CS 5330 Project 3

Kevin Heleodoro

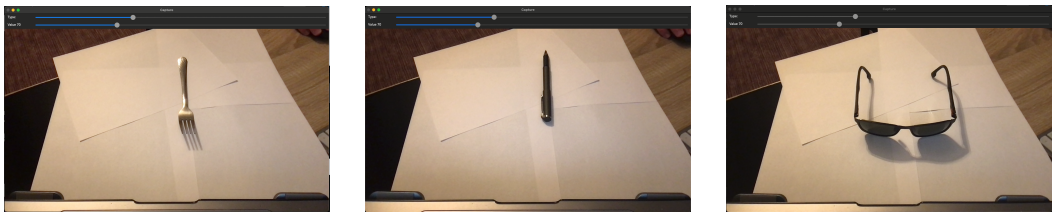
Northeastern University

February 25, 2024

Abstract

The focus of this assignment was to create a program that could detect and classify objects in real time using various image processing techniques. These include thresholding, morphological filtering, region segmentation, and feature comparison. The goal was to create a training dataset of a few objects in various positions to serve as a baseline for the objects being presented in the input video.

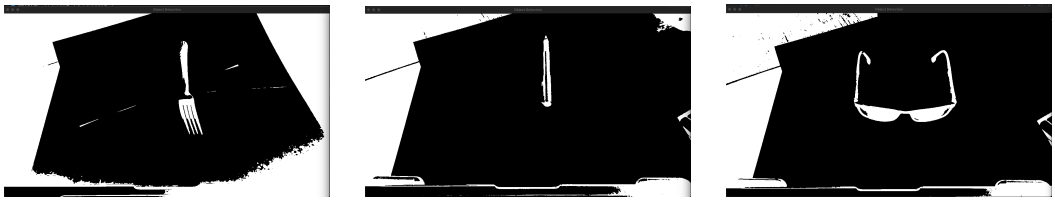
1 Thresholding Input Video



(a) Fork (b) Pen (c) Sunglasses

Figure 1: Original objects

The three starting objects I chose were a fork, pen, and sunglasses. I figured these should be easily recognizable and provide a distinct set of features for the program to work with.

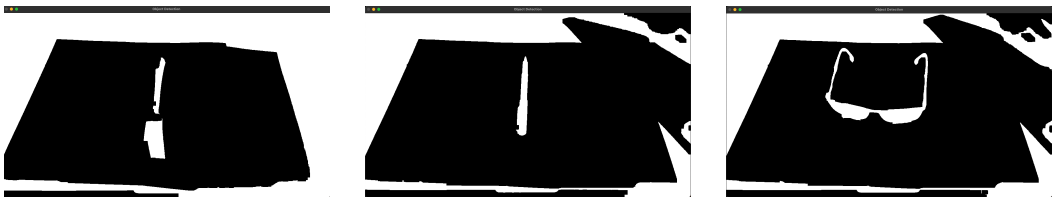


(a) Fork - Thresholded (b) Pen - Thresholded (c) Sunglasses - Thresholded

Figure 2: Thresholded objects

One of the issues I noticed right away were that the fork was a bit of challenge to threshold and I had to use about double the thresh value in order to get it to the what we see in Figure 2.a. The reflections and shine along the metallic surface made it difficult to distinguish between background and object pixels. [9] [8] [2]

2 Binary Image

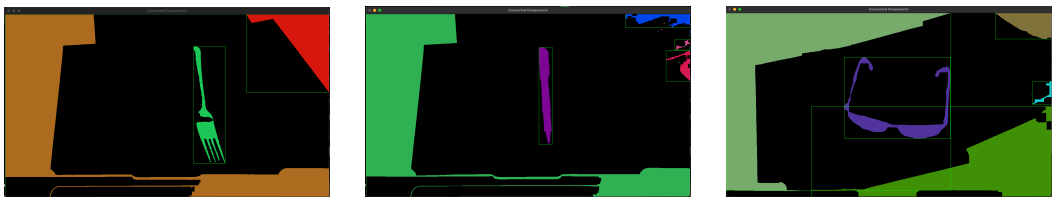


(a) Fork (b) Pen (c) Sunglasses

Figure 3: Morphological Filter on Objects

When I studied the thresholded images I noticed that there were some holes within the body of the objects. This led me to implement a closing morphological filter using OpenCV. The filter first dilates the image which increase the surface area of the object, and then performs an erosion which cleans up the borders to try and get the object back to its original size. [10] [1] [4] [6]

3 Region Segmentation

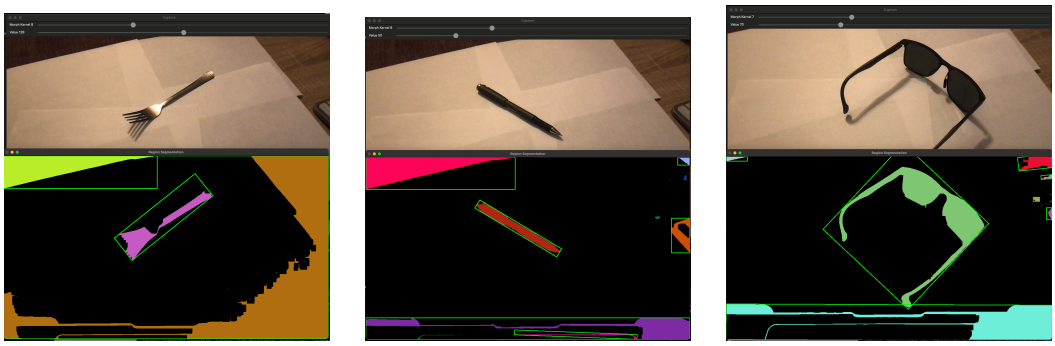


(a) Fork (b) Pen (c) Sunglasses

Figure 4: Region Segmentation on Objects

For the region segmentation I made use of OpenCV's *connectedComponentsWithStats*() function. The output from this function made it easy to distinguish between the region maps and ids, allowing me to create a border around the unique objects found within the image, while also assigning a random color to each one. [3] [7]

4 Computing Features



(a) Fork (b) Pen (c) Sunglasses

Figure 5: Region Segmentation on Objects

I utilized OpenCV's *Moments* functionality to access the features of the regions. The function calculates moments up to the third order. For my implementation, I used the area, centroid, aspect ratio,

and the contour area of the region. [11] [5]

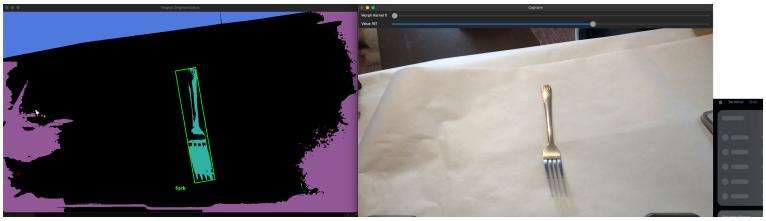
5 Collecting Training Data

For this task I used the suggestion of allowing a user to input the name of the object being displayed. The training system then calculates all the features from the previous task and stores them in a .csv file. I manually repositioned the object over many iterations and ran the training system each time. Looking back, I would have liked to have spent more time developing an efficient way to capture the feature vectors.

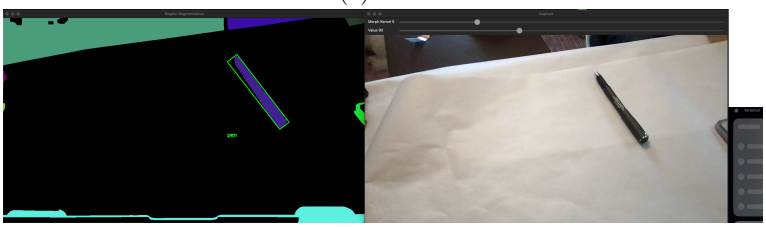
```
sunglasses,94051,452.946,48.9655,7.71084,92622
sunglasses,94261,452.57,49.0684,7.71084,92835
sunglasses,93819,450.903,49.0046,7.71084,92391.5
sunglasses,94183,451.853,49.0765,7.71084,92756.5
sunglasses,93782,450.465,49.0287,7.71084,92357
staple,100641,493.679,49.7341,7.71084,99204.5
staple,97242,475.291,49.2151,7.66467,95810
staple,97626,475.494,49.3632,7.66467,96192
staple,97467,475.568,49.3214,7.71084,96035
staple,97511,476.142,49.3073,7.71084,96080
staple,98269,477.166,49.526,7.66467,96837
```

Figure 6: Training Data

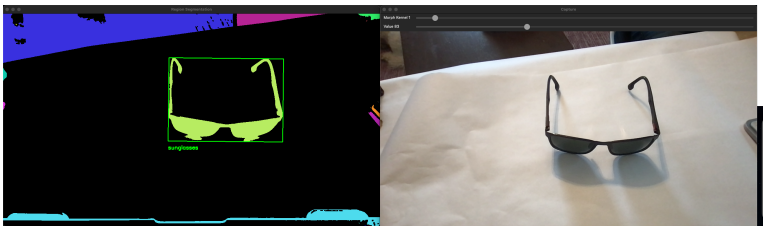
6 Classify Images



(a) Fork



(b) Pen



(c) Sunglasses

Figure 7: Classified Images

As expected, the results were very shaky in the beginning when the training data set consisted of only a handful of feature vectors for each object. As the dataset grew to over 30-40 entries per object the

results became more dependable. The fork and pen were initially the hardest to differentiate between all of the objects.

7 Evaluating Performance

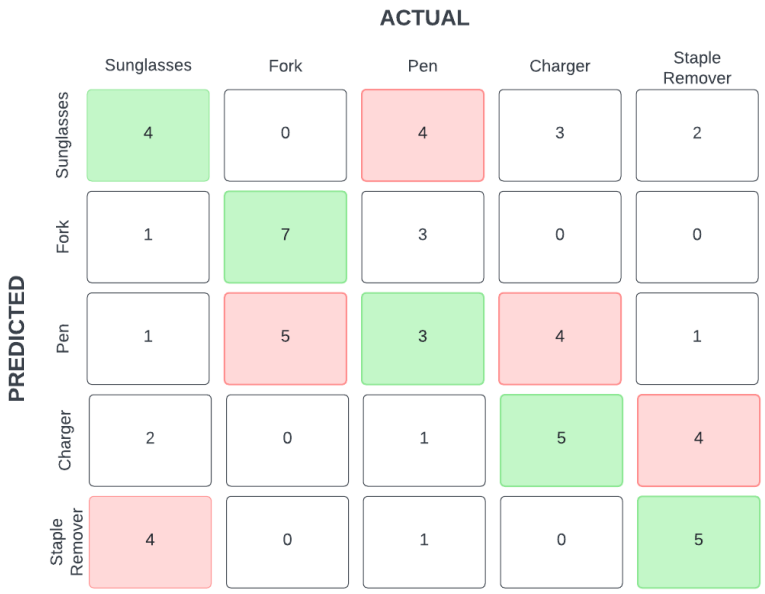


Figure 8: Confusion Matrix for the 5 object dataset

After adding a charger and staple remover to the rotation there were quite a few false classifications. In Figure 8 the green boxes represent the number of correct predictions for each object. The red boxes represent the objects most commonly predicted incorrectly for the object. What surprised me the most was that the fork was the only object that was predicted correctly more often than incorrectly.

8 Demo

You can access a copy of the demo video here.

9 Second Classification

For the second method of classification I used the K-Nearest Neighbors approach. I slightly modified my original *compareFeatures* method to sort and select the top K neighbors from the distance list. At first I had it select the object with the shortest distance, but that gave me similar results to the original method of classification. I then proceeded to select the neighbor that had the highest frequency in the top K list.

The results in the matrix represent the initial responses after implementing the new K-Nearest classification system. After conducting many more iterations, I was able to get the number of mis-classifications slightly lower.

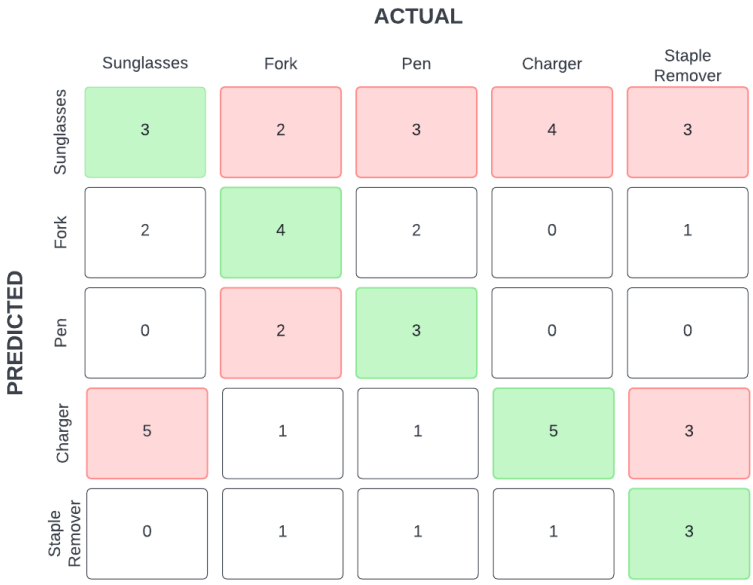


Figure 9: Confusion Matrix after K-Nearest

10 Reflection

This assignment has been my favorite so far. I would like to continue working on this after the semester is over and polishing up a lot of the logic. It would also be very interesting to implement the DNN system into this project and create a GUI for it. I learned how to segment regions in an image, extract feature vectors from them, save the vectors in a dataset and then use that dataset to compare to new images.

References

[1] Nattadet C. *Morphological Filters*. Accessed: 2024-02-19. 2018. URL: https://medium.com/nattadet-c/morphological-filters-d99860d39b85.

[2] OpenCV Developers. *Basic Thresholding Operations*. https://docs.opencv.org/4.x/db/d8e/tutorial_threshold.html. Accessed: 2024-02-18. 2021.

[3] OpenCV Developers. *Distance Transform*. https://docs.opencv.org/4.x/d2/dbd/tutorial_distance_transform.html. Accessed: 2024-02-20. 2021.

[4] OpenCV Developers. *Erosion and Dilation*. https://docs.opencv.org/4.x/db/df6/tutorial_erosion_dilatation.html. Accessed: 2024-02-19. 2021.

[5] OpenCV Developers. *Image Moments*. https://docs.opencv.org/4.x/d0/d49/tutorial_moments.html. Accessed: 2024-02-24. 2021.

[6] OpenCV Developers. *Morphological Transformations: Opening and Closing*. https://docs.opencv.org/4.x/d3/dbe/tutorial_opening_closing_hats.html. Accessed: 2024-02-19. 2021.

[7] OpenCV Developers. *Structural Analysis and Shape Descriptors*. https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e765f. Accessed: 2024-02-20. 2021.

[8] OpenCV Developers. *Thresholding Operations using inRange*. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html. Accessed: 2024-02-18. 2021.

[9] OpenCV Developers. *Tutorial on Threshold inRange*. https://docs.opencv.org/4.x/da/d97/tutorial_threshold_inRange.html. Accessed: 2024-02-18. 2021.

[10] Pitchaya Thipkham. *Image Processing Class 6 — Morphological Filter*. Accessed: 2024-02-19. 2018. URL: https://towardsdatascience.com/image-processing-class-egbe443-6-morphological-filter-e952c1ec886e.

[11] Wikipedia contributors. *Image moment*. https://en.wikipedia.org/wiki/Image_moment. Accessed: 2024-02-24. 2021.