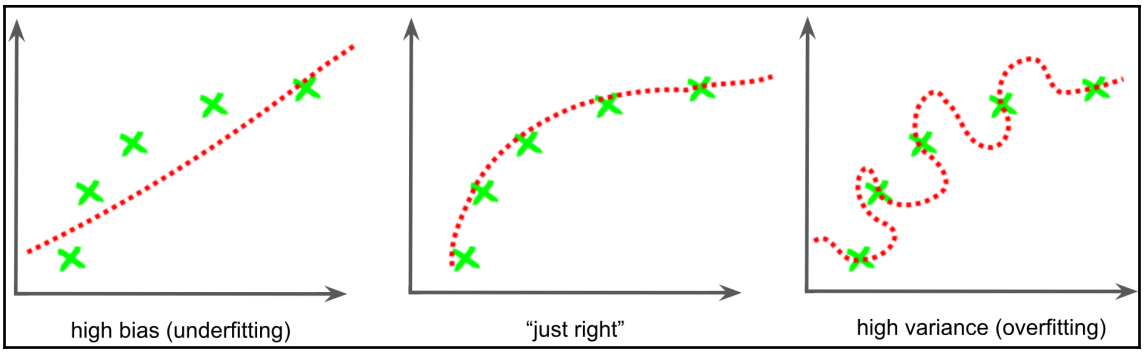


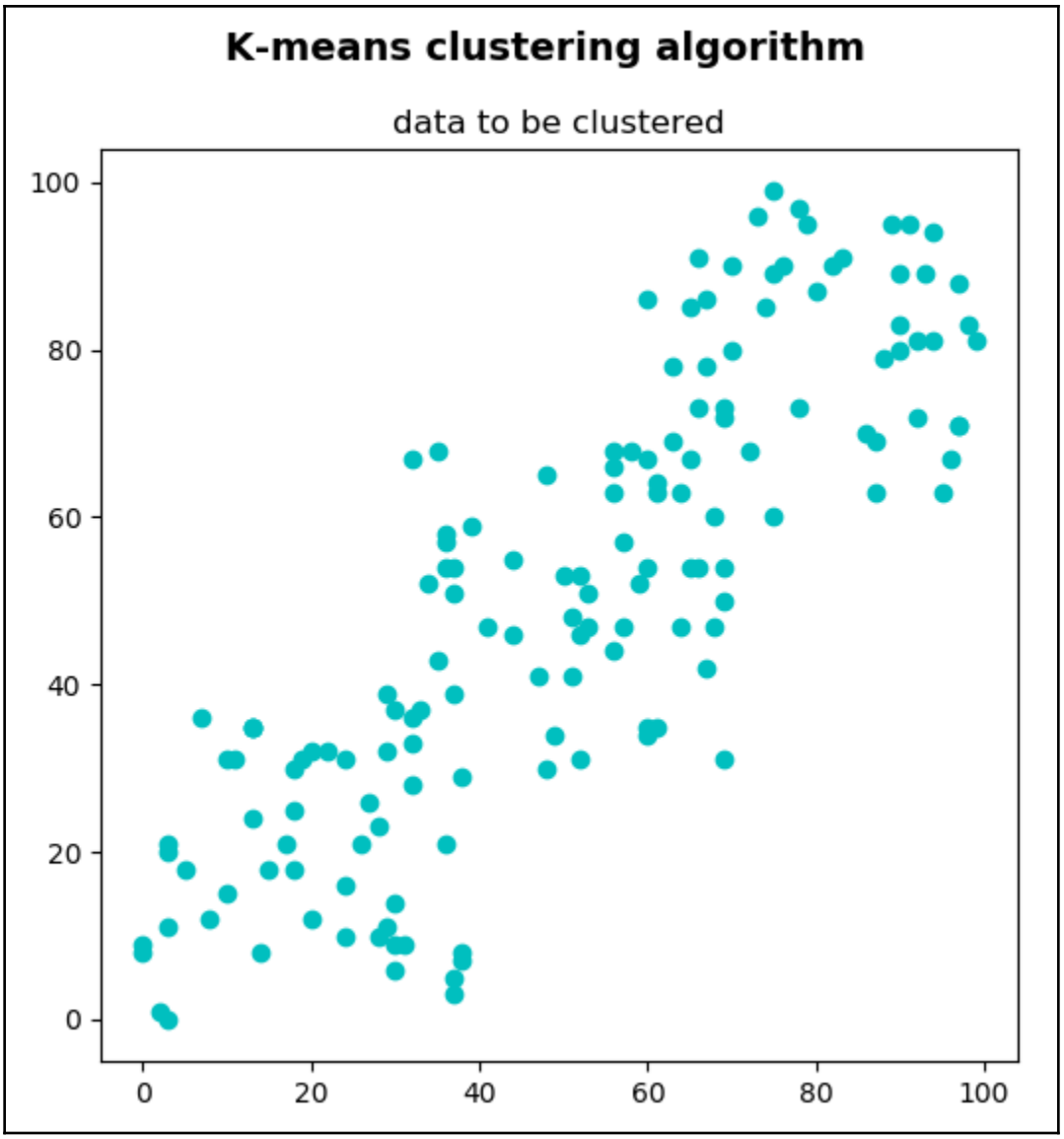
Figure 1
QR code detection
rectified QR code
decoded data: https://github.com/PacktPublishing/Mastering-OpenCV-4-with-Python

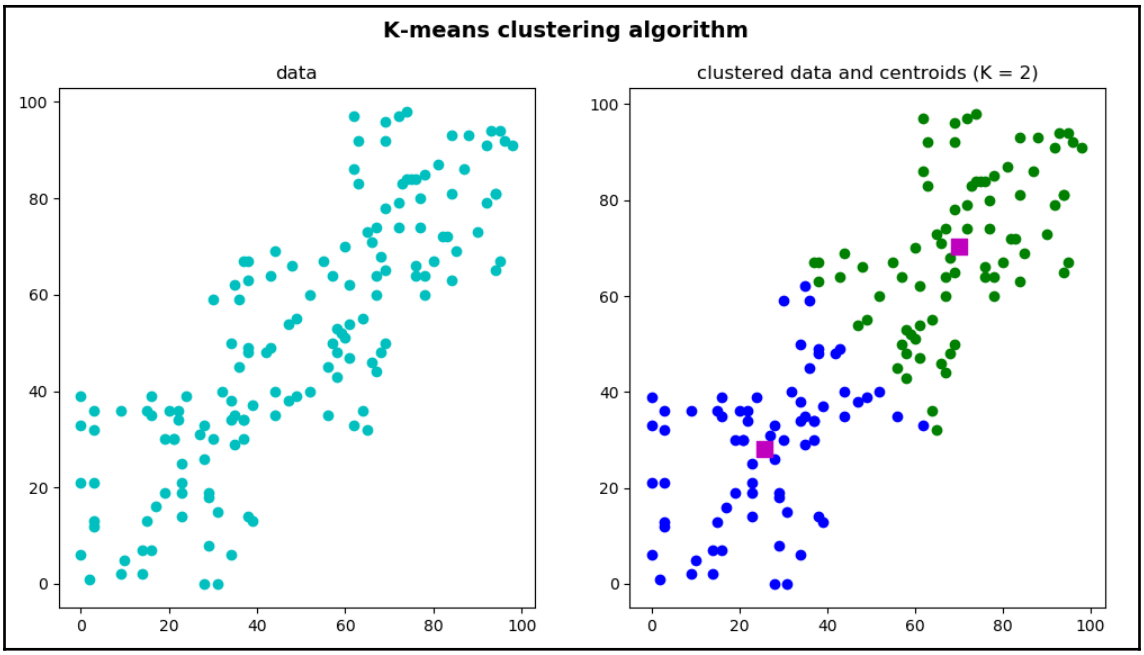
Chapter 10: Machine Learning with OpenCV

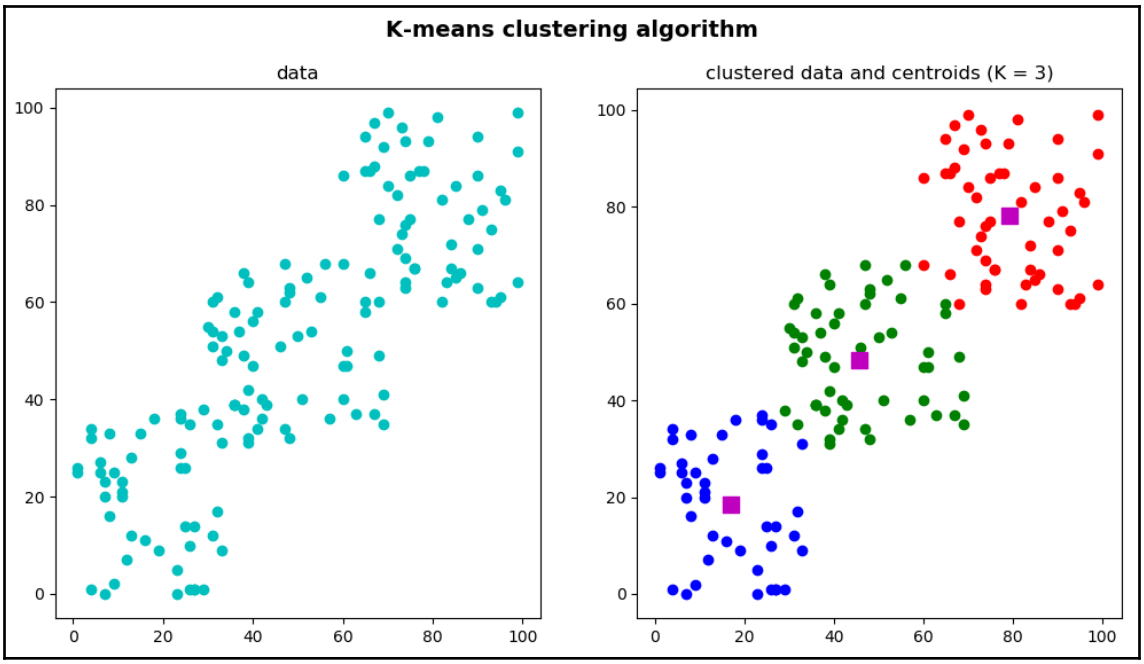


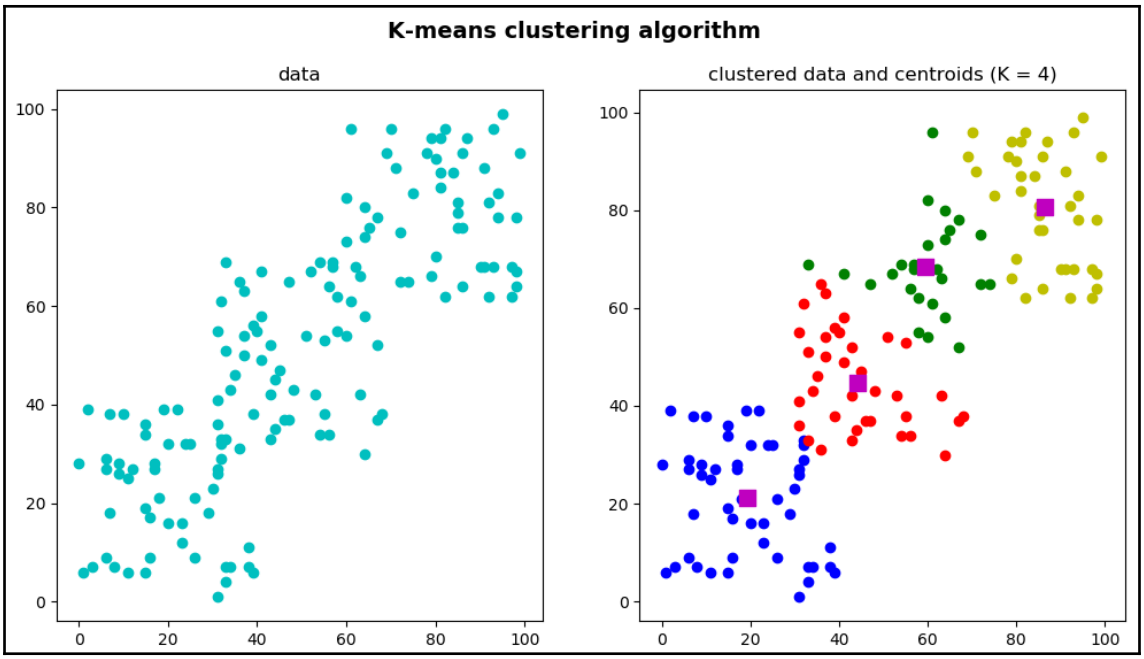


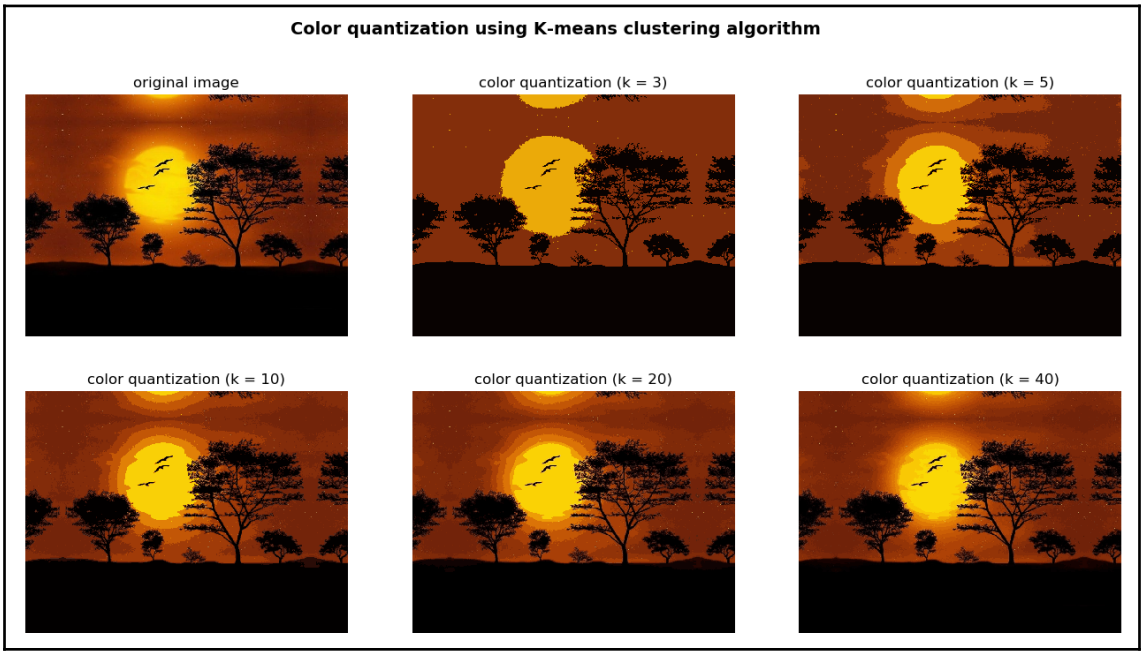

high bias (underfitting)
“just right”
high variance (overfitting)

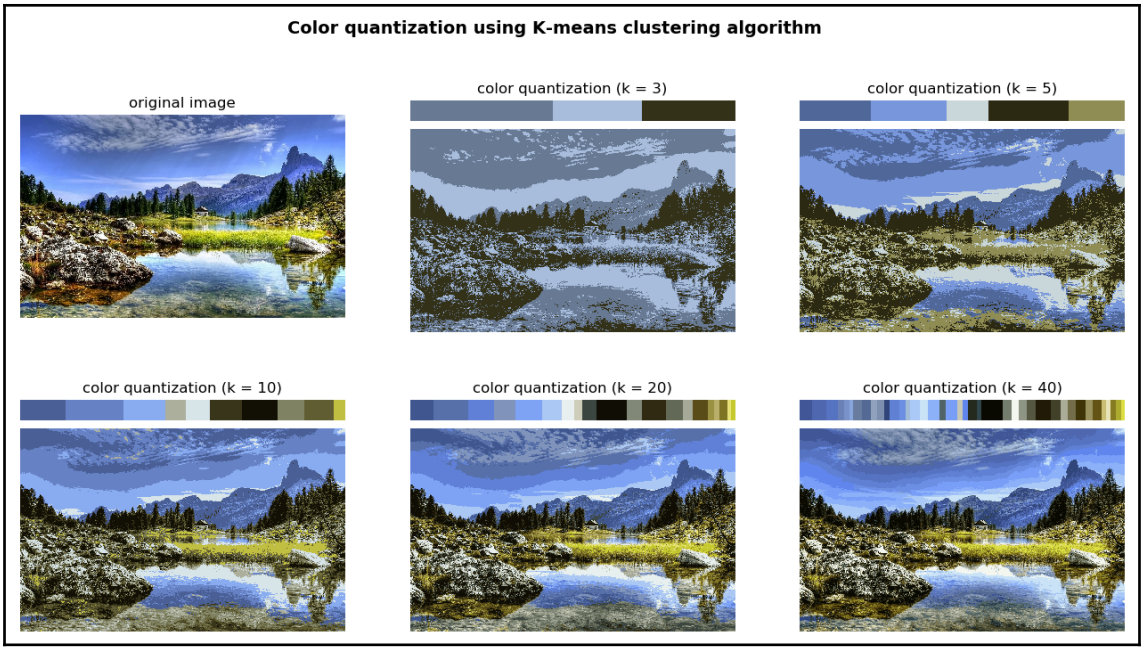

K-means clustering algorithm
data to be clustered
100
80
60
40
20
0
0
20
40
60
80
100

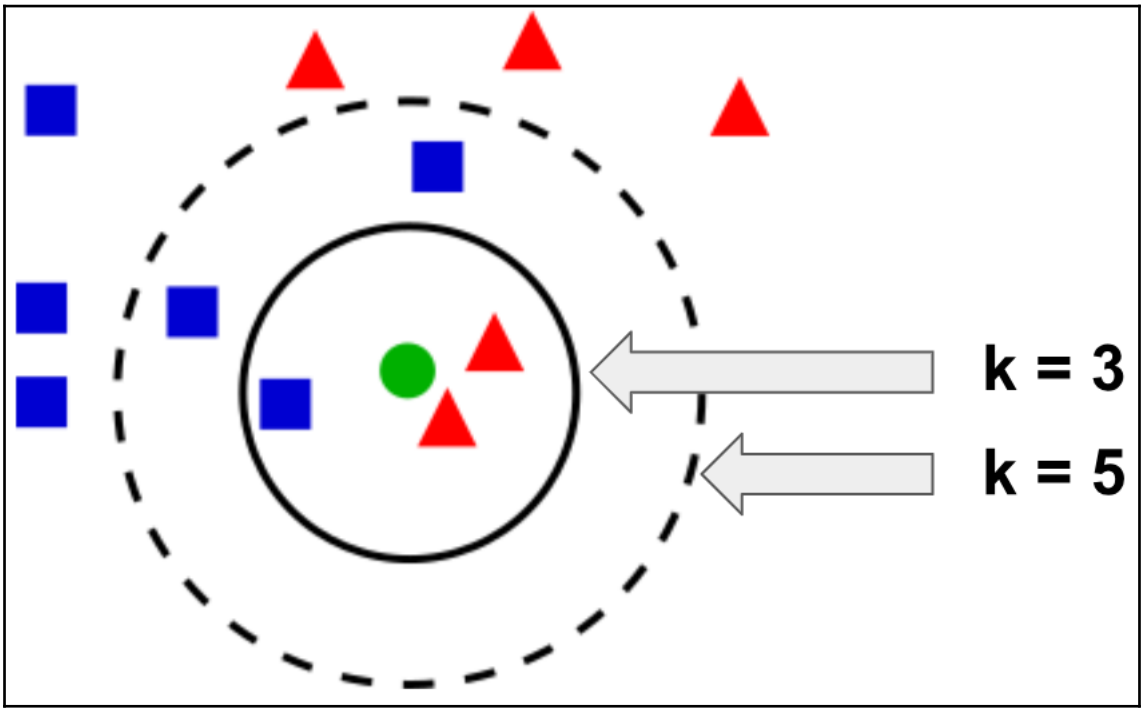

K-means clustering algorithm
data
clustered data and centroids (K = 2)
0
20
40
60
80
100

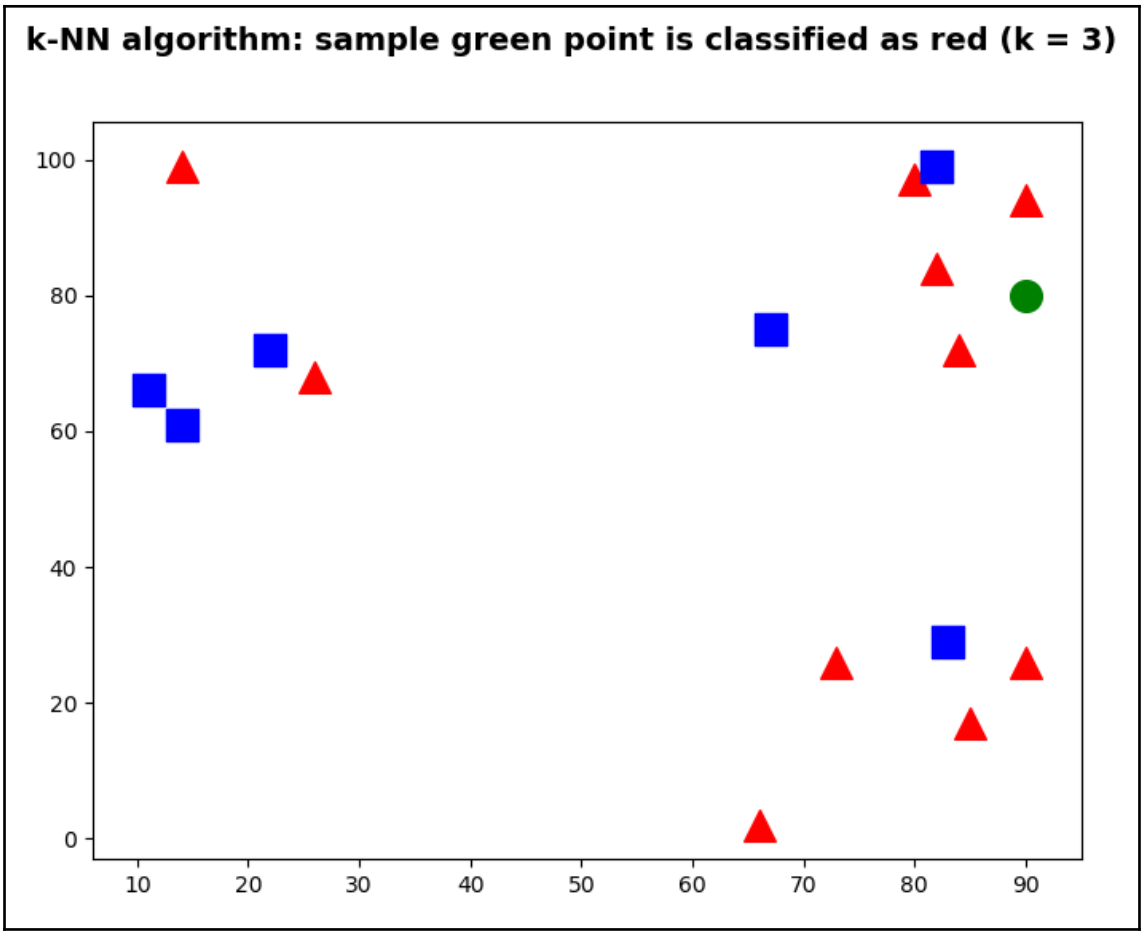

K-means clustering algorithm
data
clustered data and centroids (K = 3)
0
20
40
60
80
100

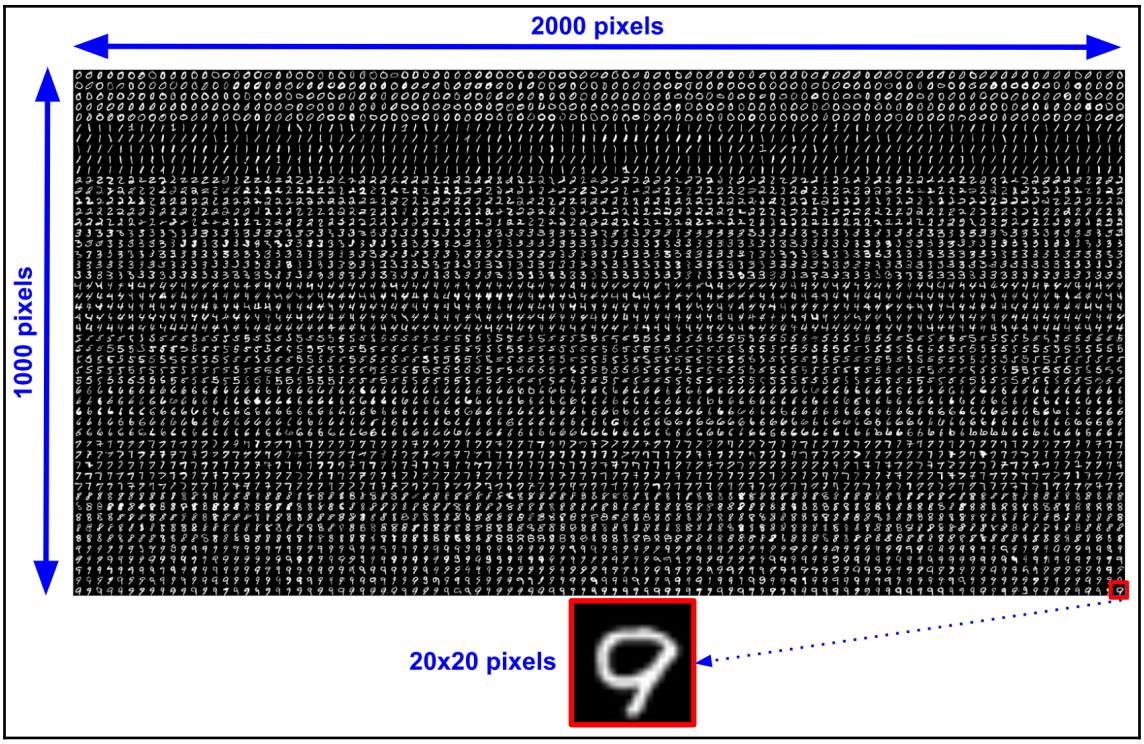

K-means clustering algorithm
data
clustered data and centroids (K = 4)
0
20
40
60
80
100

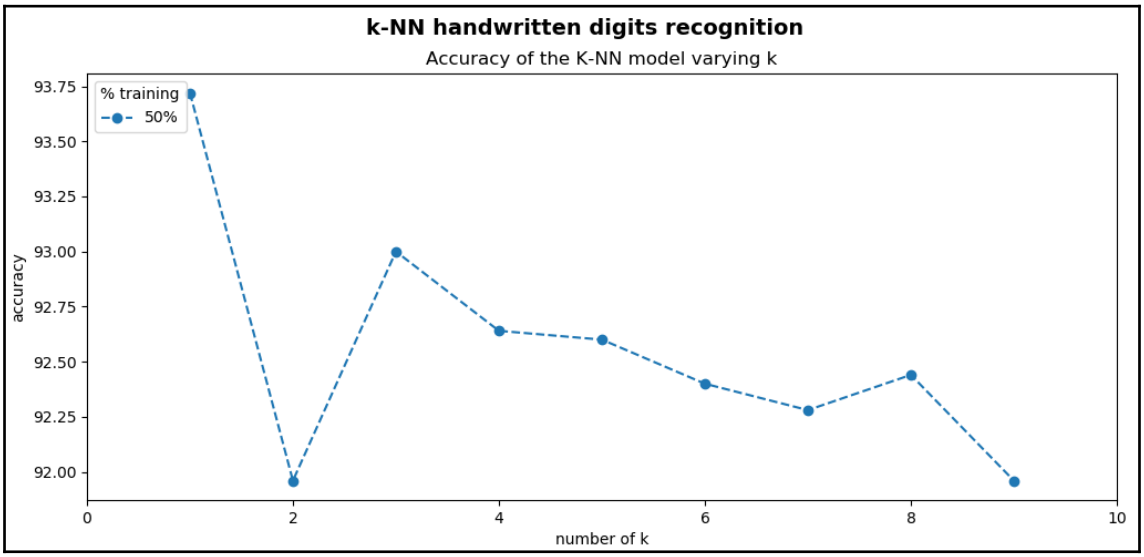

Color quantization using K-means clustering algorithm
original image
color quantization (k = 3)
color quantization (k = 5)
color quantization (k = 10)
color quantization (k = 20)
color quantization (k = 40)

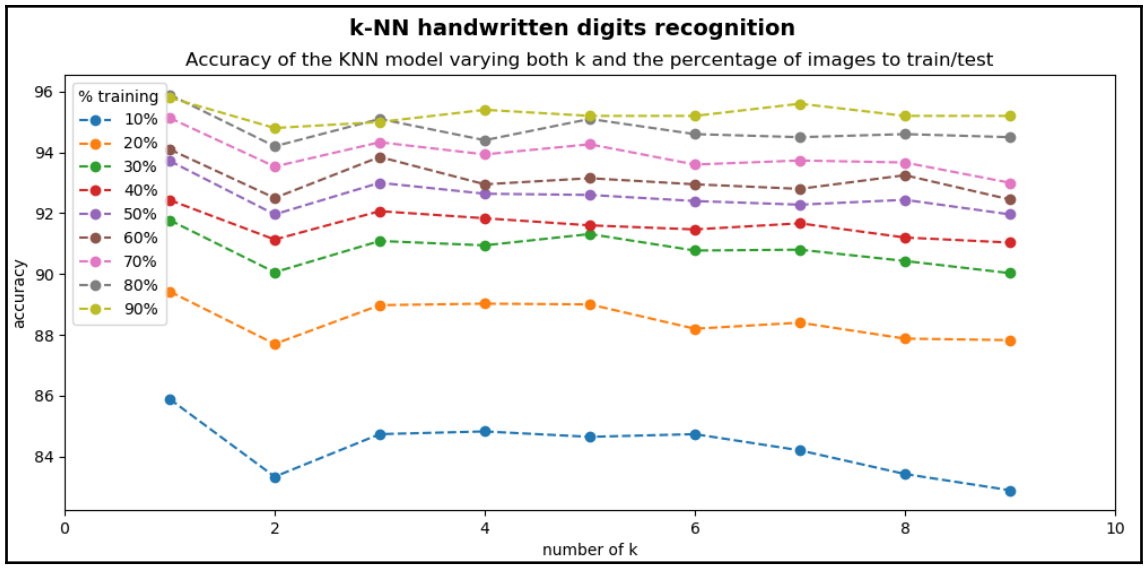

Color quantization using K-means clustering algorithm
original image
color quantization (k = 3)
color quantization (k = 5)
color quantization (k = 10)
color quantization (k = 20)
color quantization (k = 40)

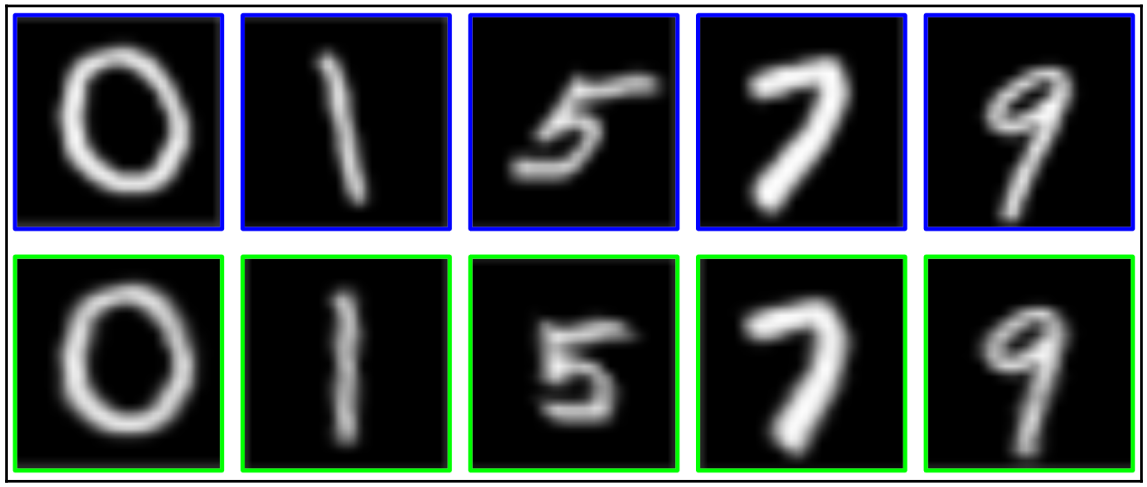

k = 3
k = 5

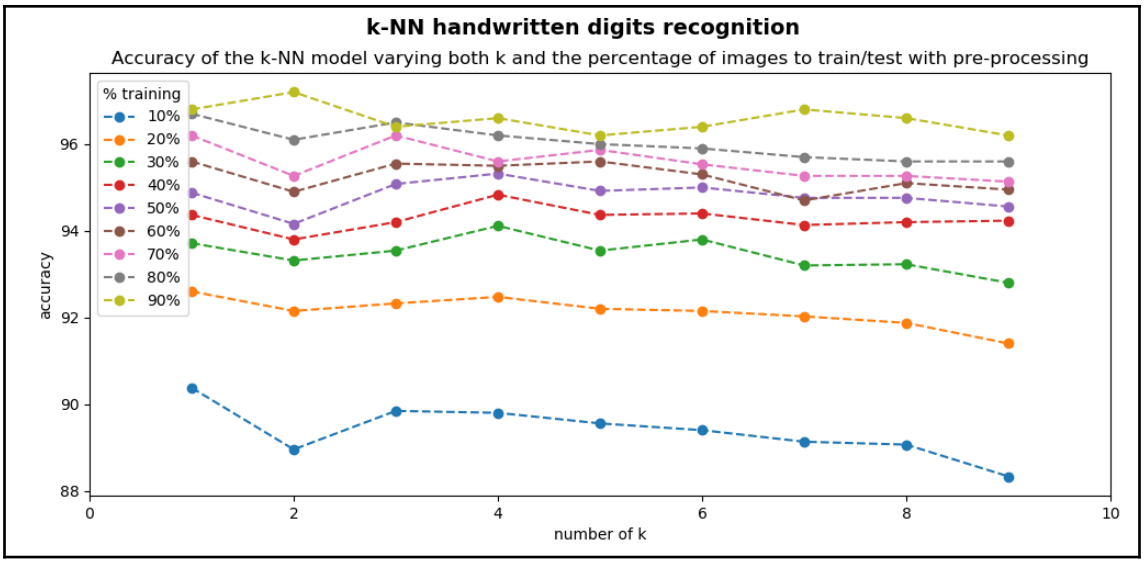

k-NN algorithm: sample green point is classified as red (k = 3)
100
80
60
40
20
0
10
20
30
40
50
60
70
80
90

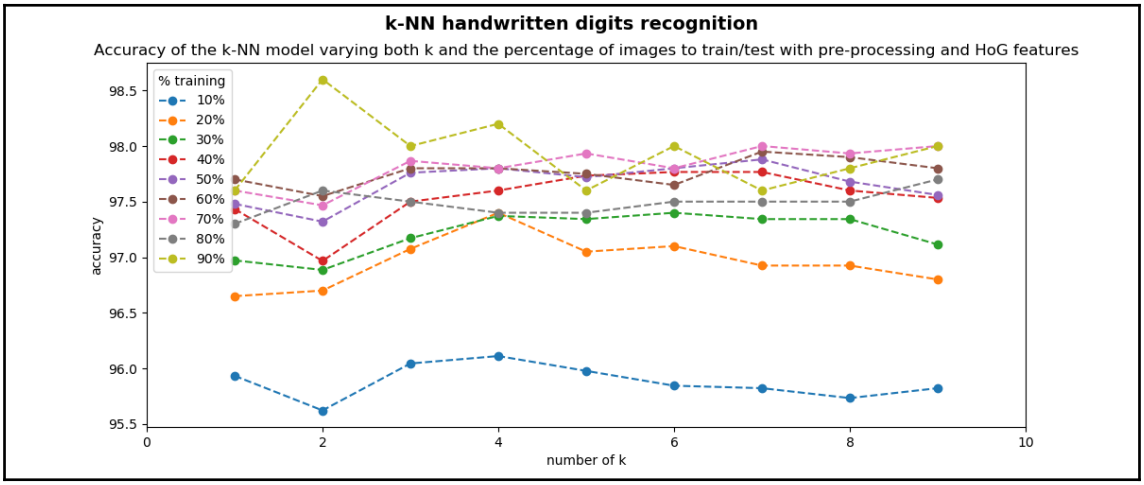

2000 pixels
1000 pixels
20x20 pixels

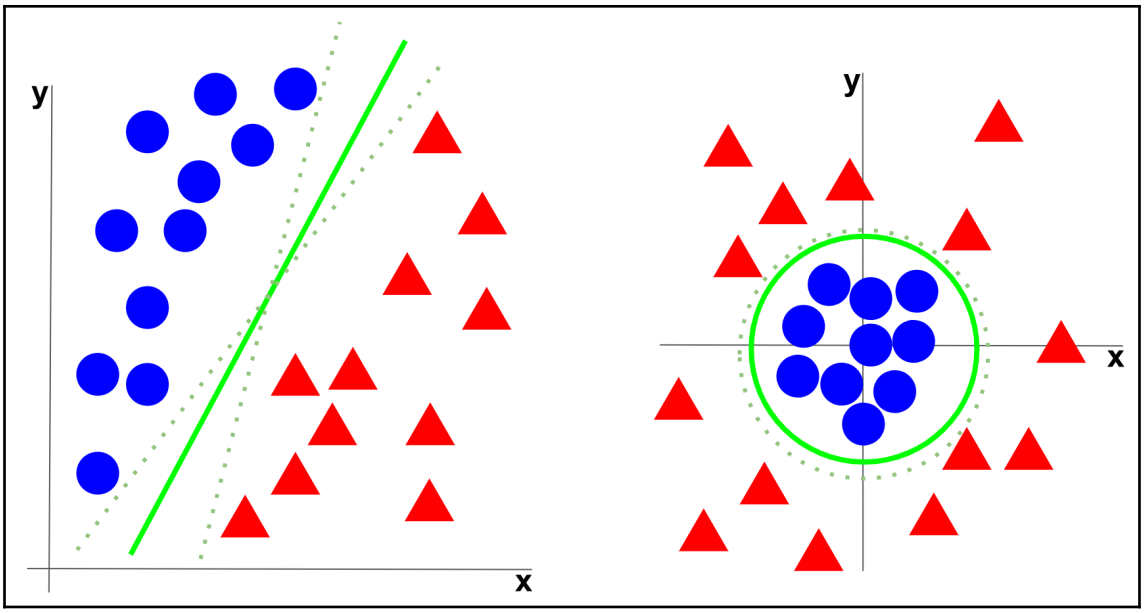

k-NN handwritten digits recognition
Accuracy of the K-NN model varying k
% training
50%
accuracy
number of k
93.75
93.50
93.25
93.00
92.75
92.50
92.25
92.00
0
2
4
6
8
10

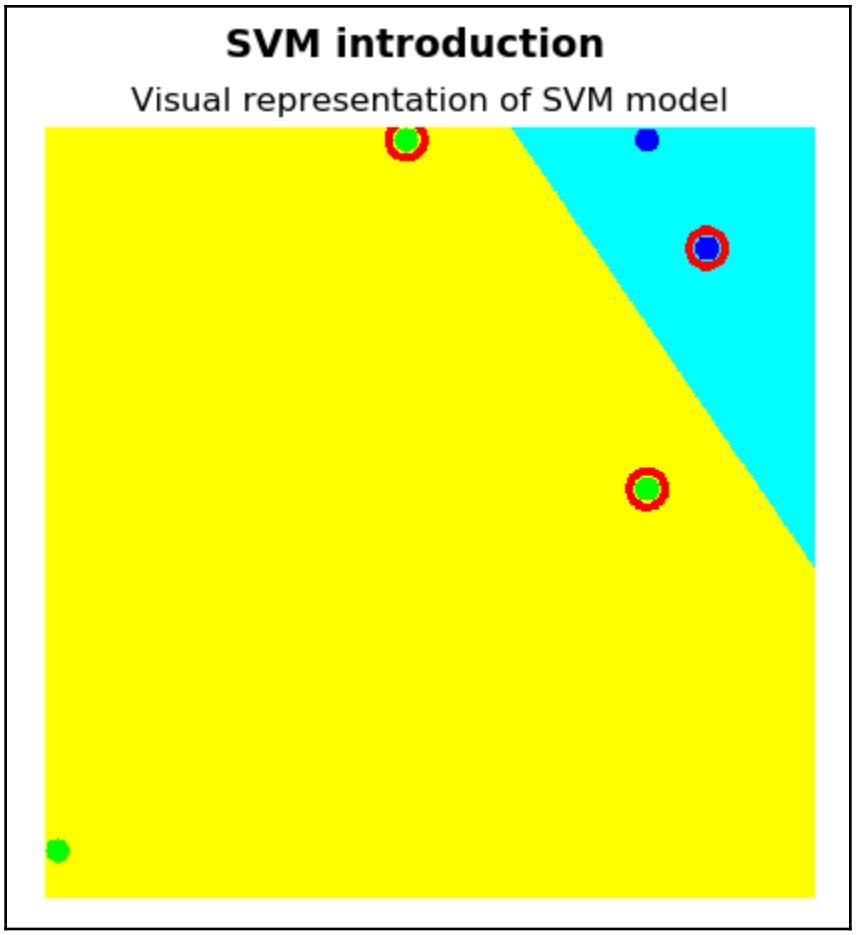

k-NN handwritten digits recognition
Accuracy of the KNN model varying both k and the percentage of images to train/test
% training
10%
20%
30%
40%
50%
60%
70%
80%
90%
accuracy
number of k
96
94
92
90
88
86
84
0
2
4
6
8
10

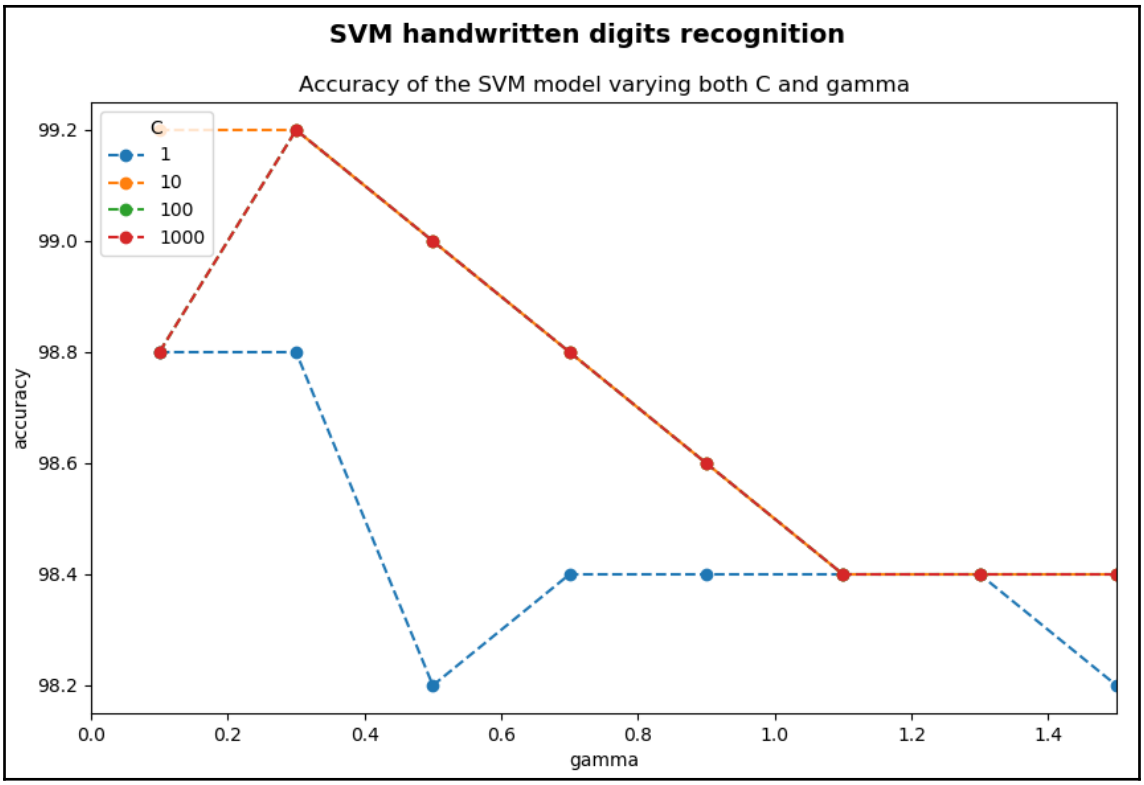



k-NN handwritten digits recognition
Accuracy of the k-NN model varying both k and the percentage of images to train/test with pre-processing
% training
10%
20%
30%
40%
50%
60%
70%
80%
90%
accuracy
number of k

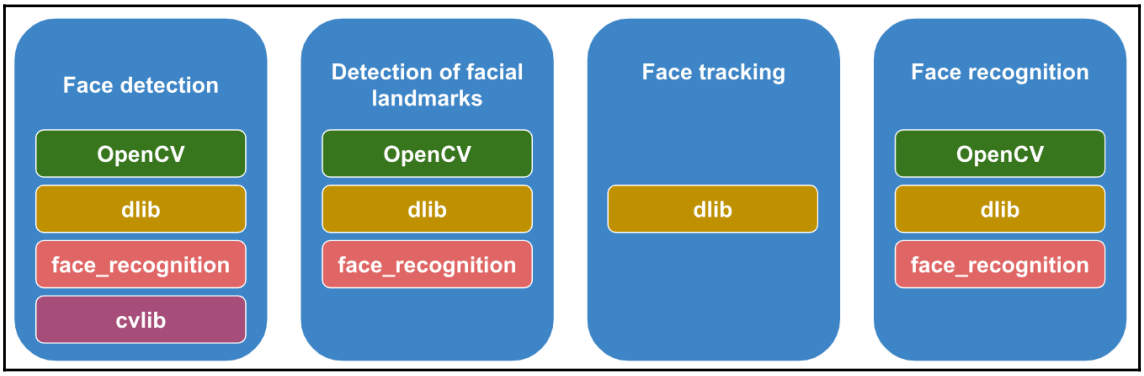

k-NN handwritten digits recognition
Accuracy of the k-NN model varying both k and the percentage of images to train/test with pre-processing and HoG features
% training
10%
20%
30%
40%
50%
60%
70%
80%
90%
accuracy
number of k

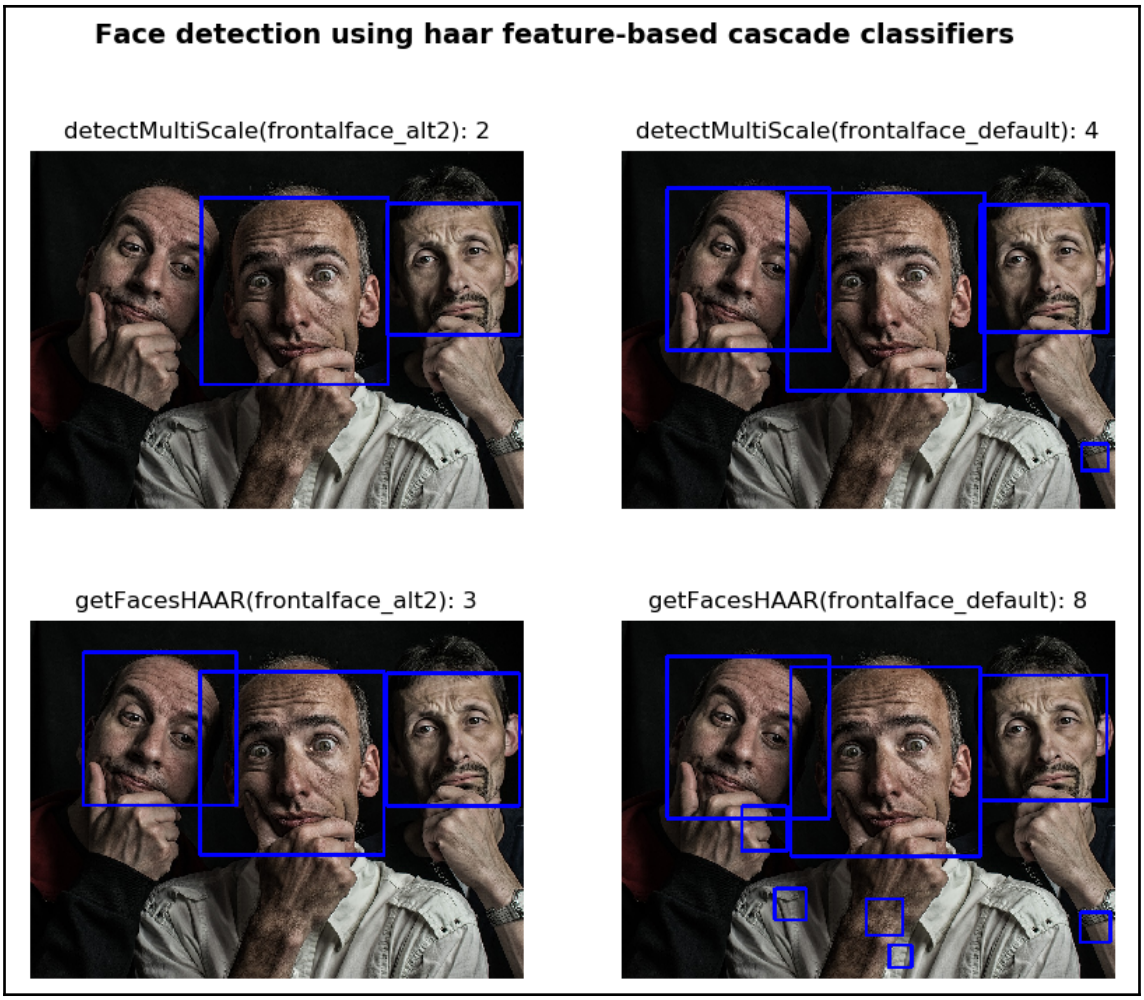

y
x
y
x

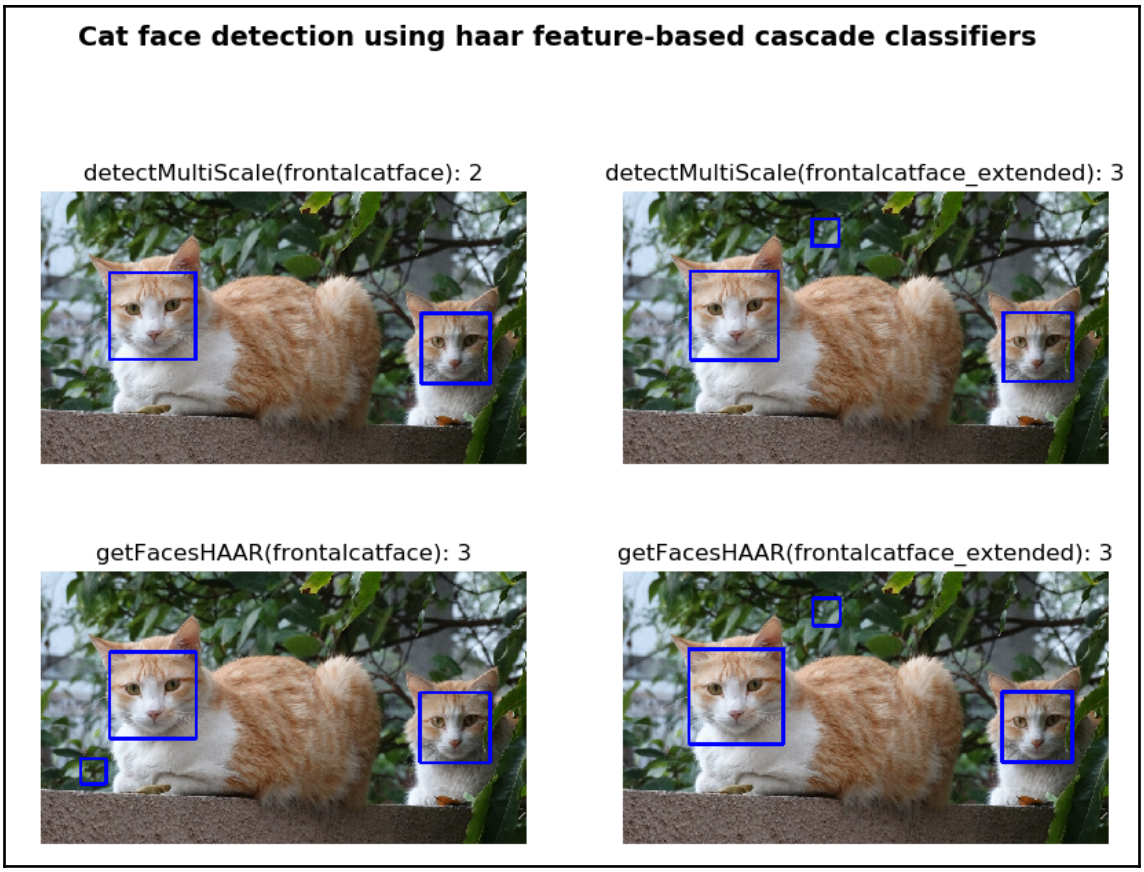

SVM introduction
Visual representation of SVM model

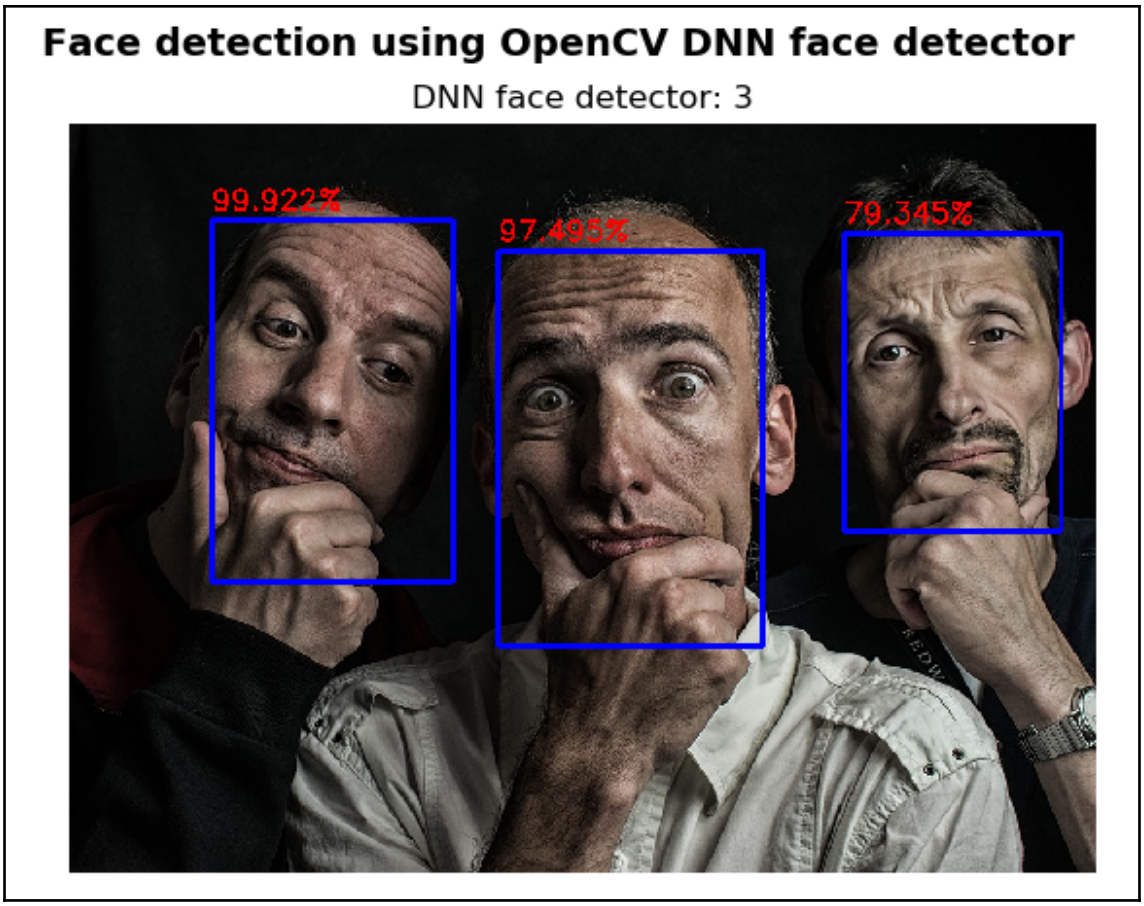

SVM handwritten digits recognition
Accuracy of the SVM model varying both C and gamma
C
1
10
100
1000
99.2
99.0
98.8
98.6
98.4
98.2
accuracy
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
gamma

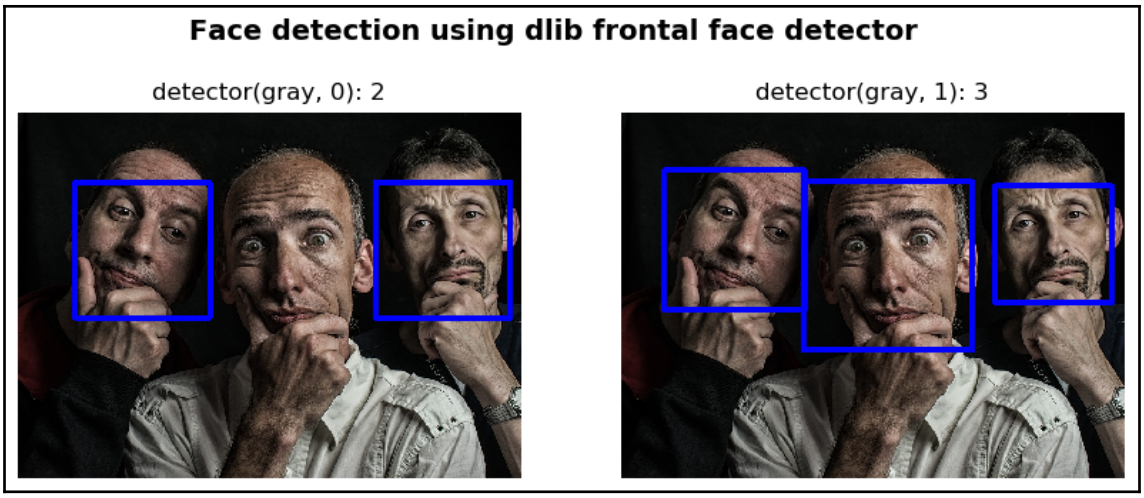
Chapter 11: Face Detection, Tracking, and Recognition

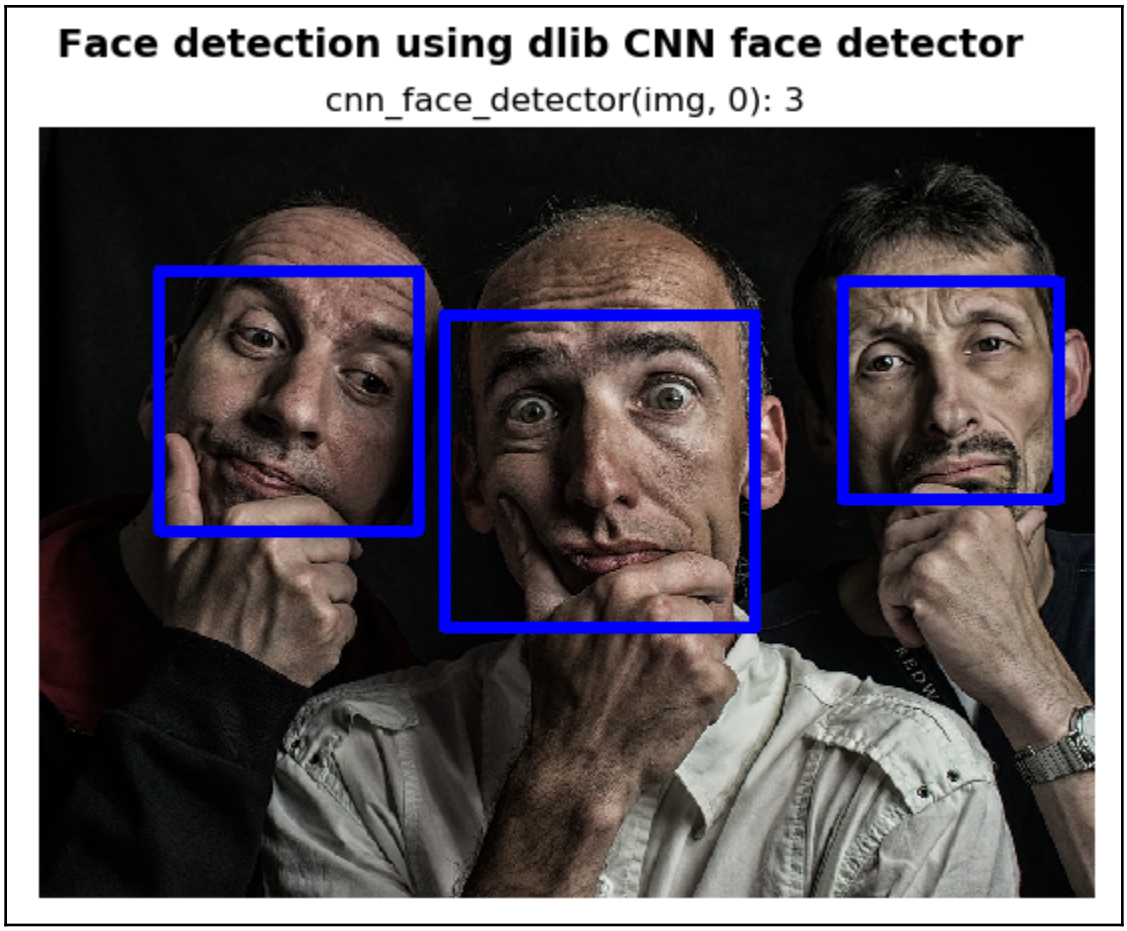


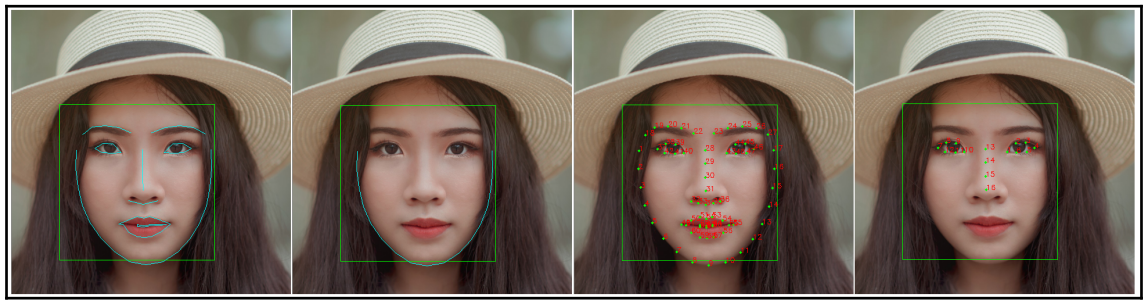

Face detection using haar feature-based cascade classifiers
detectMultiScale(frontalface_alt2): 2
detectMultiScale(frontalface_default): 4
getFacesHAAR(frontalface_alt2): 3
getFacesHAAR(frontalface_default): 8

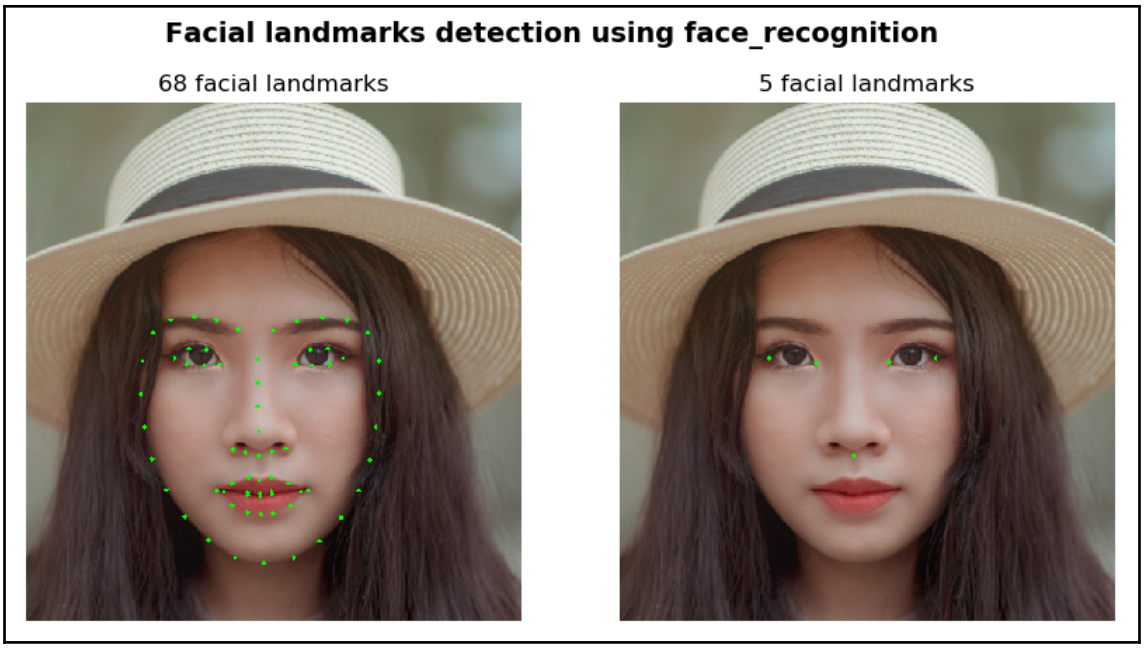

Cat face detection using haar feature-based cascade classifiers
detectMultiScale(frontalcatface): 2
detectMultiScale(frontalcatface_extended): 3
getFacesHAAR(frontalcatface): 3
getFacesHAAR(frontalcatface_extended): 3

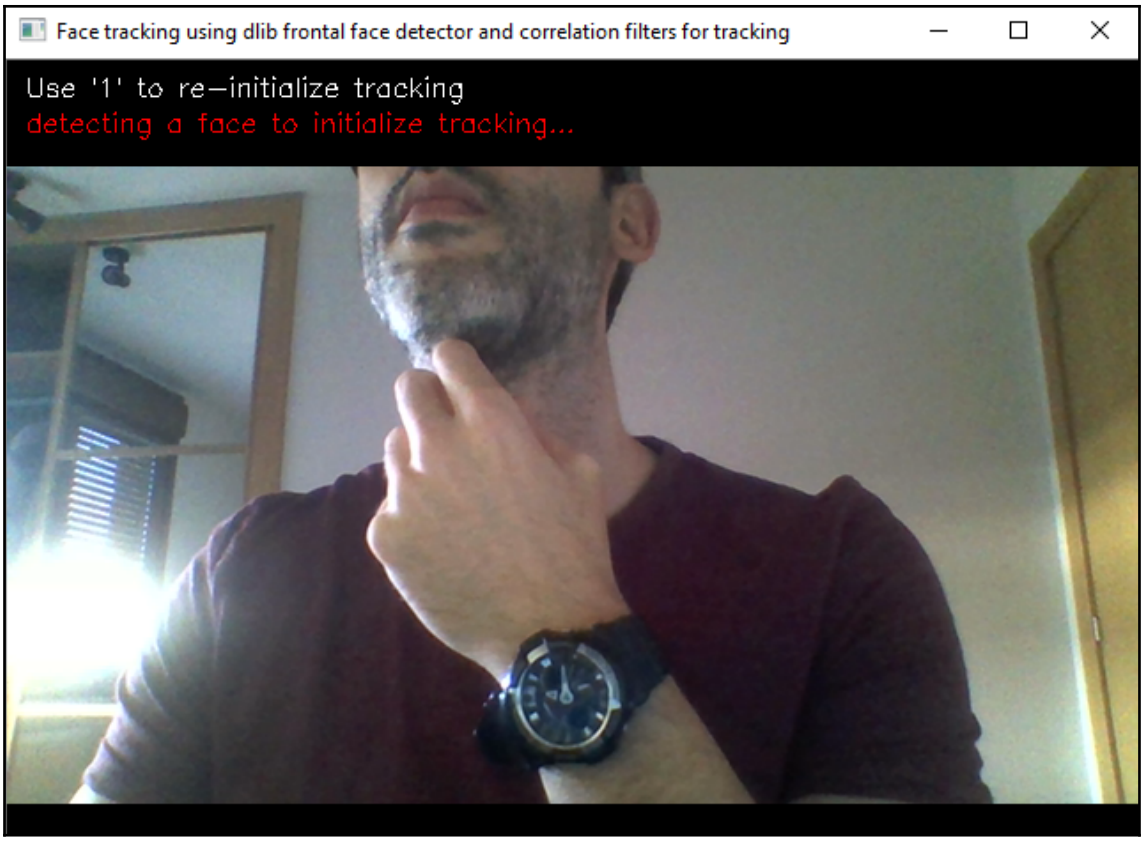

Face detection using OpenCV DNN face detector
DNN face detector: 3
99.922%
97.495%
79.345%

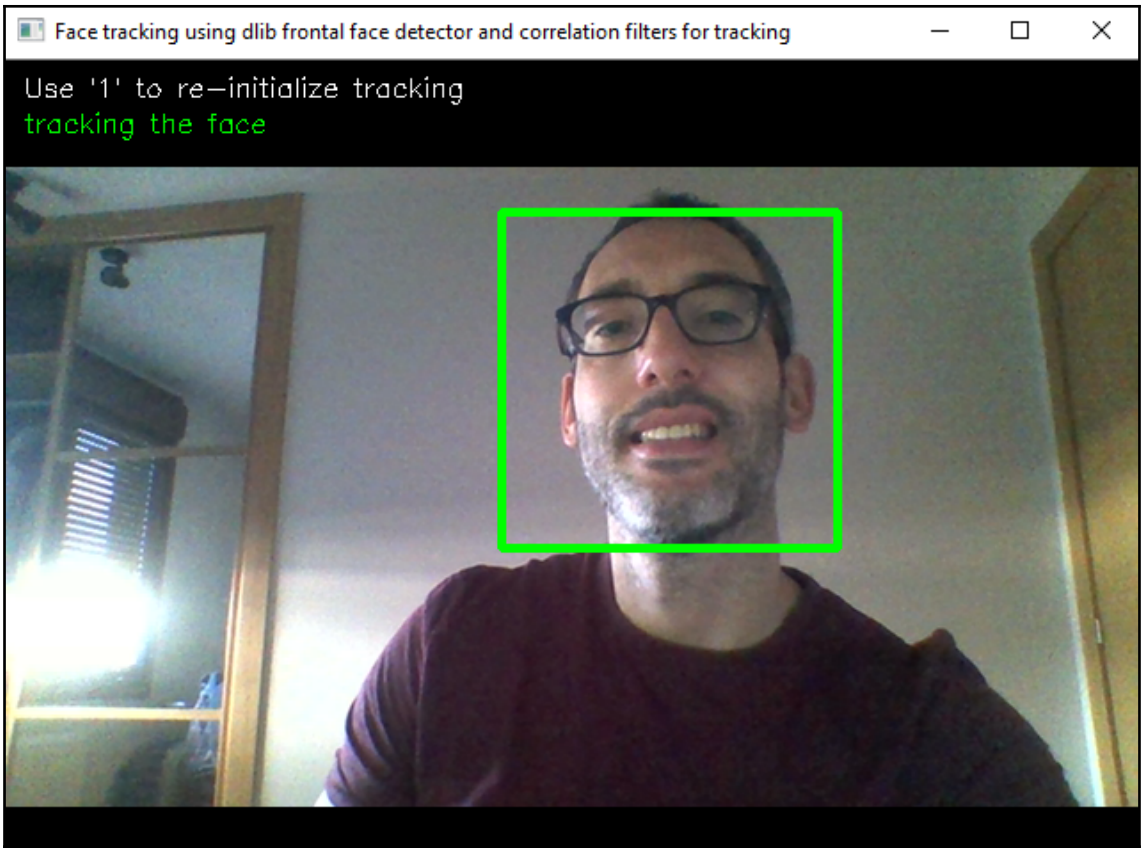

Face detection using dlib frontal face detector
detector(gray, 0): 2
detector(gray, 1): 3

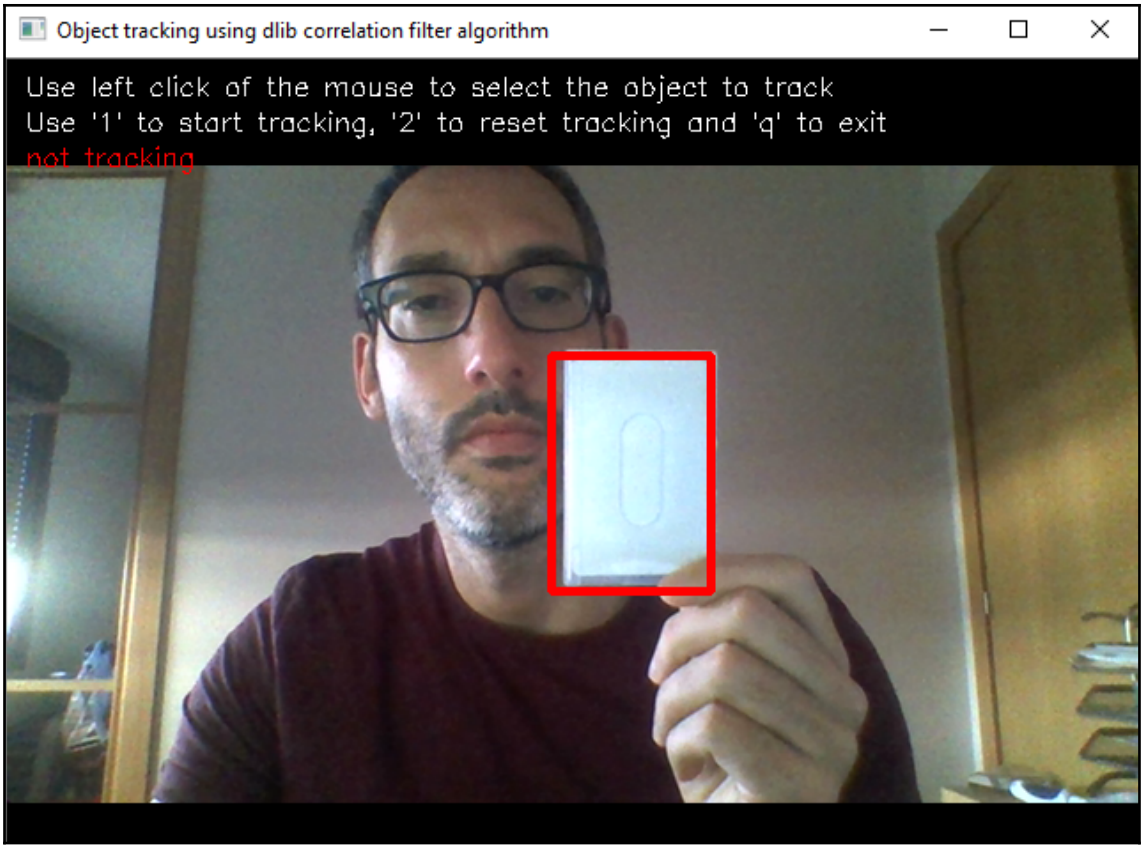

Face detection using dlib CNN face detector
cnn_face_detector(img, 0): 3

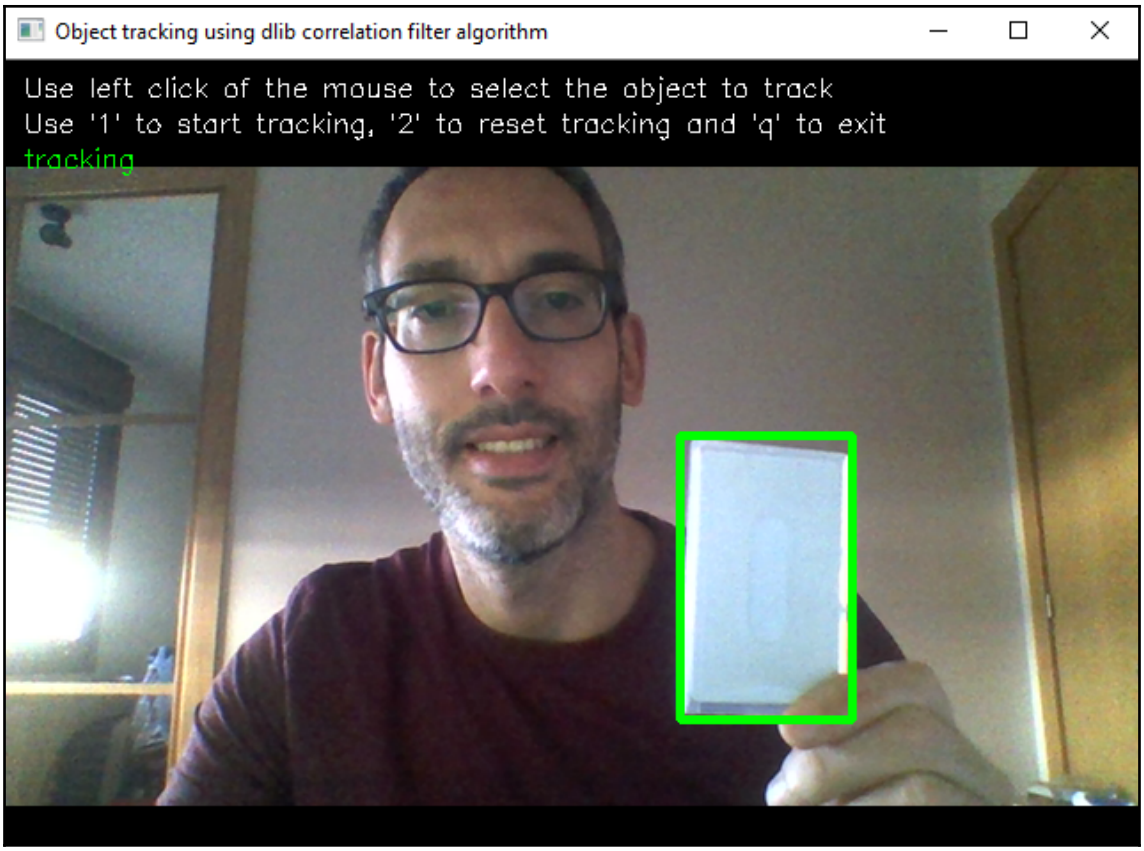



Facial landmarks detection using face_recognition
68 facial landmarks
5 facial landmarks

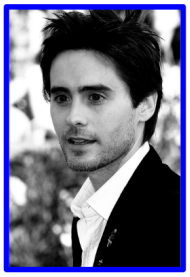

Face tracking using dlib frontal face detector and correlation filters for tracking
Use '1' to re-initialize tracking
detecting a face to initialize tracking...


Face tracking using dlib frontal face detector and correlation filters for tracking
Use '1' to re-initialize tracking
tracking the face


Object tracking using dlib correlation filter algorithm
Use left click of the mouse to select the object to track
Use '1' to start tracking, '2' to reset tracking and 'q' to exit
not tracking


Object tracking using dlib correlation filter algorithm
Use left click of the mouse to select the object to track
Use '1' to start tracking, '2' to reset tracking and 'q' to exit
tracking

jared_1.jpg
0.39983264838593896



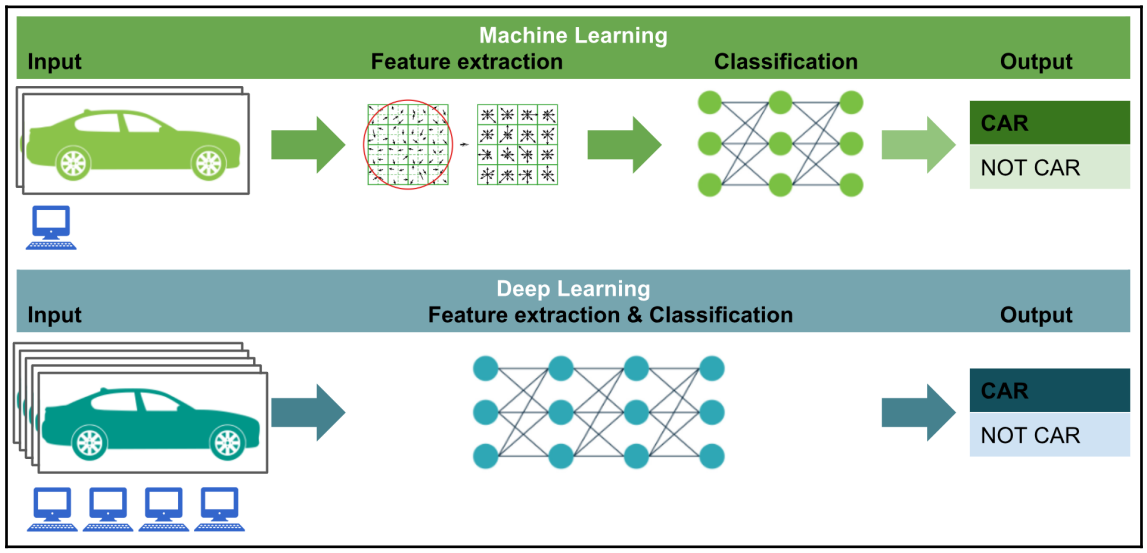
jared_2.jpg
0.4104153683230741

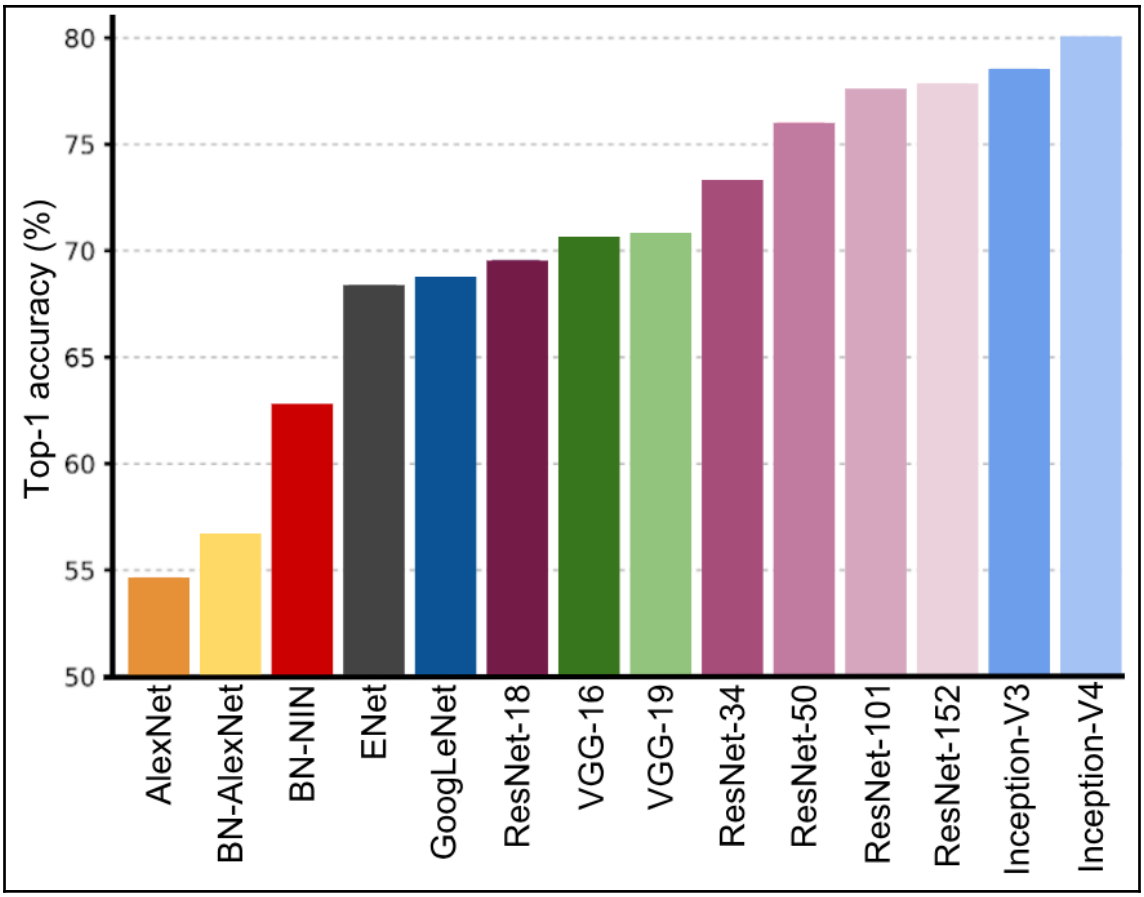
jared_3.jpg
0.3913191431497527

obama.jpg
0.9053700273411349

jared_4.jpg

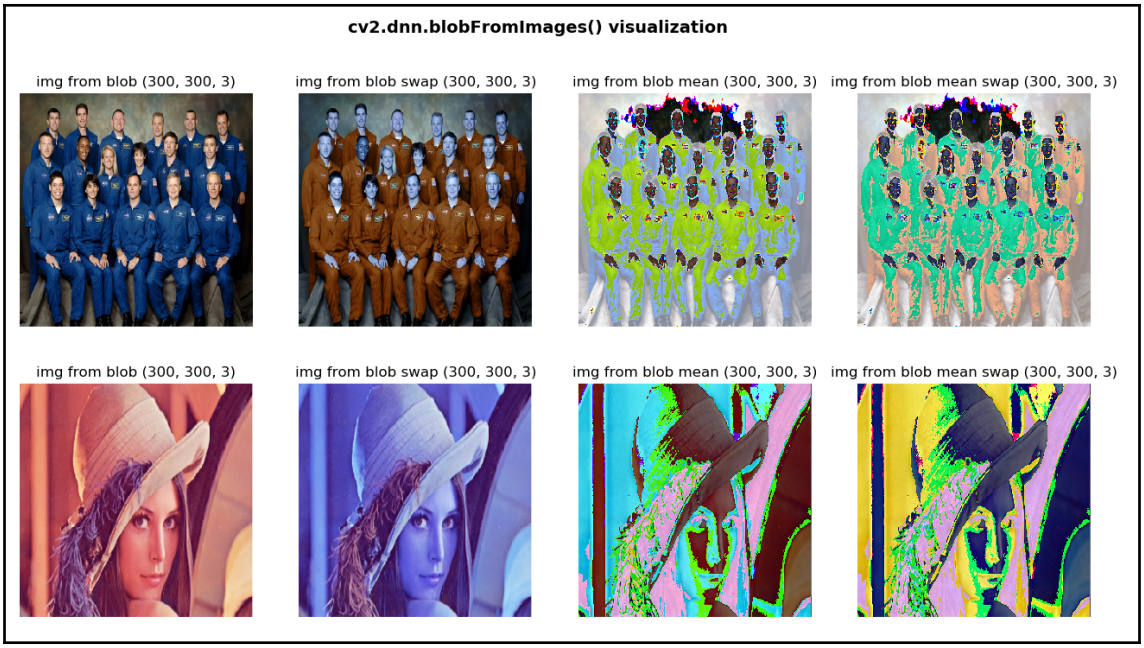
Chapter 12: Introduction to Deep Learning

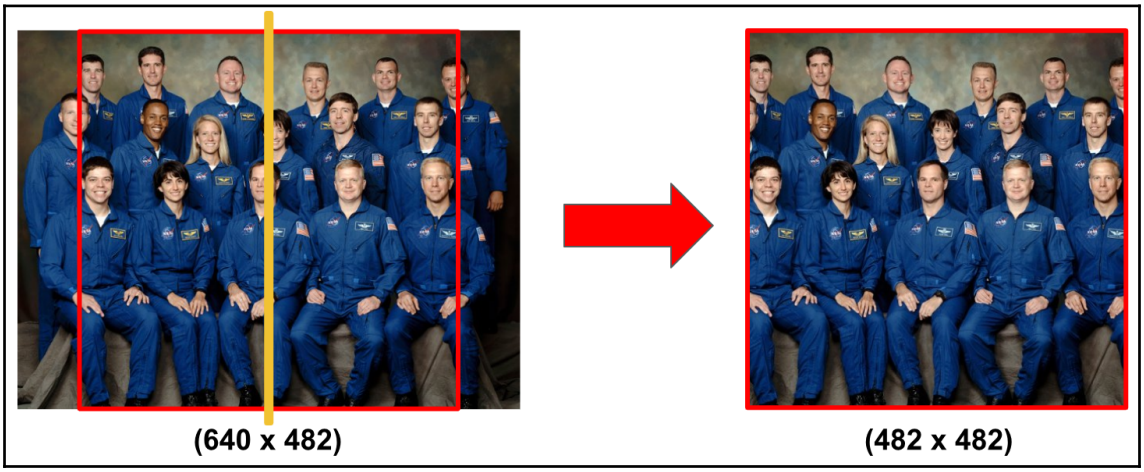



Top-1 accuracy (%)
80
75
70
65
60
55
50
AlexNet
BN-AlexNet
BN-NIN
ENet
GoogLeNet
ResNet-18
VGG-16
VGG-19
ResNet-34
ResNet-50
ResNet-101
ResNet-152
Inception-V3
Inception-V4

Model	PASCAL VOC 2007 (%)	PASCAL VOC 2010 (%)	PASCAL VOC 2012 (%)	COCO 2015 (IoU=0.5) (%)	COCO 2015 (IoU=0.75) (%)	COCO 2015 (Official Metric) (%)	COCO 2016 (IoU=0.5) (%)	COCO 2016 (IoU=0.75) (%)	COCO 2016 (Official Metric) (%)	Real Time
R-CNN (2014)	-	62.4	-	-	-	-	-	-	-	No
Fast R-CNN (2015)	70.0	**68.8**	68.4	-	-	-	-	-	-	No
Faster R-CNN (2015)	78.8	-	75.9	-	-	-	-	-	-	No
R-FCN (2016)	82.0	-	-	**53.2**	-	**31.5**	-	-	-	No
YOLO (2016)	63.7		57.9	-	-	-	-	-	-	**Yes**
SDD (2016)	**83.2**	-	**82.2**	48.5	**30.3**	**31.5**	-	-	-	No
YOLO V2 (2016)	78.6	-	-	44.0	19.2	21.6	-	-	-	**Yes**
NASNet (2016)	-	-	-	43.1	-	-	-	-	-	No
Mask R-CNN (2017)	-	-	-	-	-	-	**62.3**	**43.3**	**39.8**	No

cv2.dnn.blobFromImage() visualization


cv2.dnn.blobFromImages() visualization
img from blob (300, 300, 3)
img from blob swap (300, 300, 3)
img from blob mean (300, 300, 3)
img from blob mean swap (300, 300, 3)
img from blob (300, 300, 3)
img from blob swap (300, 300, 3)
img from blob mean (300, 300, 3)
img from blob mean swap (300, 300, 3)


(640 x 482)
(482 x 482)


cv2.dnn.blobFromImages() visualization with cropping
img 1 from blob (300, 300, 3)
img 2 from blob (300, 300, 3)
img 1 from blob cropped (300, 300, 3)
img 2 from blob cropped (300, 300, 3)

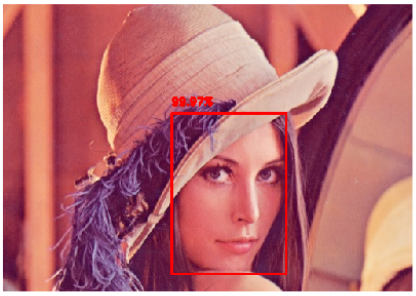
OpenCV DNN face detector when feeding several images

input img 1

input img 2

output img 1

output img 2



OpenCV DNN face detector when feeding several images and cropping

input img 1

input img 2

output cropped img 1

output cropped img 2

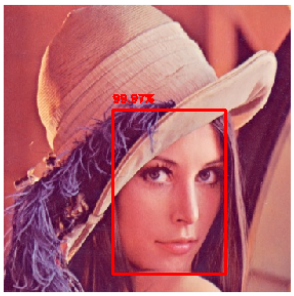


Image classification with OpenCV using AlexNet and caffe pre-trained models

AlexNet and caffe pre-trained models

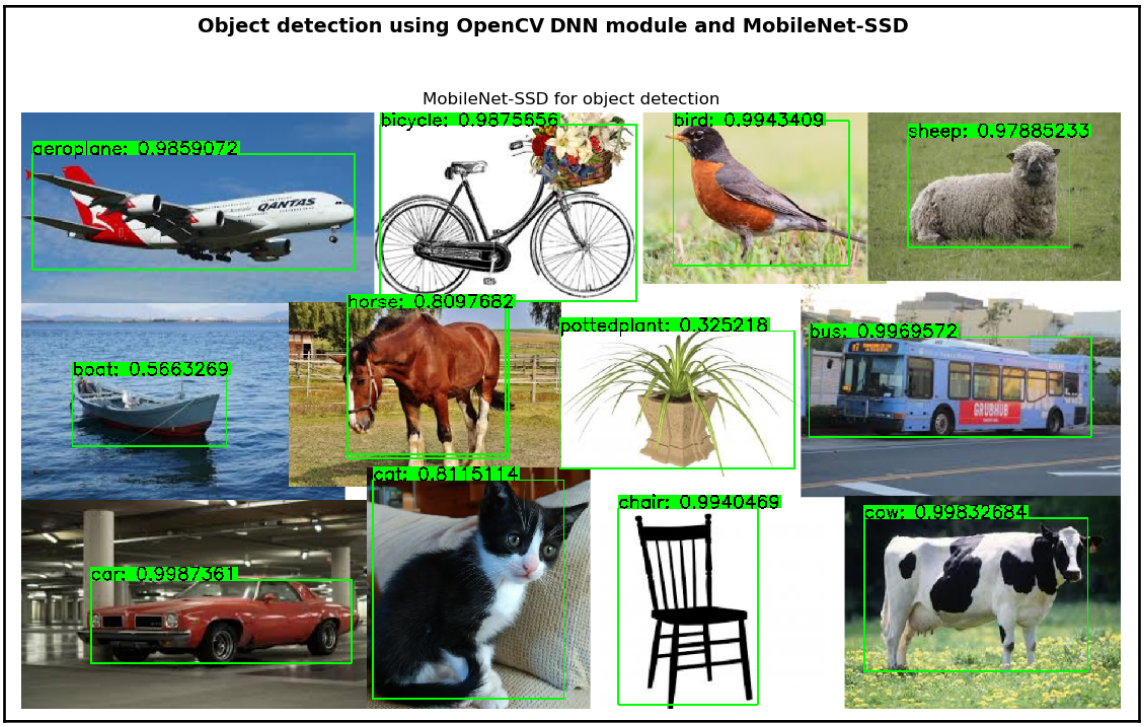
Image classification with OpenCV using GoogLeNet and caffe pre-trained models

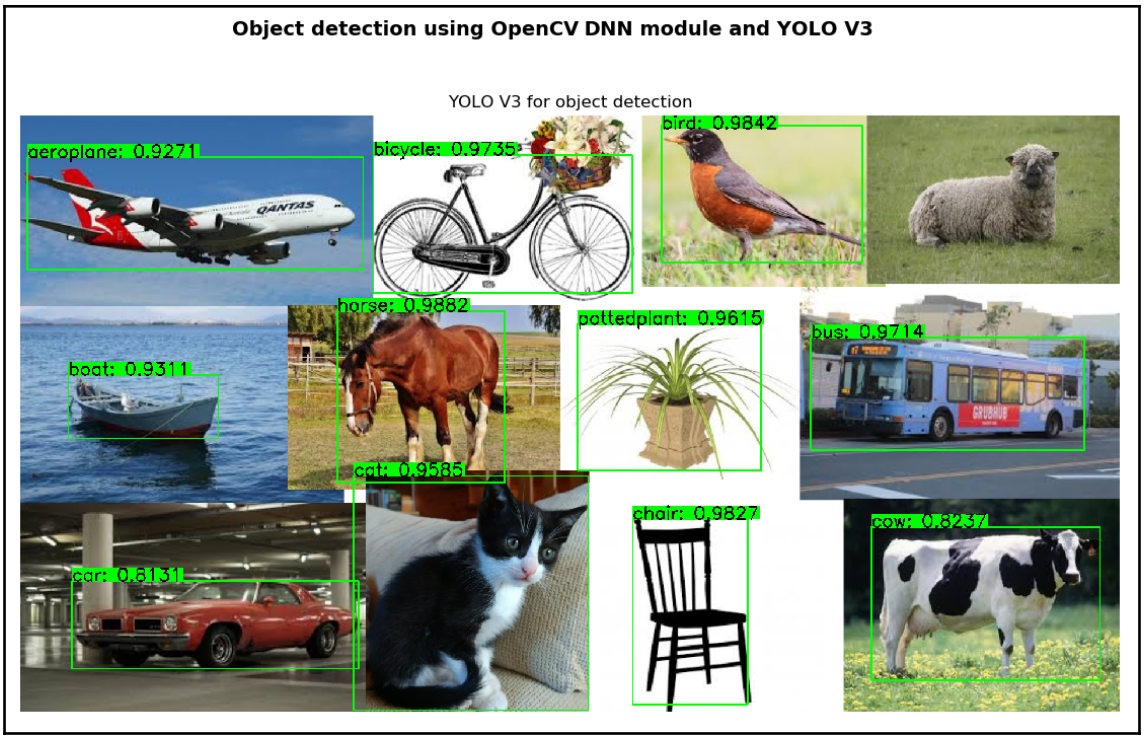
Image classification with OpenCV using ResNet-50 and caffe pre-trained models

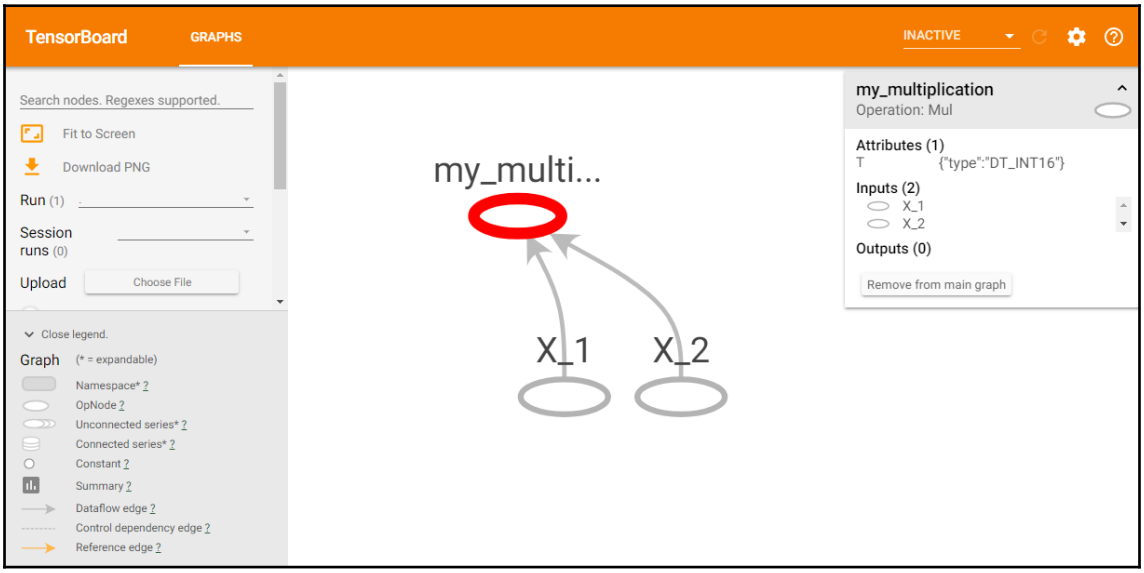
ResNet-50 and caffe pre-trained models

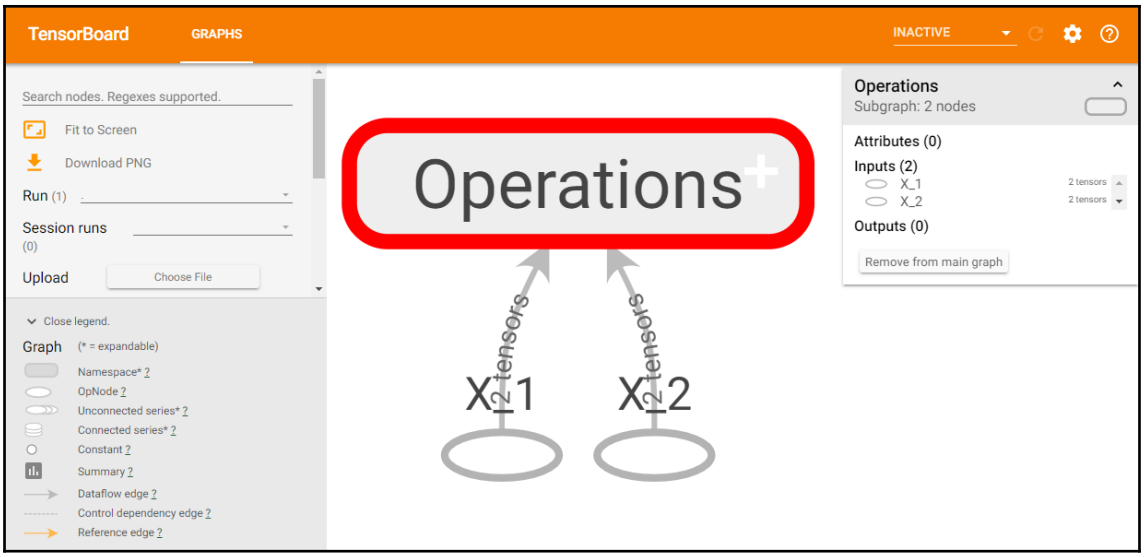
Image classification with OpenCV using SqueezeNet (v1.1) and caffe pre-trained models

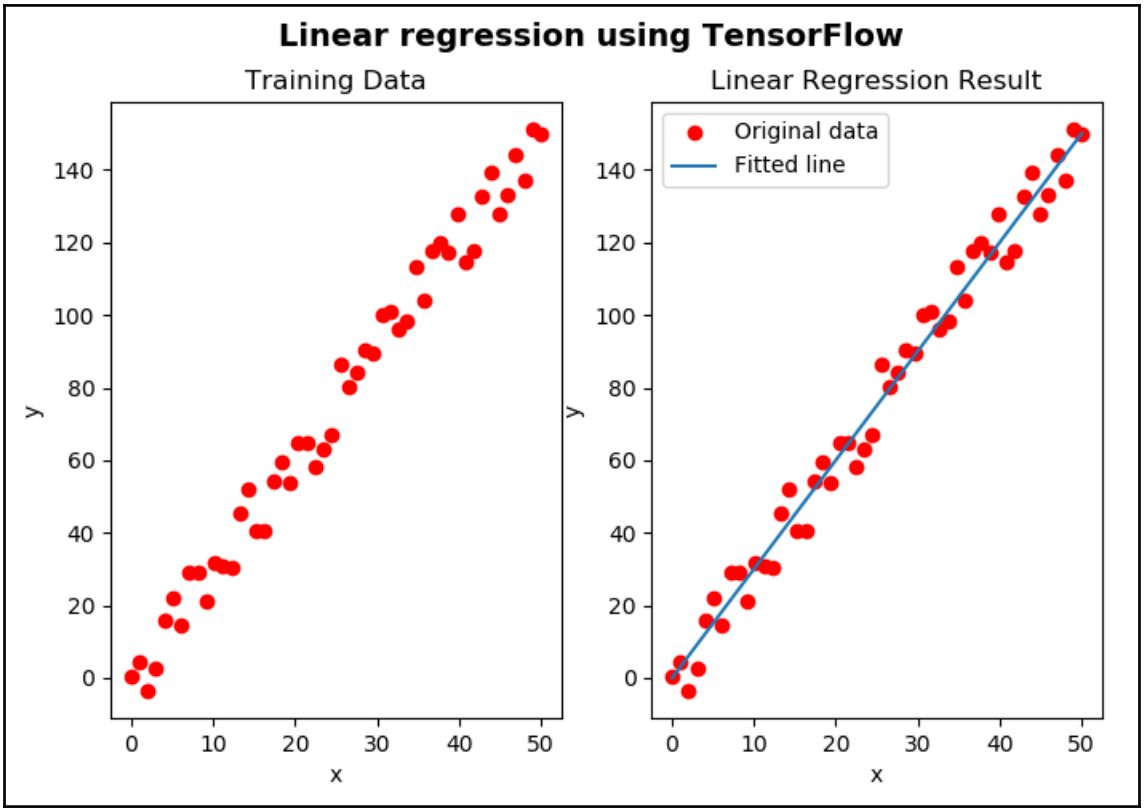
SqueezeNet (v1.1) and caffe pre-trained models

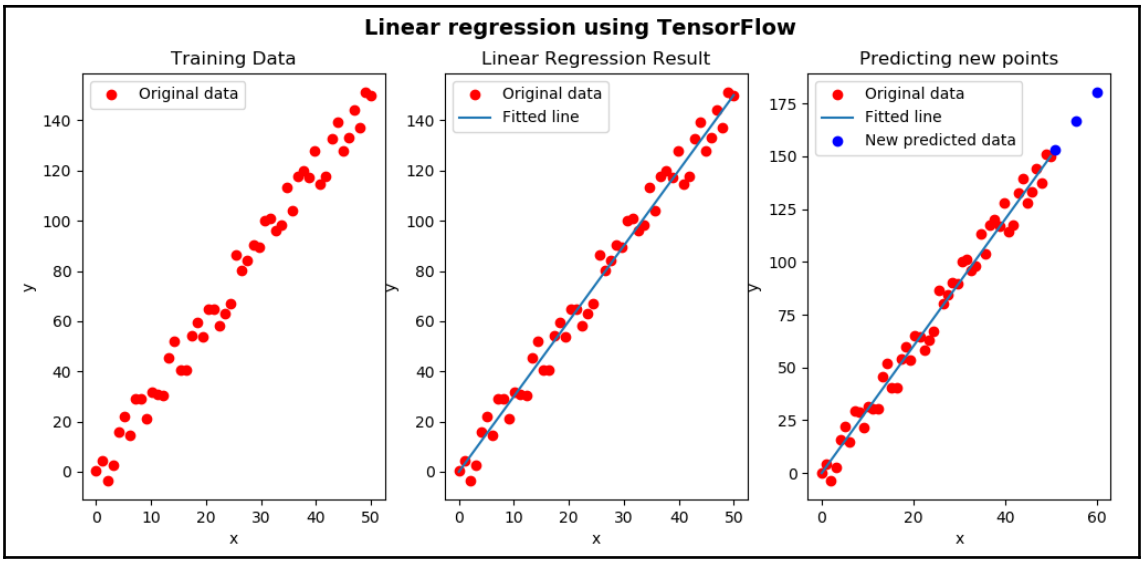

Object detection using OpenCV DNN module and MobileNet-SSD
MobileNet-SSD for object detection
aeroplane: 0.9859072
bicycle: 0.9875656
bird: 0.9943409
sheep: 0.97885233
horse: 0.8097682
boat: 0.5663269
pottedplant: 0.325218
bus: 0.9969572
cat: 0.8115114
chair: 0.9940469
cow: 0.99832684
car: 0.9987361

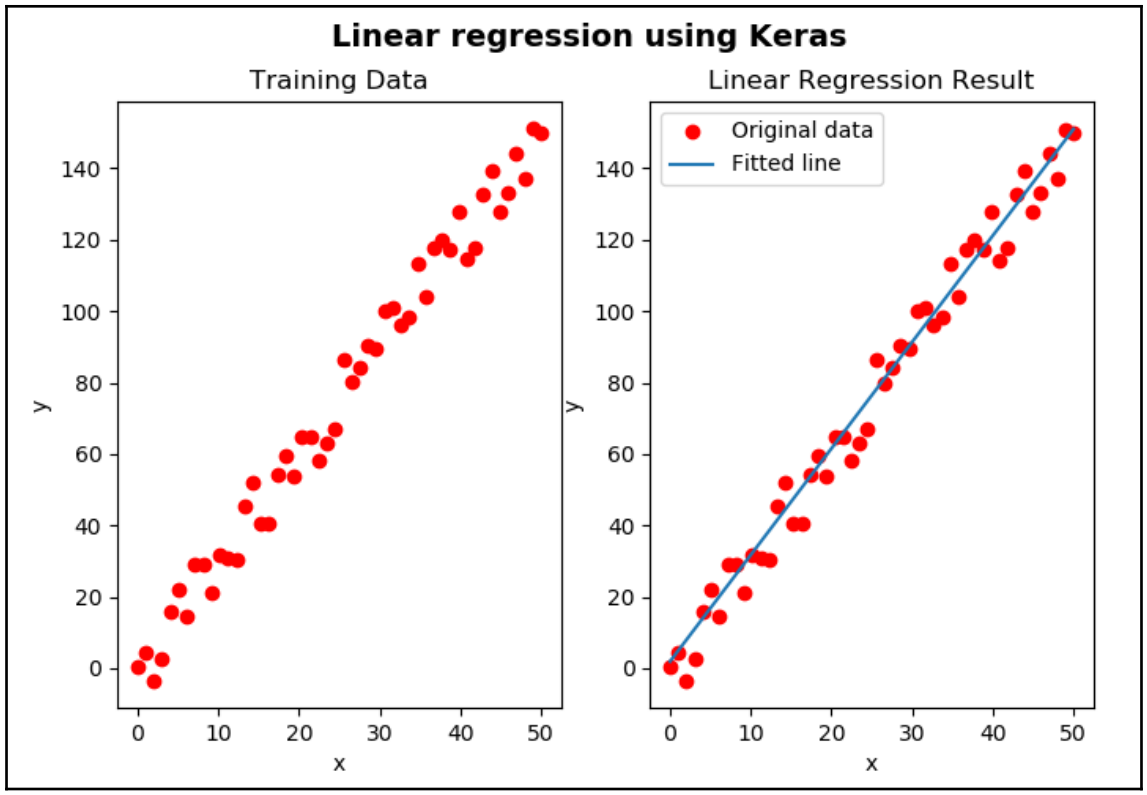

Object detection using OpenCV DNN module and YOLO V3
YOLO V3 for object detection
aeroplane: 0.9271
bicycle: 0.9735
bird: 0.9842
horse: 0.9882
pottedplant: 0.9615
bus: 0.9714
boat: 0.9311
cat: 0.9585
chair: 0.9827
cow: 0.8237
car: 0.8131

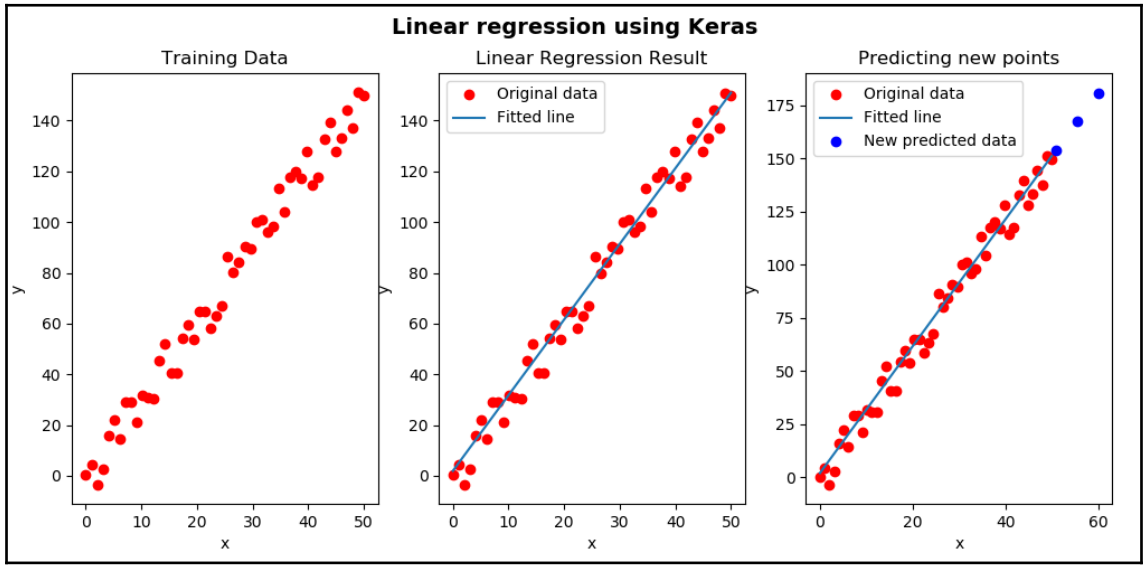

TensorBoard
GRAPHS
INACTIVE
Search nodes. Regexes supported.
Fit to Screen
Download PNG
Run (1)
Session runs (0)
Upload
Choose File
Close legend.
Graph (* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
my_multi...
X_1
X_2
my_multiplication
Operation: Mul
Attributes (1)
T {"type":"DT_INT16"}
Inputs (2)
X_1
X_2
Outputs (0)
Remove from main graph


TensorBoard
GRAPHS
INACTIVE
Search nodes. Regexes supported.
Fit to Screen
Download PNG
Run (1)
Session runs
(0)
Upload
Choose File
Close legend.
Graph (* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
Operations
X_1
X_2
2 tensors
2 tensors
Operations
Subgraph: 2 nodes
Attributes (0)
Inputs (2)
X_1
X_2
Outputs (0)
Remove from main graph

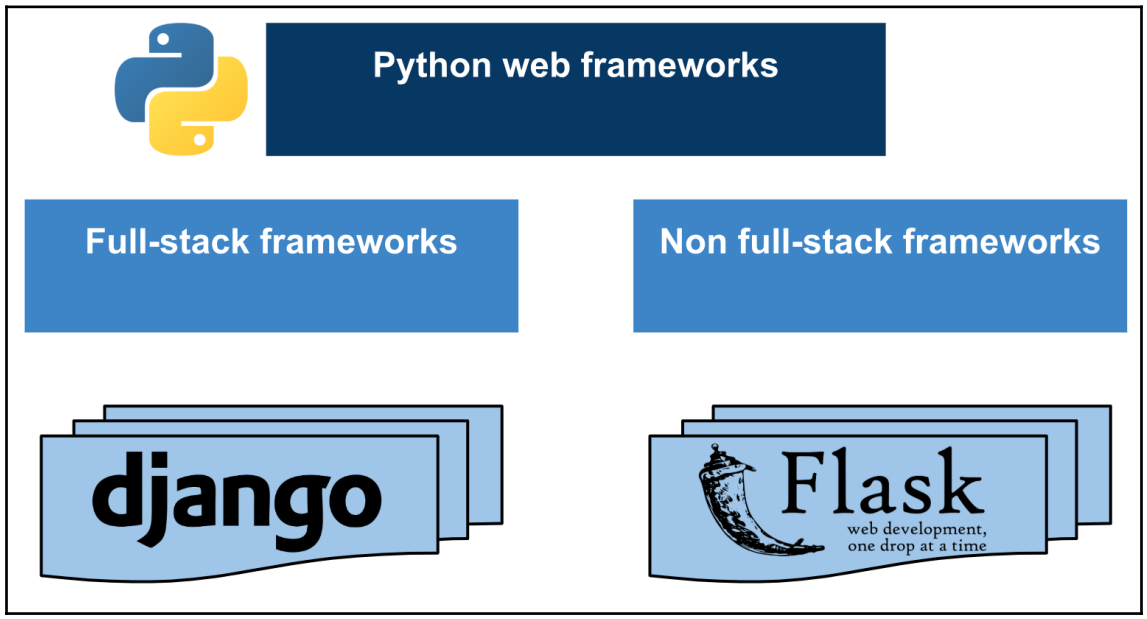

Linear regression using TensorFlow
Training Data
Linear Regression Result
Original data
Fitted line
y
x
0
20
40
60
80
100
120
140
0
10
20
30
40
50

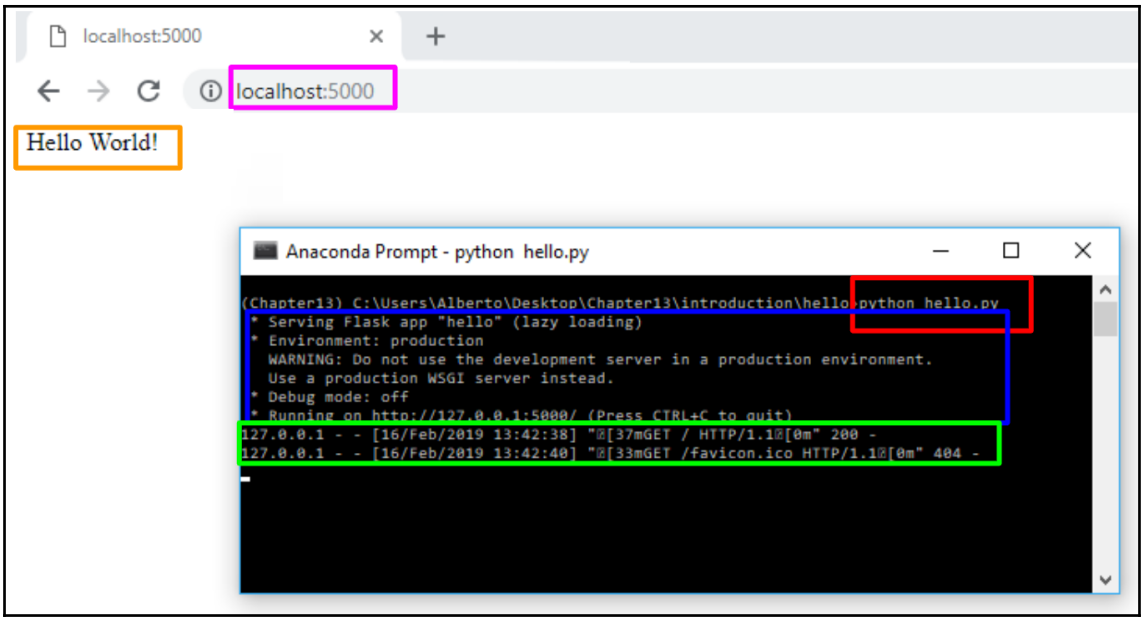

Linear regression using TensorFlow
Training Data
Original data
Linear Regression Result
Original data
Fitted line
Predicting new points
Original data
Fitted line
New predicted data
x
y

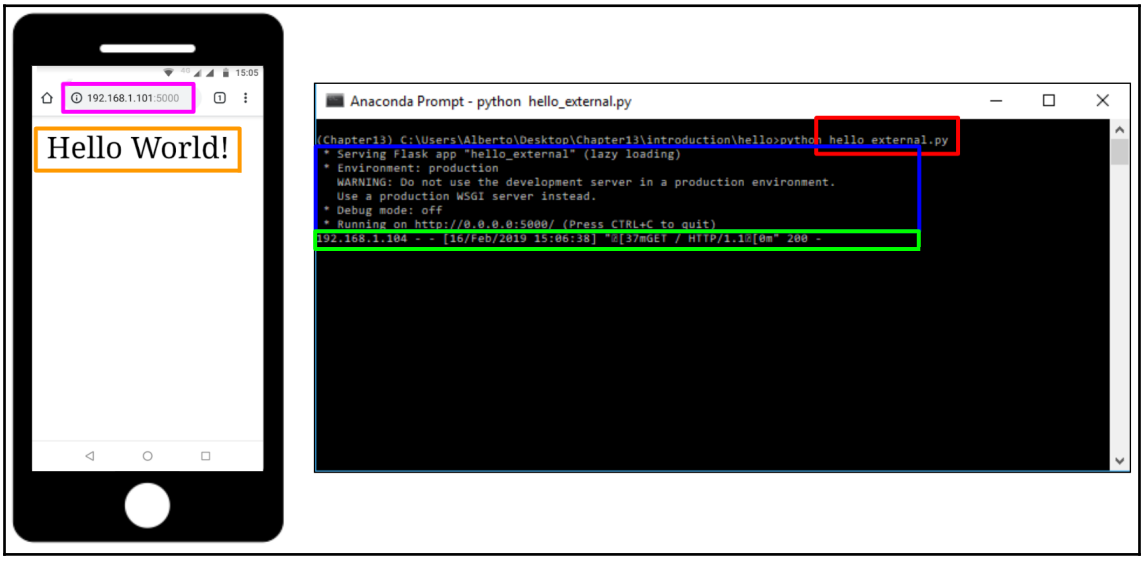

Linear regression using Keras
Training Data
Linear Regression Result
Original data
Fitted line
x
y

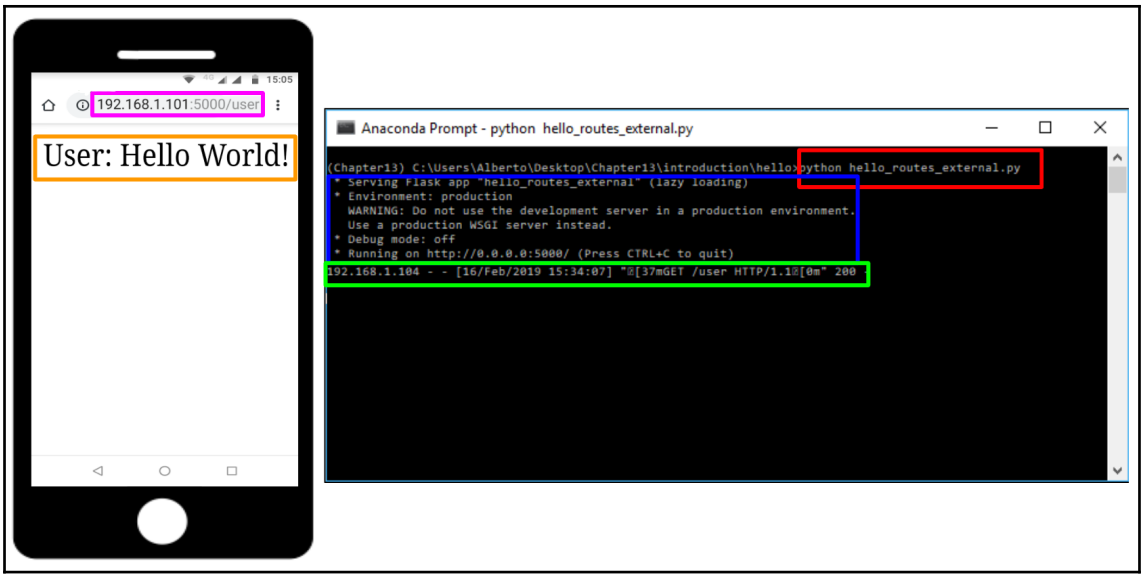

Linear regression using Keras
Training Data
Linear Regression Result
Predicting new points
Original data
Fitted line
New predicted data
x
y

Chapter 13: Mobile and Web Computer Vision with Python and OpenCV




localhost:5000
localhost:5000
Hello World!
Anaconda Prompt - python hello.py
(Chapter13) C:\Users\Alberto\Desktop\Chapter13\introduction\hello>python hello.py
* Serving Flask app "hello" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [16/Feb/2019 13:42:38] "[37mGET / HTTP/1.1[0m" 200 -
127.0.0.1 - - [16/Feb/2019 13:42:40] "[33mGET /favicon.ico HTTP/1.1[0m" 404 -

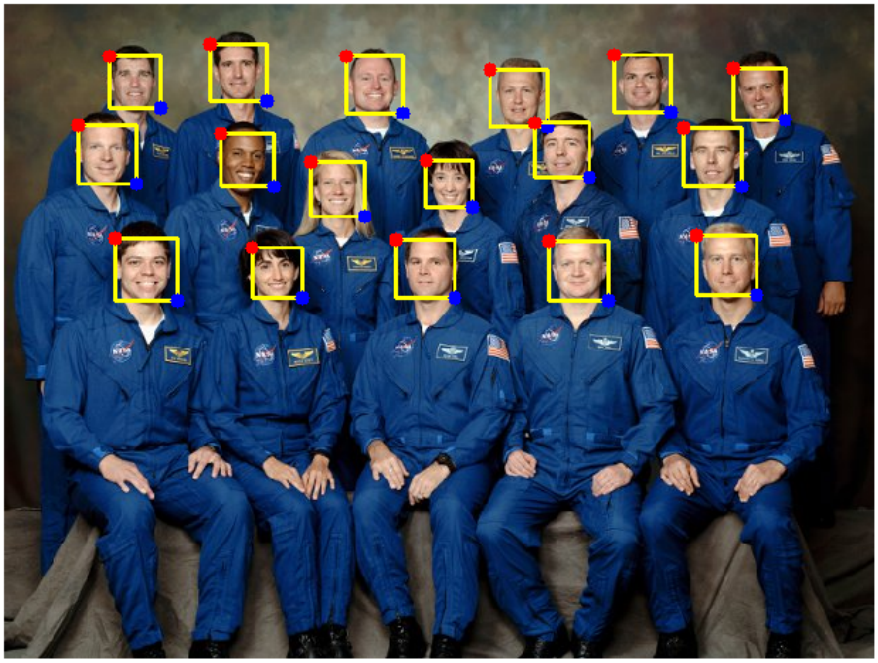

192.168.1.101:5000
Hello World!
Anaconda Prompt - python hello_external.py
(Chapter13) C:\Users\Alberto\Desktop\Chapter13\introduction\hello>python hello_external.py
* Serving Flask app "hello_external" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.1.104 - - [16/Feb/2019 15:06:38] "[37mGET / HTTP/1.1[0m" 200 -


15:05
192.168.1.101:5000/user
User: Hello World!
Anaconda Prompt - python hello_routes_external.py
(Chapter13) C:\Users\Alberto\Desktop\Chapter13\introduction\hello>python hello_routes_external.py
* Serving Flask app "hello_routes_external" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
192.168.1.104 - - [16/Feb/2019 15:34:07] "[37mGET /user HTTP/1.1[0m" 200 -

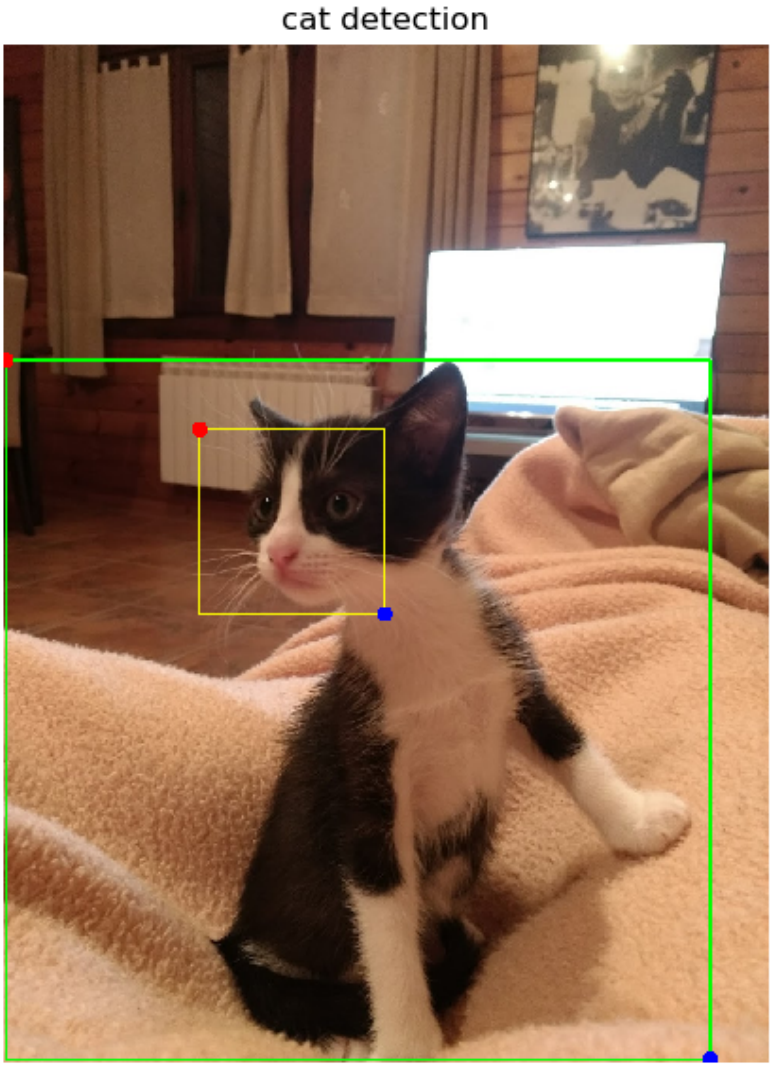

http://192.168.1.101:5000/canny?url=https://raw.githubusercontent.com/opencv/opencv/master/samples/data/lena.jpg
17:44
192.168.1.101:5000/ca

Using face API

face detection



Using cat detection API



Keras Applications: Models for image classification with weights trained on ImageNet

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Image classification in Keras using several pre-trained models

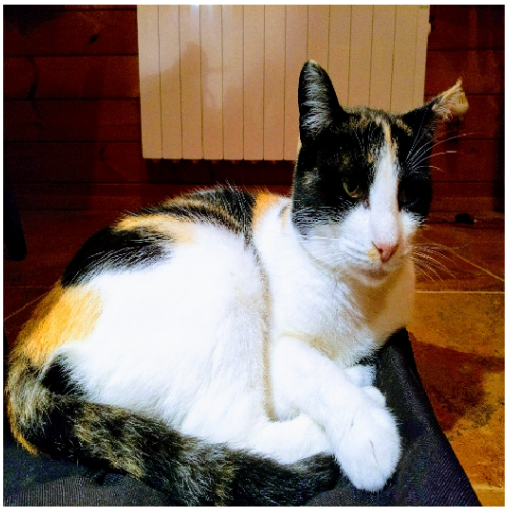
source image

classification results

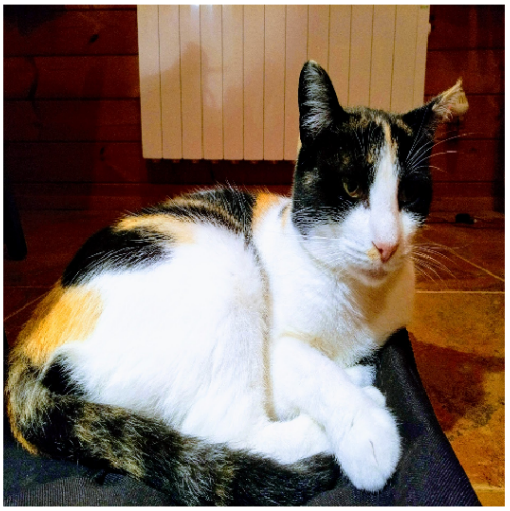
InceptionV3: sports_car, 0.53
VGG16: minivan, 0.38
VGG19: minivan, 0.23
ResNet50: sports_car, 0.29
MobileNet: sports_car, 0.29
Xception: sports_car, 0.34
NASNetMobile: sports_car, 0.55
DenseNet121: sports_car, 0.65

Image classification in Keras using several pre-trained models

source image



classification results

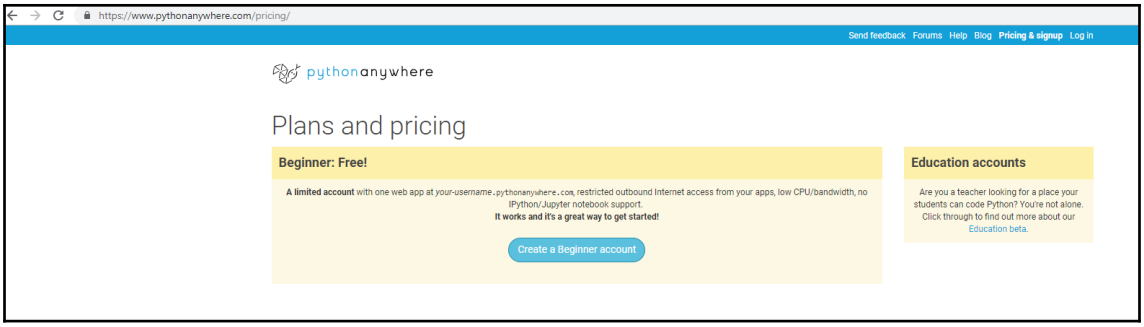


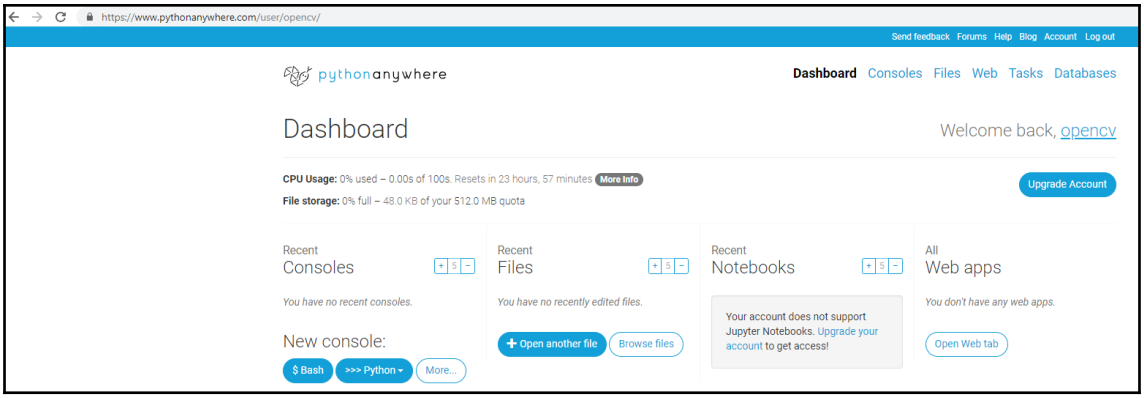
InceptionV3: tabby, 0.39
VGG16: tabby, 0.22
VGG19: Cardigan, 0.16
ResNet50: doormat, 0.29
MobileNet: tabby, 0.18
Xception: Egyptian_cat, 0.24
NASNetMobile: tabby, 0.39
DenseNet121: spotlight, 0.43

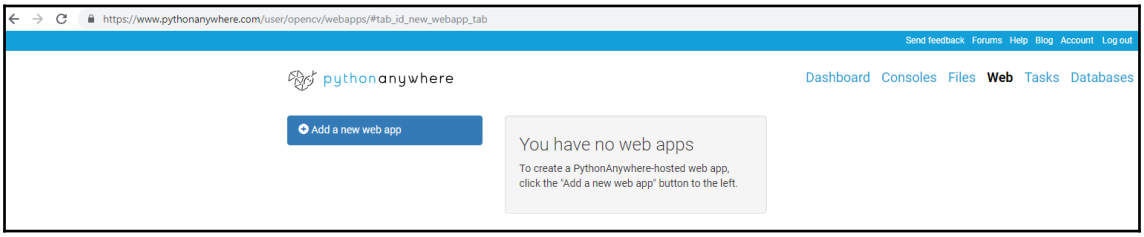
Using Keras Deep Learning REST API

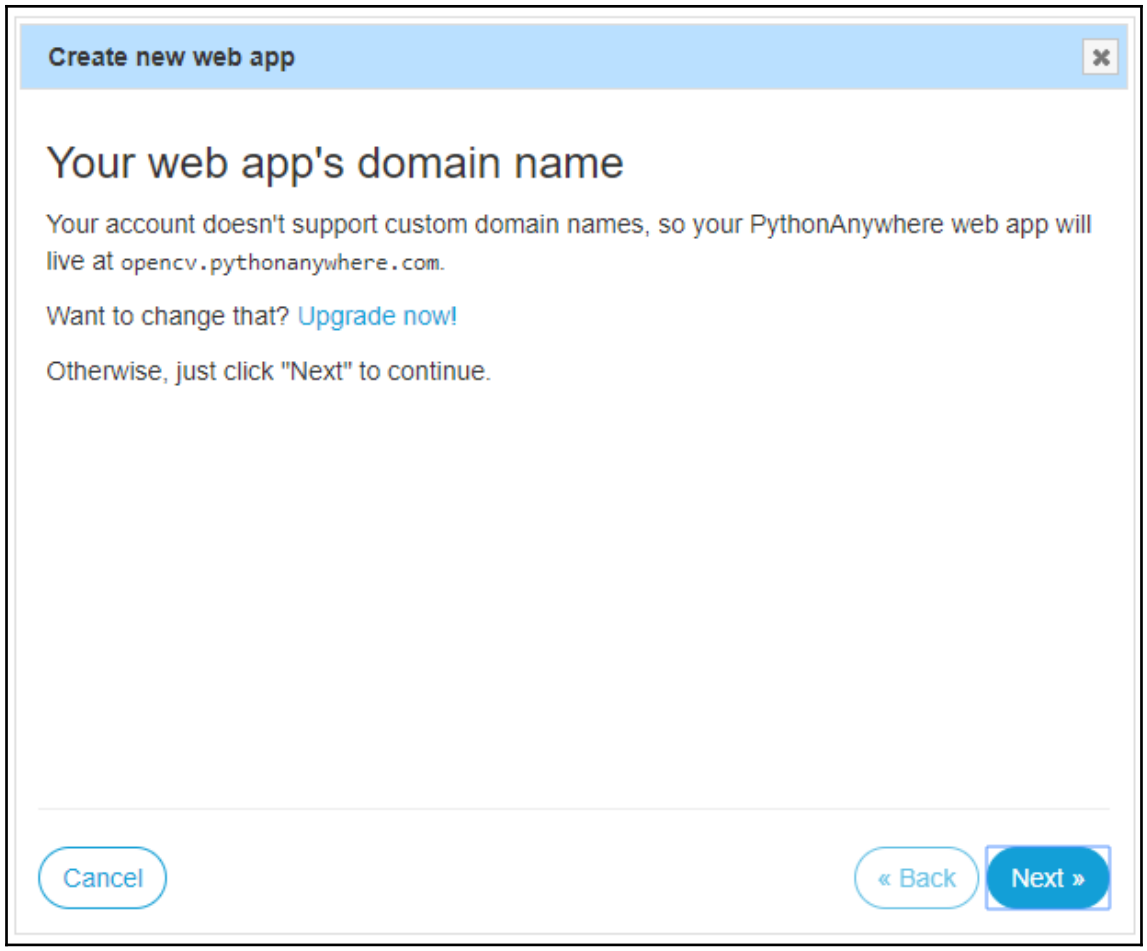
Classification results (NASNetMobile)

1. sports_car: 0.5461
2. convertible: 0.2798
3. grille: 0.0373
4. car_wheel: 0.0268
5. beach_wagon: 0.0142


https://www.pythonanywhere.com/pricing/
Send feedback Forums Help Blog Pricing & signup Log in
pythonanywhere
Plans and pricing
Beginner: Free!
A limited account with one web app at your-username.pythonanywhere.com, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.
It works and it's a great way to get started!
Create a Beginner account
Education accounts
Are you a teacher looking for a place your students can code Python? You're not alone. Click through to find out more about our Education beta.


https://www.pythonanywhere.com/user/opencv/
Send feedback Forums Help Blog Account Log out
pythonanywhere
Dashboard Consoles Files Web Tasks Databases
Dashboard
Welcome back, opencv
CPU Usage: 0% used – 0.00s of 100s. Resets in 23 hours, 57 minutes More Info
File storage: 0% full – 48.0 KB of your 512.0 MB quota
Upgrade Account
Recent Consoles
You have no recent consoles.
New console:
$ Bash >>> Python More...
Recent Files
You have no recently edited files.
Open another file Browse files
Recent Notebooks
Your account does not support Jupyter Notebooks. Upgrade your account to get access!
All Web apps
You don't have any web apps.
Open Web tab


https://www.pythonanywhere.com/user/opencv/webapps/#tab_id_new_webapp_tab
Send feedback Forums Help Blog Account Log out
pythonanywhere
Dashboard Consoles Files Web Tasks Databases
Add a new web app
You have no web apps
To create a PythonAnywhere-hosted web app, click the "Add a new web app" button to the left.


Create new web app
Your web app's domain name
Your account doesn't support custom domain names, so your PythonAnywhere web app will live at opencv.pythonanywhere.com.
Want to change that? Upgrade now!
Otherwise, just click "Next" to continue.
Cancel
« Back
Next »

Create new web app

Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. **If this file already exists, its contents will be overwritten with the new app.**

Path

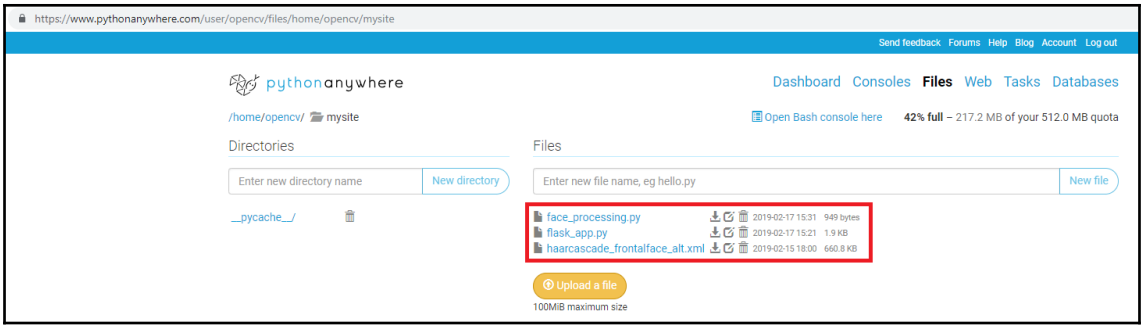
/home/opencv/mysite/flask_app.py

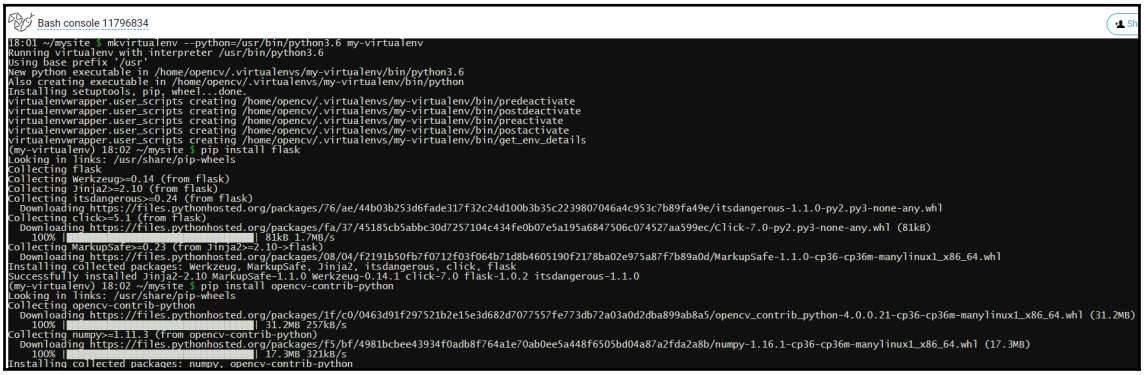
Cancel « Back Next »

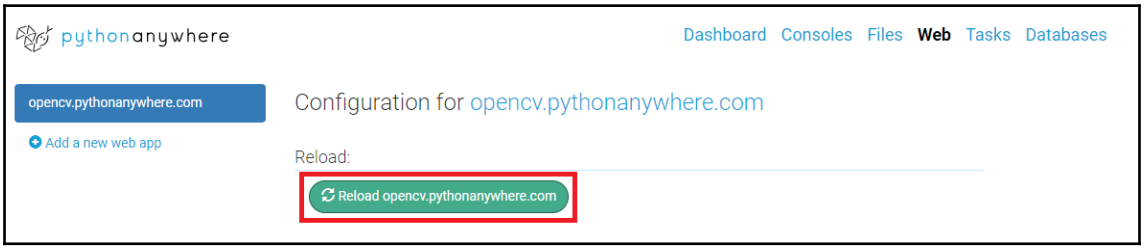
Code:

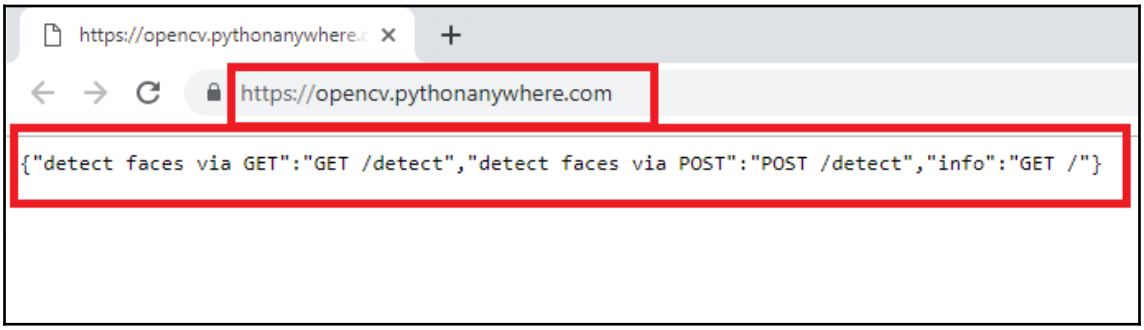
What your site is running.

Source code:	/home/opencv/mysite	Go to directory
Working directory:	/home/opencv/	Go to directory
WSGI configuration file:	/var/www/opencv_pythonanywhere_com_wsgi.py	
Python version:	3.6	


https://www.pythonanywhere.com/user/opencv/files/home/opencv/mysite
Send feedback Forums Help Blog Account Log out
pythonanywhere
Dashboard Consoles Files Web Tasks Databases
/home/opencv/ mysite
Open Bash console here
42% full – 217.2 MB of your 512.0 MB quota
Directories
Files
Enter new directory name
New directory
Enter new file name, eg hello.py
New file
__pycache__/
face_processing.py 2019-02-17 15:31 949 bytes
flask_app.py 2019-02-17 15:21 1.9 KB
haarcascade_frontalface_alt.xml 2019-02-15 18:00 660.8 KB
Upload a file
100MiB maximum size


Bash console 11796834
18:01 ~/mysite $ mkvirtualenv --python=/usr/bin/python3.6 my-virtualenv
Running virtualenv with interpreter /usr/bin/python3.6
Using base prefix '/usr'
New python executable in /home/opencv/.virtualenvs/my-virtualenv/bin/python3.6
Also creating executable in /home/opencv/.virtualenvs/my-virtualenv/bin/python
Installing setuptools, pip, wheel...done.
virtualenvwrapper.user_scripts creating /home/opencv/.virtualenvs/my-virtualenv/bin/predeactivate
virtualenvwrapper.user_scripts creating /home/opencv/.virtualenvs/my-virtualenv/bin/postdeactivate
virtualenvwrapper.user_scripts creating /home/opencv/.virtualenvs/my-virtualenv/bin/preactivate
virtualenvwrapper.user_scripts creating /home/opencv/.virtualenvs/my-virtualenv/bin/postactivate
virtualenvwrapper.user_scripts creating /home/opencv/.virtualenvs/my-virtualenv/bin/get_env_details
(my-virtualenv) 18:02 ~/mysite $ pip install flask
Looking in links: /usr/share/pip-wheels
Collecting flask
Collecting Werkzeug>=0.14 (from flask)
Collecting Jinja2>=2.10 (from flask)
Collecting itsdangerous>=0.24 (from flask)
Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting click>=5.1 (from flask)
Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl (81kB)
100% | 81kB 1.7MB/s
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10->flask)
Downloading https://files.pythonhosted.org/packages/98/94/f2191b50fb7f0712f03f064b71d8b4605190f2178ba02e975a87f7b89a0d/MarkupSafe-1.1.0-cp36-cp36m-manylinux1_x86_64.whl
Installing collected packages: Werkzeug, MarkupSafe, Jinja2, itsdangerous, click, flask
Successfully installed Jinja2-2.10 MarkupSafe-1.1.0 Werkzeug-0.14.1 click-7.0 flask-1.0.2 itsdangerous-1.1.0
(my-virtualenv) 18:02 ~/mysite $ pip install opencv-contrib-python
Looking in links: /usr/share/pip-wheels
Collecting opencv-contrib-python
Downloading https://files.pythonhosted.org/packages/1f/c0/0463d91f297521b2e15e3d682d7077557fe773db72a03a0d2dba899ab8a5/opencv_contrib_python-4.0.0.21-cp36-cp36m-manylinux1_x86_64.whl (31.2MB)
100% | 31.2MB 257kB/s
Collecting numpy>=1.11.3 (from opencv-contrib-python)
Downloading https://files.pythonhosted.org/packages/f5/bf/4981bcbee43934f0adb8f764a1e70ab0ee5a448f6505bd04a87a2fda2a8b/numpy-1.16.1-cp36-cp36m-manylinux1_x86_64.whl (17.3MB)
100% | 17.3MB 321kB/s
Installing collected packages: numpy, opencv-contrib-python


pythonanywhere
Dashboard Consoles Files Web Tasks Databases
opencv.pythonanywhere.com
Add a new web app
Configuration for opencv.pythonanywhere.com
Reload:
Reload opencv.pythonanywhere.com


https://opencv.pythonanywhere.c
https://opencv.pythonanywhere.com
{"detect faces via GET":"GET /detect","detect faces via POST":"POST /detect","info":"GET /"}

Using face API at http://opencv.pythonanywhere.com/detect

face detection