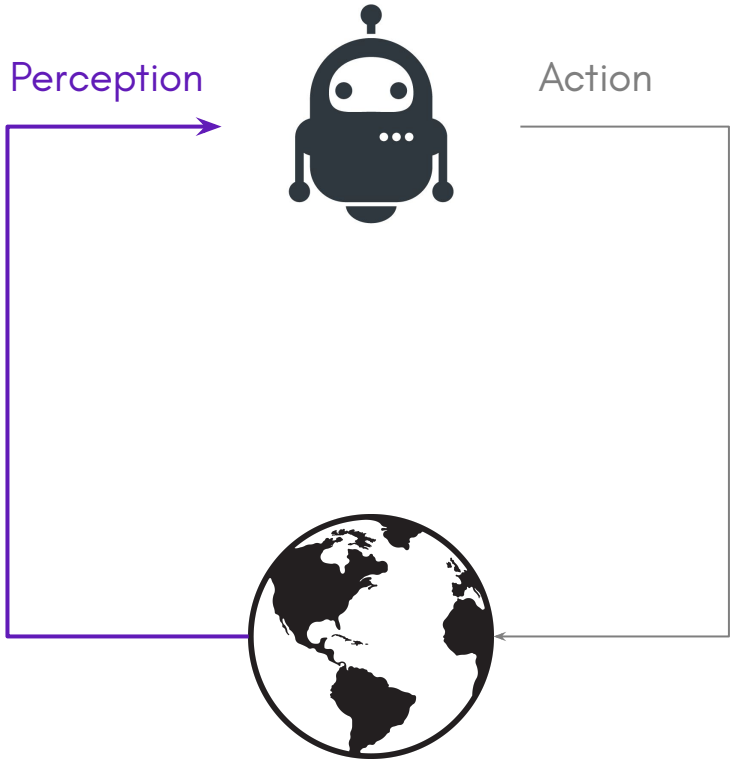
ENGR 3421: Robotics I

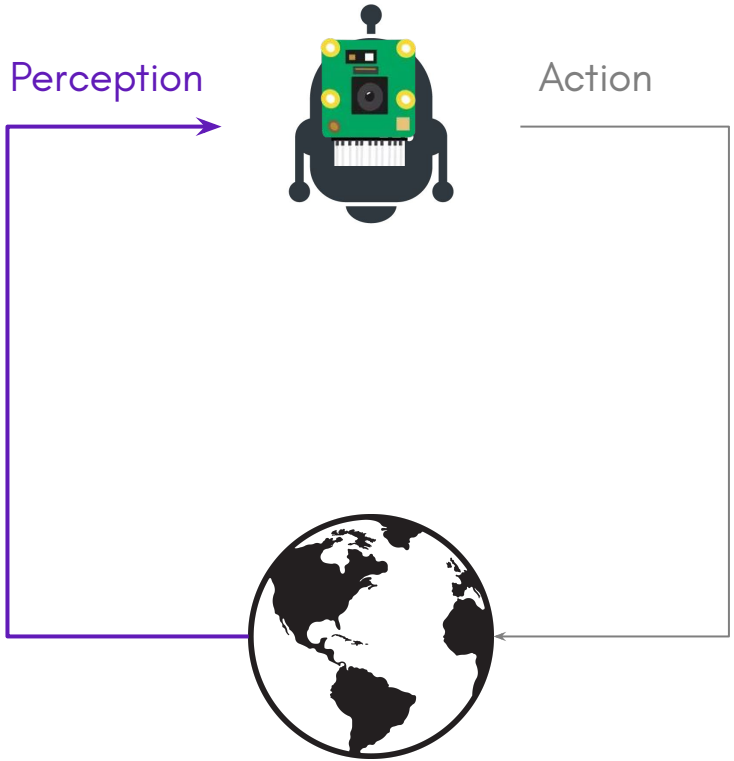
Robotic Vision

11/05/2024

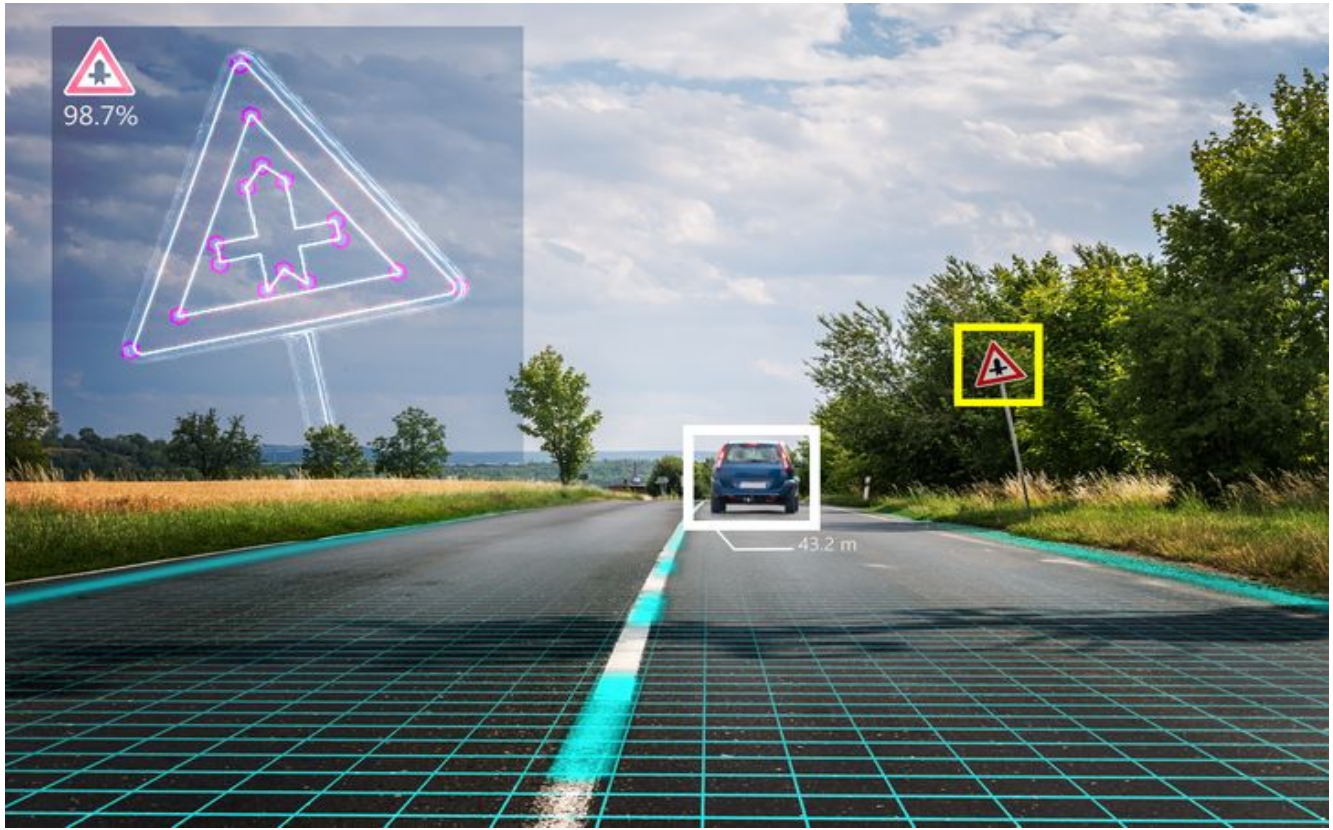
A Robot Needs to See



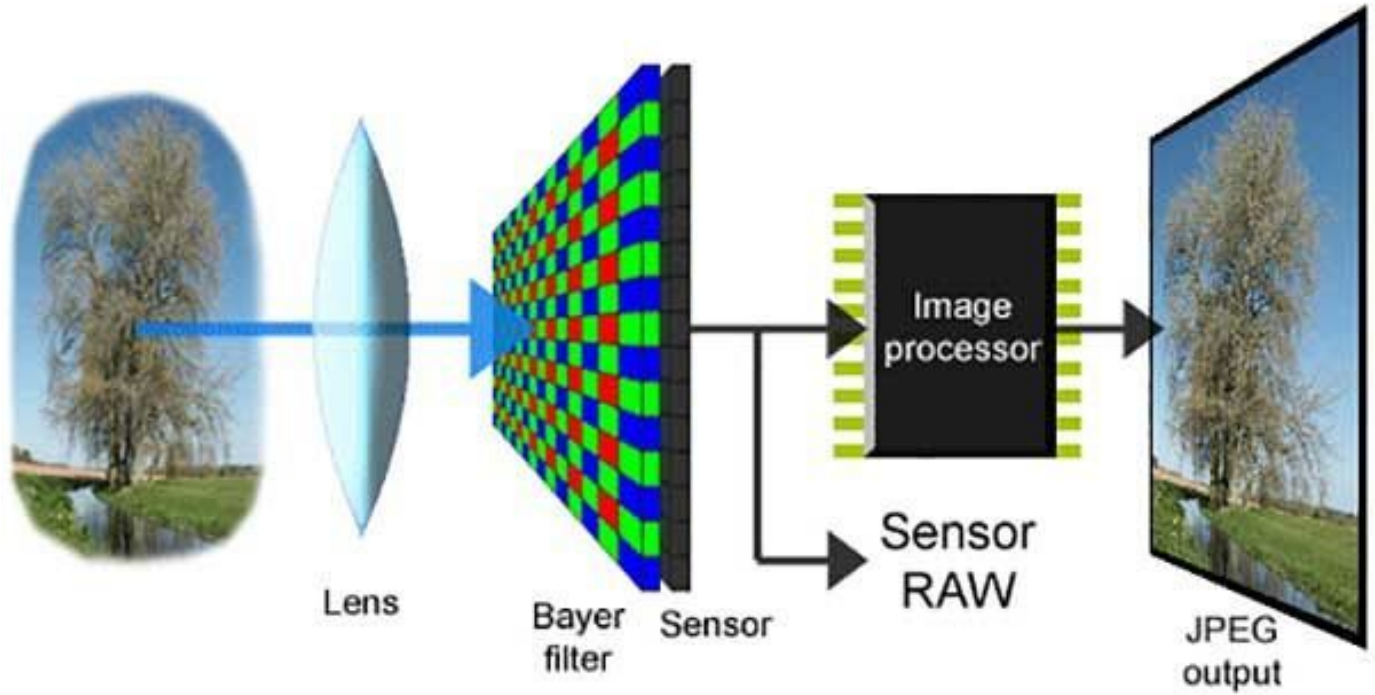
A Robot Needs to See



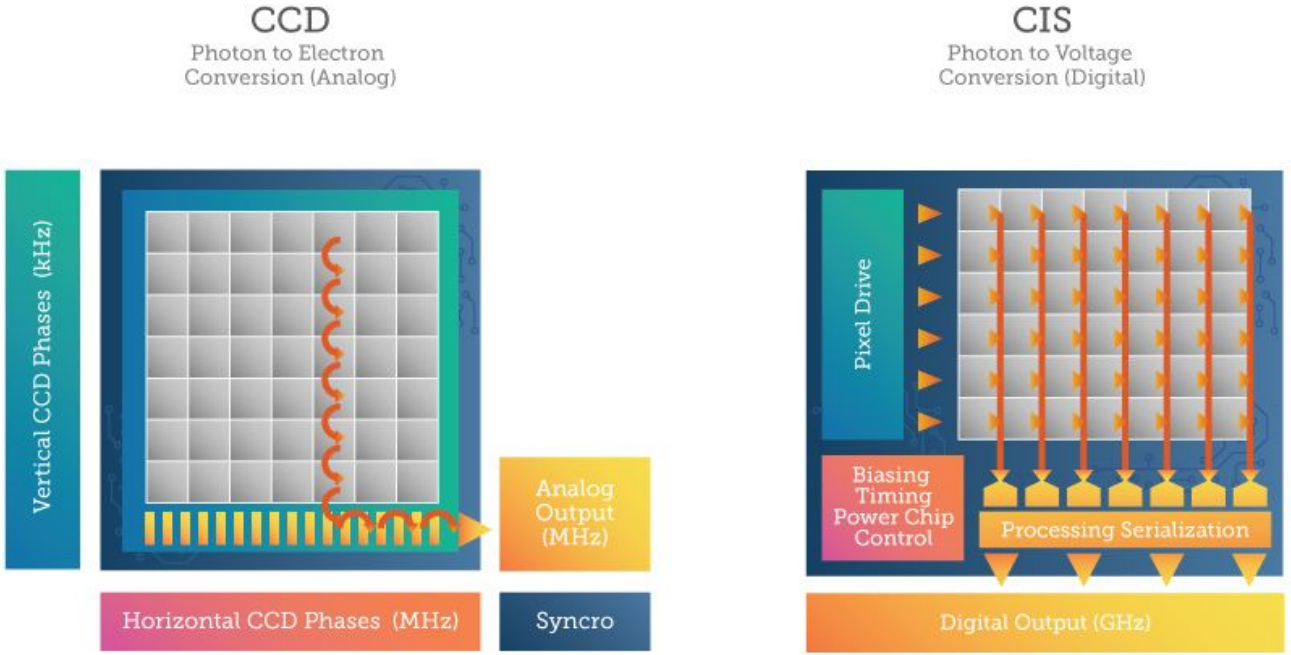
Robotic Vision



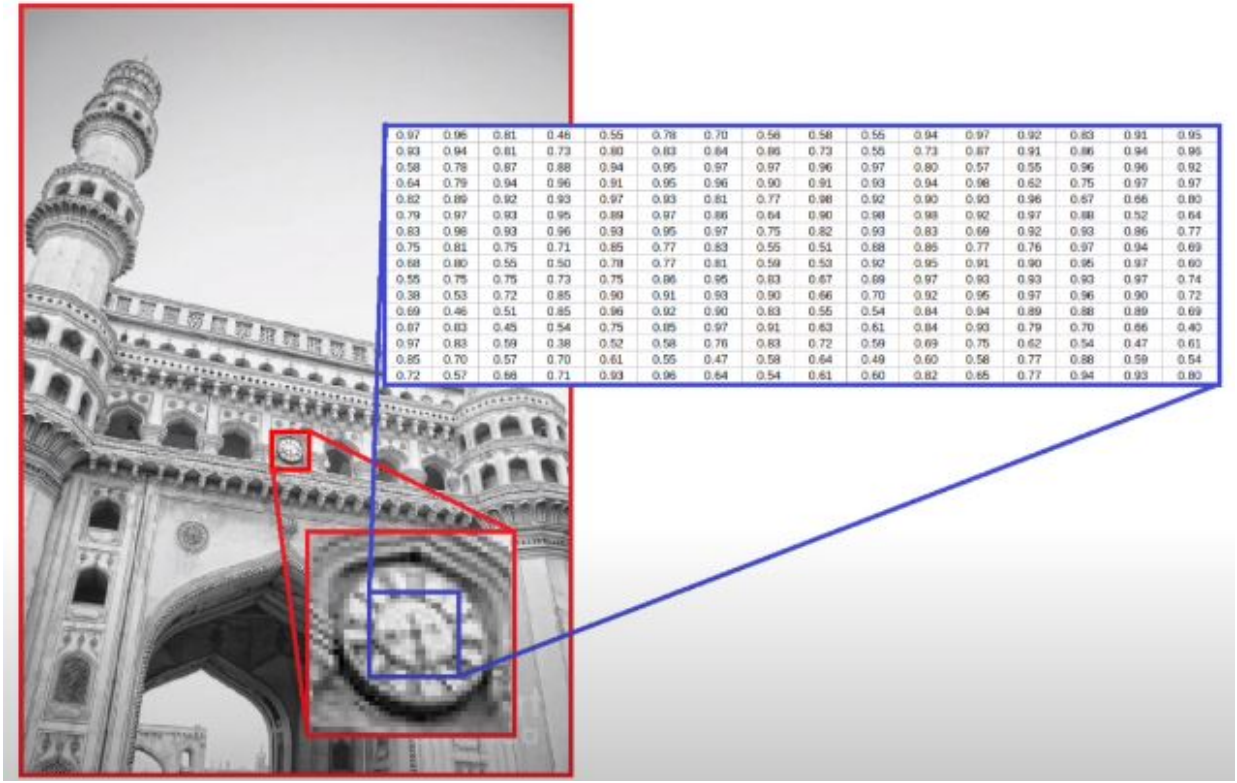
Digital Image Creation



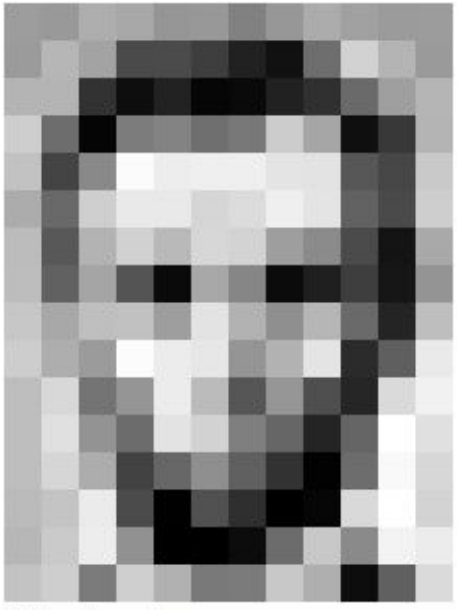
Digital Image Color Channels



Digital Image Representations

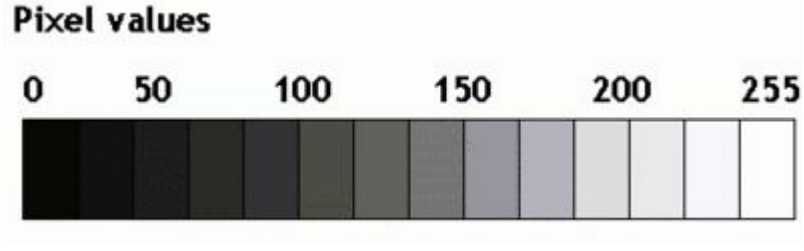


Pixel Intensity

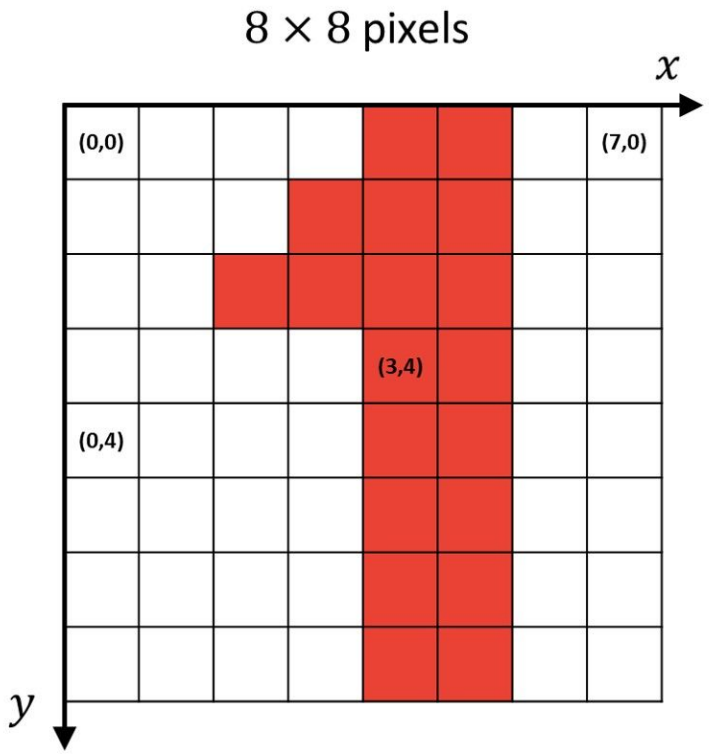


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



Pixel Localization



Grayscale Image

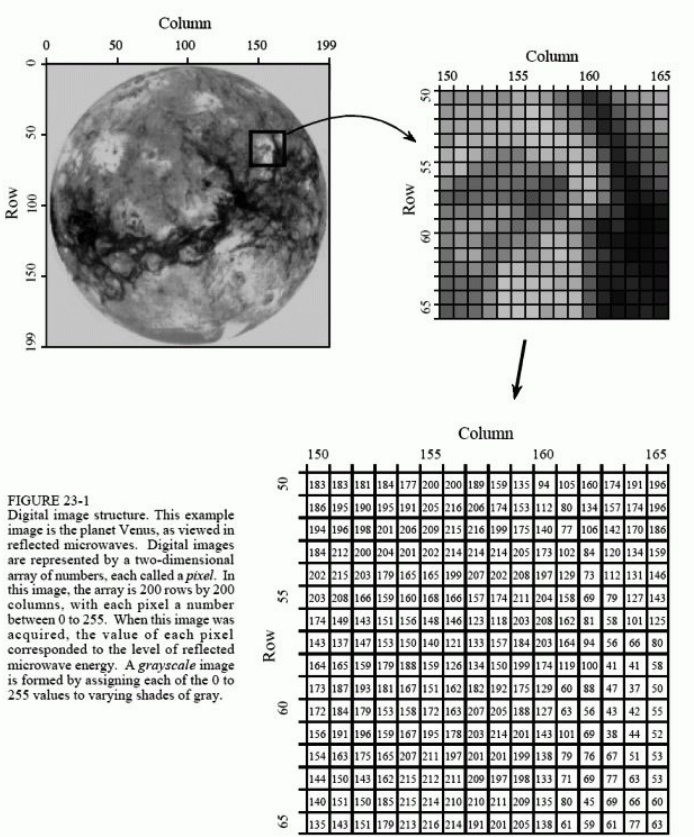
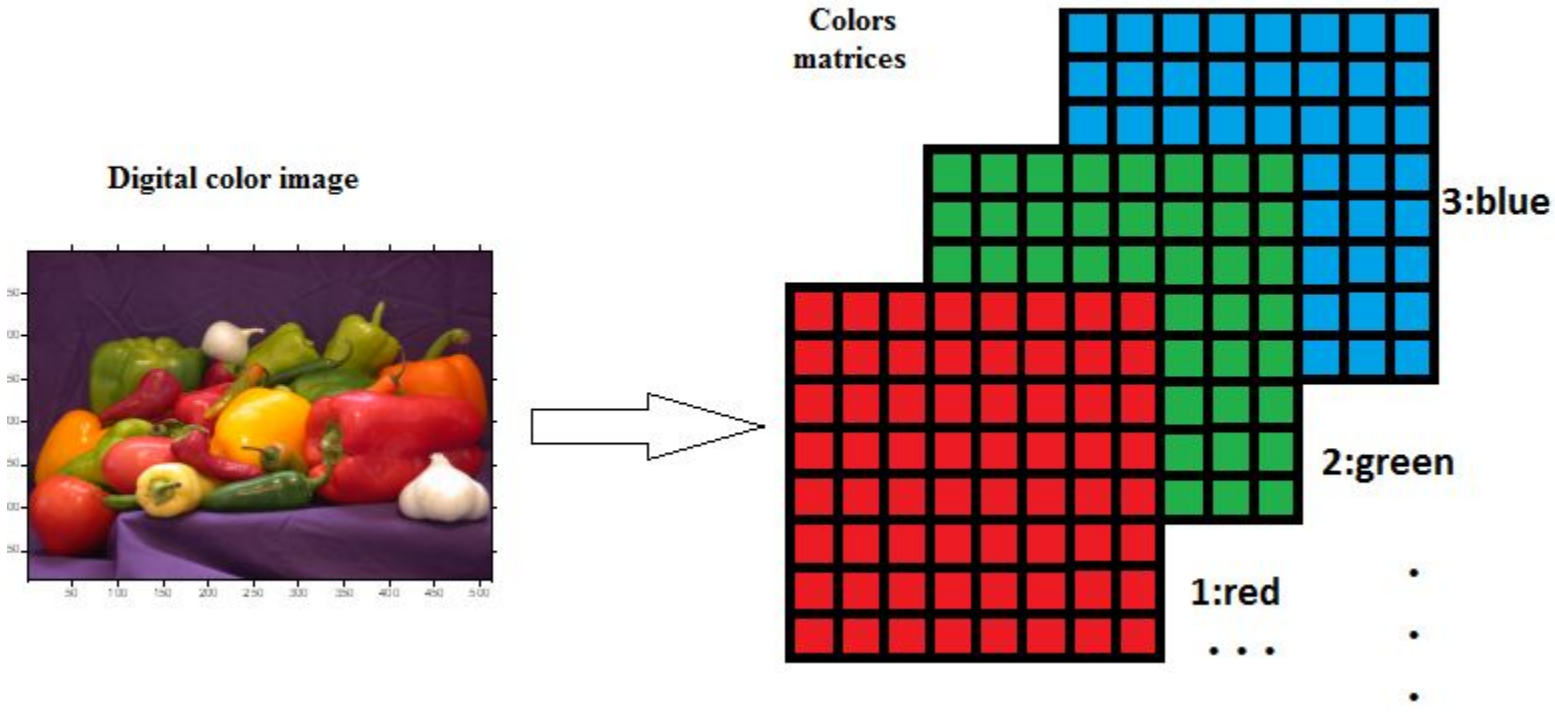


FIGURE 23-1
Digital image structure. This example image is the planet Venus, as viewed in reflected microwaves. Digital images are represented by a two-dimensional array of numbers, each called a *pixel*. In this image, the array is 200 rows by 200 columns, with each pixel a number between 0 to 255. When this image was acquired, the value of each pixel corresponded to the level of reflected microwave energy. A *grayscale* image is formed by assigning each of the 0 to 255 values to varying shades of gray.

Chapter 23, The scientist and engineer's guide to digital signal processing

Color Image



Color Channels

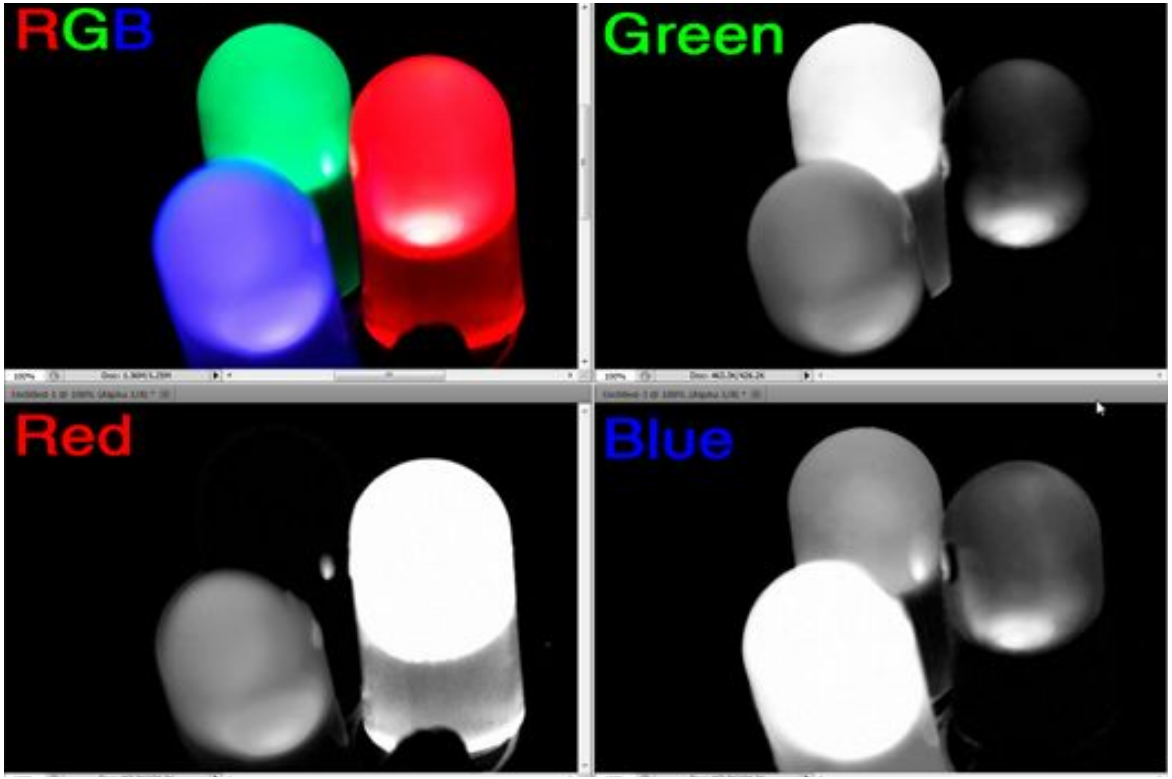


Image Resolution

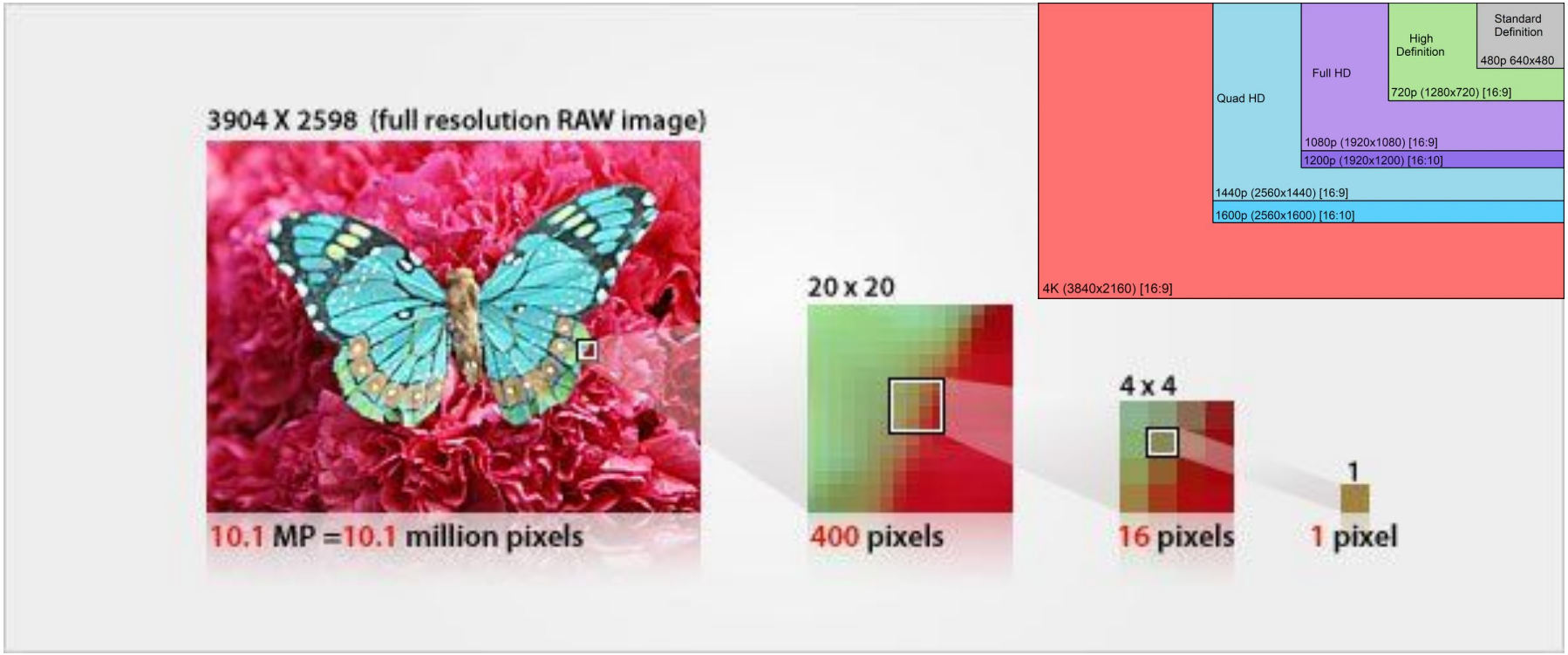


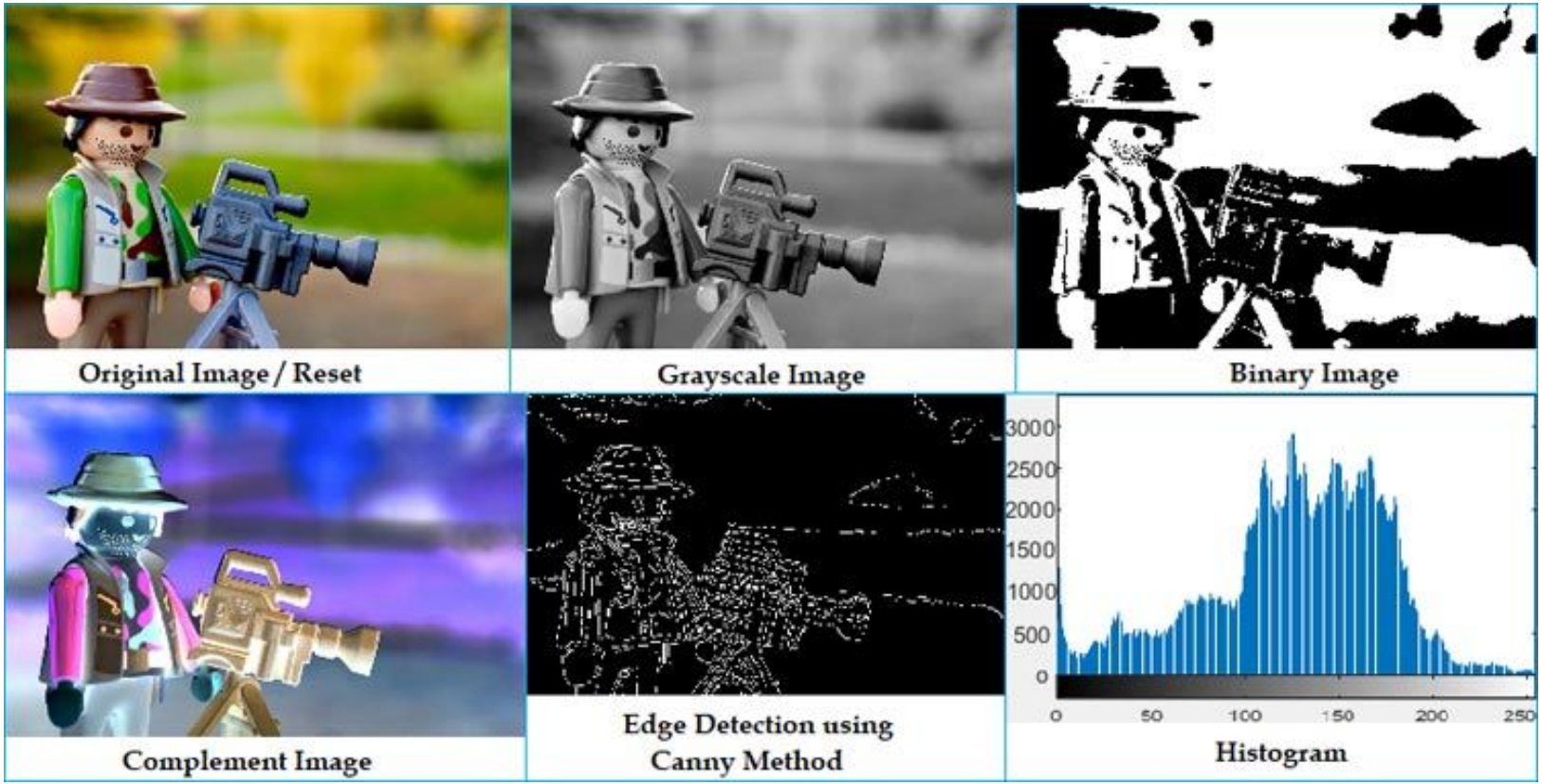
Image File Formats

web, screen printing

image format	colour model	transparency	destination	remarks
JPG	RGB	—	web, screen	generational degradation
TIFF	RGB / CMYK	✓	printing	layered images, image stacks
GIF	RGB	✓	web, screen	limited colour, animated images
PNG	RGB	✓	web, screen	lossless compression

file format	colour model	transparency	destination	remarks
SVG	RGB	✓	web, screen	interactive, scriptable
EPS	RGB / CMYK	✓	printing	PostScript document
PDF	RGB / CMYK	✓	web, screen + printing	includes PostScript, platform independent

Image Processing

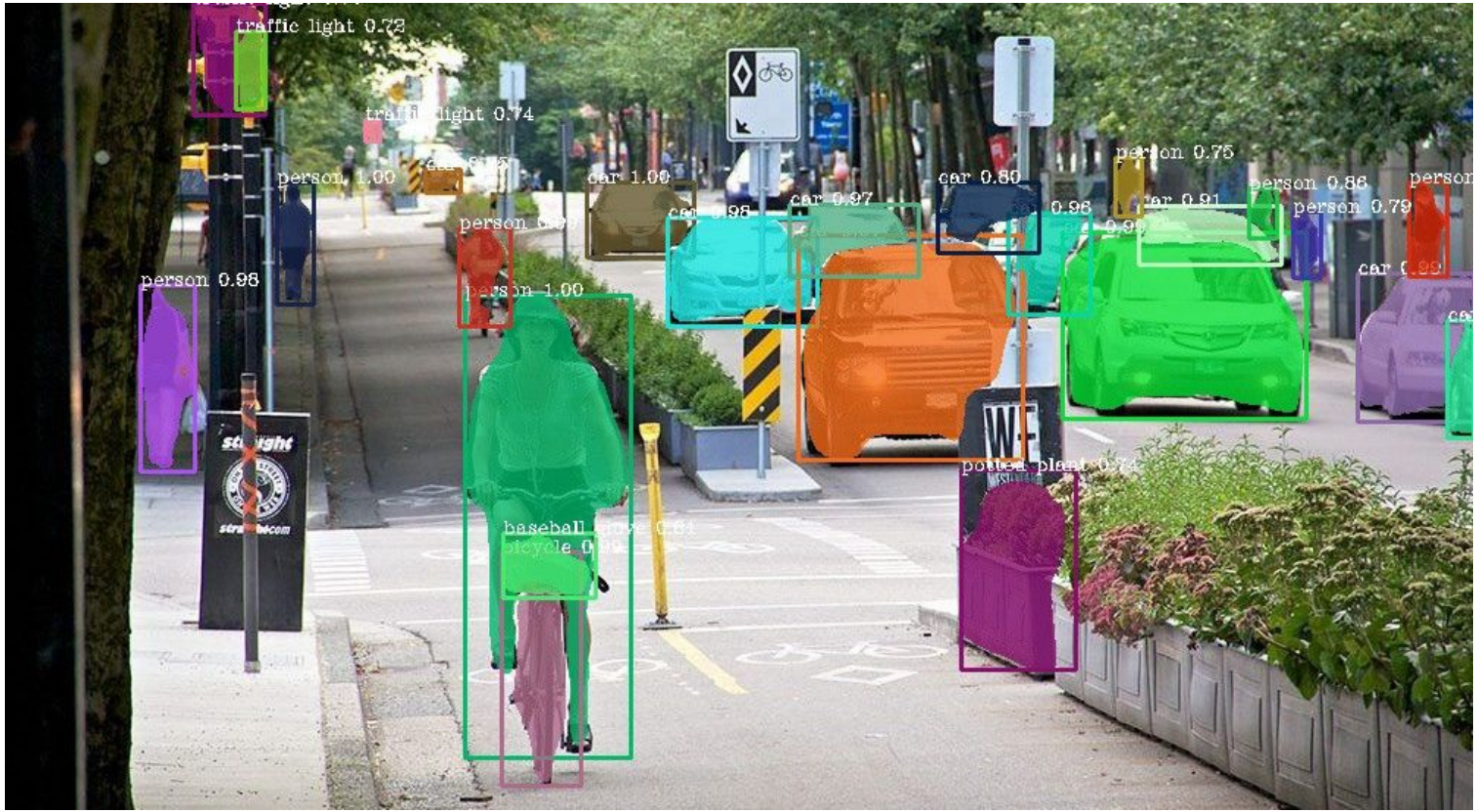


Pixel-level Image Processing

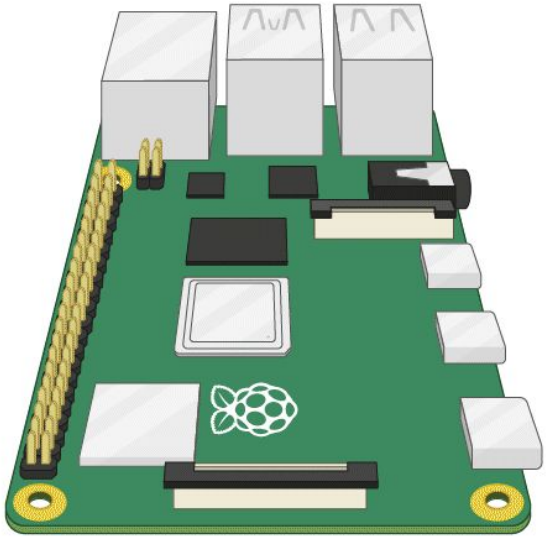
f(x,y) → *f(x,y) + 20*

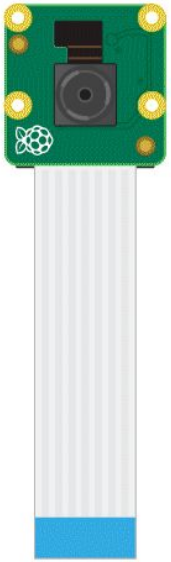
f(x,y) → *f(-x,y)*

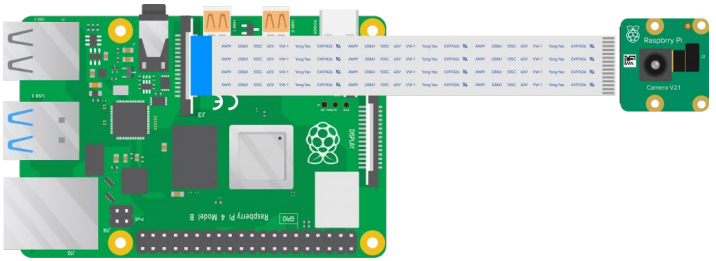
High-level Image Processing



Raspberry Pi Camera

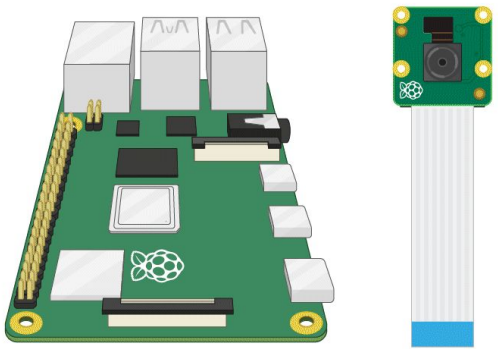






OpenCV is open source, contains over 2500 algorithms, and is operated by the non-profit Open Source Vision Foundation.

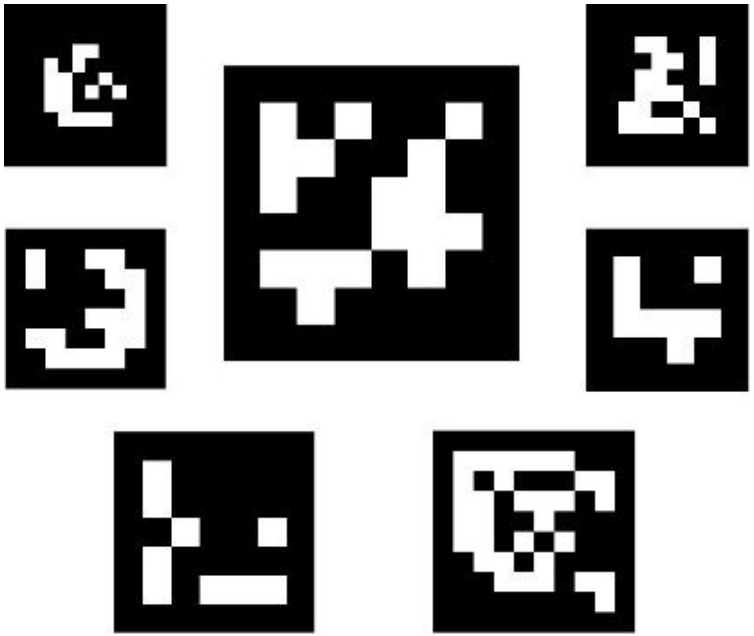
```
pip install opencv-python --break-system-packages
```

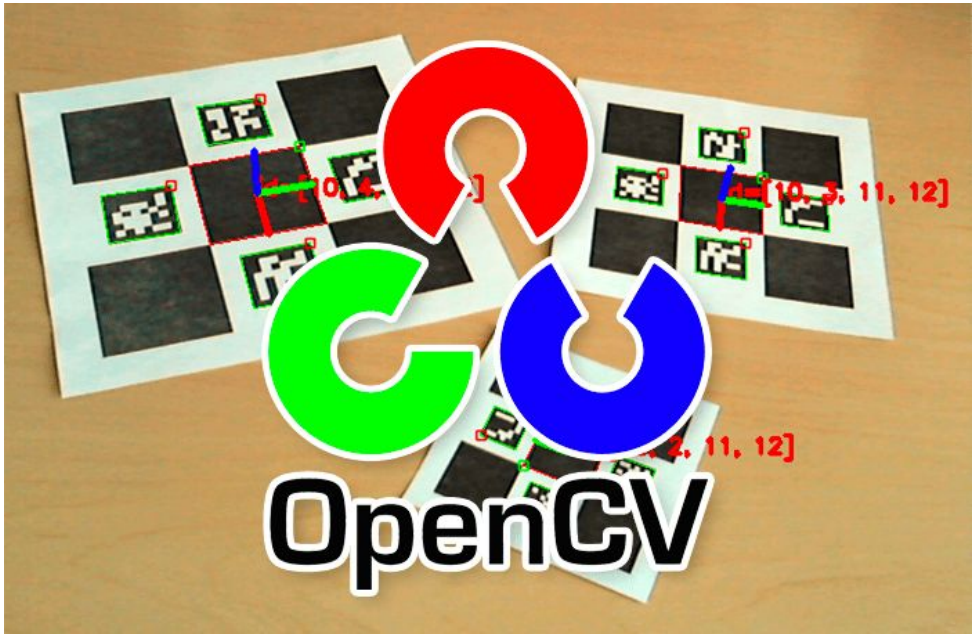


OpenCV Video Capture

```
import cv2 as cv
from picamera2 import Picamera2

# SETUP
cam = Picamera2()
config = cam.create_still_configuration()
cam.configure(config)
cam.start()

# LOOP
while True:
    im = cam.capture_array()
    im_rgb = cv.cvtColor(im, cv.COLOR_BGR2RGB)
    im_resize = cv.resize(im_rgb, (800, 600))
    cv.imshow("Camera", im_resize)
    if cv.waitKey(1) == ord('q'):
        break
```

ArUco Marker Detection



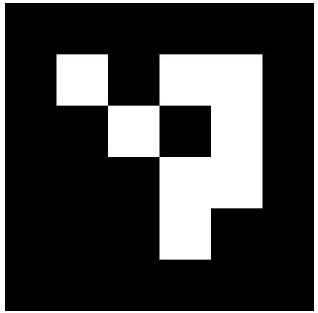
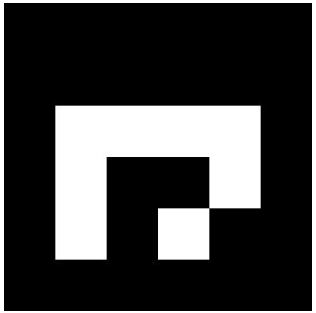
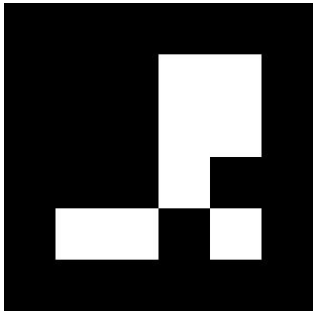
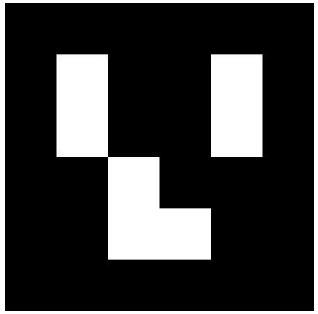
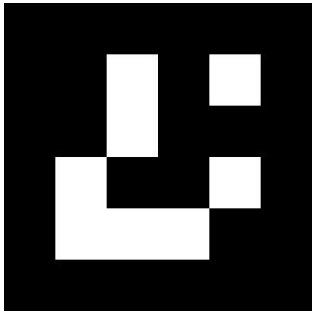
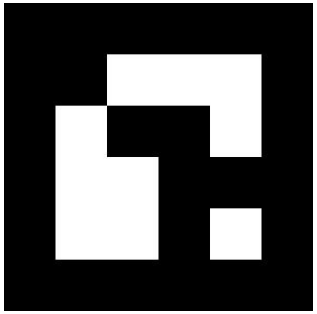
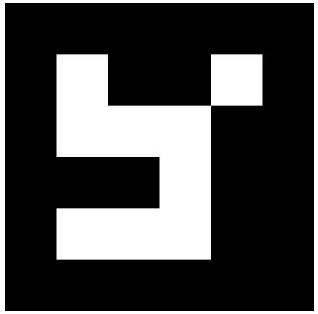
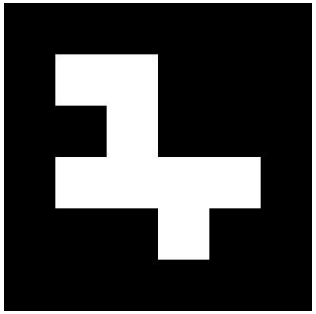
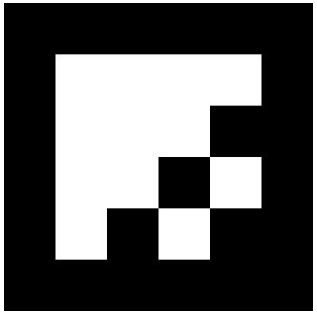


OpenCV ArUco Resources

- Official Tutorial (C++): https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
- Pyimagesearch Tutorial: https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/
- Video Tutorial: https://youtu.be/cIVZRuVdv1o

Generate ArUco Markers

```
import numpy as np

import cv2


aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_4X4_50)  # get ArUco dictionary

aruco_arr = np.zeros((300, 300, 1), dtype=np.uint8)  # create an array to save marker

for i in range(9):  # generate 9 markers

    cv2.aruco.drawMarker(aruco_dict, i, 300, aruco_arr, 1)

    cv2.imwrite('aruco/4x4_' + str(i) + '.jpg', aruco_arr)
```

Detect ArUco Markers

```
import cv2 as cv
from picamera2 import Picamera2
import numpy as np

# SETUP
cam = Picamera2()
config = cam.create_still_configuration()
cam.configure(config)
cam.start()
aruco_dict = cv.aruco.Dictionary_get(cv.aruco.DICT_4X4_50)  # aruco dictionary
aruco_params = cv.aruco.DetectorParameters_create()

# LOOP
while True:
    if cv.waitKey(1) == ord('q'):
        break
    im = cam.capture_array()
    im_rgb = cv.cvtColor(im, cv.COLOR_BGR2RGB)
    im_resize = cv.resize(im_rgb, (400, 300))
    corners, ids, reject_candidates = cv.aruco.detectMarkers(
        im_resize,
        aruco_dict,
        parameters=aruco_params,
    )
    top_left_coords = corners[0][0][0].astype(int)
    bot_right_coords = corners[0][0][2].astype(int)
    print(corners, ids)
    image = cv.rectangle(im_resize, top_left_coords, bot_right_coords, (0, 255, 0), 2)
    cv.imshow("Camera", image)
```