Measuring the size of objects with computer vision

We introduce a way to measure the size of an object using OpenCV which is implemented in Python. The overall pipeline is as follows: first, the objects to be measured is recognized and their bounding box is predicted via finetuning object detection model, in the next step we convert the image it to grayscale, and applying a segmentation method via kmeans to separate the object from the background. Then we finds the object's contours in the binary image and draws them on the original image. The code calculates the height and width of the object in pixels and converts the height and width to a real-world unit of measurement using a scale factor. Finally, it prints the size of the object in the chosen unit of measurement.

Object detection and classification

Since the scene is crowded and it may be difficult to find the contour of our main object, we used an object detection model to detect and find the bounding box over the green object.

We fine-tuned Retina Net architecture on very few examples of a novel classes (3 training sample for each class) after initializing from a pre-

trained COCO checkpoint. Training runs in estimated time less than 5 minutes.

We will start with 3 classes of data each consisting of 3 images of a rectangle, polygon and circle object. Note that the coco dataset does not contain rectangle/circle or polygon 3D printed object, so these are a novel class.

To train our own object detection model we need to annotate the data by drawing a box around them in each image manually in order to populate the groundtruth bounding boxes for training. thereafter we add the class annotations (Here, we have 3 class). We also convert everything to the format that the training loop expects (e.g., everything converted to tensors, classes converted to one-hot representations, etc.).

The figure below shows the input and output of this step.

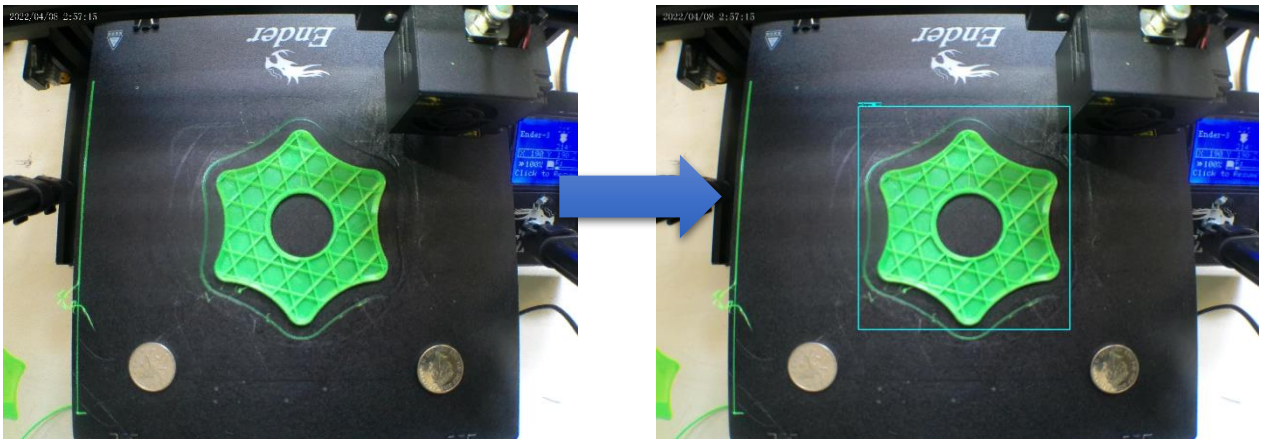
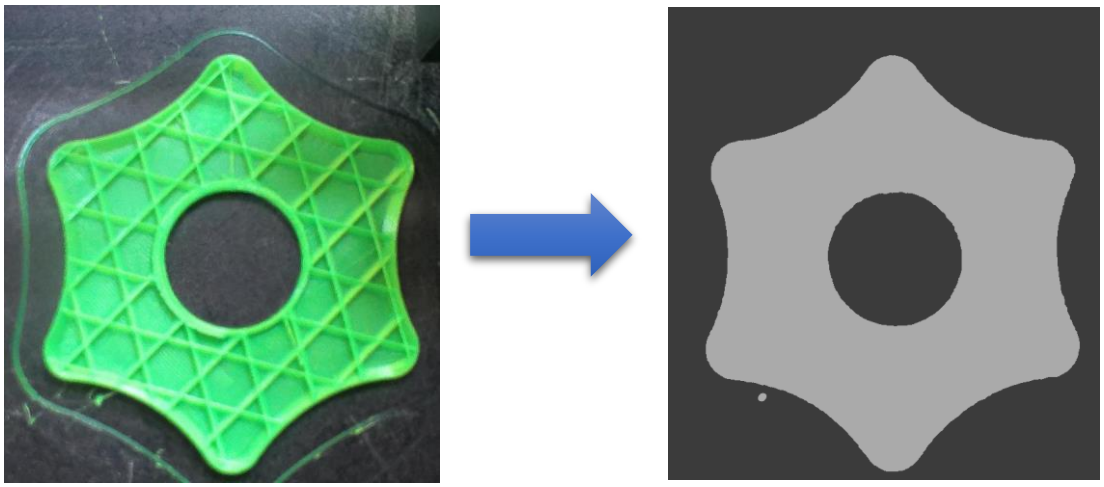


Image Segmentation using K Means Clustering

In computer vision, image segmentation is the process of partitioning an image into multiple segments. The goal of segmenting an image is to change the representation of a cropped image from previous step into something that is more meaningful and easier to analyze. It is usually used for locating objects and creating boundaries.

Therefore, by segmenting the image, we can make use of only the important segments for processing via Color clustering. The figure below shows the input and output of this step. Note that we use the detected bounding box from the previous stage to cropped the image.



The “pixels per metric” ratio

In order to determine the size of an object in an image, we first need to perform a “calibration” (not to be confused with intrinsic/extrinsic calibration) using a reference object. Our reference object should have an important property that is we should know the dimensions of this object (in terms of width or height or diameter) in a measurable unit (such as millimeters, inches, etc.)

In task example, we used the coin as our reference object and throughout all examples, ensure the distance between the camera and the screen is fixed.

By guaranteeing the fixed distance and fixed camera viewing angle we can use the coin on the right-hand side to define our pixels_per_metric, which we define as:

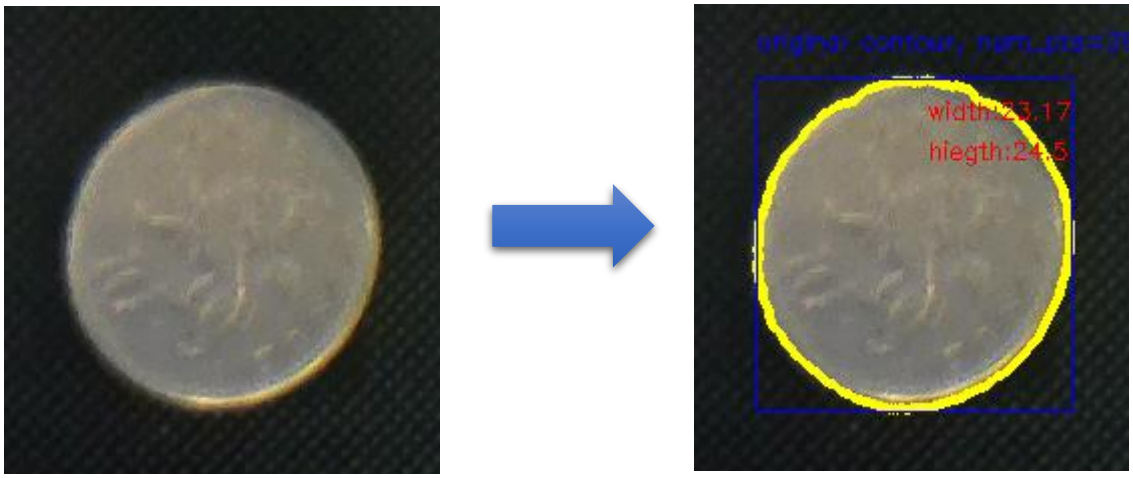
pixels_per_metric = object_diameter / know_diameter(in terms of pixel)

A coin has a known diameter of 24.5mm. Now, suppose that our object diameter (measured in pixels) is computed to be 166 pixels wide (based on its associated bounding box generated by contour).

The pixels_per_metric is therefore:

pixels_per_metric = 166px / 24.5mm = 6.77px

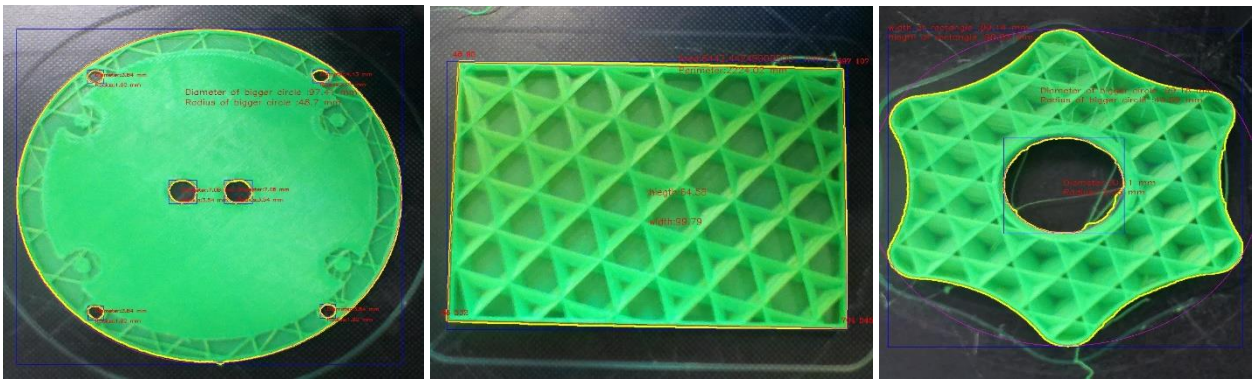
Thus, implying there are approximately 166 pixels per every 24.5mm in our image. Using this ratio, we can compute the size of objects in an image.



Results

In this task a method for object measurements present in an image using a camera was explained. The identification of an object and background removal is based on finetuning object detection model using TensorFlow.

The identified objects are distinguished by its predicted class and then based on the object classes, the specific processing will be applied for segmentation, noise removal and measurement. The measurement obtained from these images are taken as input data for the equation to calculate the real-world size. Below are some examples of outputted result.



Improving object size measurement accuracy with a proper camera calibration

We must calibrate our system before measuring the size of a object in an image. Up to now We've used simple pixels per metric in this post. However, when the extrinsic and intrinsic parameters are calculated it is possible to obtain more accurate accuracy by performing a proper calibration of the camera, there are two main distortion that we can calibrate images to have better measurement, i.e., Radial distortion and tangential distortion. You can use the link below to access the code for camera calibration in OpenCV.

https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

another point in which you have to make sure about is the place of camera that should make sure is fixed.