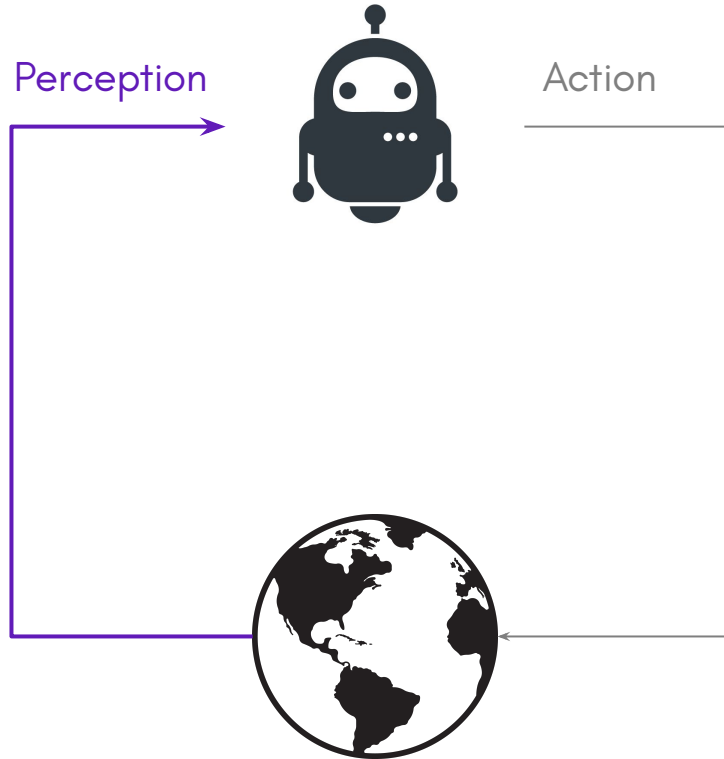


# ENGR 3421: Robotics I

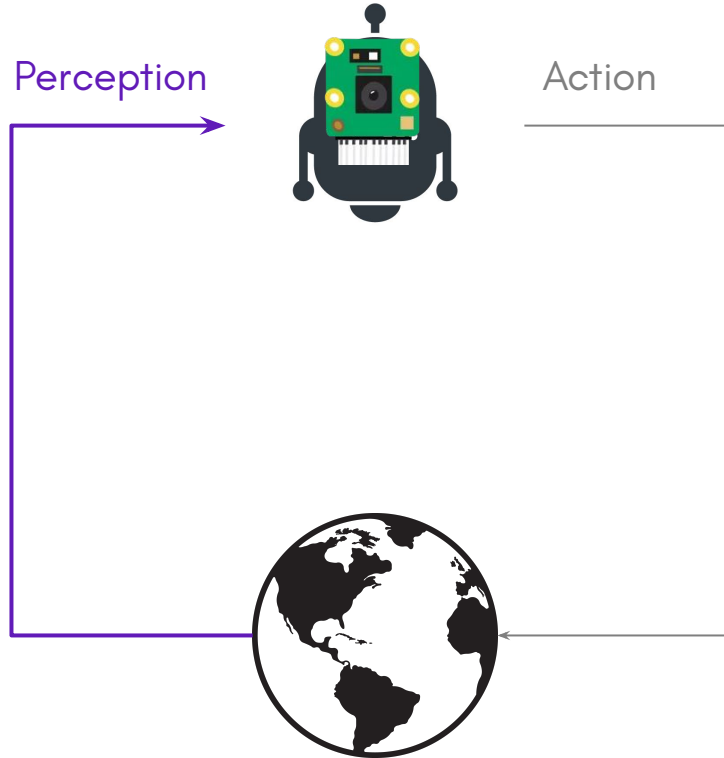
Robotic Vision

11/05/2024

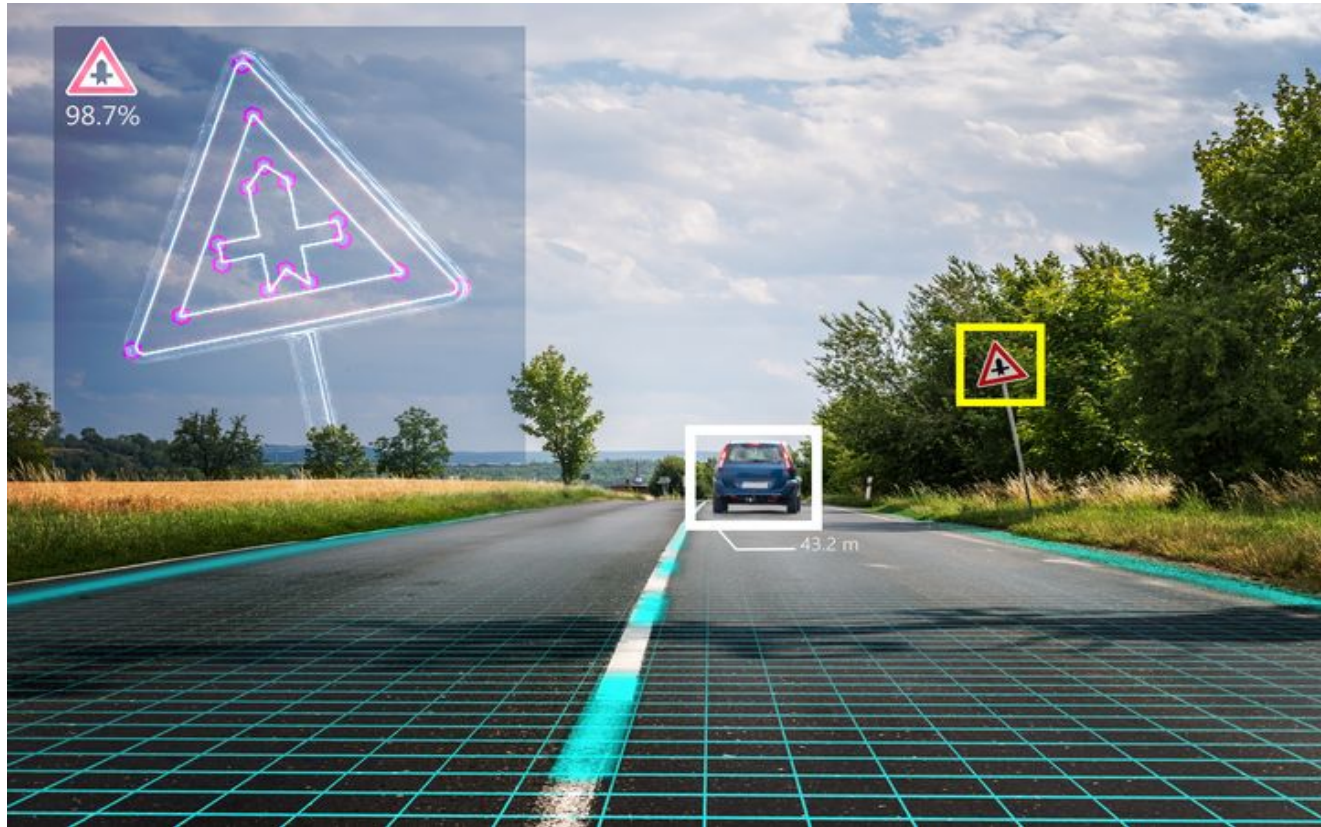
# A Robot Needs to See



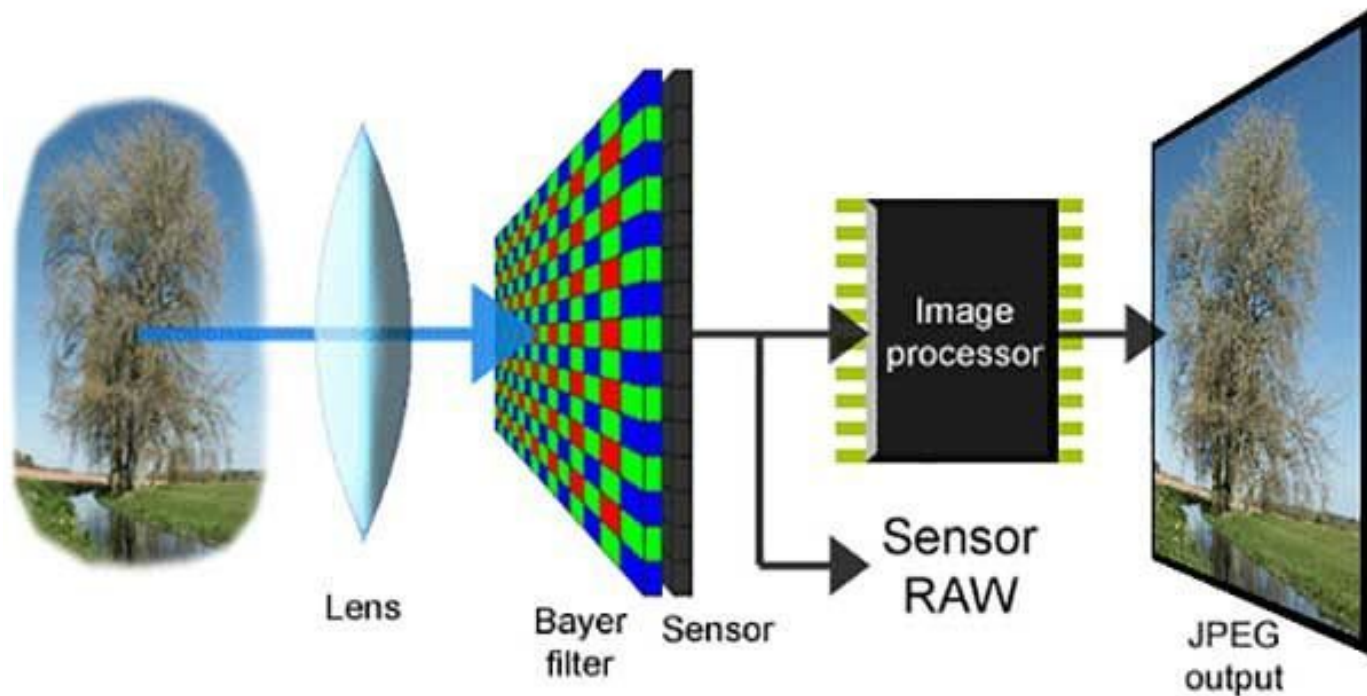
# A Robot Needs to See



# Robotic Vision

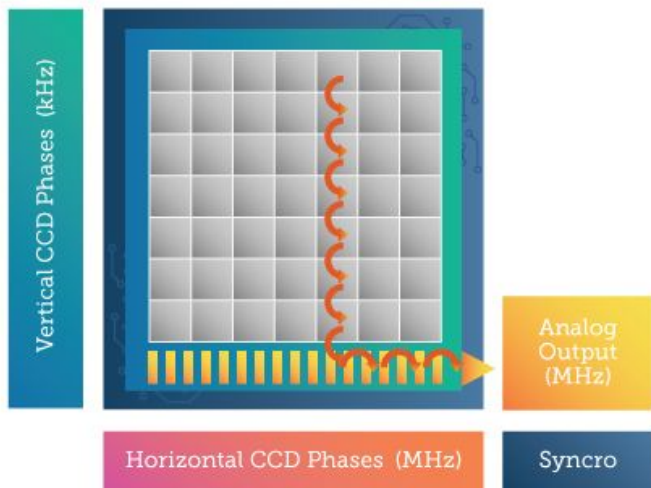


# Digital Image Creation

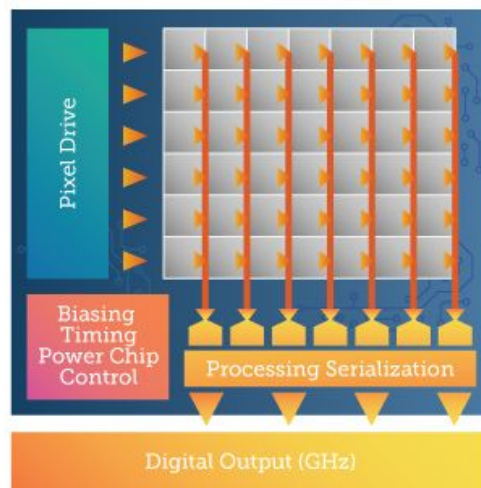


# Digital Image Color Channels

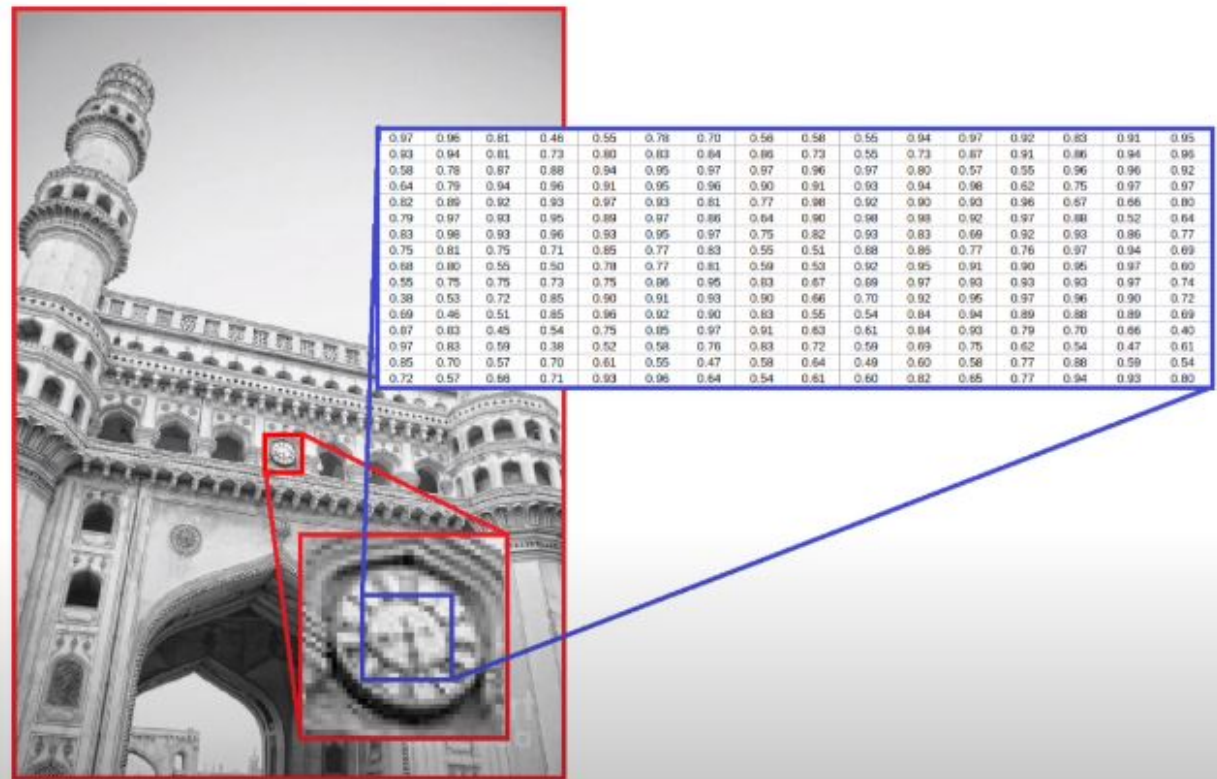
CCD  
Photon to Electron  
Conversion (Analog)



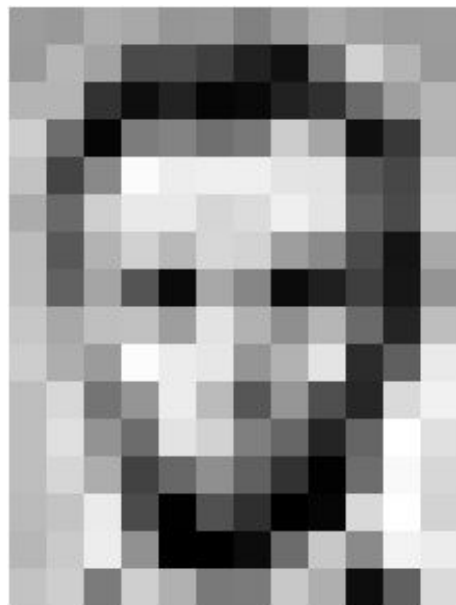
CIS  
Photon to Voltage  
Conversion (Digital)



# 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 values

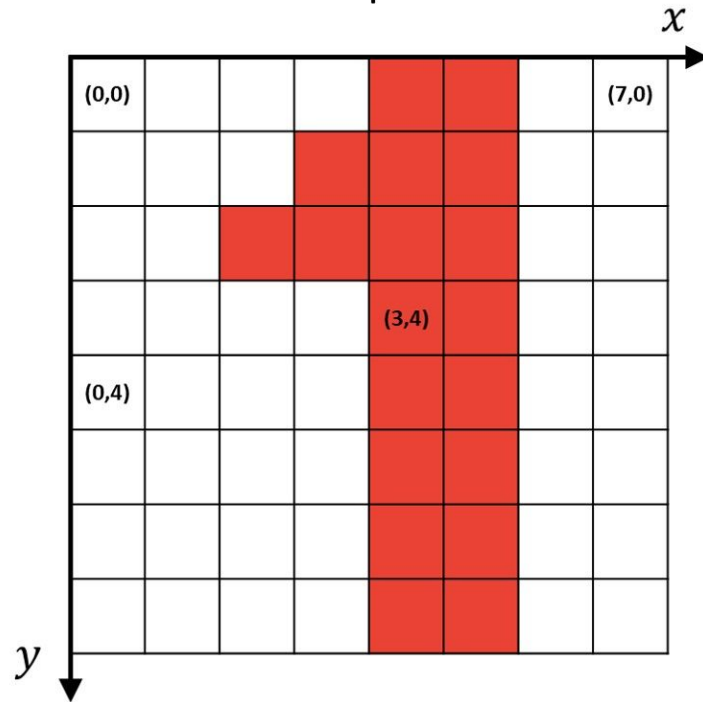
0      50      100      150      200      255





# Pixel Localization

8 × 8 pixels



# Grayscale Image

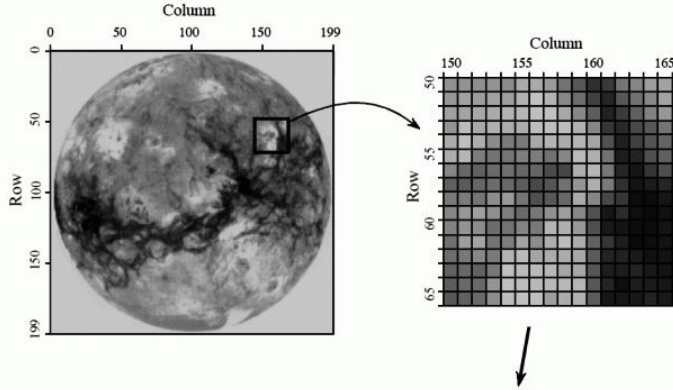
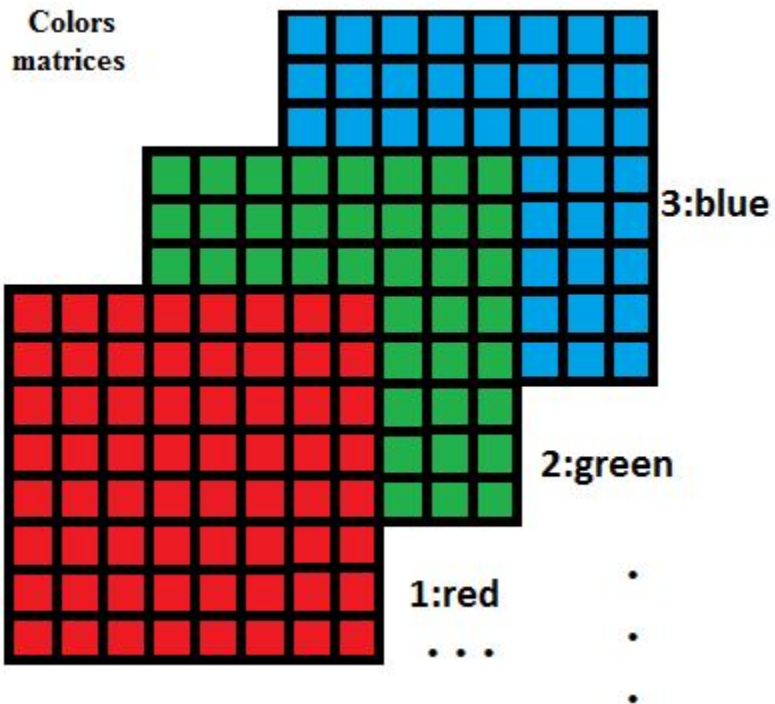
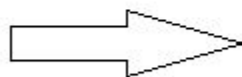


FIGURE 23-1  
Digital image structure. This example image is the planet Venus, as viewed in reflected microwaves. Digital images are represented by a two-dimensional array of numbers, each called a *pixel*. In this image, the array is 200 rows by 200 columns, with each pixel a number between 0 to 255. 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.

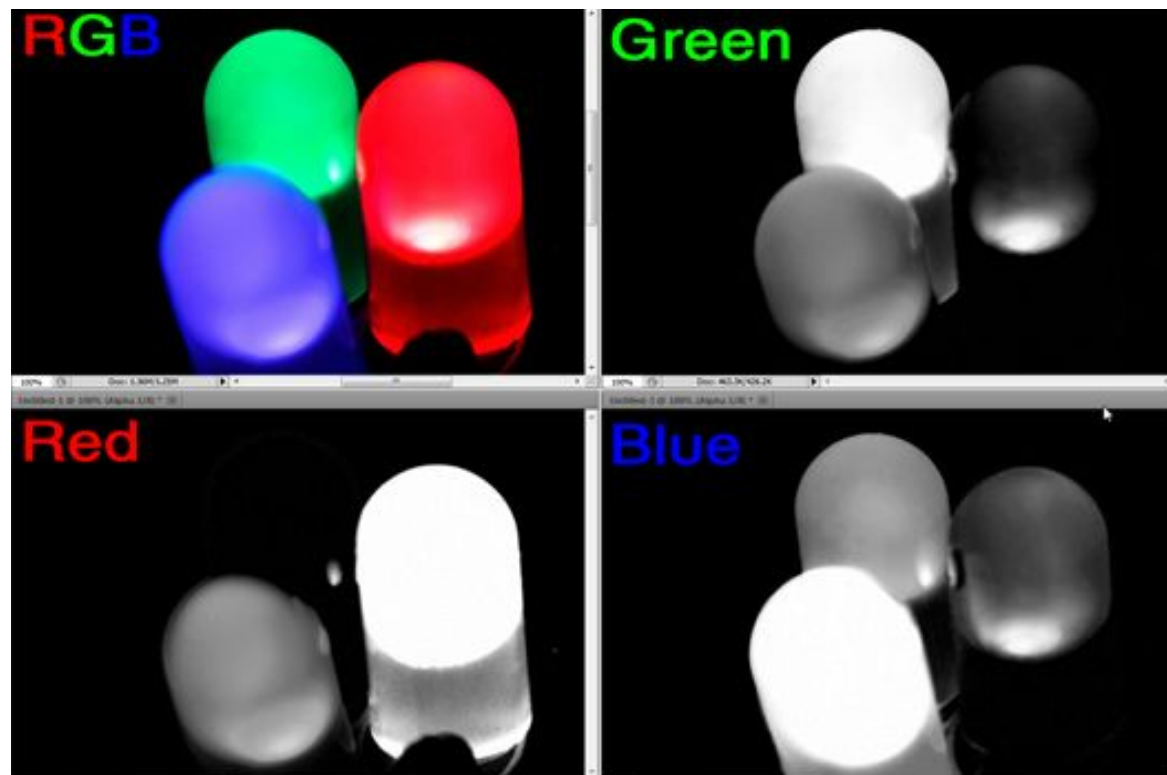
		Column															
		150	155	160	165												
Row	50	183	183	181	184	177	200	200	189	159	135	94	105	160	174	191	196
	55	186	195	190	195	191	205	216	206	174	153	112	80	134	157	174	196
60	194	196	198	201	206	209	215	216	199	175	140	77	106	142	170	186	
65	184	212	200	204	201	202	214	214	214	205	173	102	84	120	134	159	
70	202	215	203	179	165	165	199	207	202	208	197	129	73	112	131	146	
75	203	208	166	159	160	168	166	157	174	211	204	158	69	79	127	143	
80	174	149	143	151	156	148	146	133	118	203	208	162	81	58	101	125	
85	143	137	147	153	150	140	121	133	157	184	203	164	94	56	66	80	
90	164	165	159	179	188	159	126	134	150	199	174	119	100	41	41	58	
95	173	187	193	181	167	151	162	182	192	175	129	60	88	47	37	50	
100	172	184	179	153	158	172	163	207	205	188	127	63	56	43	42	55	
105	156	191	196	159	167	195	178	203	214	201	143	101	69	38	44	52	
110	154	163	175	165	207	211	197	201	201	199	138	79	76	67	51	53	
115	144	150	143	162	215	212	211	209	197	198	133	71	69	77	63	53	
120	140	151	150	185	215	214	210	210	211	209	135	80	45	69	66	60	
125	135	143	151	179	213	216	214	191	201	205	138	61	59	61	77	63	

# Color Image

Digital color image



# Color Channels



# Image Resolution

3904 X 2598 (full resolution RAW image)



10.1 MP = 10.1 million pixels

20 x 20



400 pixels

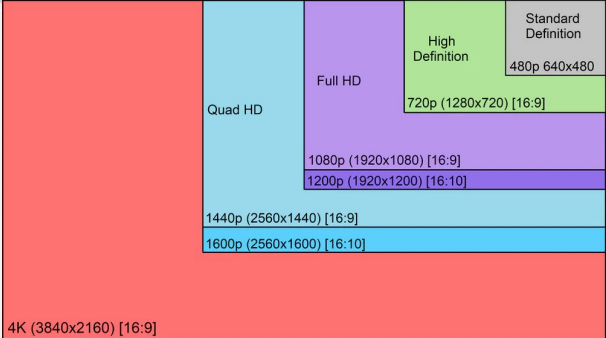
4 x 4







16 pixels

1




1 pixel



# Image File Formats

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

© IlluScientia

file format	colour model	transparency	destination	remarks
<b>SVG</b>	RGB	✓		interactive, scriptable
<b>EPS</b>	RGB / CMYK	✓		PostScript document
<b>PDF</b>	RGB / CMYK	✓		includes PostScript, platform independent

# Image Processing



Original Image / Reset



Grayscale Image



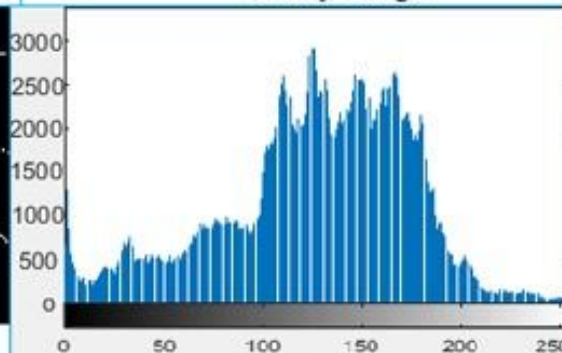
Binary Image



Complement Image



Edge Detection using  
Canny Method



Histogram

# Pixel-level Image Processing



$f(x,y)$



$f(x,y) + 20$



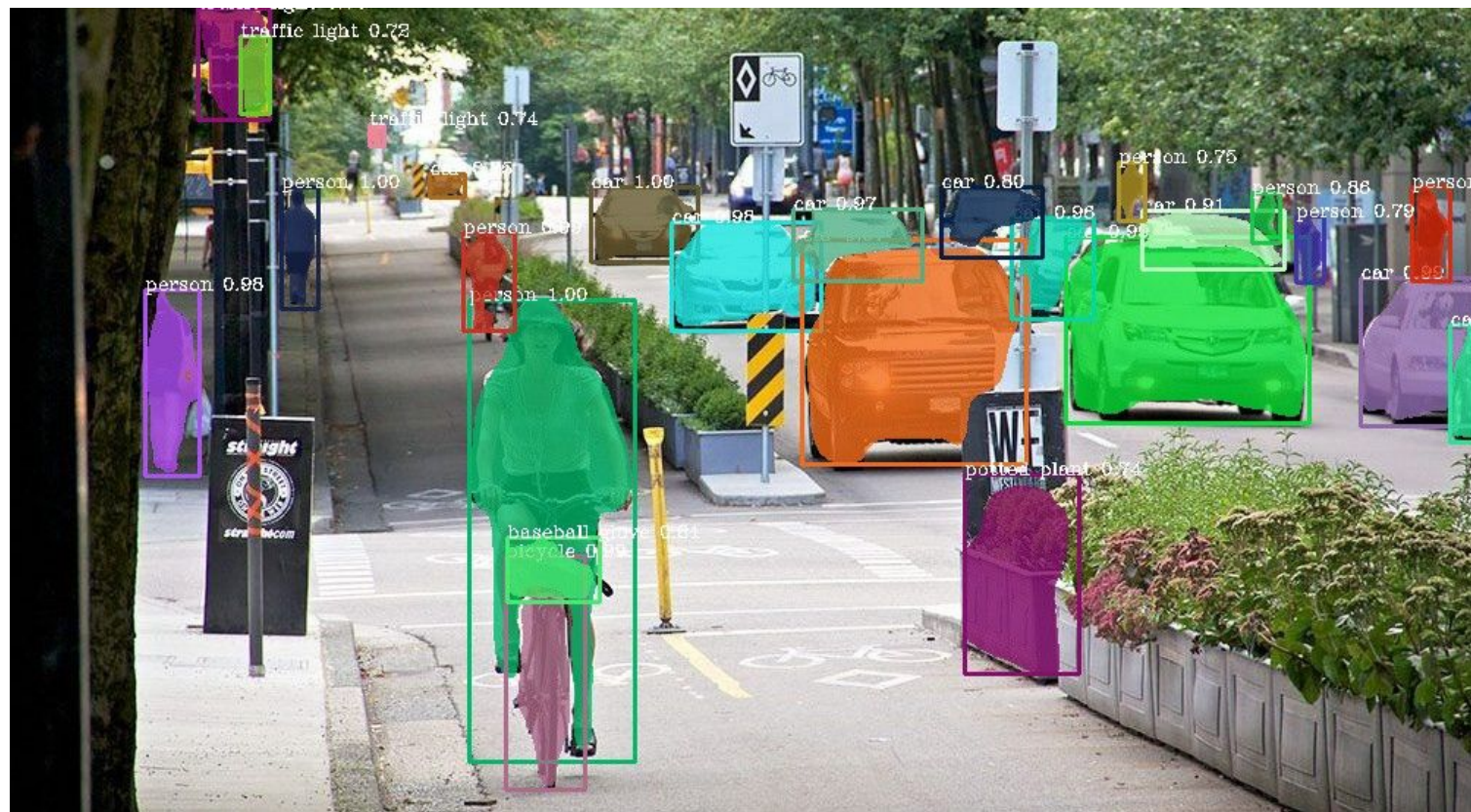
$f(x,y)$



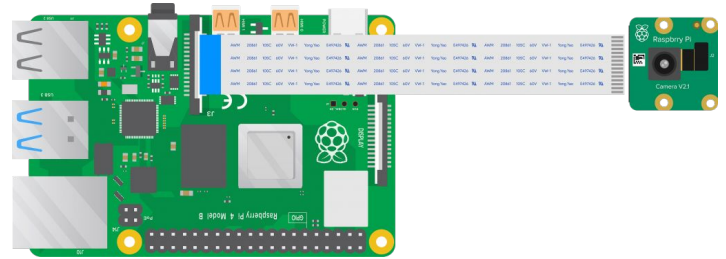
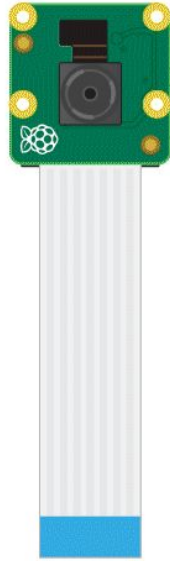
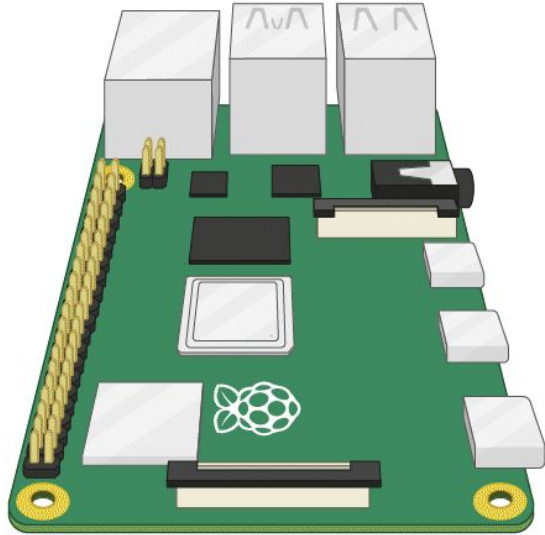
$f(-x,y)$



# High-level Image Processing



# Raspberry Pi Camera

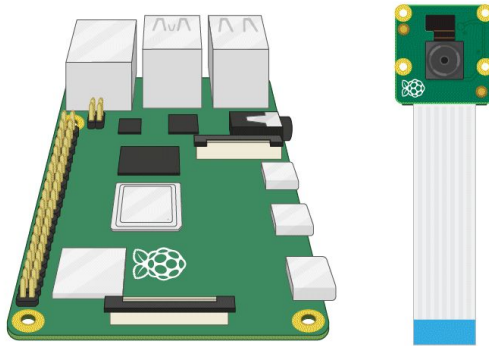


OpenCV



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
```



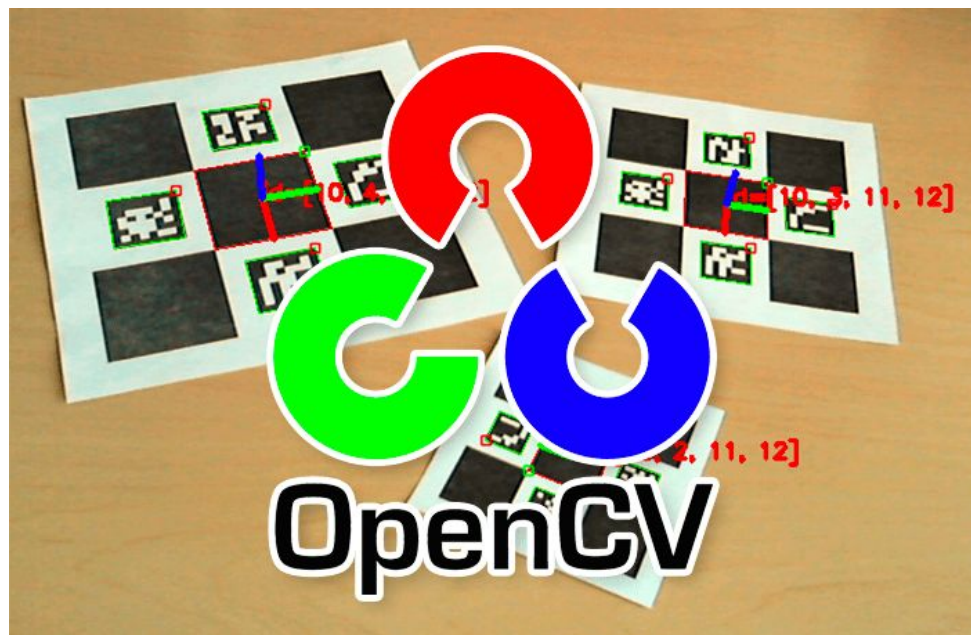
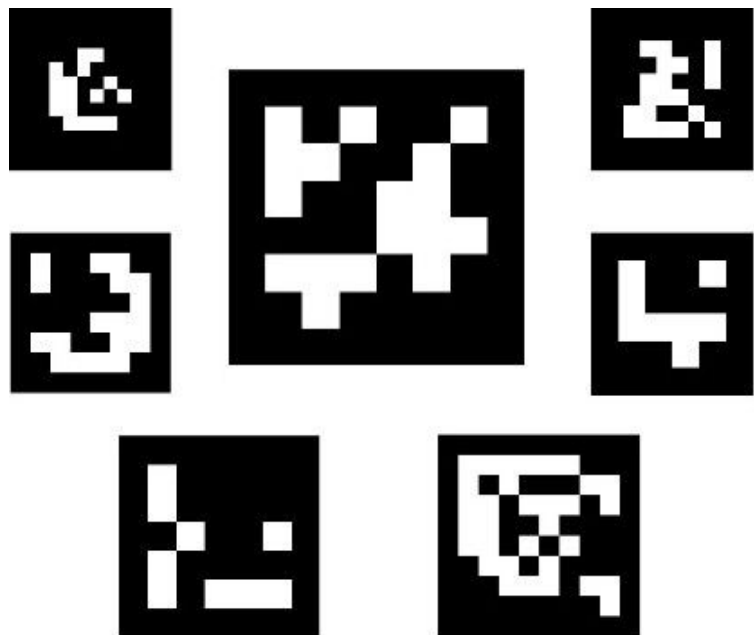
# OpenCV Video Capture

```
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](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
```

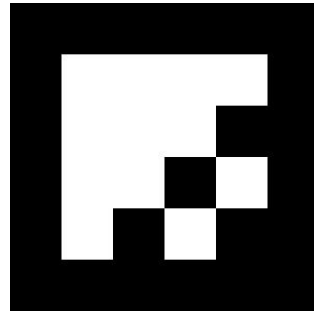
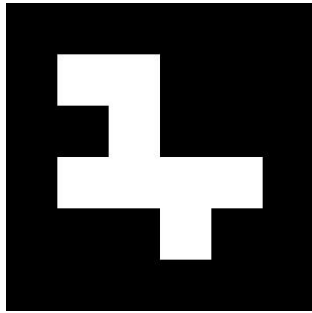
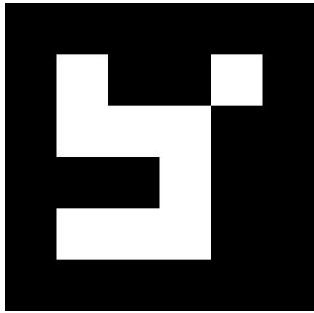
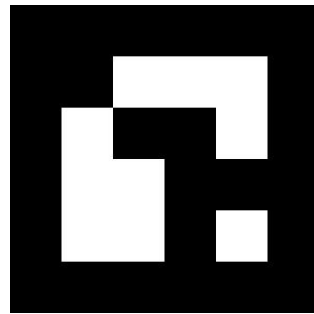
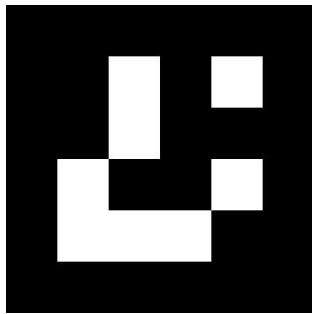
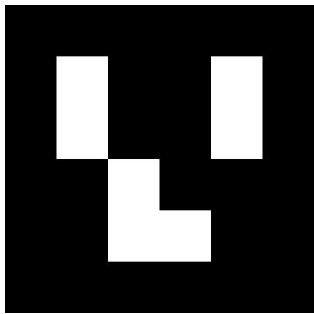
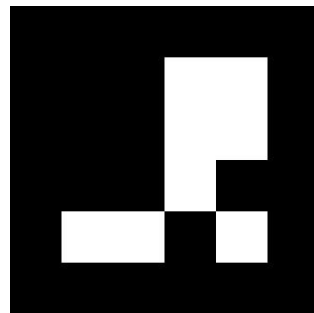
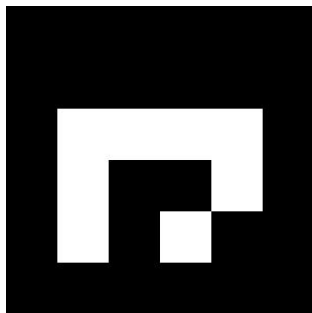
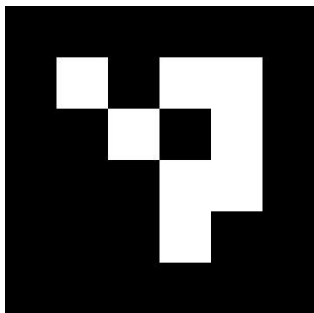
```
aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_4X4_50) # get ArUco  
dictionary
```

```
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)
```