Computer Vision

CVI620

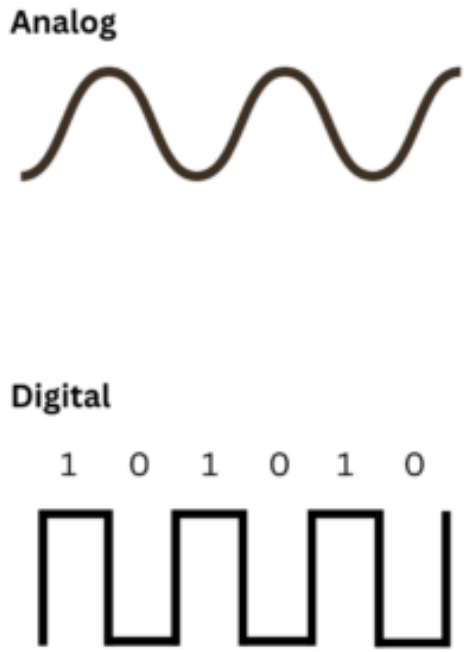
Session 2
01/2025

cameras

Digital vs Analog

- Continuous signals
- Infinite values in a range

- Discrete signals
- Finite values (0s and 1s)



Digital Images

Everything is signal

A representation of visual information stored digitally.

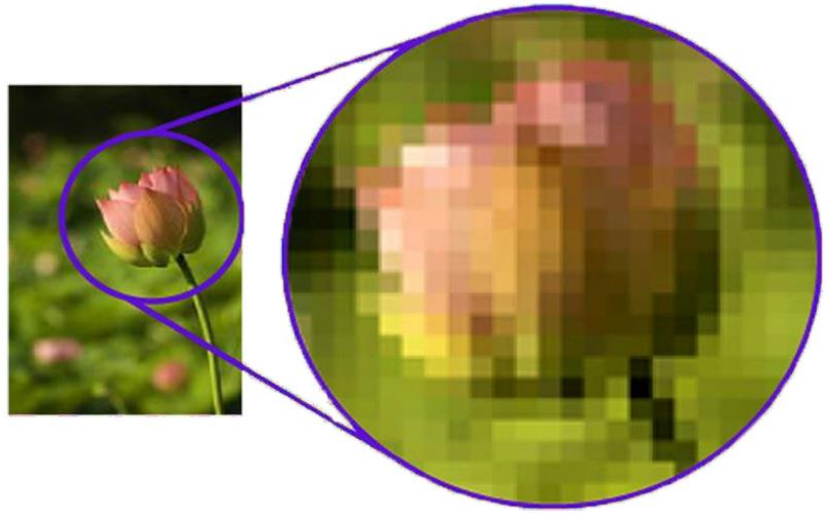
Humans are sensitive to light at wavelengths between 400 nm and 700 nm and hence most camera image sensors are

designed to be sensitive at those wavelengths.

Composed of pixels (picture elements).

Pixels

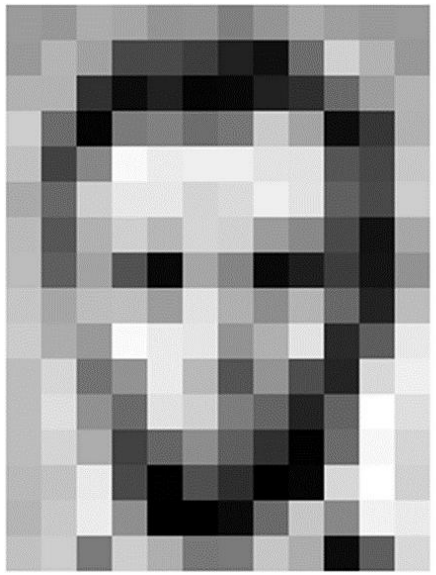
- Smallest unit of a digital image.
- Represents color or intensity information.
- Images are made of a grid of pixels.
- Higher resolution = more pixels = better detail.
- Pixels are typically organized in a 2D array (image matrix), with rows and columns corresponding to their position in the image.
- Size of the picture is the number of pixels in rows and columns (width*height)
- Resolution is number of pixels



Pixel Values

Each pixel can have a value between 0 to 255

Reduced processing costs

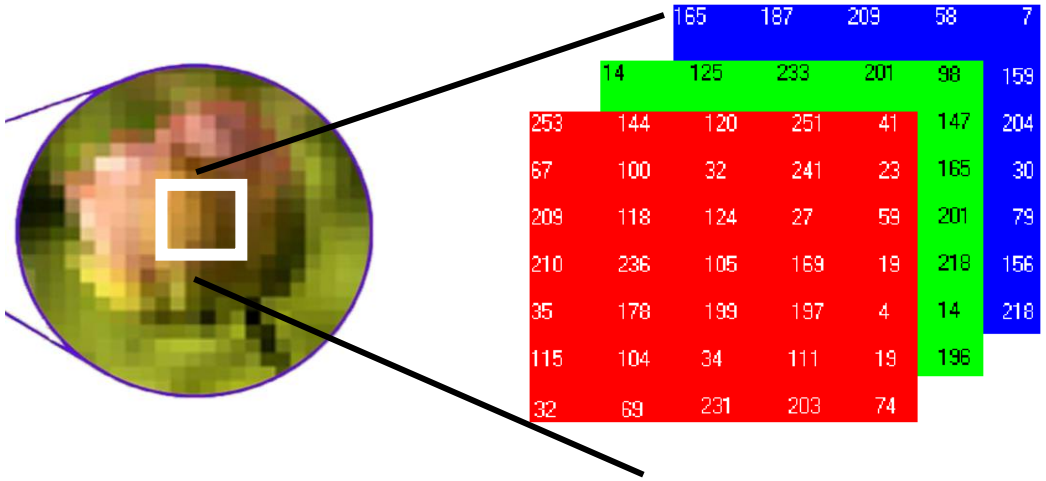


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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

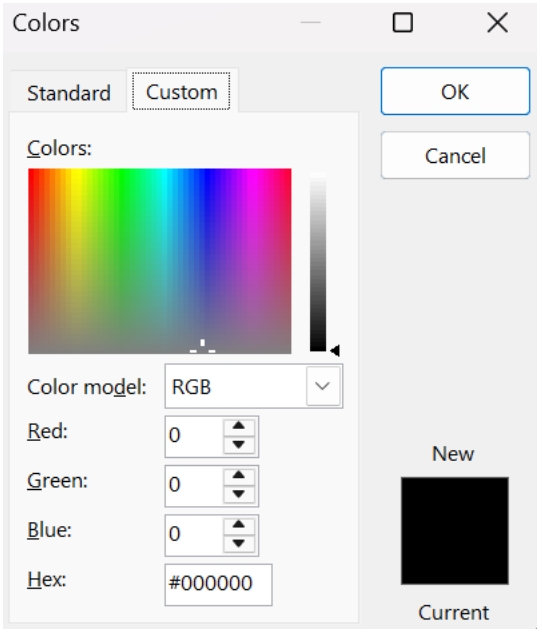
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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

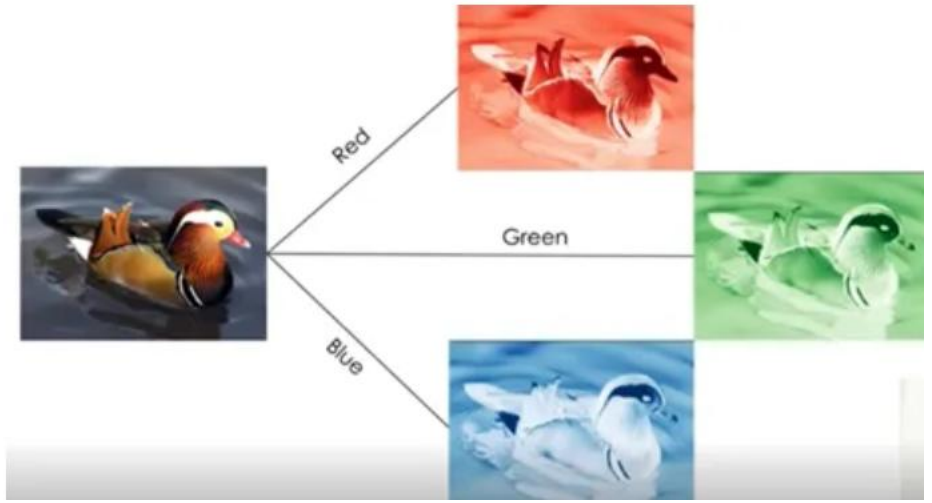
Channels

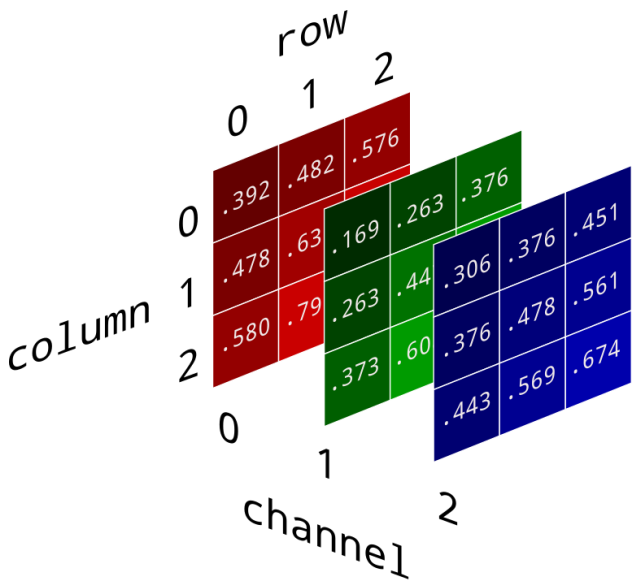
- 3 main pigmentation were introduced to represent color: Red, Green, Blue
- We call each one a channel
- 256*256*256 = 2^24 different possible colors



Channels


Colors
Standard
Custom
OK
Cancel
Colors:
Color model:
RGB
Red:
0
Green:
0
Blue:
0
Hex:
#000000
New
Current


Red
Green
Blue


row
column
channel
0
1
2
.392
.482
.576
.478
.63
.169
.263
.376
.580
.79
.263
.44
.306
.376
.451
.373
.60
.376
.478
.561
.443
.569
.674

Binary Image

- 0 or 255
- 255 is represented with a logical 1
- 1 or no channels

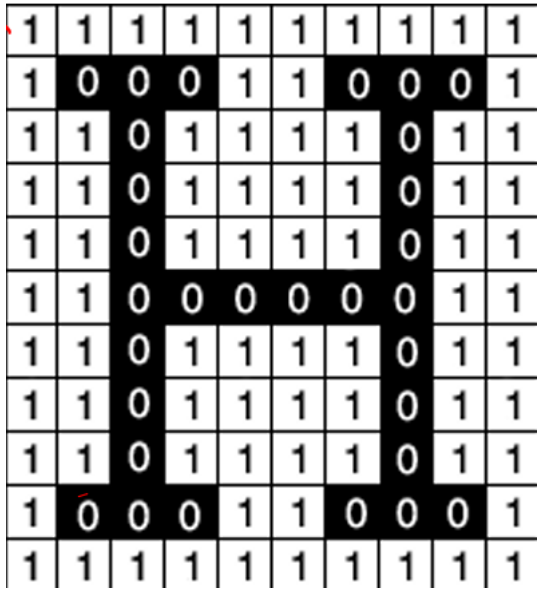
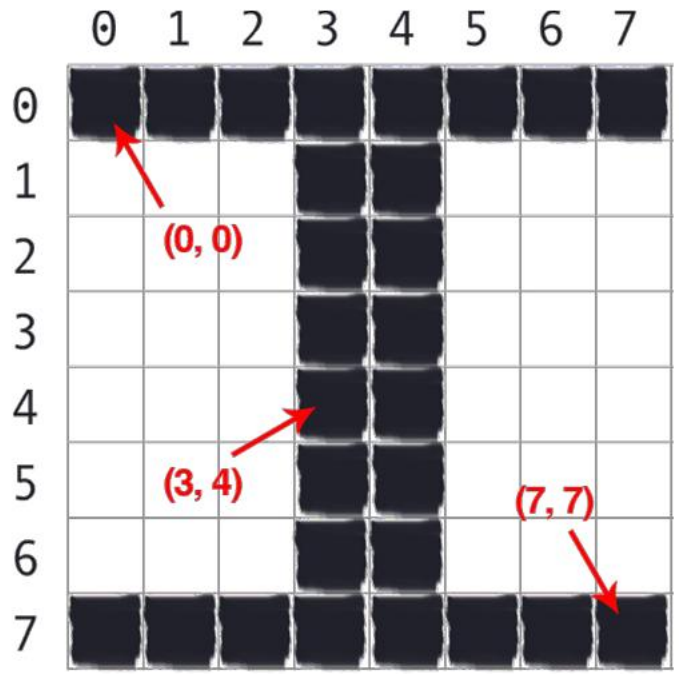


Image Coordinates

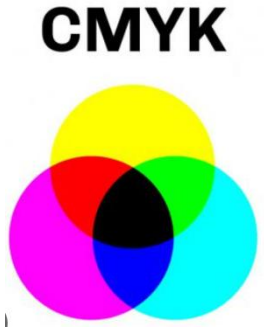


Color Models

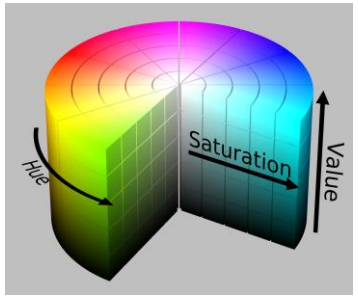
Binary: 0 or 1 – black or white

Grayscale: 0 (black) to 255 (white)

RGB: Combination of Red, Green, Blue (RGB)



CMYK: Cyan, Magenta, Yellow, and Key (Black)



HSV: Hue, Saturation, Value

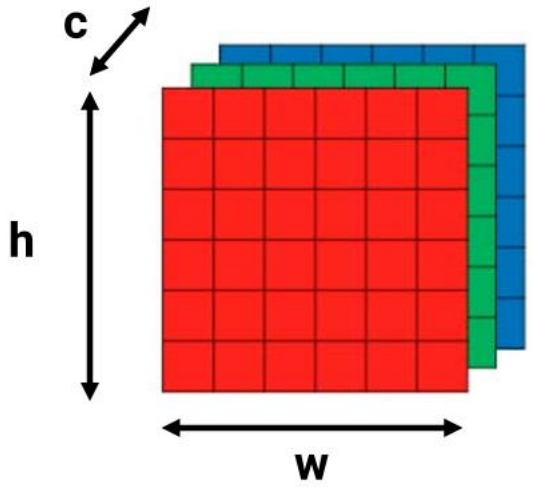
Reading Image

```python
import cv2
import numpy as np

image_array = cv2.imread('Lucy.jpg')
print(f"Image shape: {image_array.shape}")
```

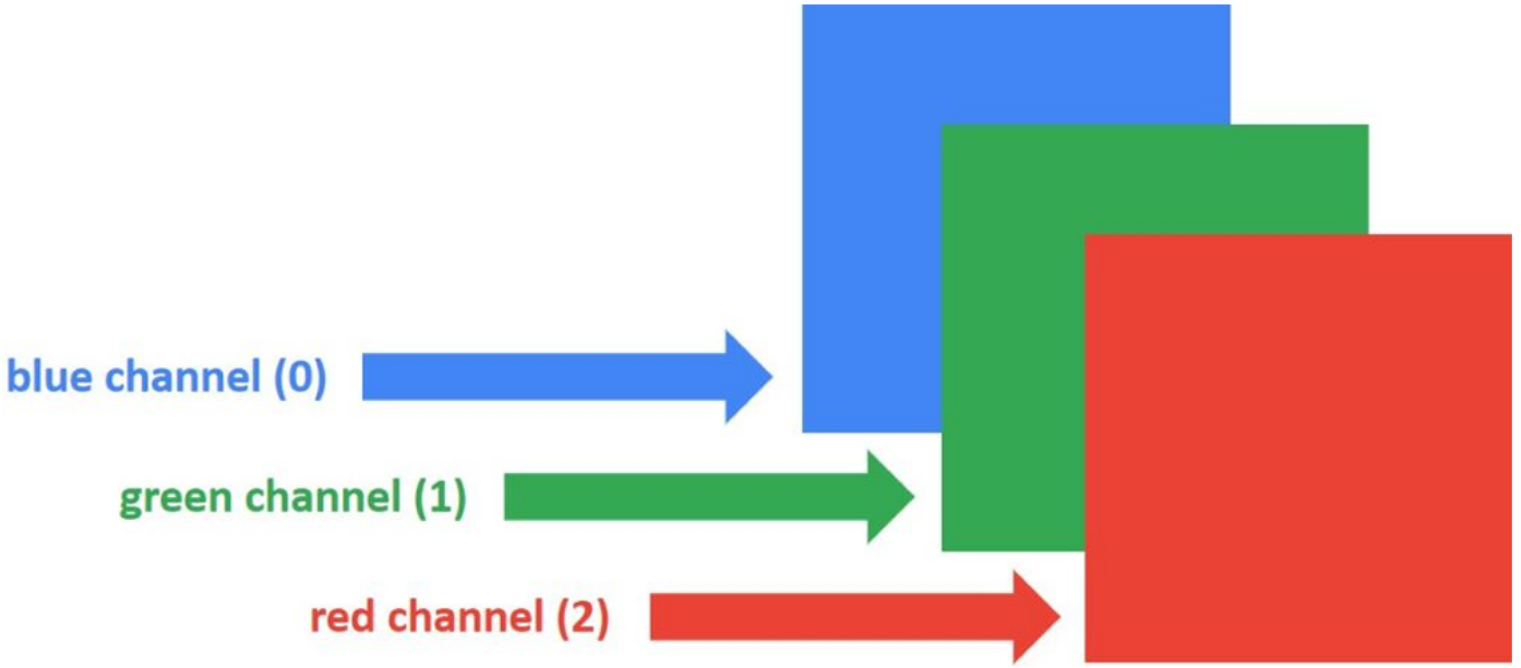
https://opencv24-python-tutorials.readthedocs.io/en/latest/index.html

Image Shape

```
h, w, c = img.shape
```



Channel Extraction



Saving Image

- Store processed images for later use or analysis

```python
import cv2

image = cv2.imread("input.jpg")
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imwrite("output.jpg", gray_image)

```

Saving Image

- Compresses image to specific format
- Writes compressed image to file

cv2.imwrite(filename, image[, params])

filename: The path and name of the file where the image will be saved.
Example: 'output.jpg'
image: The image array to be saved (e.g., a NumPy array).
params (optional): Format-specific save parameters, such as compression quality for .jpg or .png.

```
import cv2

image = 255 * np.ones((100, 100, 3), dtype=np.uint8)

cv2.imwrite('output.jpg', image)
```

Showing Image

```
1  cv2.imshow('img', image_array)
2  cv2.waitKey()
3  # cv2.destroyAllWindows(0)
```

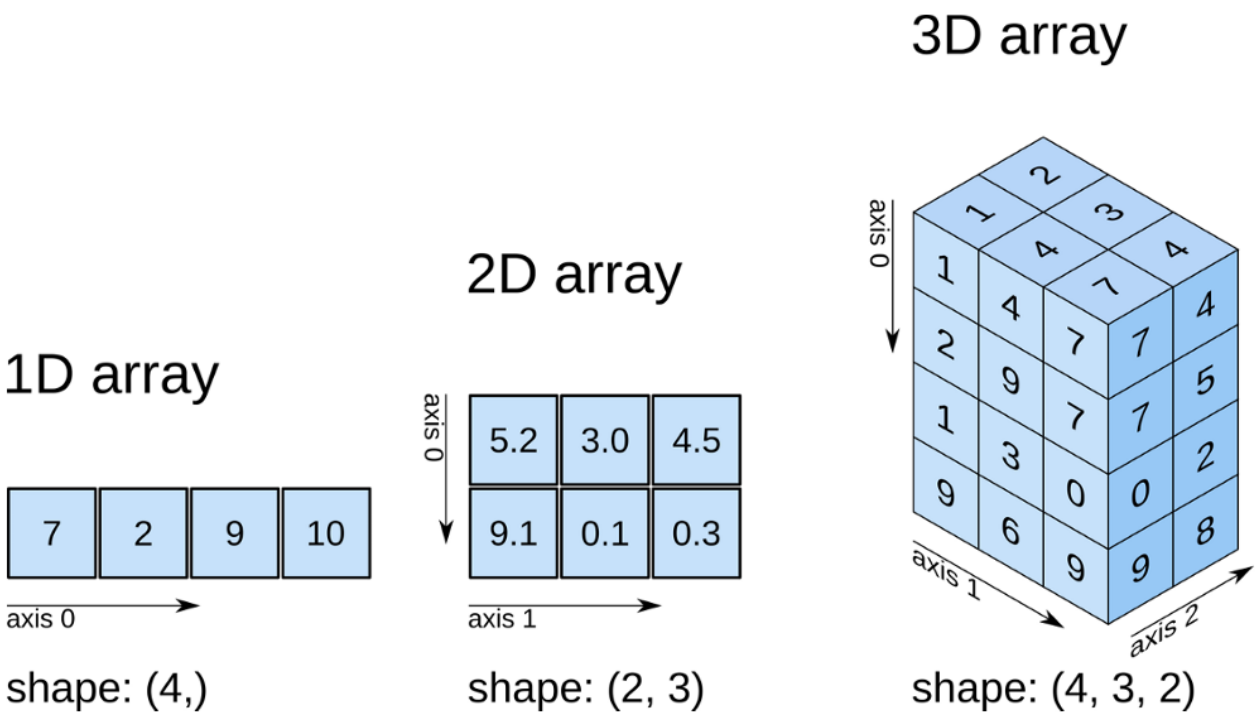
https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html

Numpy Arrays

- Core data structure for images in OpenCV.
- Represent images as multi-dimensional arrays:
- Grayscale: 2D (Height x Width)
- Color: 3D (Height x Width x Channels).

```
1  pixel = image_array[60, 17]  # (B, G, R) for color images
2  image_array[5, 80] = [255, 0, 0]  # Set pixel to blue
3  roi = image_array[50:200, 100:300]
```

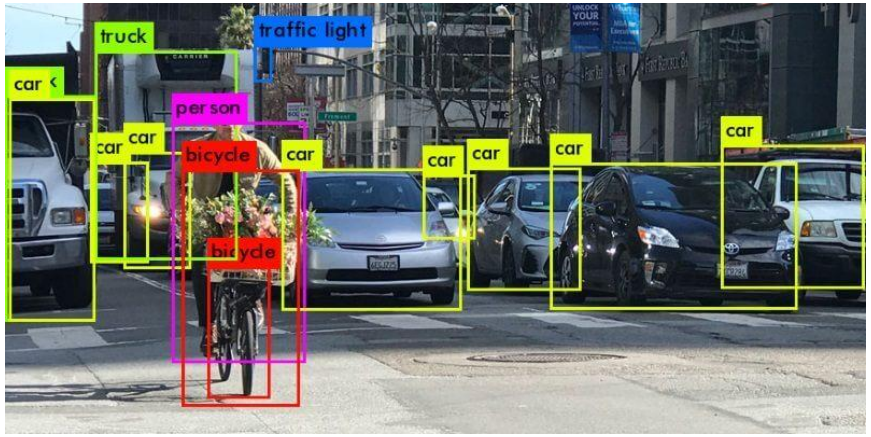
Numpy Coordinates



Region of Interest (ROI)

- A specific part of an image for focused processing
- Enhance efficiency by processing only the necessary area
- Common in object detection, image cropping, and analysis

- uint8



Slicing

- Extracting a portion of a NumPy array

```
array[start:stop:step]
```

```python
import numpy as np

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
slice = matrix[0:2, 1:3]

```

```python
import cv2
image = cv2.imread("image.jpg")
roi = image[50:200, 100:300]  #crop
cv2.imshow("ROI", roi)
cv2.waitKey(0)

```

Cropping

```
cropped = img[411:1560, 1700:3000]
```

Exercise

- Copy an ROI from your image, and paste it in another region

Convert Channels

Changing the color format or extracting individual color channels.

```
cv2.cvtColor(image, conversion_code)
```

cv2.COLOR_BGR2GRAY – Convert to grayscale
cv2.COLOR_BGR2RGB – Convert to RGB
cv2.COLOR_BGR2HSV – Convert to HSV

```
b, g, r = cv2.split(image)  # Blue, Green, Red channels
```

```
merged_image = cv2.merge([b, g, r])
```

Image Formats

- File types that store visual information in various ways.
- For communicating with Hardware. Like a “protocol”.
- They determine how images are saved, compressed, and displayed.

- To balance quality and file size for different use cases.
- Specific formats cater to unique needs like transparency, animation, or high-resolution printing
- Optimize images for web, print, or storage purposes

- Storage formats (e.g., BMP, JPEG, PNG) compress and encode pixel data for efficient use.

Format Examples

JPEG (JPG):

- Best for: Photographs and complex images.
- Features: Lossy compression (small size, lower quality at high compression).
- No transparency support.

PNG:

- Best for: Graphics, logos, and transparent images.
- Features: Lossless compression and transparency support.

TIFF:

- Best for: High-quality printing and professional imaging.
- Features: Lossless or lossy compression, supports layers, and large file sizes.

Padding

cv2.copyMakeBorder(src, top, bottom, left, right, borderType[, value])

src: Input image.
top, bottom, left, right: Pixels to add on each side.
borderType: Type of border:
cv2.BORDER_CONSTANT – Fixed color border.
cv2.BORDER_REFLECT – Mirrored border.
cv2.BORDER_REPLICATE – Extends edge pixels.
cv2.BORDER_WRAP – Wraps image around.
value (optional): Border color for BORDER_CONSTANT.

```
padded_image = cv2.copyMakeBorder(image, 20, 20, 20, 20, cv2.BORDER_CONSTANT, value=(255, 0, 0))
```

Video

- Videos are sequences of images
- A class to capture video streams from:
- Webcam
- Video files (e.g., .mp4, .avi)
- IP cameras or other sources.

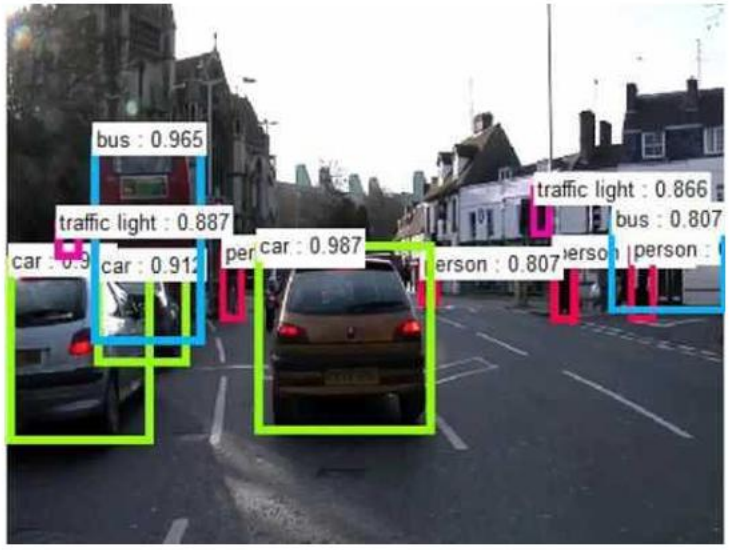
- object being created

```python
cap = cv2.VideoCapture(0)

# 0 for the default webcam
# Path to a video file for playback
# 1, 2, ... for external cameras
# IP
```

```python
ret, frame = cap.read()
# ret: Boolean, True if frame is read successfully
# frame: Captured image array
cap.release()
cv2.destroyAllWindows()
```

```python
cap = cv2.videoCapture("filename.mp4")

while True:
    ret, frame = cap.read()

    if frame is None: break

    cv2.imshow("frame", frame)

    if cv2.waitKey(30) == ord('q'):
```

Drawing Shapes



Line

- draw a straight line on an image

cv2.line(image, pt1, pt2, color, thickness)

image: Input image where the line will be drawn
pt1: starting point (x1, y1) (W, H)
pt2: ending point (x2, y2)
color: line color in BGR format (e.g., (255, 0, 0) for blue)
thickness: line thickness (integer)

```python
import cv2
import numpy as np

#blank image
image = np.zeros((400, 400, 3), dtype=np.uint8)

cv2.line(image, (50, 50), (350, 350), (255, 255, 255), thickness=3)

cv2.imshow("line example", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Rectangle

- Draw rectangle on an image

cv2.rectangle(image, pt1, pt2, color, thickness)

image: Input image where the rectangle will be drawn.
pt1: Top-left corner (x1, y1).
pt2: Bottom-right corner (x2, y2).
color: Rectangle color in BGR format (e.g., (0, 255, 0) for green).
thickness: Border thickness (integer). Use -1 to fill the rectangle.

```
5  image = np.zeros((400, 400, 3), dtype=np.uint8)
6
7  cv2.rectangle(image, (50, 50), (350, 300), (0, 255, 0), thickness=5)
8
9  cv2.imshow("Rectangle Example", image)
10 cv2.waitKey(0)
11 cv2.destroyAllWindows()
```

Circle

- Draw circle

cv2.circle(image, center, radius, color, thickness)

image: input image where the circle will be drawn
center: center of the circle (x, y)
radius: radius of the circle (integer)
color: circle color in BGR format (e.g., (0, 0, 255) for red)
thickness: circle thickness (integer). Use -1 for a filled circle

```
7  cv2.circle(image, (200, 200), 100, (0, 0, 255), thickness=5)
```

Text

- Add text on an image

```
cv2.putText(image, text, org, font, font_scale, color, thickness, line_type)
```

image: Input image where text will be added.
text: The string to display.
org: Bottom-left corner of the text (x, y).
font: Font type (e.g., cv2.FONT_HERSHEY_SIMPLEX).
font_scale: Scale factor for font size.
color: Text color in BGR format (e.g., (255, 255, 255) for white).
thickness: Thickness of the text stroke.
line_type: Type of line for the text (e.g., cv2.LINE_AA).

```
cv2.putText(image, "Hello OpenCV!", (50, 200), cv2.FONT_HERSHEY_SIMPLEX,
            1, (255, 255, 255), thickness=2, lineType=cv2.LINE_AA)
```