OBJECT DETECTION USING IMAGEPROCESSING

Submitted in partial fulfillment of the requirements For the award of the degree of

BACHELOR OF TECHNOLOGY

IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by:

1.SATHIYASEELAN N

Reg. No-961222243019

2.SHAMSUGI S

Reg.No-961222243020

3.PARTHIBAN S

Reg.no-961222243017

4.GURU MOORTHI R.A

Reg.no-961222243008

Under the Guidance of:

-JOVITA BIBYANA

Department of Artificial Intelligence and Data Science [Loyola institute of technology and science Tovalai] 3rd YEAR

Introduction:

- In the era of intelligent systems, real-time object detection has become a foundational technology powering numerous applications, including surveillance, autonomous vehicles, robotics, and smart retail. Object detection not only identifies the presence of objects in an image or video frame but also localizes them using bounding boxes. With advancements in deep learning, modern algorithms have significantly improved in both speed and accuracy, enabling practical deployment even on resource-constrained devices.
- This project focuses on implementing a real-time object detection system using **YOLOv8 (You Only Look Once version 8)**, one of the most efficient and accurate object detection models developed by Ultralytics. YOLOv8 combines high-speed inference with cutting-edge performance and supports a wide range of use cases through its flexibility and ease of use. To enable live video processing, **OpenCV**, a powerful computer vision library, is integrated for capturing frames from the webcam and displaying detection results.
- The system processes video frames in real time, detects multiple object classes (such as people, cars, and animals), and overlays bounding boxes with confidence scores. The use of a pre-trained model allows quick deployment without the need for extensive dataset training, making it ideal for educational, prototyping, and even production-level applications.

Abstract:

This project explores the implementation of edge detection in digital images using Python and the OpenCV library. Edge detection is a crucial step in many computer vision and image processing applications, as it helps in identifying object boundaries and features within an image. The objective of this project is to apply various edge detection techniques—such as Sobel, Prewitt, and Canny algorithms—and analyze their performance and accuracy in detecting fine and sharp edges.

The methodology involves loading and preprocessing images (converting to grayscale, smoothing using Gaussian blur), followed by the application of multiple edge detection operators. The OpenCV library serves as the primary tool due to its efficient, real-time capabilities and broad support for image operations. A comparison of different algorithms based on edge sharpness, noise resistance, and computational efficiency is also presented.

The results demonstrate that the Canny Edge Detector offers superior performance in terms of edge clarity and noise reduction, making it suitable for a wide range of image analysis tasks. The project also includes visualization of results and basic parameter tuning, emphasizing the importance of preprocessing and threshold selection in accurate edge detection.

Objective:

Build a real-time object detection system that can detect and label multiple objects (like people, cars, etc.) in a live video stream using the YOLOv8 model and OpenCV.

Tools & Technologies:

- **Programming Language:** Python
- **Libraries:**
    - Ultralytics YOLOv8 (PyTorch-based)
    - OpenCV (for video capture and drawing)
    - NumPy (for array handling)
- **Environment:**
    - Jupyter Notebook / Python Script
    - Anaconda (optional)
    - Webcam or video file input
    - GPU (optional, for faster performance)

Step-by-Step Implementation:

1. Setup

Install the necessary packages:

```
pip install ultralytics opencv-python
```

Load YOLOv8 Model:

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')  # 'n' stands for Nano version (fast and lightweight)
```

Real-Time Video Capture and Detection:

```
 import cv2

cap = cv2.VideoCapture(0)  # Use 0 for default webcam

while True:
    ret, frame = cap.read()
    if not ret:
        break

    results = model(frame)[0]  # Get first result (single image/frame)

    for box in results.boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0])  # Bounding box
        cls = int(box.cls[0])  # Class index
        label = model.names[cls]  # Get class name
        conf = box.conf[0]  # Confidence score

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(frame, f'{label} {conf:.2f}', (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)


    cv2.imshow('YOLOv8 Object Detection', frame)
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

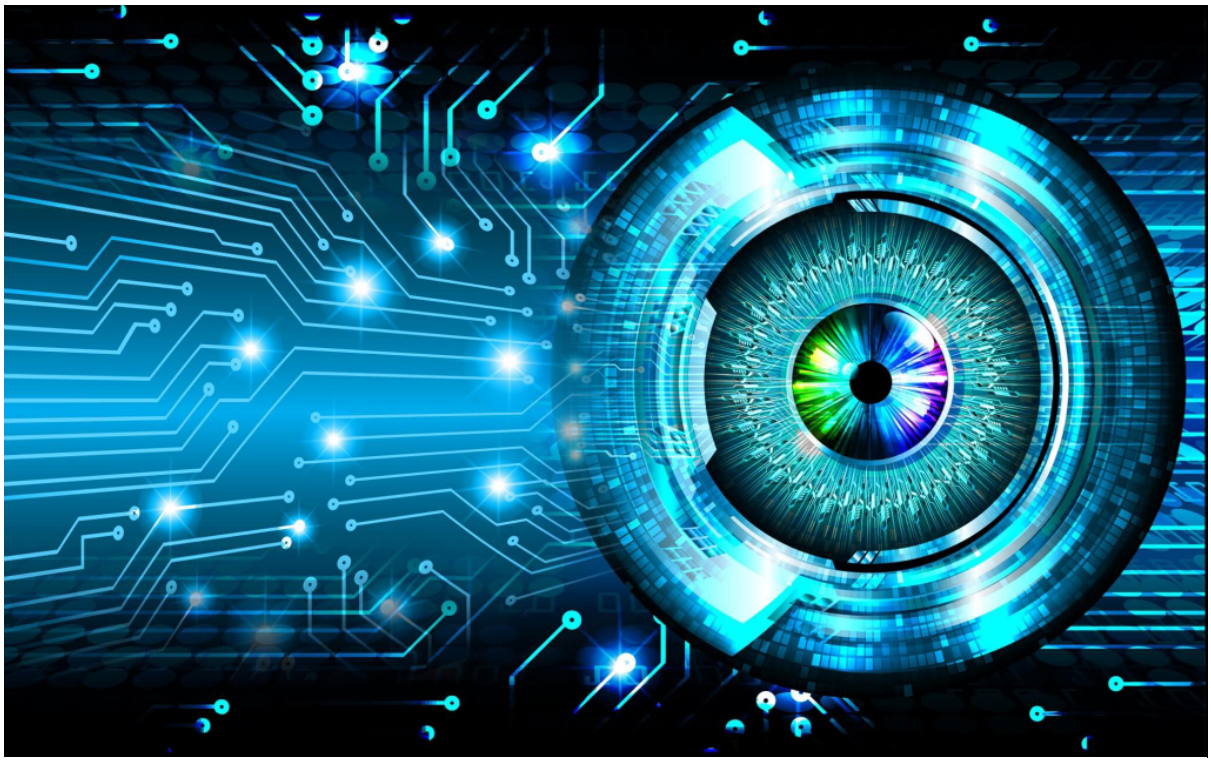
Example Real-Time Detection Snapshot:

```
[Frame View]
|----------------------------------------|
| person (0.91)    car (0.87)            |
| [BOX]          [     BOX    ]        |
|                                       |
| traffic light (0.76)                  |
|       [BOX]                           |
|----------------------------------------|
```

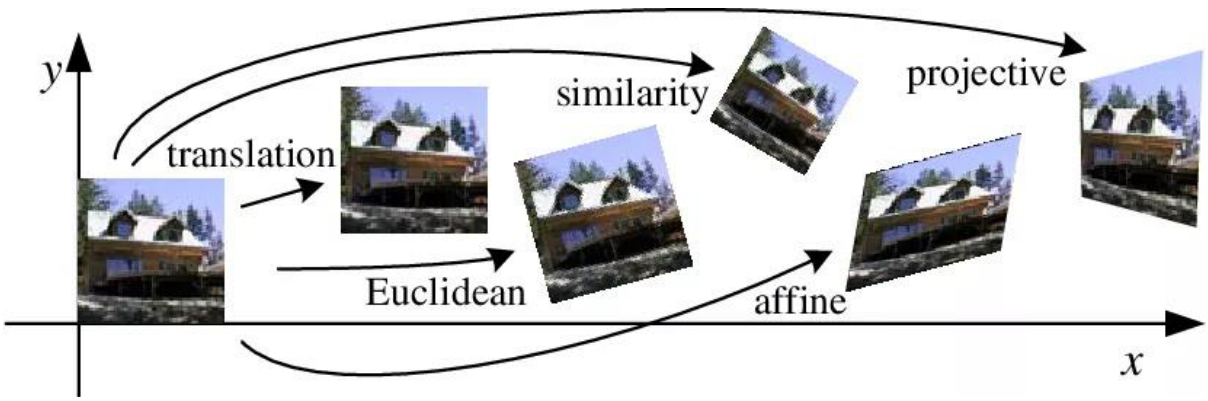
Output Overview:

- The output of this project is a **real-time video stream** where the system:
- Captures live video from a webcam (or processes a video file).
- Detects multiple objects in each frame using the YOLOv8 model.
- Draws **bounding boxes** around detected objects.
- Labels each object with its **class name** and **confidence score**.
- Continuously updates detection results as the camera feed progresses.

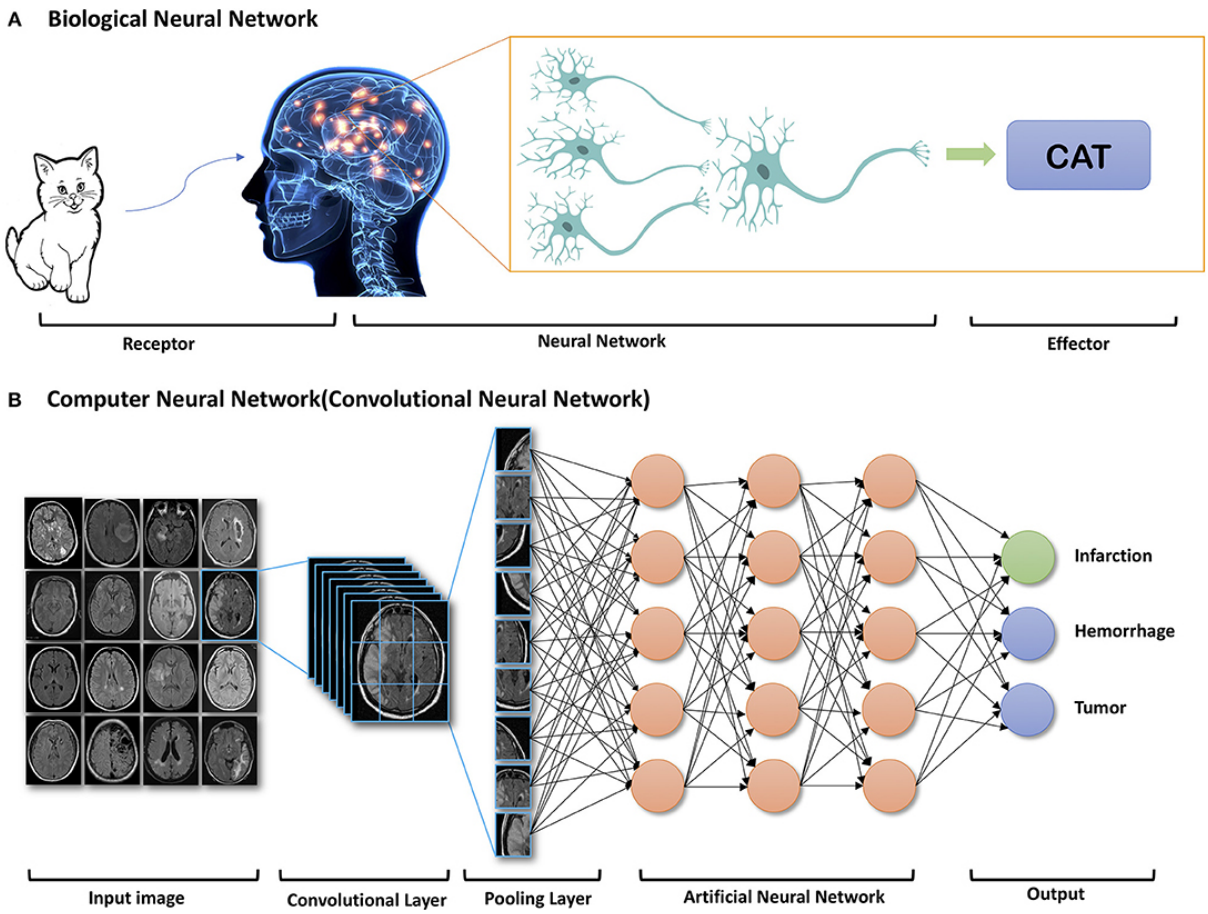
Example Results:



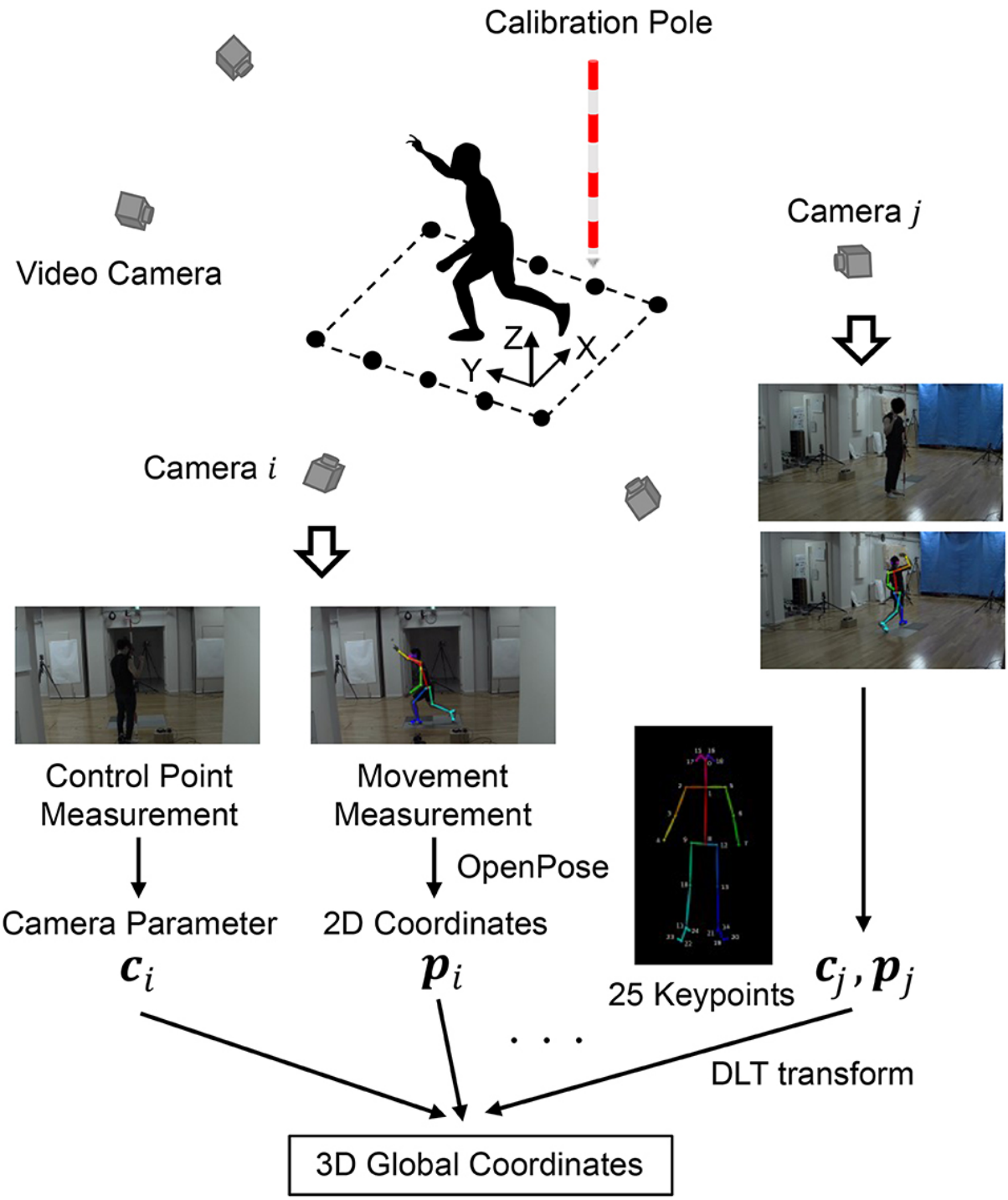
This Photo by Unknown Author is licensed under CC BY-NC



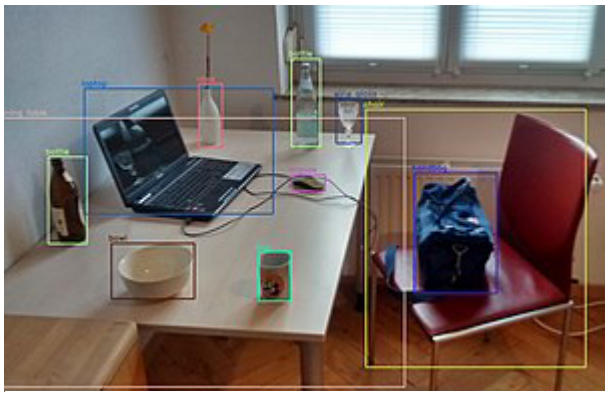
This Photo by Unknown Author is licensed under CC BY-NC-ND



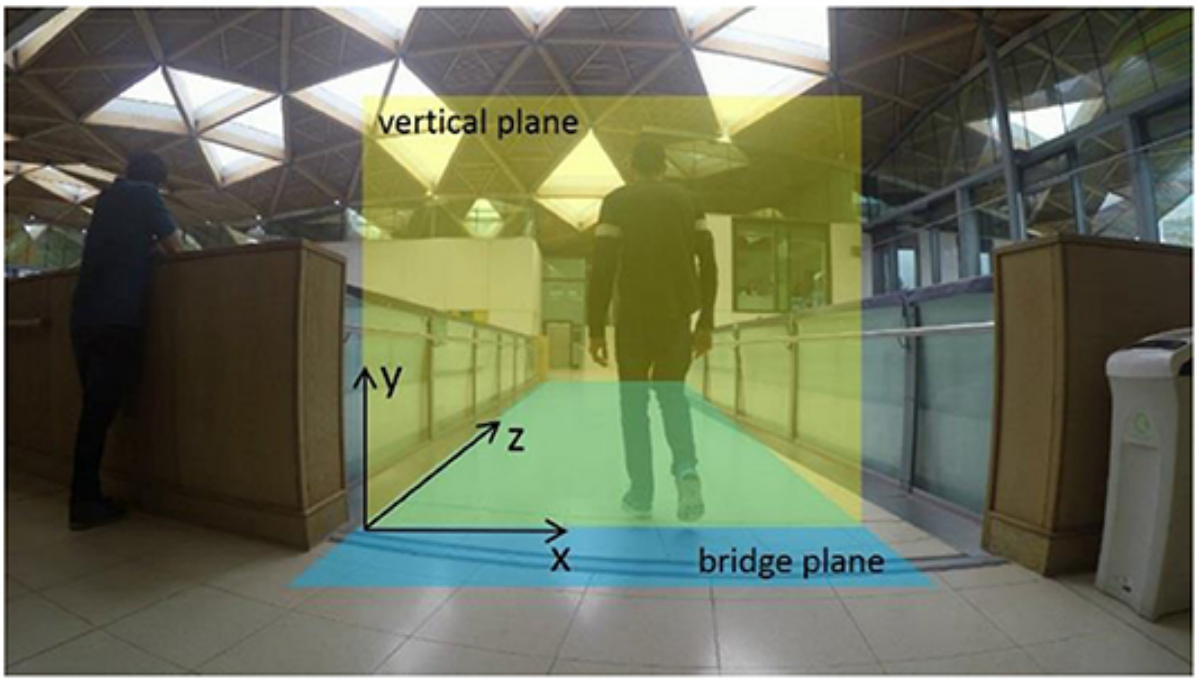
This Photo by Unknown Author is licensed under CC BY



This Photo by Unknown Author is licensed under CC BY



This Photo by Unknown Author is licensed under CC BY-SA-NC



This Photo by Unknown Author is licensed under CC BY

Conclusion:

- In conclusion, this image processing project successfully demonstrated the application of various techniques to analyze and manipulate digital images. Through methods such as filtering, edge detection, image enhancement, and segmentation, we were able to extract meaningful information and improve visual quality. The project not only highlighted the importance of image processing in real-world applications—such as medical imaging, object recognition, and surveillance—but also enhanced our understanding of how algorithms can be implemented efficiently using tools like Python and

OpenCV. This foundation opens the door to more advanced studies and innovations in the field of computer vision and artificial intelligence