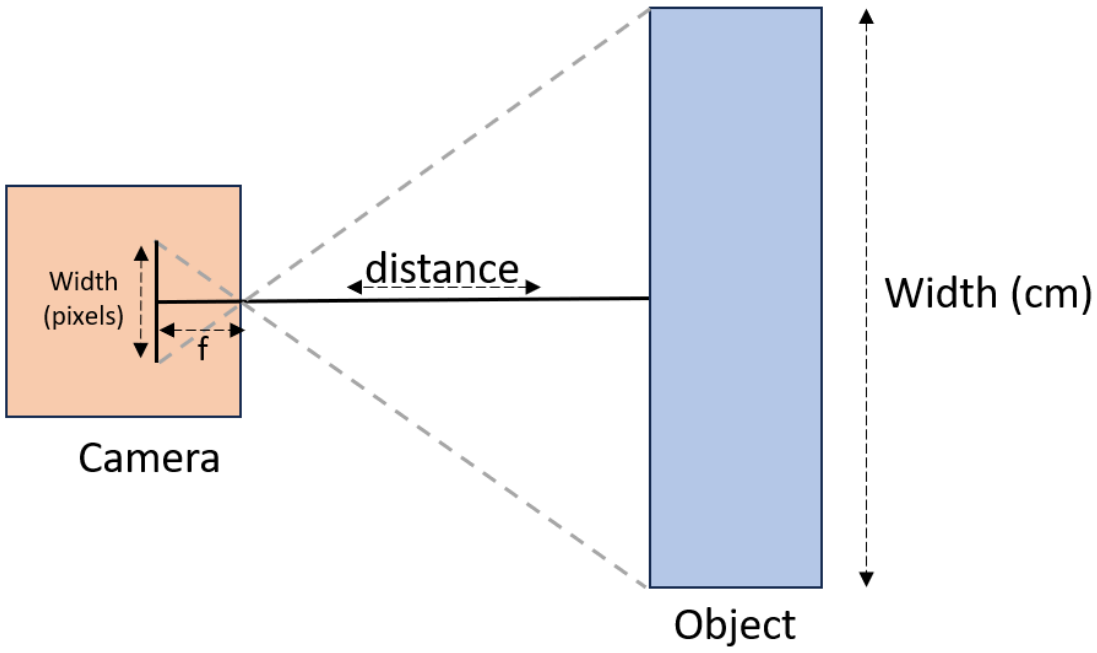
# OBJECT DETECTION and DISTANCE CALCULATION using YOLOv8

## Calculating the Distance of Detected Object

To calculate the distance of the detected object following method is used:




Relation between all the variables is given by:

$$f = \frac{distance \ * width \ (in \ pixels)}{width \ (in \ cm)}$$

Rearranging above formula:

$$Distance = \frac{width \ (in \ cm) * f}{width \ (in \ pixels)}$$

## Steps to calculate distance:

1. First, f (focal point) is calculated
   a. A known-width object is placed at a known distance to calculate the focal point.
   b. Then using YOLO, the object is detected, and its width in pixels is calculated from the boundary box dimensions.

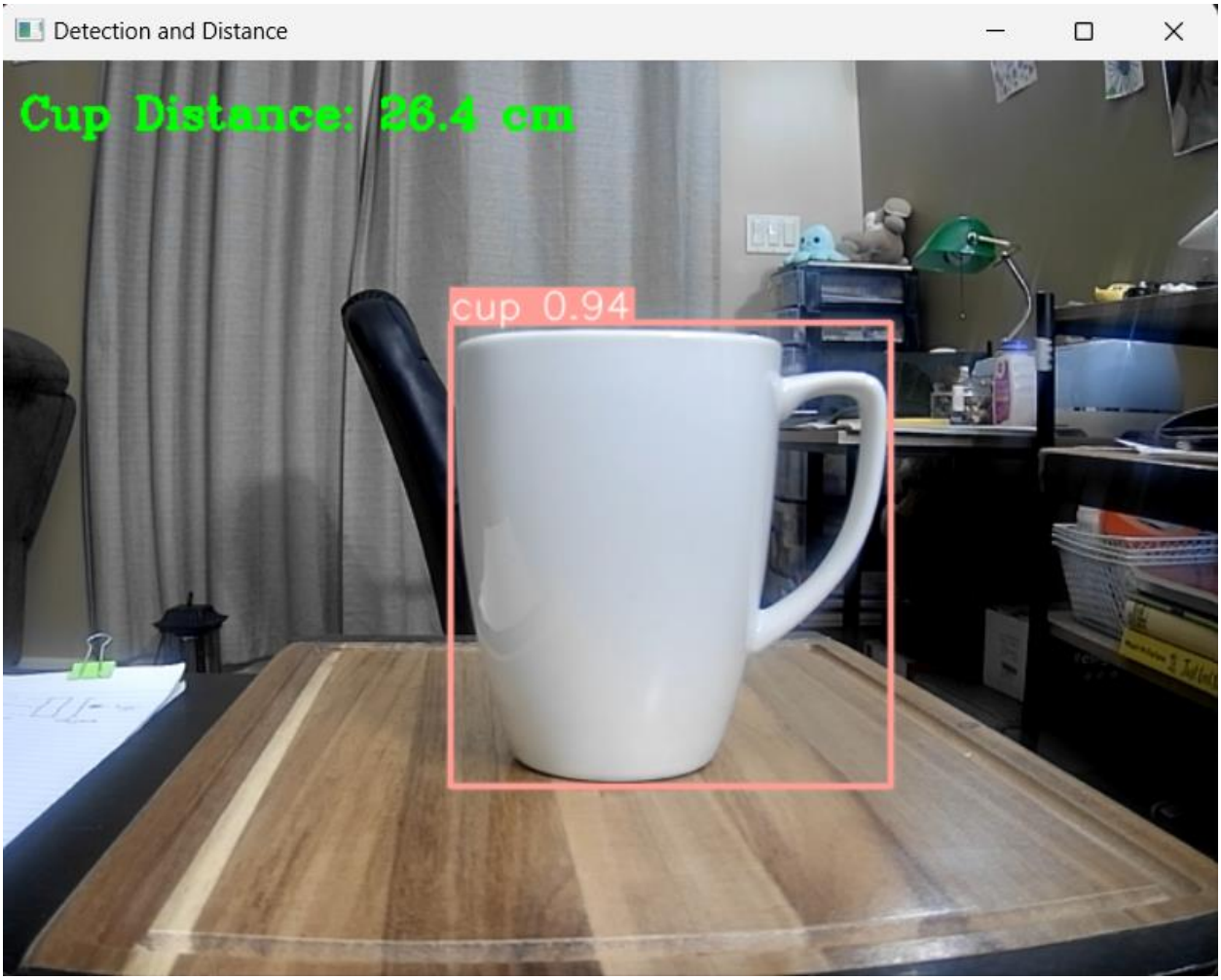
2. Calculating the distance
    a. Once the focal point is calculated, it is provided to the distance calculation formula as a constant, and the only thing that is not a constant is the width of the detected pixels, which is calculated continuously and fed to the distance formula.
    b. Focal point is calculated once.

## Code:

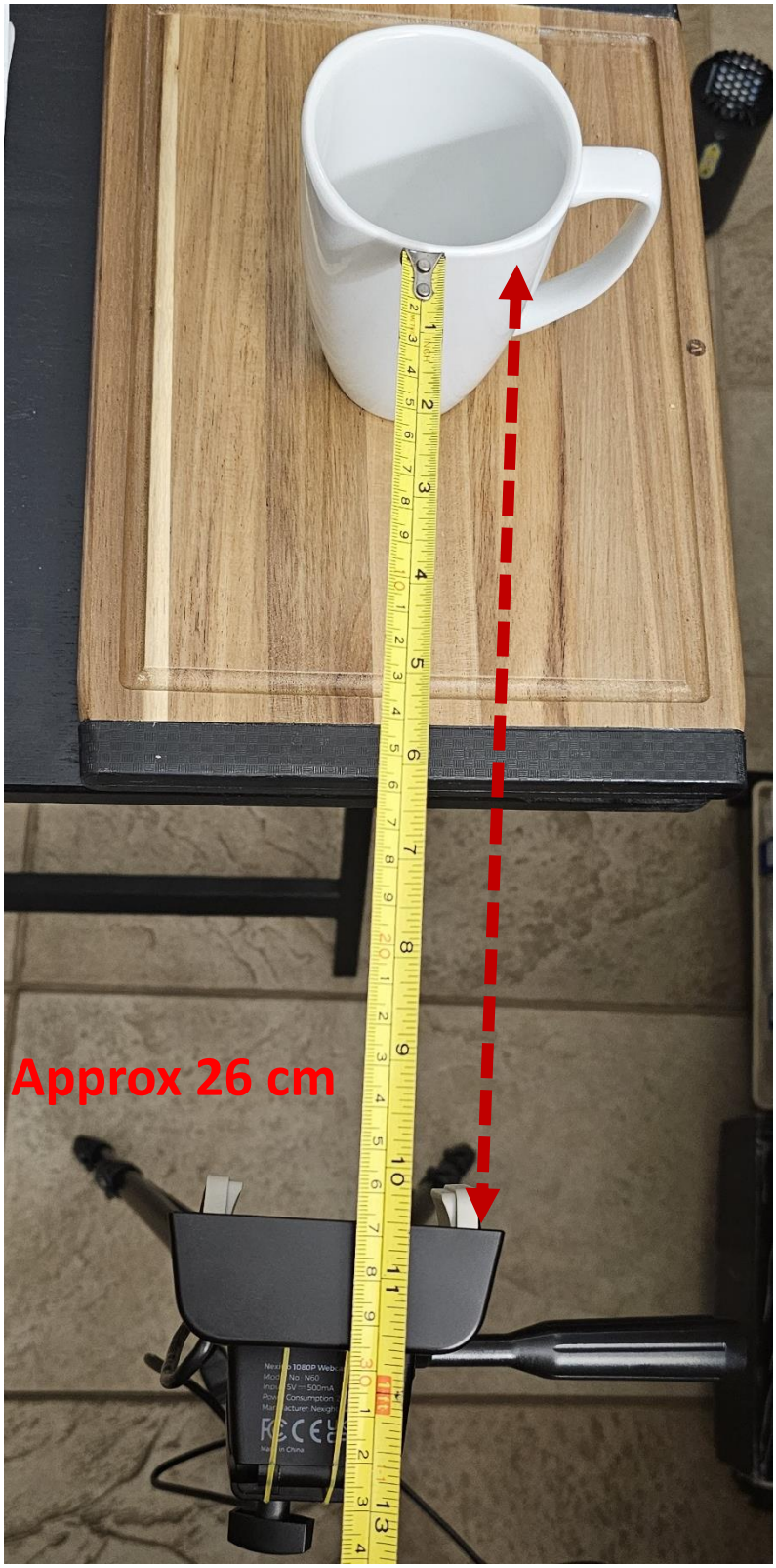
```python
import cv2
from ultralytics import YOLO
import torch

# To calculate Focal Length
widthOfObject_cm = 11           # Cup width
distanceOfObject_cm = 30
objectToDetecte_CUP = 41        # 41: CUP
objectToDetecte_Apple = 47      # 47: Apple

# Access webcam
cap = cv2.VideoCapture(0)   # 0 refers to webcam

# Load the YOLOv8 model
model = YOLO('yolov8n.pt')

while cap.isOpened():

    # Read from webcam
    success, frame = cap.read()

    # if Read from webcam is successful
    if success:
        # Run YOLO on captured video (object is frame) AND  Return detected objects with confidence of >=80%
        # to reduce detected objects
        results = model.predict(frame, conf = 0.8)

        # .cpu().numpy() for easy calculations
        objectClass = results[0].boxes.cls
        objectxyxy = results[0].boxes.cpu().numpy().xyxy    # Boundary Box Coordinates XYXY
        objectxywh = results[0].boxes.cpu().numpy().xywh    # Boundary box Coordinated XY + w (width) and h (hight)

        # Detected Objects Class ID
        output = torch.tensor(objectClass)

        #detected object
        detectedObject = results[0].plot()

        # Check for Objects and Calculate distance
        # Checking for Cup
        if objectToDetecte_CUP in output:
            print("ID 41: CUP detected")
            # Calculate Distance of CUP
            widthOfObject_px = objectxywh[0][3].astype(int)

            # Calculate Focal Length        (Calculate Once to calibrate the system)
            #f = (widthOfObject_px * distanceOfObject_cm) / widthOfObject_cm
            #print(f)
            f = 580.9090                                    # Calculated
            # Calculated to be 580.9090909090909

```

```
            # Distance Calculations
            d = (widthOfObject_cm * f ) / widthOfObject_px
            #print(d)

            # Distance Text
            cv2.putText(detectedObject, f'Cup Distance: {round(d,2)} cm', (10,35),
cv2.FONT_HERSHEY_COMPLEX, 0.75, (0,255,0),2)

        # Display Captured Video with detected objects
        cv2.imshow("Detection and Distance", detectedObject)

    # loop every 1000ms (main loop: while)
    cv2.waitKey(1000)
```

## Output:



Physical Calculation:



## Code Explanation:

- Code line 6 and 7 corresponds to step 1 (First, f (focal point) is calculated). As an example, the known width of a cup and distance from the camera are provided here.
- From line 47, the focal point is calculated.
- The code is commented because the focal point is calculated at capture time. It can be seen on line 49, which comes out to be 580.9090
- Then, in line 44, the width of the detected object (cup) is calculated and is calculated in every iteration of the loop.
- Here, we have all the required values to calculate the distance. Now, as seen in line 53, distance is calculated.
- From line 57, the calculated distance is placed onto the output video.

Further:

- Any object's distance can be calculated (provided its class ID is in the YOLO database) by adding another if statement in the above-provided code.
- Source for YOLO Object Data Class: https://github.com/ultralytics/ultralytics/blob/main/ultralytics/datasets/coco.yaml
- The code for detecting the distance of the cup is from line 47; this 'if' statement is responsible for detecting the cup and calculating its distance. As seen in the code, the class ID for a cup is 41. Also, the class ID for an apple is provided as an example, i.e., 47 in lines 8 and 9.
- Similarly, more condition statements can be added to the above code to detect a particular object and further compute its distance.