Import numpy as np

Import imutils

Import cv2

Import time

Import serial

# … (other imports and variables)

# Define camera calibration matrix (replace with your actual values)

Camera\_matrix = np.array([[focal\_length, 0, principal\_point\_x],

[0, focal\_length, principal\_point\_y],

[0, 0, 1]])

# Define ROI for vehicle detection (adjust based on your camera setup)

Roi\_x\_min, roi\_y\_min, roi\_width, roi\_height = 100, 50, 500, 300

While True:

Ret, frame = vs.read()

Frame = imutils.resize(frame, width=1000)

# Apply ROI

Frame = frame[roi\_y\_min:roi\_y\_min+roi\_height, roi\_x\_min:roi\_x\_min+roi\_width]

# … (object detection, confidence thresholding, etc.)

For I in np.arange(0, detShape):

Confidence = detections[0, 0, I, 2]

If confidence > confThresh:

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”)

# Calculate pixel dimensions within ROI

Pixel\_width = endX – startX

Pixel\_height = endY – startY

# Convert pixel dimensions to real-world units using camera calibration

Real\_width = pixel\_width \* camera\_matrix[0, 0] / (pixel\_width + camera\_matrix[0, 2]) # Assuming camera plane is parallel to ground

Real\_height = pixel\_height \* camera\_matrix[1, 1] / (pixel\_height + camera\_matrix[1, 2])

# … (display bounding box, print dimensions, etc.)

# … (calculate and send speed based on dimensions, using your desired control mechanism)

# … (other processing)

# …