Computer Vision - Decoding the Visual world

Premanand S

Assistant Professor,
School of Electronics and Engineering,
Vellore Institute of Technology, Chennai

premanand.s@vit.ac.in

September 12, 2024

Technology?

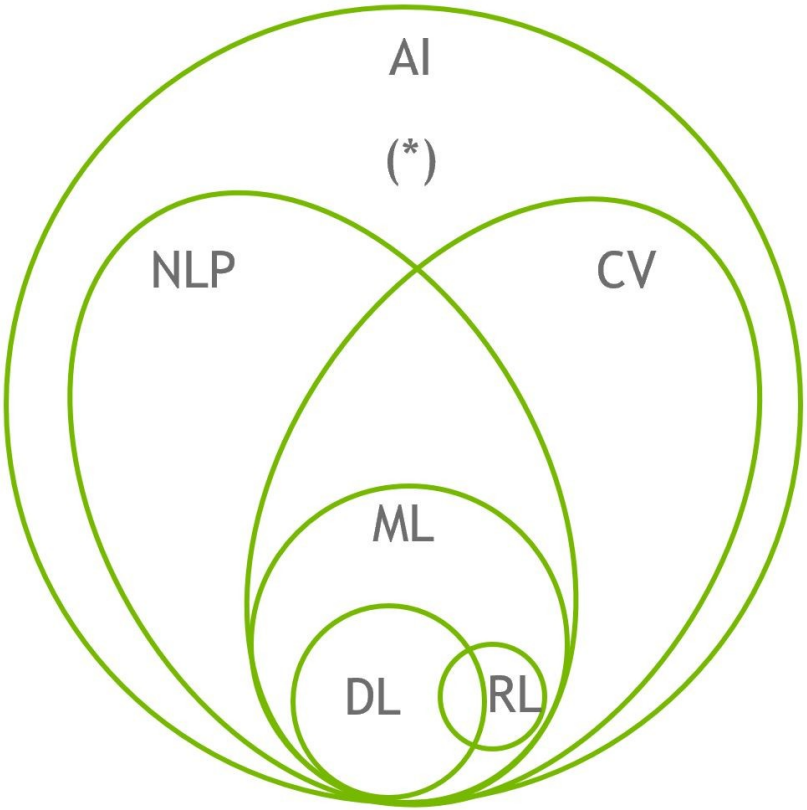
Technology

- Technology applies scientific knowledge, tools, and techniques to solve problems, improve processes, and create new solutions.
- It includes everything from the wheel to computers to medicines to zippers and buttons on clothes.

Trending Technologies

- Artificial Intelligence
- Artificial Intelligence of Things
- Quantum Computing
- Blockchain
- Cybersecurity
- Augmented Reality and Virtual Reality
- Robotics and many more...

AI - ML - DL - DS - GenAI - CV - NLP



AI = Artificial Intelligence
NLP=Natural Language Processing
CV=Computer Vision
ML=Machine Learning
DL=Deep Learning
RL=Reinforcement Learning

(*)=We would have more ellipses there (similar to NLP or CV) representing Robotics, Expert Systems, Speech, and Planning, Scheduling & Optimization systems. But it would look very messy. So, go ahead and imagine they are there too.

Computer Vision

Computer Vision

What is NOT Computer Vision?

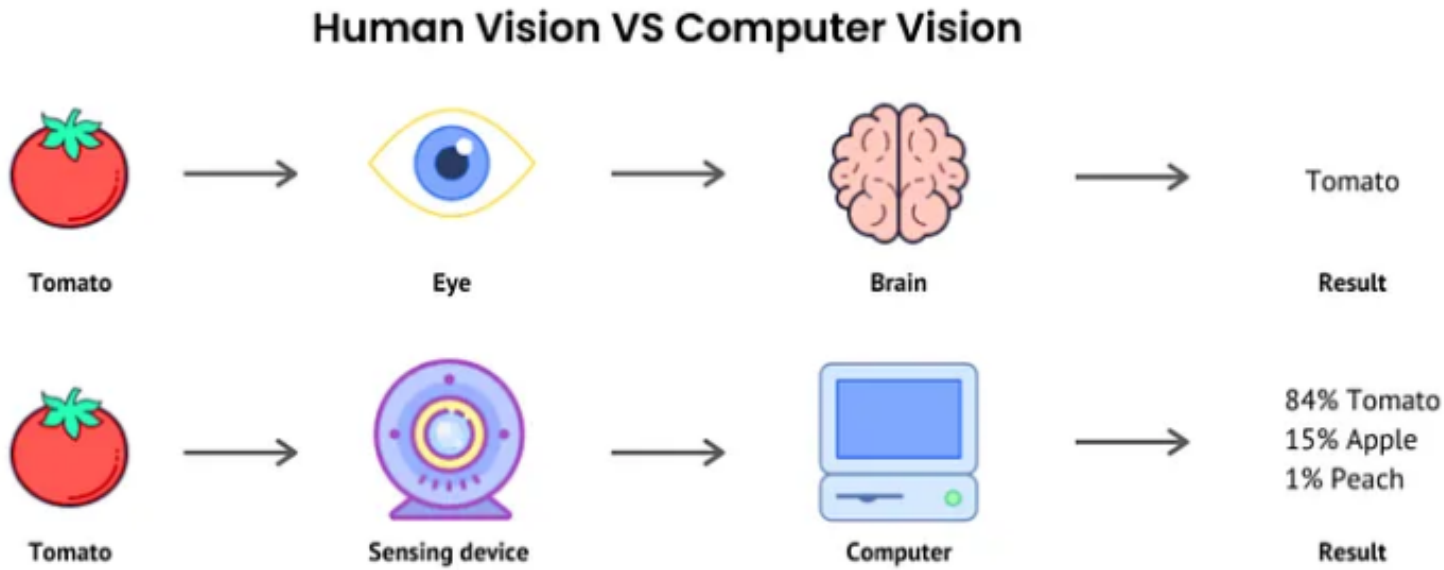
- Non-visual data processing - NLP, Audio processing
- Data Storage and Retrieval - Involves saving and managing files in a database or storage system, but not interpreting
- Mathematical Computation: Tasks like matrix operations, solving algebraic equations, or statistical analysis that do not deal with visual information.
- Networking: Covers the communication and transfer of visual data over networks, like streaming videos, but doesn't involve interpreting or analyzing this visual data.
- Operating System Interaction: Involves file management and system operations
- Automation without intelligence
- Computer Graphics: Involves generating and rendering graphics from 3D models or simulations

What is Computer Vision?

- Vision - Discovering from images what is present in the scene and where it is
- Computer + Vision - Field of AI that enables machines to see and understand the world and the things in it
- Computer Vision - the technology that enables computers to "see" and understand images or videos, just like how humans do. It allows machines to identify objects, recognize faces, and interpret visual information from the real world. Ex: Facial recognition, Cameras in self-driving car to identify sign boards, objects

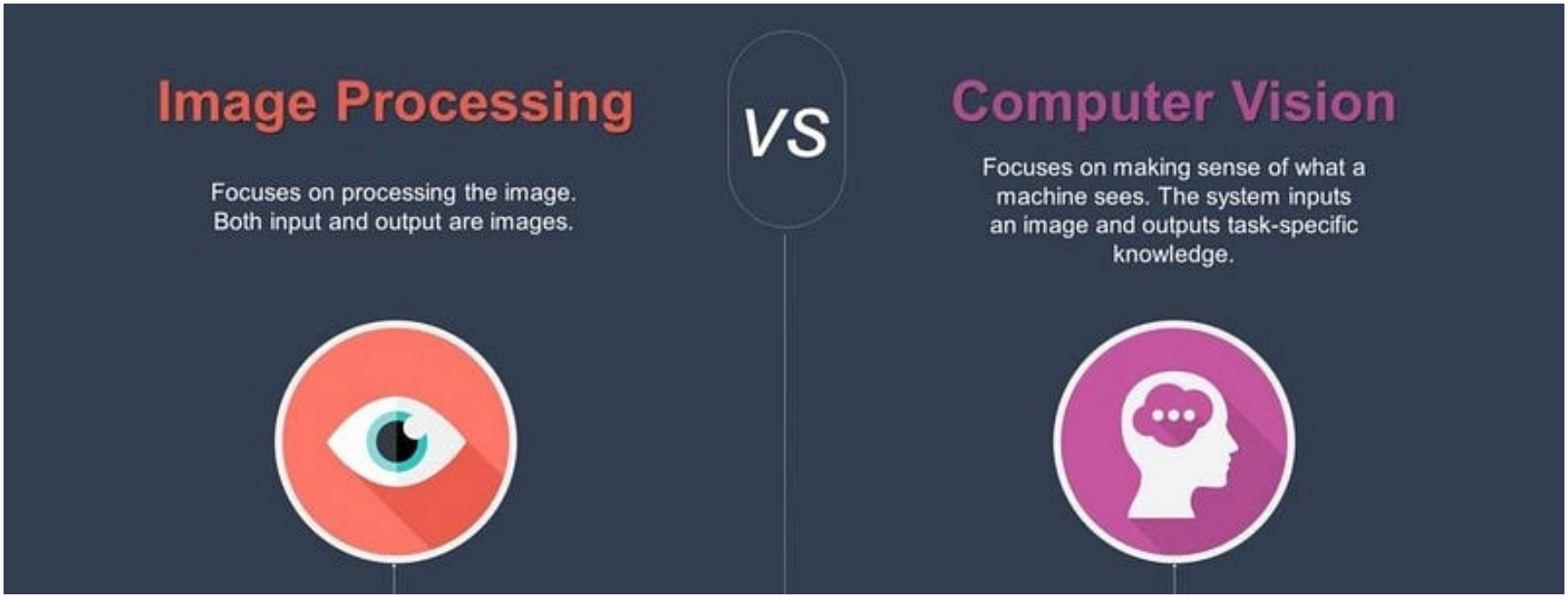
Computer Vision Vs Human Vision

- Uses algorithms and software to process and analyze visual data from cameras and sensors.
- Uses the eyes to capture visual information and the brain to interpret and understand it.



Computer Vision vs Image Processing

- Involves manipulating and enhancing images to improve their quality or to extract useful information.
- Involves understanding and interpreting the content of images or videos, mimicking human visual perception.



Applications

- Healthcare - AR / 3D in Medical Imaging and Surgical Assistance
- Agriculture - Crop Monitoring, Precision farming
- Sports Analytics - Performance tracking, Game Strategy
- Security and Surveillance - Facial Recognition, Anomaly Detection
- Manufacturing - Quality Control, Automated Robotics
- Augmented Reality

Coding CV - OpenCV?

1. What are Images?

Images

- Images are digital representations of visual information that algorithms analyze and interpret.
- Images are numpy arrays
- Numpy - Mathematical operation, numerical operation
- Image shape is given by height, width, no. of channels

Images

Example (Python snippet)

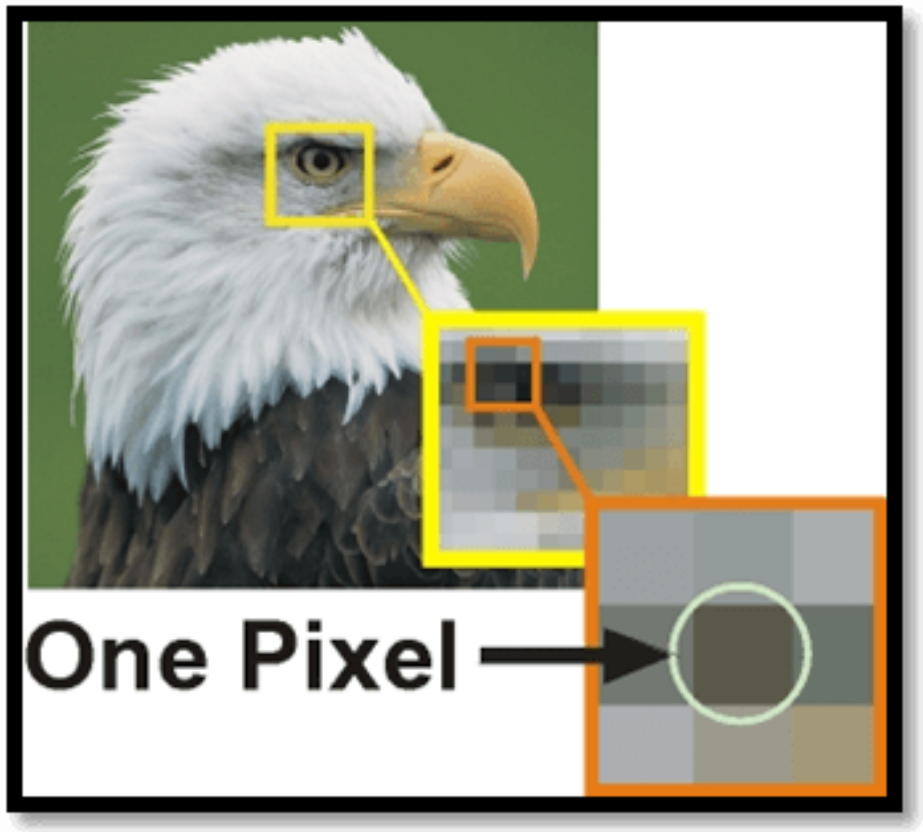
```
import cv2
image = cv2.imread('python_.png')

print(type(image))      // <class 'numpy.ndarray'>

print(image.shape)     // (769, 767, 3)
```

Images - Pixel

- Image is made by 'PIXELS'
- **Pixel**: The smallest unit of a digital image, represents a single point.
- In a grayscale image, each pixel has a single value; in a color image, each pixel has multiple values (e.g., RGB).

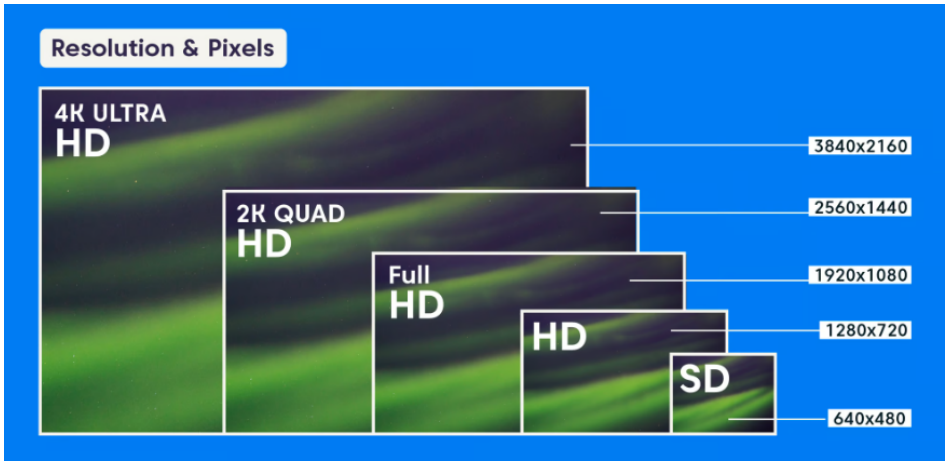


Images - Pixel

- 8-bit Grayscale Images - Most of the pixel values range from 0 to 255
- Binary images pixels values - [0,1] or [0,255]
- 16 bit images pixel values - 0 to 65535

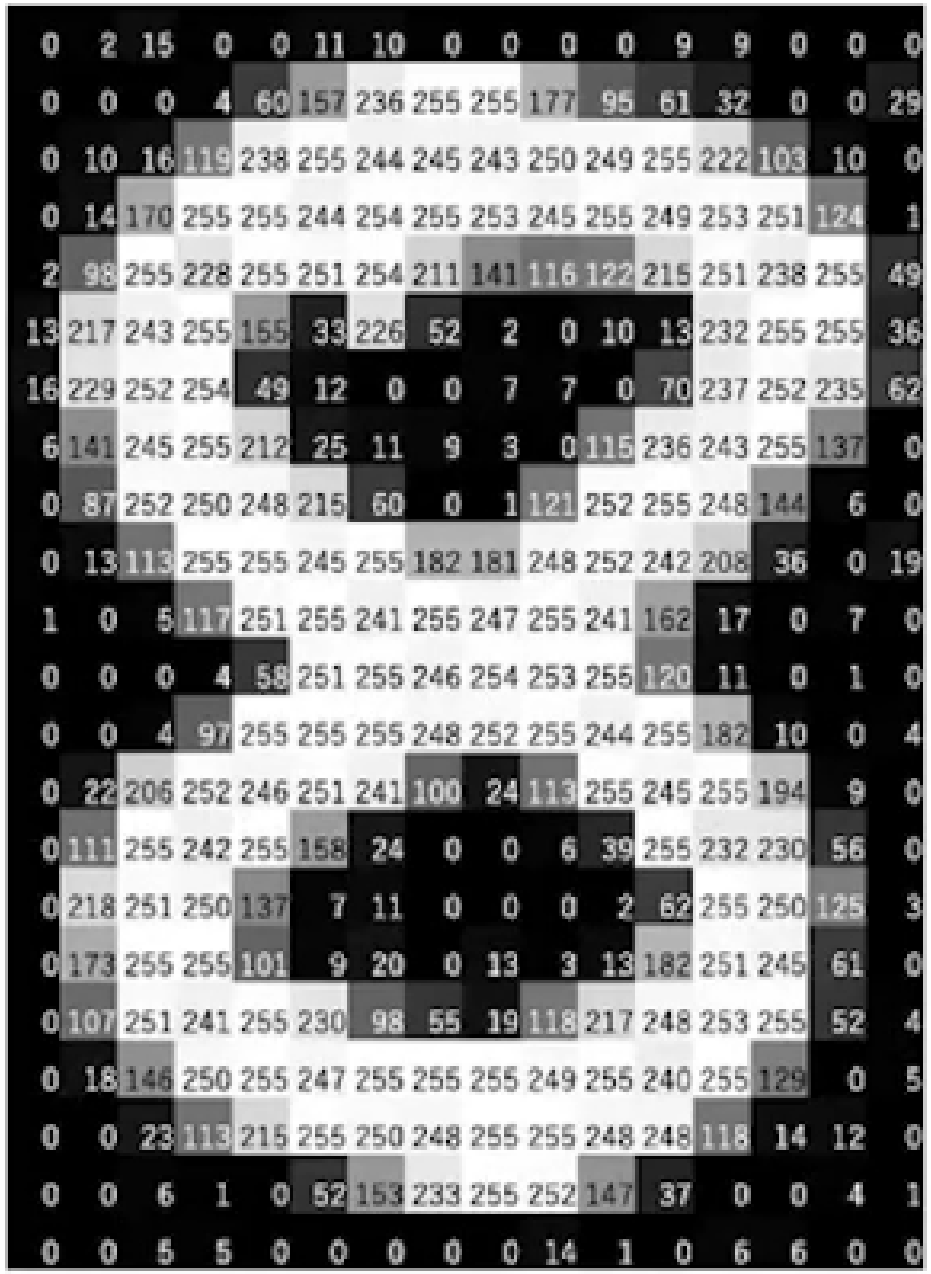
Images - Resolution

- The dimensions of an image, often described in pixels (e.g., 1920x1080).
- Higher-resolution images have more detail but require more processing power.



Images - Grid Structure

- **Grid Structure**: A digital image is a matrix of these pixels, where each pixel has a specific value corresponding to its color and intensity.



2. Input / Output

Python - Read and Write an Image

Example (Python snippet)

```
import cv2

img = cv2.imread('python_.png')

cv2.imwrite('python_out.png', img)

cv2.imshow('image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Python - Read Video

Example (Python snippet)

```
import cv2

video = cv2.VideoCapture('cat_videos.mp4')

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

    if ret:
        cv2.imshow('frame', frame)
        cv2.waitKey(40)

video.release()
cv2.destroyAllWindows()
```

Python - Read WebCam

Example (Python snippet)

```
import cv2

webcam = cv2.VideoCapture(0)

if not webcam.isOpened():
    print("Error: Could not open webcam.")
else:
    while True:
        ret, frame = webcam.read()

        if not ret:
            print("Error: Failed to capture frame.
                        Check webcam index.")
            break
```

Python - Read WebCam (Contd...)

Example (Python snippet)

```
        if frame is not None and frame.shape[0] > 0
                                and frame.shape[1] > 0:
            cv2.imshow('frame', frame)
        else:
            print("Error: Captured frame is empty
                                                or invalid.")
            break

        if cv2.waitKey(40) & 0xFF == ord('q'):
            break

webcam.release()
cv2.destroyAllWindows()
```

3. Basic Operations

Resizing and Cropping

- Resizing is a basic preprocessing step in computer vision, often used to standardize the size of images before feeding them into models or algorithms. It ensures uniformity across the dataset and may affect the performance of certain algorithms.
- Cropping is used to focus on specific regions of interest in an image. This is especially important in object detection, where only a portion of the image may contain the relevant object.

Resizing

Example (Python snippet)

```
import cv2

img = cv2.imread('python_.png')

resized_img = cv2.resize(img, (640, 640))

print("Original image shape:", img.shape)
print("Resized image shape:", resized_img.shape)

cv2.imshow('Original Image', img)
cv2.imshow('Resized Image', resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Cropping

Example (Python snippet)

```
import cv2

img = cv2.imread('python_.png')

print("Original image shape:", img.shape)

cropped_img = img[220:740, 320:940]

cv2.imshow('Original Image', img)
cv2.imshow('Cropped Image', cropped_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

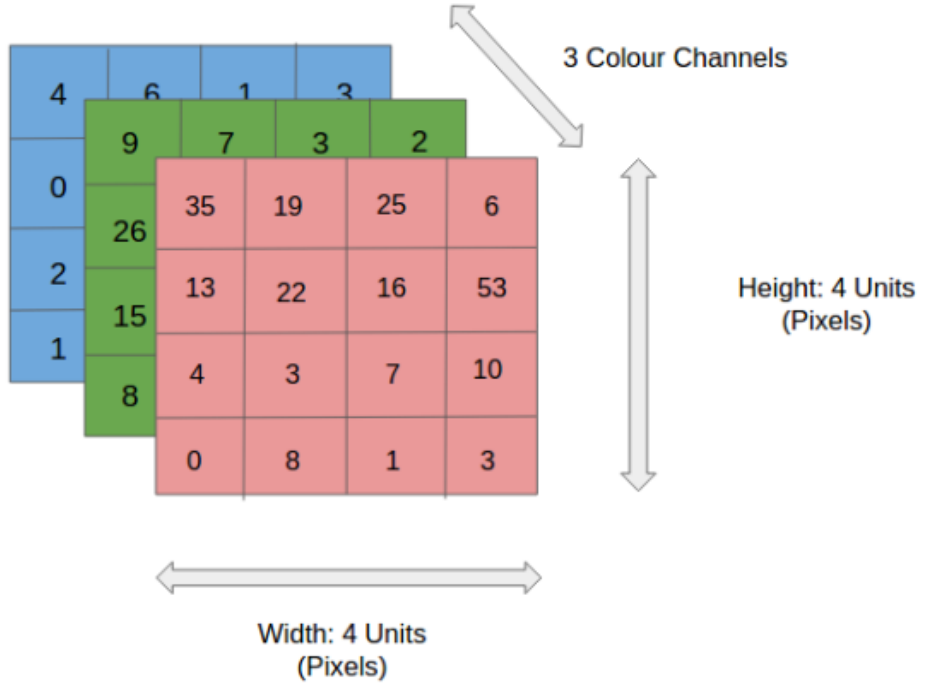
4. Color-Spaces

Color-Spaces

- Changing color spaces (e.g., from RGB to grayscale or HSV) is a common operation in computer vision.
- Grayscale is used to simplify processing when color is not important, while HSV can be useful for color-based segmentation and detection.
  - RGB (Red, Green, Blue) - Computer monitors, digital cameras, and televisions
  - CMYK (Cyan, Magenta, Yellow, Key/Black) - Magazines and Brouchers
  - HSV (Hue, Saturation, Value) and HSL (Hue, Saturation, Lightness)- Intuitive color adjustments.
  - YCbCr - Luminance and chrominance - JPEG and MPEG

Digital Images - Channel

- **Channel**: Refers to a specific component of the color representation of a pixel.
- Each channel corresponds to a particular color or intensity level, and together, these channels define the overall color of the pixel.



Color-Spaces

Example (Python snippet)

```
import cv2

img = cv2.imread('python_.png')

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

cv2.imshow('Original Image', img)
cv2.imshow('HSV Image', img_hsv)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. Smoothing

Smoothing

- Smoothing is often used in preprocessing to reduce noise and detail.
- Gaussian, for example, smooths images and is commonly used before edge detection or thresholding.
- Median is effective at removing salt-and-pepper noise.

Smoothing

Example (Python snippet)

```
import cv2
img = cv2.imread('cow-salt-peper.png')
k_size = 7
img_blur = cv2.blur(img, (k_size, k_size))
img_gaussian_blur = cv2.GaussianBlur(img, (k_size, k_size),
                                                        5)
img_median_blur = cv2.medianBlur(img, k_size)

cv2.imshow('Original Image', img)
cv2.imshow('Blur Image', img_blur)
cv2.imshow('Gaussian Blur Image', img_gaussian_blur)
cv2.imshow('Median Blur Image', img_median_blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6. Threshold

Threshold

- Thresholding converts images to binary (black-and-white) by setting pixel intensity limits.
- It's a critical step in segmentation, object detection, and contour detection.
- Thresholding helps isolate objects from the background.

Thresholding

Example (Python snippet)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image_path = 'dhoni.png'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Global Thresholding
_, global_thresh = cv2.threshold(image, 128, 255, cv2.THRESH_

# Adaptive Thresholding
adaptive_thresh = cv2.adaptiveThreshold(image, 255, cv2.ADAPTI
                                        cv2.THRESH_BINARY, 11,
```

Thresholding (Contd...)

Example (Python snippet)

```
# Otsu's Thresholding
_, otsu_thresh = cv2.threshold(image, 0, 255, cv2.THRESH_BINAR

plt.figure(figsize=(10, 7))

plt.subplot(2, 2, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.title('Global Thresholding')
plt.imshow(global_thresh, cmap='gray')
plt.axis('off')
```

Thresholding (Contd...)

Example (Python snippet)

```
plt.subplot(2, 2, 3)
plt.title('Adaptive Thresholding')
plt.imshow(adaptive_thresh, cmap='gray')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.title("Otsu's Thresholding")
plt.imshow(otsu_thresh, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```

7. Edge Detection

Edge Detection

- Edge detection is fundamental in identifying object boundaries and shapes in images.
- It helps locate significant changes in intensity, which are often where edges of objects reside.
- This is crucial for tasks like object detection, feature extraction, and image segmentation.

Edge Detection

Example (Python snippet)

```
import cv2
import numpy as np

img = cv2.imread('dhoni.png')
img_edge = cv2.Canny(img, 100, 200)
img_edge_d = cv2.dilate(img_edge, np.ones((3, 3),
                                          dtype=np.int8))
img_edge_e = cv2.erode(img_edge_d, np.ones((3, 3),
                                          dtype=np.int8))
cv2.imshow('Original Image', img)
cv2.imshow('Edge Detected Image', img_edge)
cv2.imshow('Dilated Edges', img_edge_d)
cv2.imshow('Eroded Edges', img_edge_e)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8. Drawings

Drawings

- Drawing shapes and text is used for annotation, visualization, and highlighting regions of interest.
- For example, after detecting an object, you might draw a bounding box or label it with text to visualize the results.

Drawings

Example (Python snippet)

```
import cv2

img = cv2.imread('whiteboard.png')

print(img.shape)

cv2.line(img, (100, 150), (300, 450), (0, 255, 0), 3)

cv2.rectangle(img, (200, 350), (450, 600), (0, 0, 255), -1)

cv2.circle(img, (800, 200), 75, (255, 0, 0), 10)
```

Drawings (Contd...)

Example (Python snippet)

```
cv2.putText(img, 'Hey you!', (600, 450),
            cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 0), 10)

cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

9. Contours

Contours

- Contours represent the boundaries of objects.
- They are useful in shape analysis, object detection, and image segmentation.
- Contour detection is often preceded by edge detection or thresholding.

Contours

Example (Python snippet)

```
import cv2

img = cv2.imread('birds.jpg')

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

ret, thresh = cv2.threshold(img_gray, 127, 255,
                            cv2.THRESH_BINARY_INV)

contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
                                cv2.CHAIN_APPROX_SIMPLE)
```

Contours (Contd...)

Example (Python snippet)

```
for cnt in contours:
    if cv2.contourArea(cnt) > 200:
        x1, y1, w, h = cv2.boundingRect(cnt)
        cv2.rectangle(img, (x1, y1), (x1 + w, y1 + h),
                                        (0, 255, 0), 2)

cv2.imshow('Original Image', img)
cv2.imshow('Threshold Image', thresh)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Key task - Computer Vision

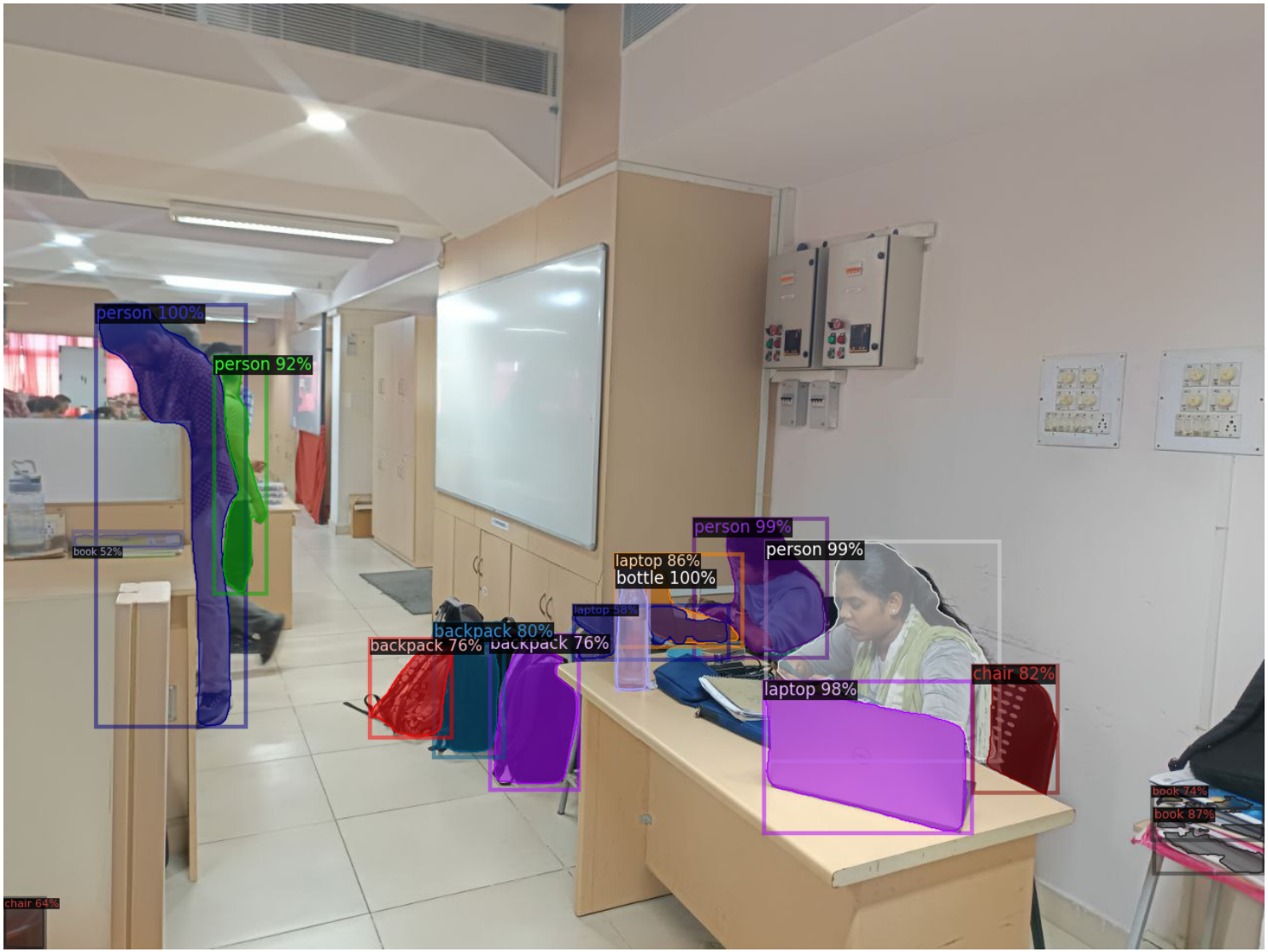
Some key task - Computer Vision

- Image classification
- Object detection
- Facial recognition
- Pose estimation
- Motion analysis
- Segmentation
  - Semantic Segmentation
  - Instance Segmentation
  - Panoptic Segmentation

Segmentation

Segmentation - KeyPoint detection

Instance Segmentation and Object detection



Semantic Segmentation

Panoptic Segmentation

Motion analysis

Zoox - Driverless car

▸ Zoox - Autonomous Vehicle

mail me: er.anandprem@gmail.com / premanand.s@vit.ac.in
ring me: +91 73586 79961
follow me: Linkedin
author at Analytics Vidhya: premanand17
author at Medium: Premanand S

Predicting the future isn't magic, it's artificial intelligence!