EXERCISE NO. 4
REAL-TIME OBJECT DETECTION

AIM:
To implement object detection in real-time.

ALGORITHM:

1. Import the necessary libraries.
2. Load the pre-trained SSD model and the configuration file.
3. Define the object class labels.
4. Start video capture using a web camera.
5. Store the processed frames as time is being tracked.
6. Detect the objects in the processed frame by drawing bounding boxes and assigning labels with confidence.
7. Display the last frame with object(s) within bounding boxes.

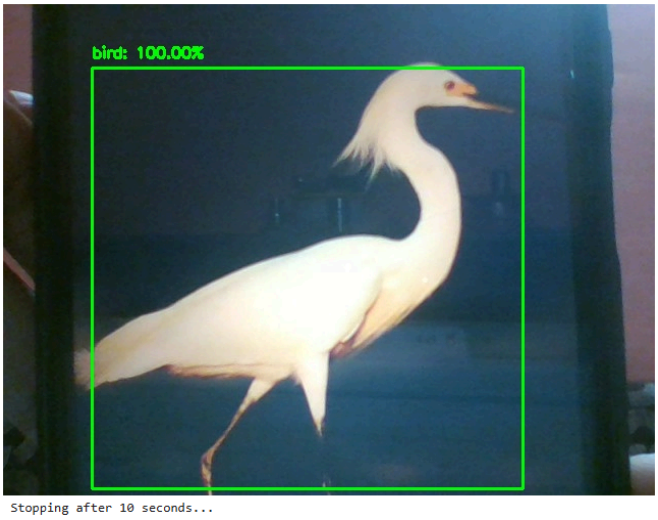
PROGRAM:

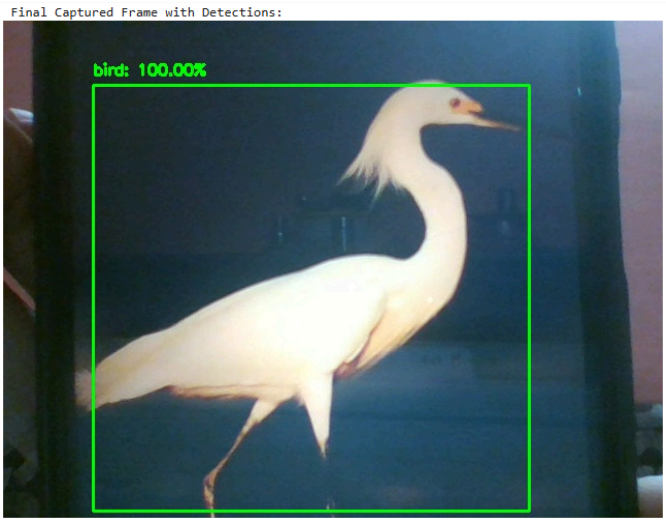
```
import cv2
import numpy as np
import time
import PIL.Image
import io
from IPython.display import display, clear_output

prototxt_path = ".../MobileNetSSD_deploy.prototxt"
caffemodel_path = ".../MobileNetSSD_deploy.caffemodel"

net = cv2.dnn.readNetFromCaffe(prototxt_path, caffemodel_path)

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat", "bottle", "bus",
        "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike",
        "person", "pottedplant", "sheep", "sofa", "train", "tvmonitor", 'fox']

cap = cv2.VideoCapture(0)

if not cap.isOpened():
   print("Error: Could not open camera.")
```

```
    exit()

start_time = time.time()
final_frame = None

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: Frame not captured.")
        break

    print("Frame Captured!")

    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)
    net.setInput(blob)
    detections = net.forward()

    print("Detections Processed...")

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.2:  # Confidence threshold
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            label = f"{CLASSES[idx]}: {confidence * 100:.2f}%"

            cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 255, 0), 2)
            cv2.putText(frame, label, (startX, startY - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,
0), 2)

            print(f"Detected: {CLASSES[idx]} with {confidence * 100:.2f}% confidence")

    _, buffer = cv2.imencode(".jpg", frame)
    img = PIL.Image.open(io.BytesIO(buffer))
```

```
    clear_output(wait=True)
    display(img)

    final_frame = img

    if time.time() - start_time > 20:
        print("Stopping after 10 seconds...")
        break

cap.release()
cv2.destroyAllWindows()

if final_frame:
    print("\nFinal Captured Frame with Detections:")
    display(final_frame)
```

OUTPUT:





RESULT:

Thus the program has been successfully implemented and verified.