Problem Statement:

An Approach of dimensioning and measuring distance from a reference object to other objects in an image with perspective transformation.

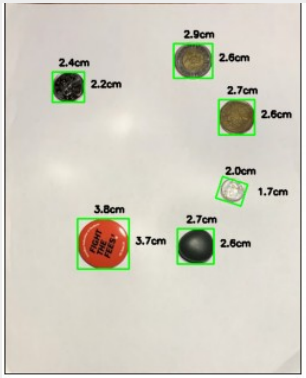
The project also compares performance of proposed system with a system without perspective transformation.

- **With Perspective Transformation**

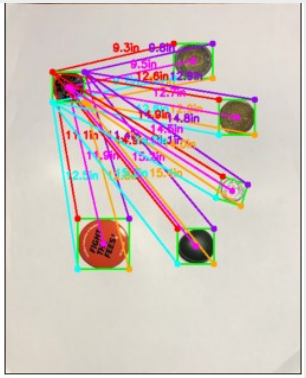
Input Image

Transformed Image

Dimensioning



Distance Measurement



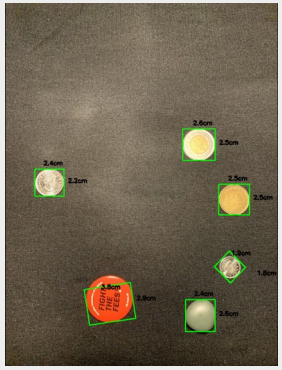
Accuracy Test

Object Name	AH (cm)	PH (cm)	AW (cm)	PW (cm)	Avg. Accuracy (%)
Quarter Coin	2.4	2.2	2.4	2.4	95.83
Toonie	2.8	2.6	2.8	2.9	94.70
Dime	1.8	1.7	1.8	2	92.22
Loonie	2.7	2.6	2.7	2.7	98.15
Badge	3.7	3.7	3.7	3.8	98.68
Stone	2.5	2.5	2.6	2.7	98.15

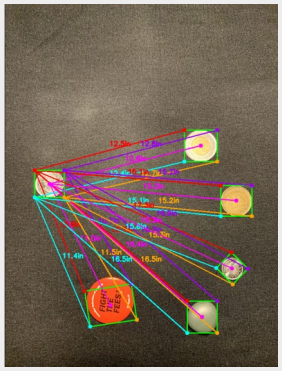
- **Without Perspective Transformation**

Input Image

Dimensioning



Distance Measurement

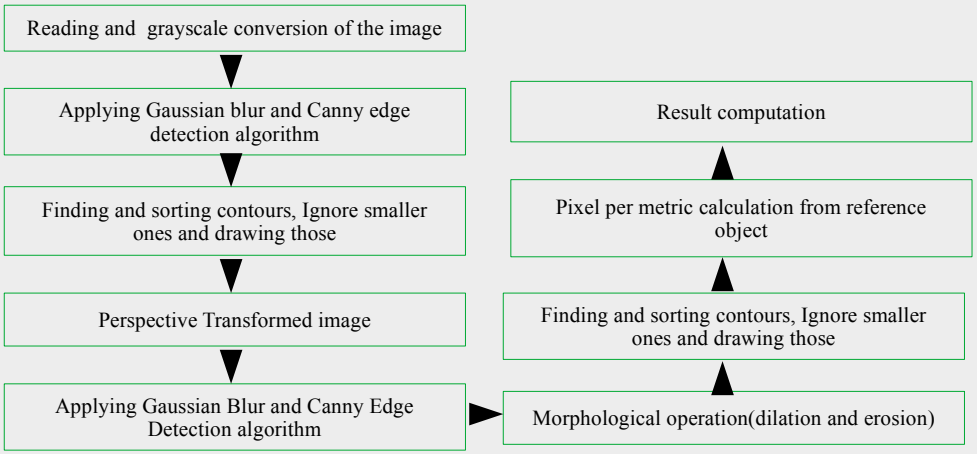


Accuracy Test

Object Name	AH (cm)	PH (cm)	AW (cm)	PW (cm)	Avg. Accuracy (%)
Quarter Coin	2.4	2.2	2.4	2.4	95.83
Toonie	2.8	2.5	2.8	2.5	89.29
Dime	1.8	1.8	1.8	1.9	97.37
Loonie	2.7	2.5	2.7	2.5	92.59
Badge	3.7	2.9	3.7	3.8	87.87
Stone	2.5	2.5	2.6	2.4	96.15

We can observe that, system with perspective transformation performs better that without transformation in dimensioning objects and distance measurement.

Processes of the System:



Brief Description of Processes:

A. Image Pre-processing and Perspective Transformation:

- Reading and input image, resizing it for quicker process, doing grayscale conversion , next steps are to apply 5X5 Gaussian filter to remove high-frequency noise and call Canny edge detection algorithm with threshold 75 and 200 to smooth edge detection process.
- Finding, sorting contours then considering the significant contours only, Drawing those with OpenCV function cv2.getPerspectiveTransform to get transformed image.
- Then again applying a 15x15 Gaussian filter to blur the image to pass through Canny edge detector with 50 and 70 threshold. Then applying morphological operator (dilation + erosion) with iteration 1 to close any gaps in the edge map.

B. Object Segmentation:

- After getting external contours from the edge map with cv2.RETR_EXTERNAL, next steps are to sort those contours from left to right and detect the reference object.

C. Reference Object:

- This reference object will help to count number of pixels captured by each object in an image.
- This pixel number provides the approximate number of pixels per every unit measurement of its width, which is known as pixel per metric. We can express pixels per metric as follows:

  pixel per metric = width of the object in pixel / actual width of the object.

D. Computation of Results:

- Then scanning through the previously found contour list, next steps are to cancel out insignificant contours compute and order rotated bounding box with cv2.boxPoint function from top-left to bottom-left (clockwise).
- Then an object's height and width are calculated from the division of Euclidean distance between the set of midpoints of the object and pixels per metric parameter
- Lastly, our model computes the distance for the bounding box coordinates and the centroid coordinates by dividing the Euclidean distance between the ordered bounding box points of the reference object and the targeted object by pixels per metric.