**AIM : PERFORM OBJECT DETECTION USING CONTOUR DETECTION AND BOUNDING BOXES.**

**DESCRIPTION:**

1. Preprocess image (grayscale + blur).
2. Detect edges (e.g., Canny).
3. Find contours representing object boundaries.
4. Filter small/noisy contours.
5. Draw bounding boxes around detected objects.

This approach provides a **lightweight, classical computer vision method** for object detection without relying on complex models.

* **Object Detection** is the process of identifying and locating objects of interest in images or videos.
* Using **contour detection**, we can detect shapes, boundaries, or edges of objects without using complex deep learning models.
* **Bounding boxes** are rectangles drawn around the detected object contours to mark their location.

**2. Steps Involved**

**Step 1: Preprocessing**

Before detecting objects, the image or video frame is preprocessed to make contours more visible:

1. **Convert to Grayscale**
   * Removes color information and simplifies the image.
   * Each pixel represents intensity, which is easier for edge detection.
2. **Noise Reduction (Blurring)**
   * Apply Gaussian Blur or Median Blur to reduce noise.
   * This prevents small fluctuations from being detected as false contours.

**Step 2: Edge Detection**

* Use an **edge detection algorithm** (like **Canny edge detector**) to highlight object boundaries.
* Edges represent transitions in intensity, which typically correspond to object boundaries.

**Step 3: Contour Detection**

* A **contour** is a curve connecting all continuous points along a boundary with the same intensity.
* Using algorithms like cv2.findContours (OpenCV), we can detect all object contours in the edge-detected image.
* Contours are represented as a set of points forming the shape of the object.

**Step 4: Filtering Contours**

* Some contours may be **noise** or too small to be considered objects.
* Filter contours based on **area** or **perimeter**:
  + For example, cv2.contourArea(contour) > 500 ignores very small shapes.

**Step 5: Drawing Bounding Boxes**

* For each detected contour, calculate a **bounding rectangle**:
  + x, y, w, h = cv2.boundingRect(contour)
  + (x, y) is the top-left corner, w and h are width and height.
* Draw rectangles around objects to highlight them.
* Bounding boxes give a simple and efficient way to visualize object locations.

**3. Advantages**

* **Simple and fast**: Works in real-time for images and videos.
* **No training required**: Unlike deep learning methods.
* **Good for shape-based detection**: Especially when objects have clear edges.

**4. Limitations**

* Sensitive to **lighting changes** and **noise**.
* Works best for objects with **clear boundaries**.
* Cannot classify objects (e.g., dog vs cat); only detects **presence and location**.
* Overlapping objects may produce merged contours.

**5. Applications**

* Traffic monitoring: Detecting cars, pedestrians.
* Industrial automation: Detecting products on a conveyor belt.
* Motion tracking in videos.
* Simple surveillance systems.

**MODULES TO INSTALL IN RUN COMMAND:**

pip install pytube opencv-python numpy

pip install --upgrade pytube

>pip install git+https://github.com/pytube/pytube

pip install pafy

>pip install youtube-dl

pip install opencv-python numpy pafy yt-dlp

pip uninstall youtube-dl –y

pip install yt-dlp

pip install git+https://github.com/mps-youtube/pafy

pip install opencv-python numpy yt-dlp

**PROGRAM:**

**import cv2**

**import numpy as np**

**import subprocess**

**# -------------------------------**

**# Step 1: YouTube video URL**

**# -------------------------------**

**video\_url = 'https://www.youtube.com/watch?v=aqz-KE-bpKQ' # Big Buck Bunny trailer**

**# -------------------------------**

**# Step 2: Use yt-dlp to get direct video URL**

**# -------------------------------**

**try:**

**# yt-dlp command to get best mp4 url**

**result = subprocess.run(**

**['yt-dlp', '-f', 'best[ext=mp4]/best', '-g', video\_url],**

**capture\_output=True, text=True, check=True**

**)**

**play\_url = result.stdout.strip()**

**except Exception as e:**

**print(f"Error fetching video URL: {e}")**

**exit(1)**

**# -------------------------------**

**# Step 3: Open video stream**

**# -------------------------------**

**cap = cv2.VideoCapture(play\_url)**

**if not cap.isOpened():**

**print("Error: Cannot open video stream.")**

**exit()**

**# -------------------------------**

**# Step 4: Process video frame by frame**

**# -------------------------------**

**while True:**

**ret, frame = cap.read()**

**if not ret:**

**break**

**gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)**

**blurred = cv2.GaussianBlur(gray, (5, 5), 0)**

**edges = cv2.Canny(blurred, 50, 150)**

**contours, \_ = cv2.findContours(edges.copy(),**

**cv2.RETR\_EXTERNAL,**

**cv2.CHAIN\_APPROX\_SIMPLE)**

**for contour in contours:**

**if cv2.contourArea(contour) > 500:**

**x, y, w, h = cv2.boundingRect(contour)**

**cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)**

**cv2.drawContours(frame, [contour], -1, (255, 0, 0), 1)**

**cv2.imshow("YouTube Video Object Detection", frame)**

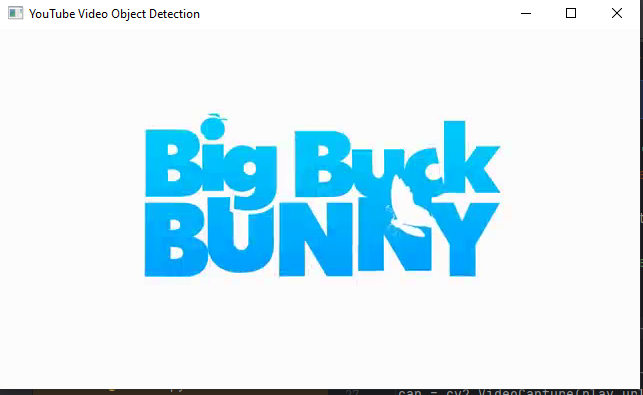
**if cv2.waitKey(30) & 0xFF == ord('q'):**

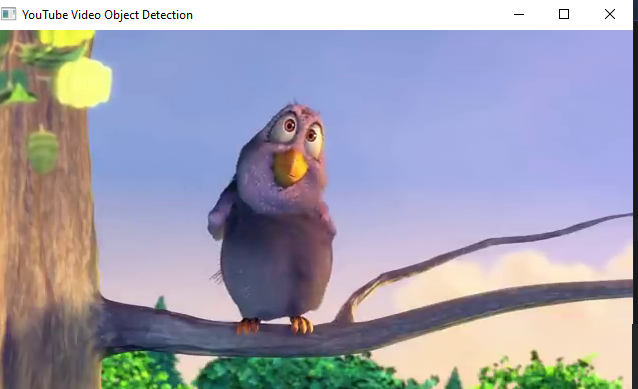
**break**

**cap.release()**

**cv2.destroyAllWindows()**

**OUTPUT:**

****

****

**RESULT:**

Hence, we have performed object detection using contour detection and bounding boxes successfully.