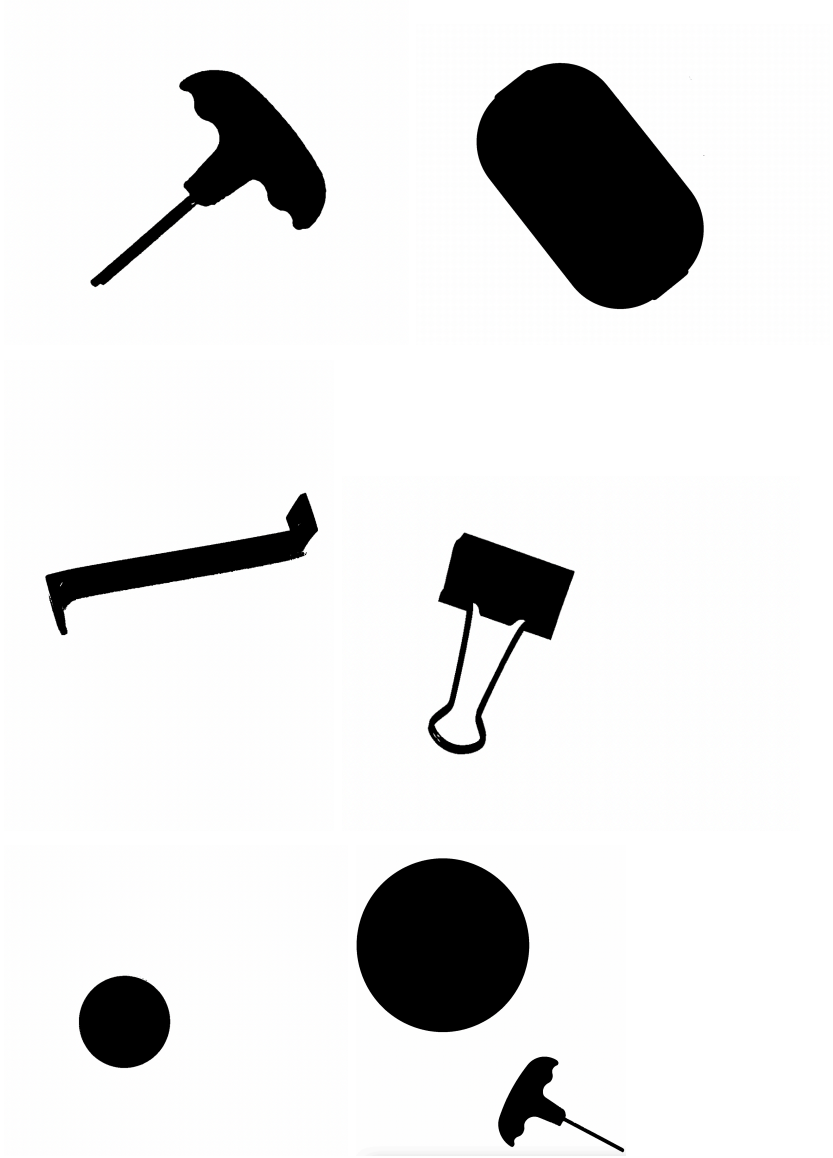
Real Time 2-d Object Recognition

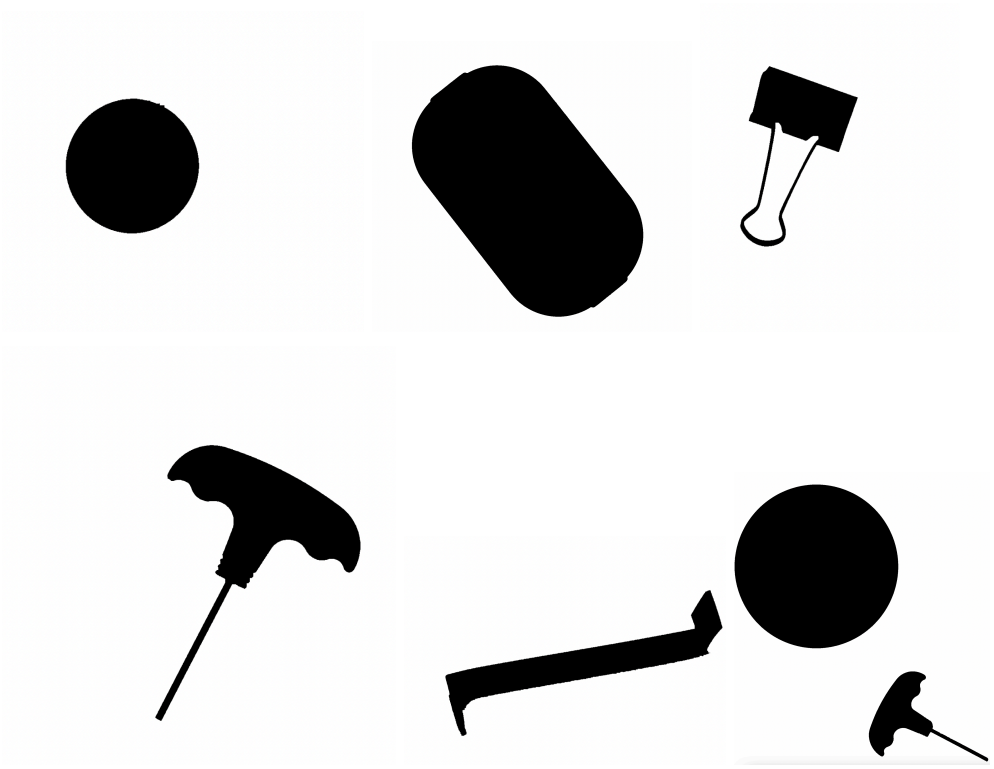
In this Project, I have implemented real time 2-d object recognition of 5 objects by processing the image accordingly to identify appropriate objects in the Image. All the images are read and the csv files are read using the methods provided by the professor.

1. **Threshold the input Value(Written from scratch):**
   To threshold the input Image, First I convert the given image into Grayscale. Now, I create a histogram of the greyscale with 255 bins to find the 2 peaks to separate foreground and background. Now, I chose the middle point in the histogram between two peaks and I consider only the pixel values above that threshold as foreground region pixels. The last image consists of two regions.

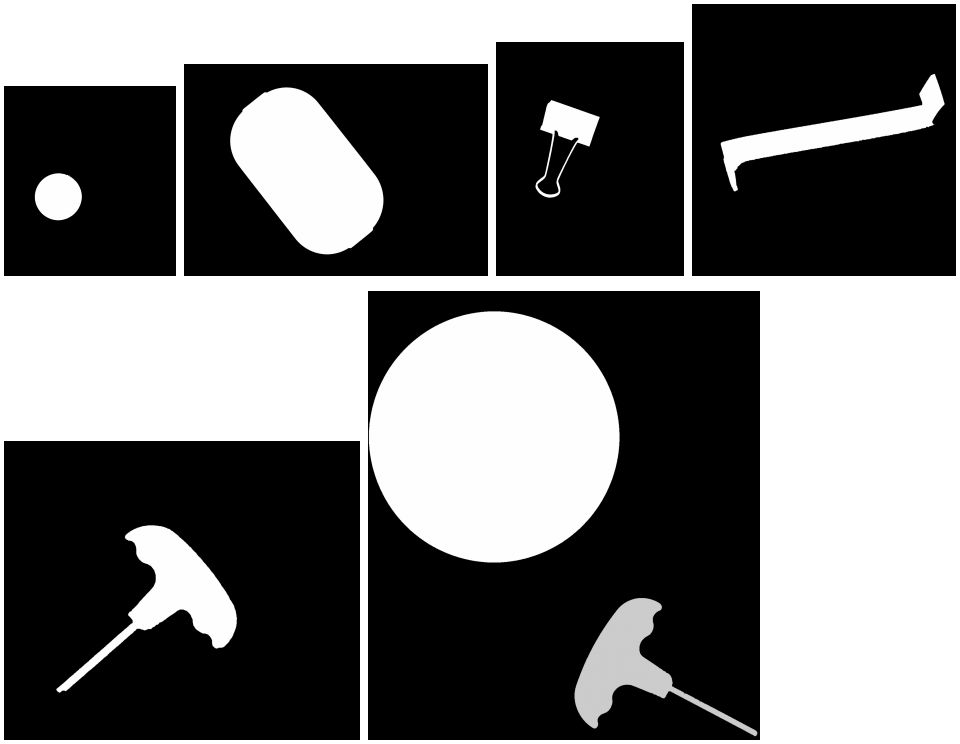


2. **Clean up the binary Image(Used the library to implement Cleaning):**
   The threshold image is then cleaned up using a morphological process of the threshold image.I'm using computer vision morphologyEx methods to apply shrinking and growing

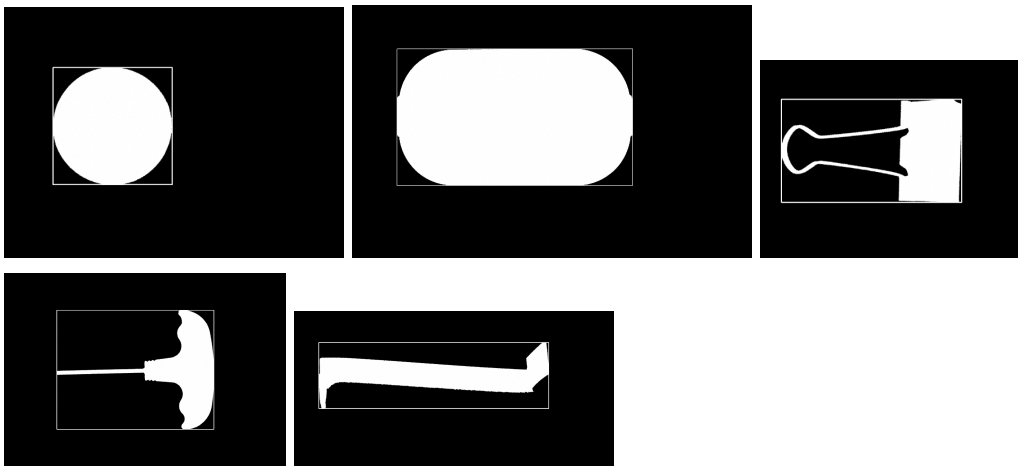
matrices. I'm shrinking the image first using a 4*4 matrix and then Im growing using a 8*8 matrix.



3. **Segment the images into regions(Written from scratch):**
   After cleaning the image, I'm creating the region map for each different region in the image. I'm doing this by identifying the undefined region pixels and using a stack to find the connected components to that pixel and assigning a region to that. I'm also calculating the stats of the region like the position, total pixels, centroid, etc.



4. **Compute features of each major region:**
   For each region, the features like width, height , total pixels, pixel positions, centroid etc are all calculated while segmenting the image itself. I'm also using huMoments as features for the regions. I'm using the computer vision huMoments method to compute the huMoments. HuMoments features help in making the image invariant to translation, rotation and scale. First central moments are calculated. It helps in getting information of pixel distribution in the region. I have calculated this manually and also verified by using the moments method of the cv library. This will also help us get the centroid and the orientation of the image. I use the orientation angle to rotate the image to align with the horizontal axis and then apply the bounding box. I'm rotating the matrix using the cv::getRotationMatrix2D method. The features considered are two features of percentageFilled and width/height ratio and the 7 huMoments values which are invariant to scaling, translation and rotation. Here, I have rotated the object along the x axis and hence, haven't given the axis of least central moment as after rotation, that will be along x axis.



featureFile

cap	0.1592	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.7801	0.5016
Box	0.1834	0.0077	0.0000	0.0000	0.0000	0.0000	0.0000	0.6363	0.5195
clip	0.3588	0.0190	0.0693	0.0449	0.0025	0.0062	0.0000	0.3012	0.6030
Wrench	0.3382	0.0005	0.0587	0.0202	0.0007	0.0004	0.0000	0.2545	0.5726
divider	0.8419	0.6582	0.0020	0.0002	0.0000	-0.0002	0.0000	0.2234	0.3418
Wrench	0.3293	0.0027	0.0524	0.0170	0.0005	0.0009	0.0000	0.2672	0.4800

5. **Collect Training Data:**
   For training, from command line, if the argument passed is training, it iterates over all the images present in a specific directory and then applies thresholding, cleaning and segmentation to get regions and computes features as described above and displays the rotated bounded box and asks the user to enter the label. When the user enters the label, it stores the feature vector computed into a csv file. The reading of images from directory and storing into csv code is taken from the code provided by the professor.

6. **Classification of New Images:**
   The image regions are extracted from the video/images and then features are computed. Now, using scaled euclidean distance, we compute the distance of each region from all the objects trained. Now using the nearest neighbor classifier, we classify the region as the one to which the distance is closest.

7. **Confusion Matrix:**

Actual

	Box	Wrench	Clip	Divider	Cap
Box	8		1		
Wrench		12	1		
Clip			4		
Divider				7	
Cap					4

The row wise objects are the predicted values.

We can see from the above confusion matrix that the model does pretty well in recognizing the objects. Only in a few cases it mis recognized as a wrench because of their similar shape and when the clips silver part is not recognized, then it identifies the clip as a box as both are similar if the silver part of the clip is removed. In one scenario,

8. **Video Link:**
   Video link:
   https://drive.google.com/file/d/1kdHp7bsC3SUJn1-rWE45TZH8jqKYzlzO/view?usp=drive_link

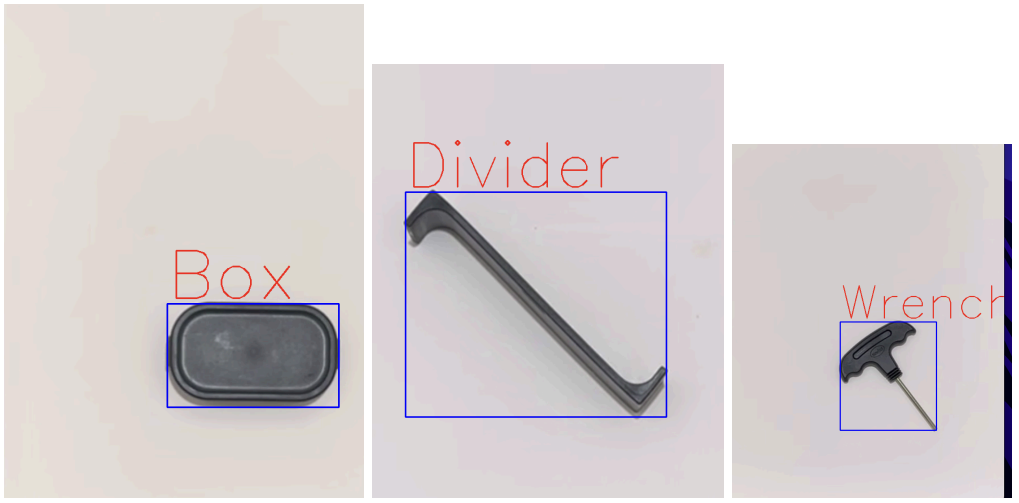
9. **Second Classification Implementation:** I have used the DNN model given by the professor to get the embeddings as feature vectors for each object. The reason is that deep neural networks can perform better in analyzing the patterns in the object. I iterate over all the training images and apply threshold, clean and segment the regions of the image and then pass each region of the image into the model to get the embedding values. I'm using that embedding values as a feature vector and ask the user for labeling the object and then store this information in a csv file. This is then used while classifying the unknown object, which is also segmented to regions and then passed to model to get the embedding values. Then, euclidean distance is used to compute the distance of it from all the trained images and classified as the one closest to the trained images.
   The Classification using DNN Embeddings approach performs badly compared to the first implementation. It incorrectly classifies clips and wrenches most of the time.

Actual

	Box	Wrench	Clip	Divider	Cap
Box	7		1		1
Wrench		9	2		
Clip		3	3		
Divider		1		7	
Cap	1				3

The row wise objects are the predicted values.
We can see from the above confusion matrix that it doesn't identify the objects as accurately as our previous approach. However, it identifies the objects accurately the majority of the time.

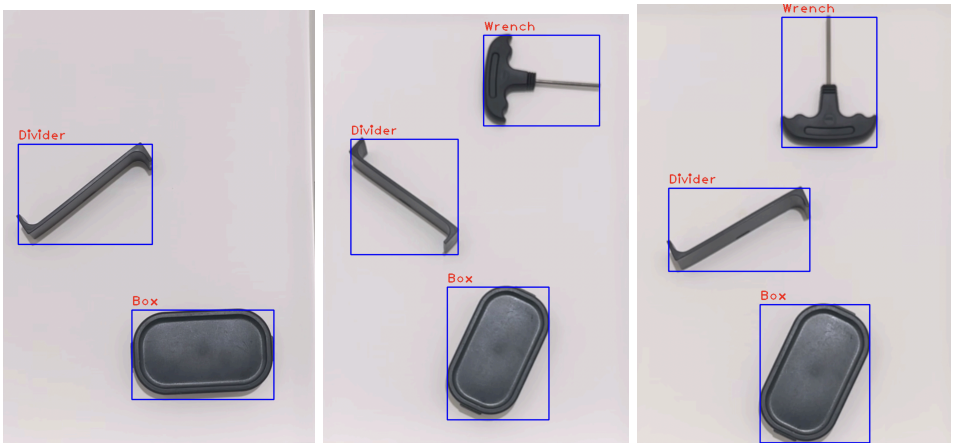
Correct Classification Images:



Incorrect Classification:

Extensions:

- Apart from Thresholding, I have implemented Segmentation and also calculation of statistical features and moments from scratch.
- I have integrated multiple object detection in a single image of the video stream. Below are the results(I have also added a video demo of it as well):



- I have written another classification algorithm, which is the K nearest neighbor algorithm. In this algorithm, I'm identifying the distinct objects in the training set and in the set of each object, I'm finding the top K objects closest to the input object to be classified. Now, I sum the distances and repeat this for each distinct object category present in the training set. Now, I find the shortest distance among them and classify the result as that object.
  This also performs almost similarly to the first approach, which is essentially K=1.
  The below confusion matrix is when K=3.

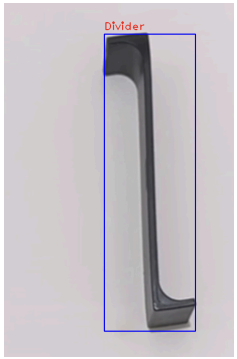
Actual

	Box	Wrench	Clip	Divider	Cap
Box	8		1		1
Wrench		12	1		
Clip			4		
Divider				7	
Cap					4

The row wise objects are the predicted values.
We can see from the above confusion matrix that it also does pretty well and is almost similar to our first approach. When k is one, it's exactly the same as our first approach. But when K is greater than 1, it also considers other nearest neighbors distances as well. For K=3 also, the results are almost the same as it also predicts almost all objects accurately.


Box


Divider


Divider