IMAGE PROCESSING PROJECT

Object Measurement Using OpenCV

This project reads an image and based on the dimensions of a reference object; it finds the dimensions of other objects in a scene. The reference object must be the leftmost, topmost object in the scene. In this case, a box of dimension 2cm x 2cm is taken as a reference object.

For any other reference object, we can provide the actual width of the object.

Prerequisites:

1. Python 3
2. OpenCV
3. NumPy
4. SciPy
5. VS Code, Google Colab (Any IDE)

Constraints

1. Shadows can cause incorrect prediction. This can be avoided by making sure there is enough light.
2. To get accurate object boundary, dark background is used.

Algorithm

1. Image pre-processing
    - Read an image and convert it to grayscale
    - Blur the image using Gaussian Kernel to remove unnecessary edges/ reduce noises.
    - Edge detection using Canny edge detector is performed on the blurred image using a threshold value.
    - Create a kernel for morphological operations (dilation and erosion) using NumPy.
    - Dilate the edges obtained from Canny edge detection to make them more contiguous.
    - Erode the dilated image to further refine the edges.

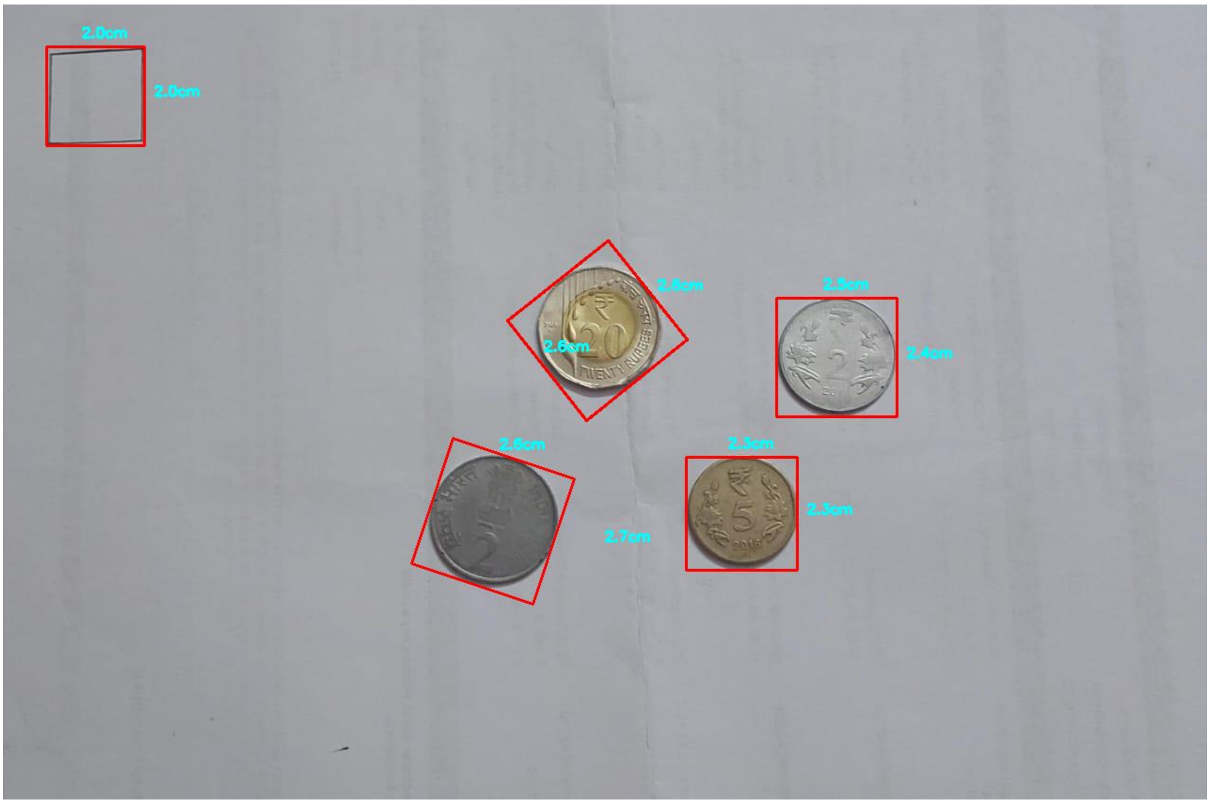
3. Reference object
    - Calculate how many pixels are there per metric (centimeter is used here)
4. Compute results
    - Draw bounding boxes around each object and calculate its height and width

Code

```
import imutils
import cv2
from scipy.spatial.distance import euclidean
from imutils import perspective_transform as persp_trans
from imutils import contours as contours_util
import numpy as np

def display_images(img_list):
    for i, img in enumerate(img_list):
        cv2.imshow("image_" + str(i), img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

image_path = "images/c.jpg"

#Image pre-processing
image = cv2.imread(image_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (9, 9), 0)
edged = cv2.Canny(blurred, 50, 100)
edged = cv2.dilate(edged, None, iterations=1)
edged = cv2.erode(edged, None, iterations=1)

#Image segmentation
contours = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(contours)
(contours, _) = contours_util.sort_contours(contours)
contours = [x for x in contours if cv2.contourArea(x) > 500]

#We have taken a 2cm x 2cm square as a reference object
ref_object = contours[0]
box = cv2.minAreaRect(ref_object)
box = cv2.boxPoints(box)
box = np.array(box, dtype="int")
box = persp_trans.order_points(box)
(top_left, top_right, bottom_right, bottom_left) = box
dist_in_pixel = euclidean(top_left, top_right)
dist_in_cm = 2
pixel_per_cm = dist_in_pixel / dist_in_cm

#Calculating contours and sizes
for contour in contours:
    box = cv2.minAreaRect(contour)
    box = cv2.boxPoints(box)
    box = np.array(box, dtype="int")
    box = persp_trans.order_points(box)
    (top_left, top_right, bottom_right, bottom_left) = box
    cv2.drawContours(image, [box.astype("int")], -1, (0, 0, 255), 2)
    horizontal_midpoint = (top_left[0] + int(abs(top_right[0] -
top_left[0])/2), top_left[1] + int(abs(top_right[1] - top_left[1])/2))
    vertical_midpoint = (top_right[0] + int(abs(top_right[0] -
bottom_right[0])/2), top_right[1] + int(abs(top_right[1] -
bottom_right[1])/2))
    width = euclidean(top_left, top_right) / pixel_per_cm
    height = euclidean(top_right, bottom_right) / pixel_per_cm
    cv2.putText(image, "{:.1f}cm".format(width), (int(horizontal_midpoint[0] -
15), int(horizontal_midpoint[1] - 10)),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 2)
    cv2.putText(image, "{:.1f}cm".format(height), (int(vertical_midpoint[0] +
10), int(vertical_midpoint[1])),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 2)

display_images([image])
```

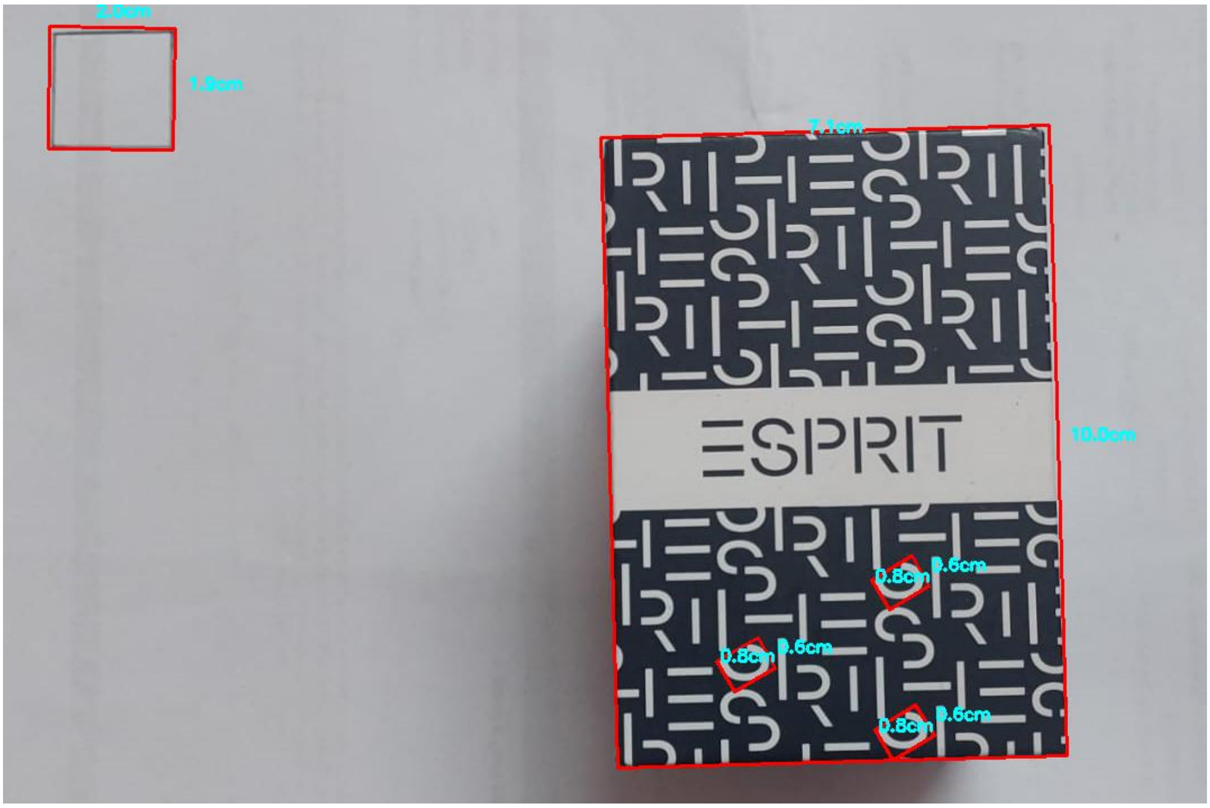
RESULTS-

Case 1:



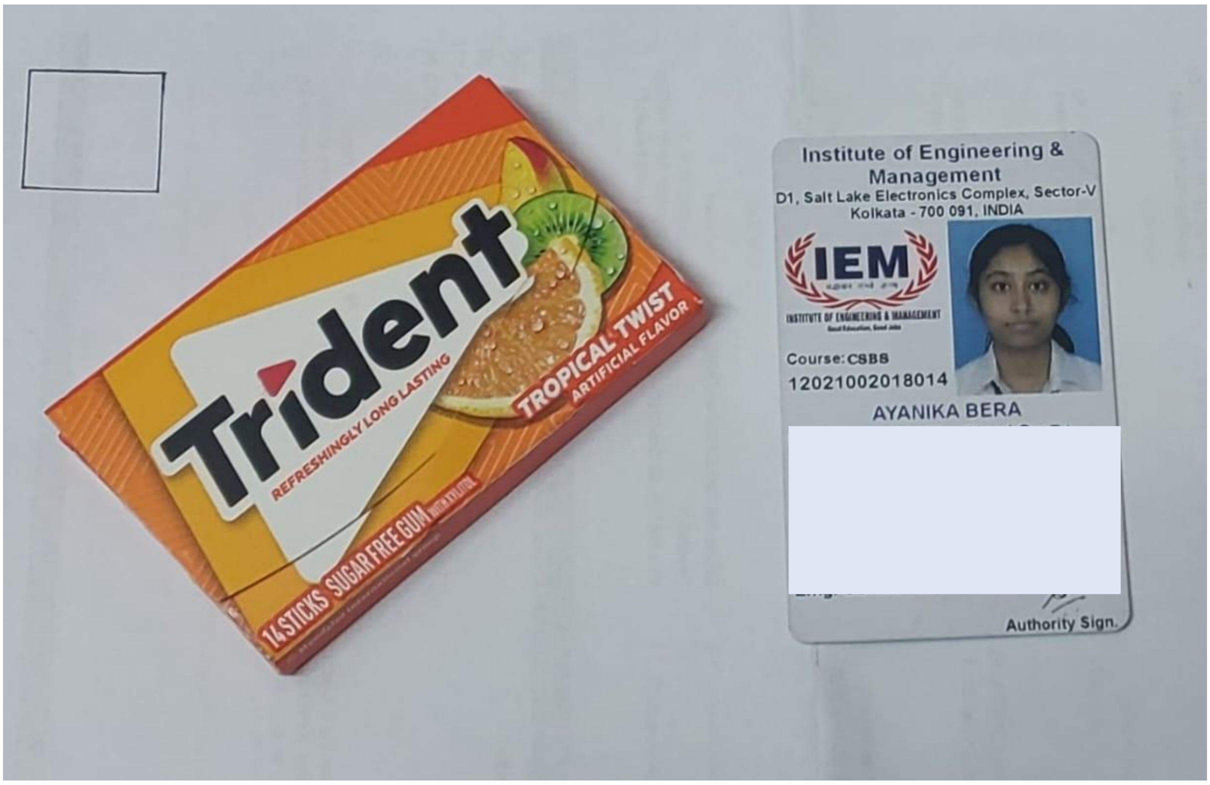
#Accurate Results.

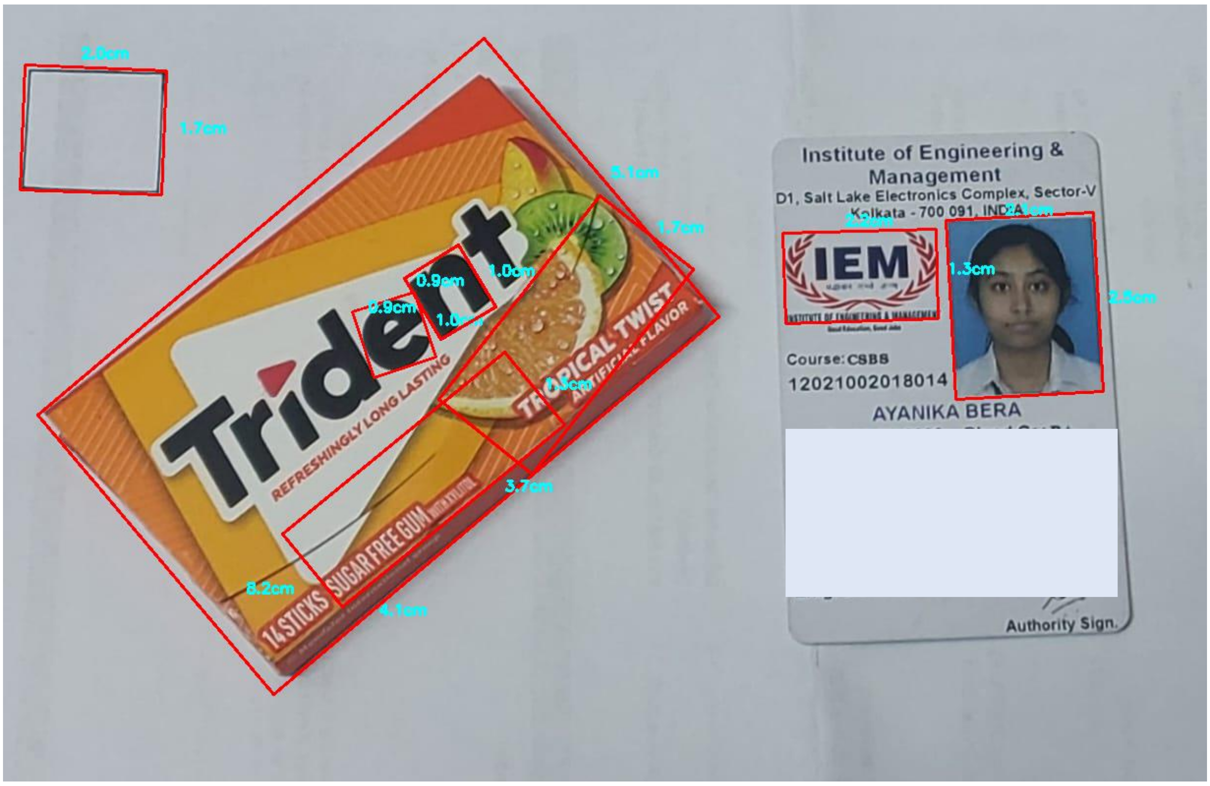
Case 2:



#Accurate Results.

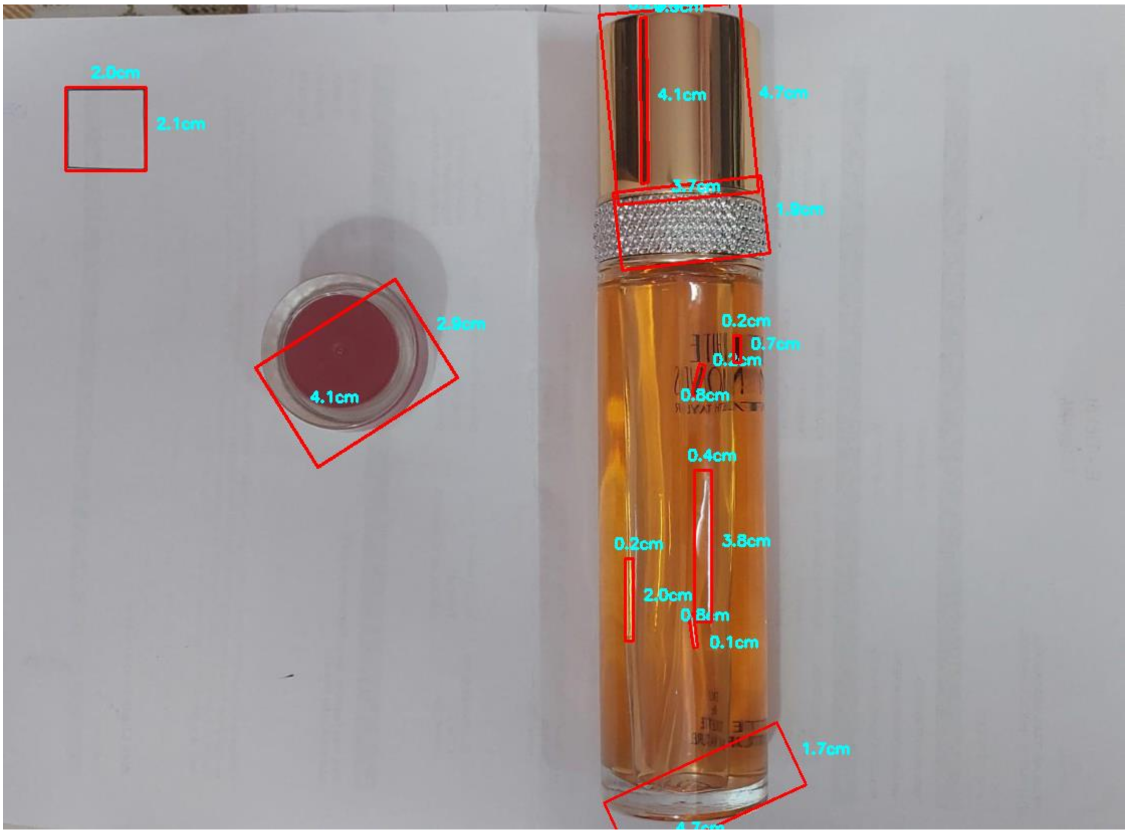
Case 3:





#To prove how one of our constraints affects the result- because the ID card and background are not contrast- we don’t see the main object border (which we see in other cases)

Case 4:



#Too many noises in the image lead to extra measures inside the perfume bottle.

Project completed and compiled by-

1.	Shreyansh Agarwal	(12021002018009)	10
2.	Ayanika Bera	(12021002018014)	12
3.	Ankana Ghosh	(22022002018010)	63