Samuel Lee and Anjith Prakash Chathan Kandy

CS5330

Project 3

2/24/2024

Project Overview

Through this project, we were able to create a Real-Time Object Recognition program. To demonstrate object recognition in our program, we first needed to find regions in the image in an efficient way. First, we passed the image through a custom threshold to create a binary image. With this binary image, the program cleans up the image via custom morphological filters to reduce noise by shrinking (eroding) first then growing (dilating) the pixels. With this cleaned up image, we used connected components analysis to separate the image into regions. Each region can be identified via a different color. With the region map created, we were able to overlay this with the original image to identify objects. The objects identified are matched based on feature vectors stored in a local CSV file and labeled accordingly. The program has a "training mode" where the user can train the program on objects by entering in a label name and storing this name with the features in the CSV file.

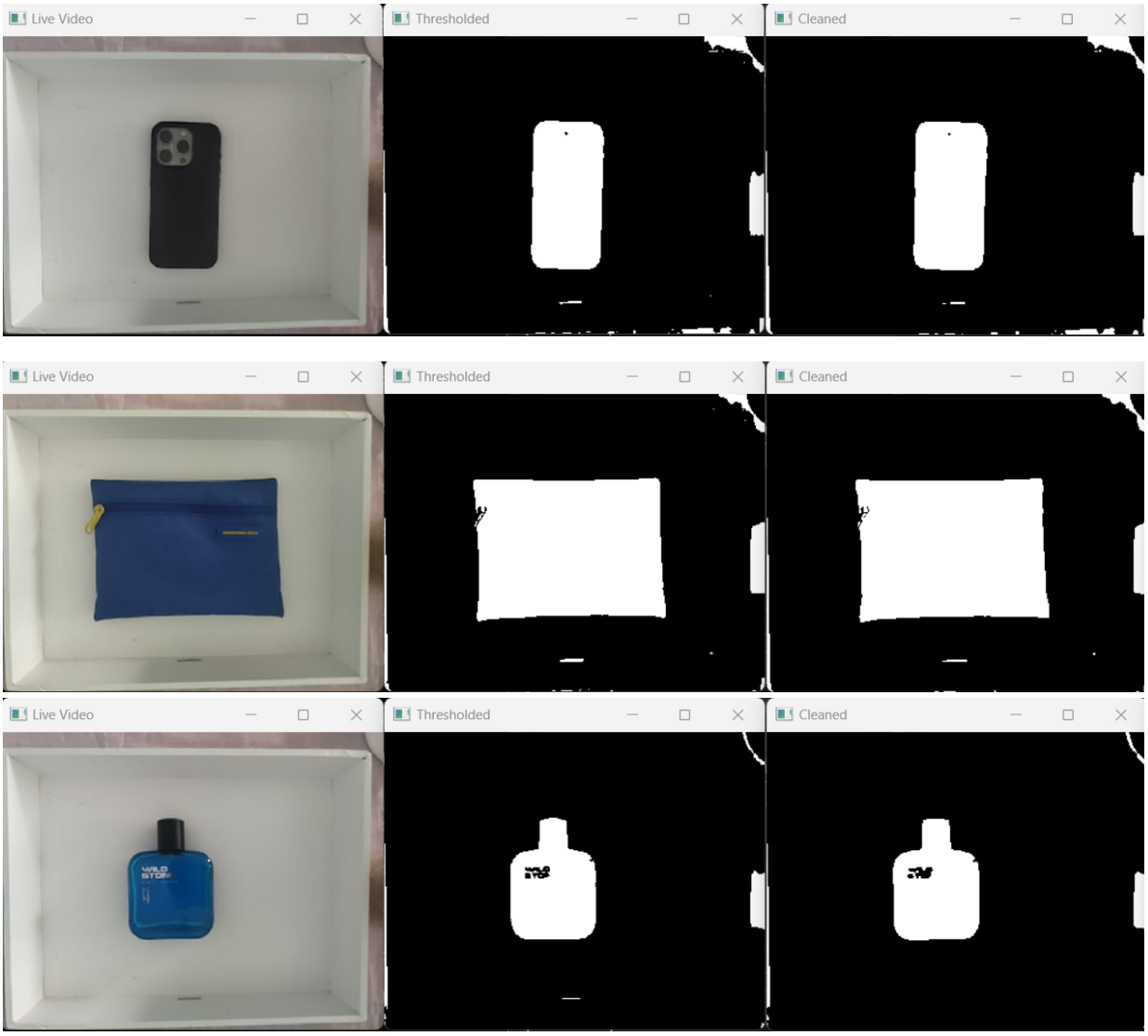
Overall, this program allows users to create a trained program to identify objects in 2-D with feature vectors either imported externally or created internally via the training mode. For demonstration, we picked the following 10 objects:

- Phone
- Mug
- Watch
- Perfume
- Box
- Plug
- Band
- Cup
- Pouch
- Vicks

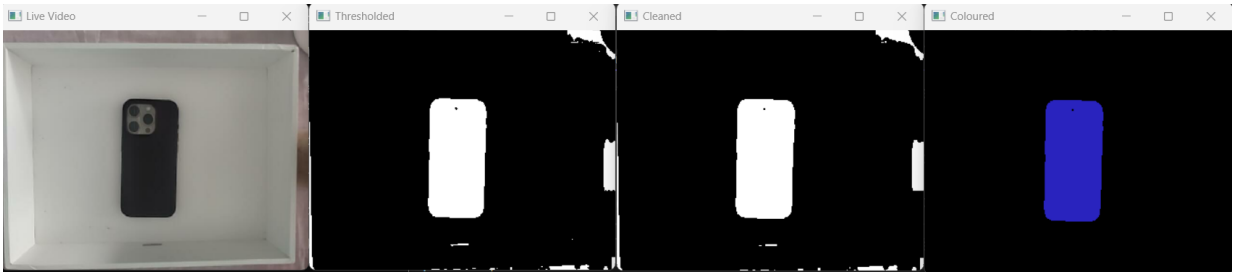
Required Images

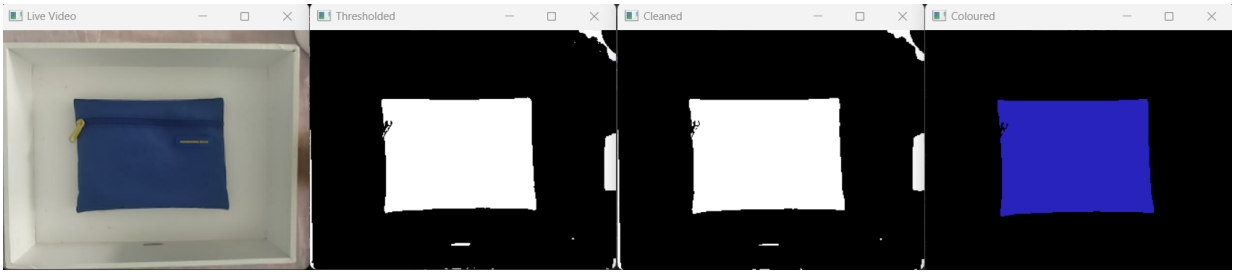
Task #1: Thresholded Images

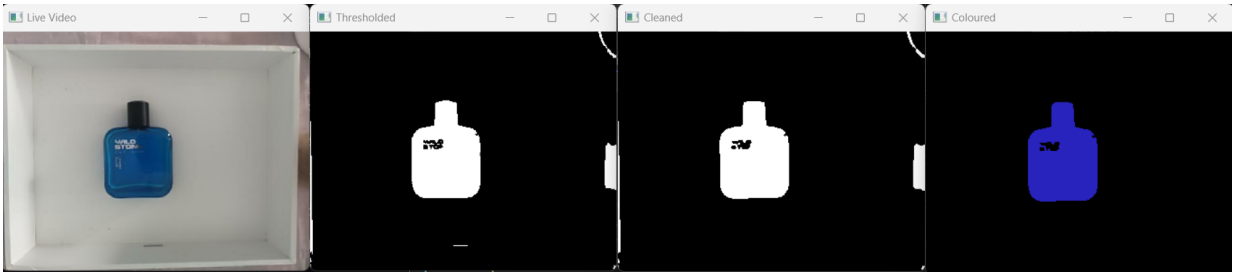
Task #2: Cleaned Up Images

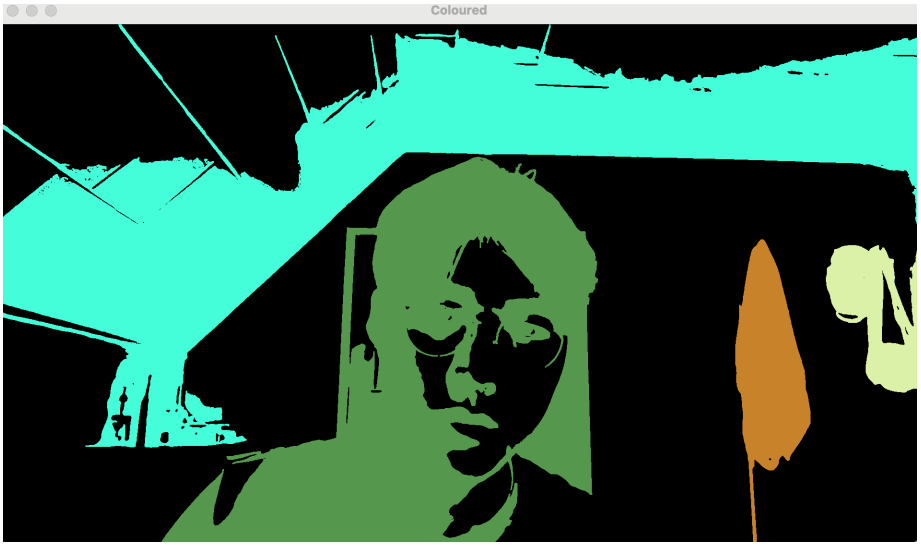


Task #3: Region Maps



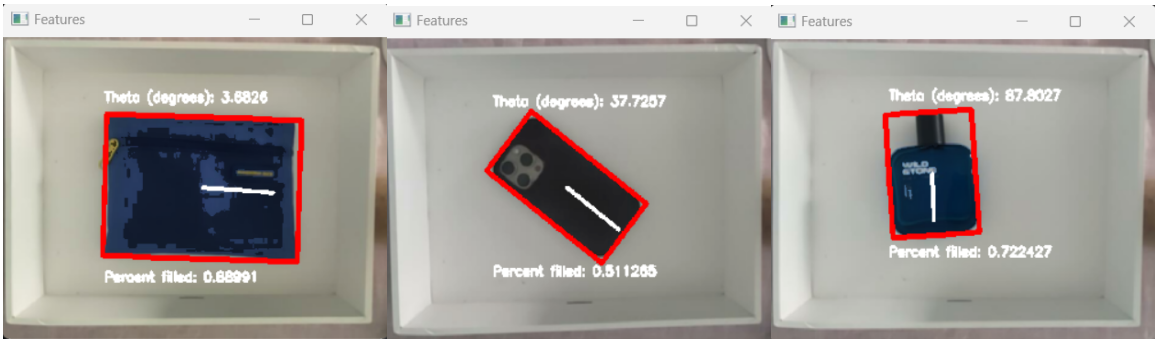






- This is to just show that if there are multiple objects, they will be labeled as different colors

Task #4: Axis of Least Central Moment + Oriented Bounding Box




```
Pouch 0.901978 0.735849
Pouch 0.893046 1.39914
Pouch 0.913272 0.712074
```

```
Perfume 0.745492 1.34247
Perfume 0.773962 0.728205
Perfume 0.767483 0.729592
```

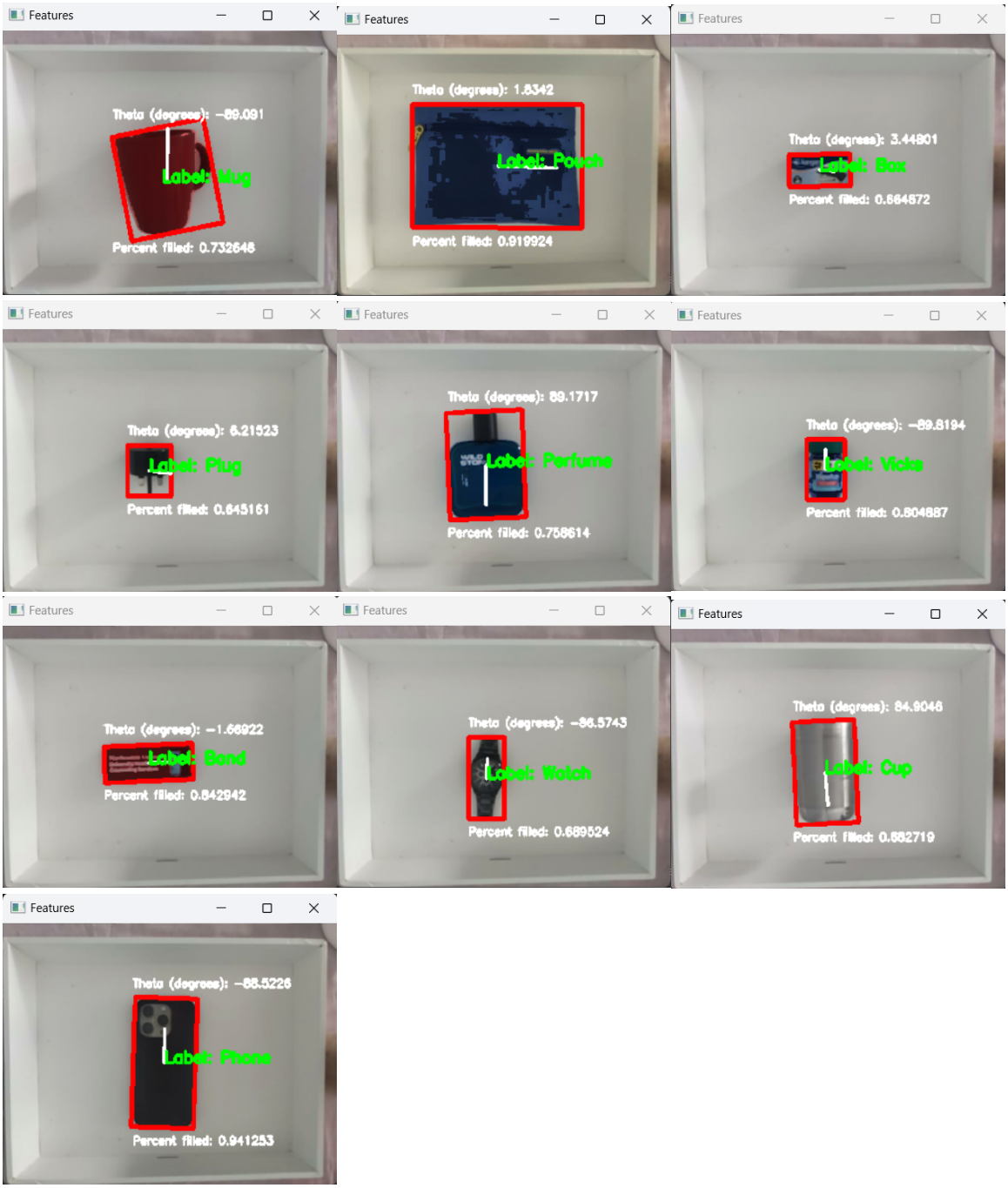
```
Phone 0.948726 0.504167
Phone 0.958018 1.96694
Phone 0.513962 0.949791
```

Task #5: Training System

At the start of the program, all the feature vectors and labels in the csv training file are read into a data structure. When the user types “T”, the training mode is activated. Typing “T” again toggles it off. In training mode, only the object closest to the center is identified. Once the desired object is identified by the bounding box, the user can type “N” to capture the feature vector of the most central object/region. Once the user exits the program by typing “q”, the feature vectors are stored in the CSV file.

This is how we set up for training ☺

Task #6: Categorized Image



Task #7: Performance of System

		Predicted Label									
		Phone	Mug	Watch	Perfume	Box	Plug	Band	Cup	Pouch	Vicks
True Label	Phone	10	0	0	0	0	0	0	0	0	0
	Mug	0	6	0	1	0	2	0	0	0	0
	Watch	0	0	3	0	4	2	3	0	0	0
	Perfume	0	0	0	6	0	0	0	0	2	1
	Box	0	0	0	3	5	0	1	0	0	3
	Plug	0	0	0	0	0	6	0	0	0	0
	Band	0	0	2	0	1	0	4	0	0	0
	Cup	0	0	0	0	0	0	0	7	0	0
	Pouch	0	0	0	0	0	0	0	0	9	0
	Vicks	0	0	0	3	0	1	0	0	0	5

Task #8: Link to Demo

- Link to Demo: Project 3 Demo

Task #9: Second Classification Method

- We implemented KNN with matching K > 1. The function we implemented returns the closest K labels as a vector of string values. Using the scaled Euclidean distance, we find the closest K labels and sort it by smallest to greatest value, with smallest corresponding to the closest label.

```
Time taken for closestLabelScaledEuclidean: 7 microseconds
Closest label found with nearest neighbor (euclidean distance) - Perfume
Time taken for closestLabelCosineSimilarity: 5e-06 microseconds
Closest label found with nearest neighbor (cosine similarity) - Perfume
Time taken for kNearestLabels: 15 microseconds
Closest label found with KNN - Perfume
```

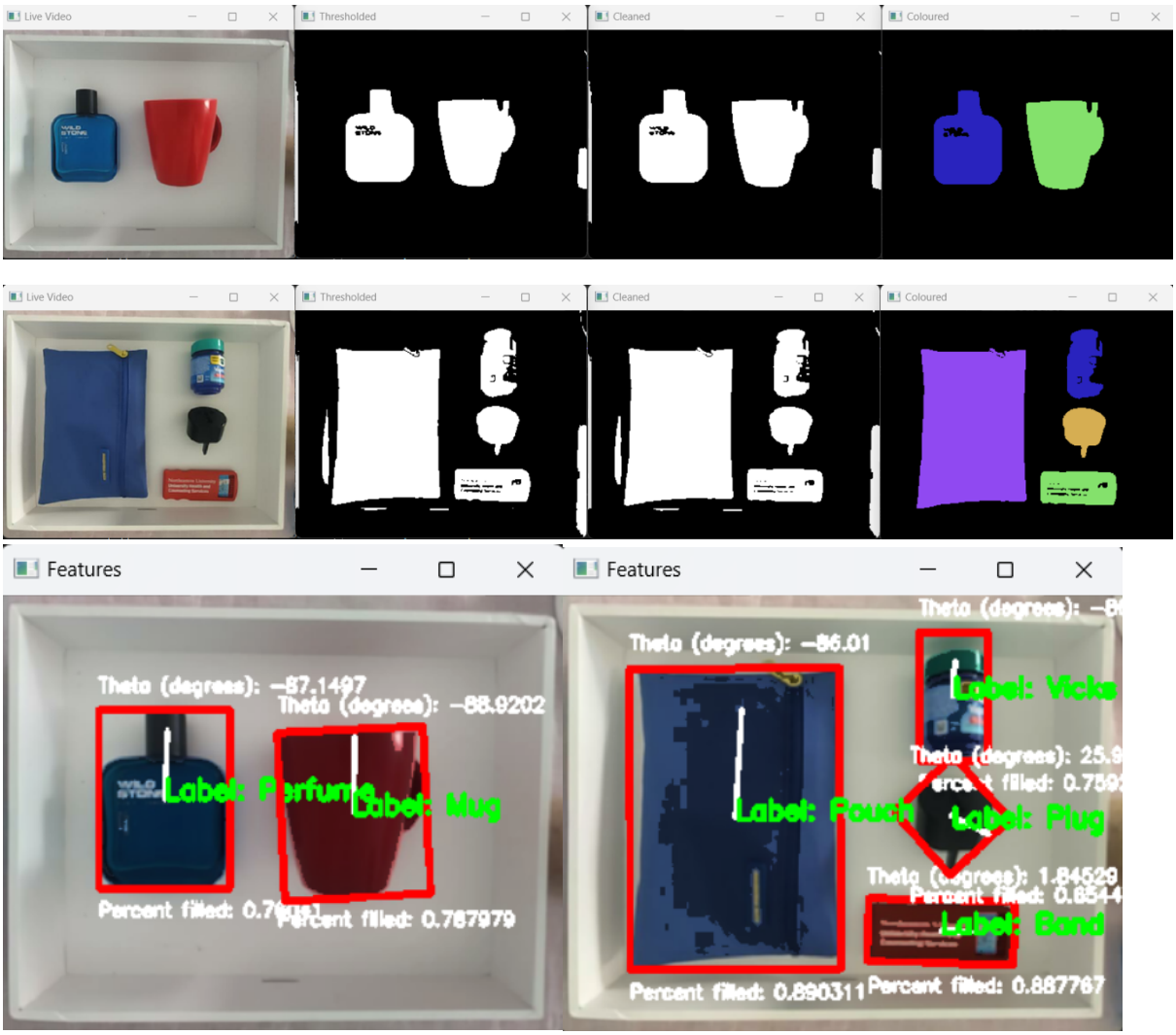
- Scaled Euclidean distance is more efficient for our program as of now because our database isn't too large. The larger the database, KNN will be more efficient for finding K labels. Scaled Euclidean finds just the closest 1 label. If we want multiple labels and get the mode of the labels to classify a label, then KNN is more efficient overall. For a small database set, the difference between 15 microseconds and 7 microseconds is very minimal.

Extensions

We implemented four extensions: multiple object recognition, unknown object recognition, cosine distance metric, and 10 objects trained in our model.

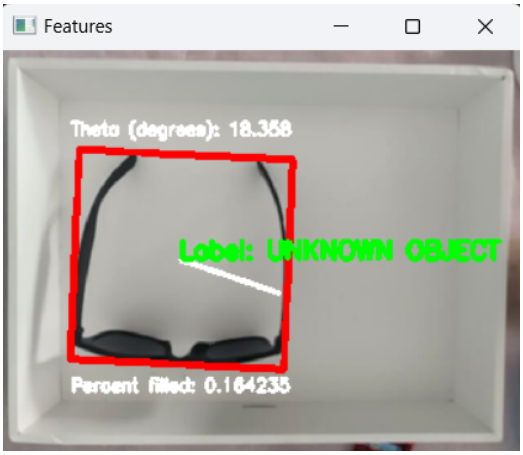
Multiple Object Recognition

Our program can recognize multiple objects in the video frame at a time. It chooses the top N objects (needs to be changed in the code).



Unknown Object Recognition

If an object is not recognized based on a certain threshold, then the object says UNKNOWN OBJECT.



Cosine Distance Metric

In addition to KNN and scaled Euclidean distance, we implemented Cosine Distance. It's not directly implemented in main.cpp, but we can use this if we wanted to.

```
/**
 * Distance formula between two vectors via cosine similarity
 * curr - feature vector #1
 * other - feature vector #2
 */
double cosineSimilarity(const FeatureData &curr, const FeatureData &other) {
    double dotProduct = (curr.f1 * other.f1) + (curr.f2 * other.f2);

    double magnitudeCurr = sqrt(pow(curr.f1, 2) + pow(curr.f2, 2));
    double magnitudeOther = sqrt(pow(other.f1, 2) + pow(other.f2, 2));

    if (magnitudeCurr == 0 || magnitudeOther == 0) {
        return 0.0;
    }

    return dotProduct / (magnitudeCurr * magnitudeOther);
}
```

10 Objects Trained

Instead of the 5 objects recommended, we doubled the amount of objects trained to 10. This allowed us to stress-test our program more, and we generated more data. The first value is the label name. The second value is the first feature (percentage filled). The third value is the second feature (height/width ratio).

Project 3 > build > trainingdata.csv

```
Mug 0.76883 1.08523
Mug 0.768063 1.03889
Perfume 0.745492 1.34247
Perfume 0.773962 0.728205
Perfume 0.767483 0.729592
Perfume 0.766115 1.35172
Mug 0.745192 1.06286
Mug 0.736979 0.93401
Mug 0.796895 0.910995
Plug 0.629858 1.10976
Plug 0.629539 1.09756
Plug 0.623023 1.10976
Plug 0.665673 0.931034
Plug 0.624228 0.84375
Watch 0.69878 2.16176
Watch 0.697933 2.17647
Watch 0.695124 0.446667
Watch 0.687924 0.435065
Box 0.554446 0.491379
Box 0.552747 0.508621
Box 0.542735 1.8871
Box 0.544165 2.01754
Pouch 0.901978 0.735849
Pouch 0.893046 1.39914
Pouch 0.913272 0.712074
Cup 0.389945 1.36496
Vicks 0.791166 1.54167
Vicks 0.791166 1.54167
Vicks 0.790939 0.784
Band 0.846386 0.39759
Band 0.834713 2.45588
Phone 0.968207 1.98333
Phone 0.946384 0.504167
Phone 0.948726 0.504167
Phone 0.958018 1.96694
Phone 0.513962 0.949791
Box 0.682621 0.512821
Cup 0.687698 1.575
Pouch 0.79386 0.739003
```

Reflections

Sam

Learning about eigenvectors and region algorithms seemed very theoretical. Through this project, however, I was able to apply the concepts learned in class to practice, which allowed me to understand the concepts better. I now know that it's important to put a threshold on an image to create a binary image, on which we can apply morphological filters to erode/dilate the image so that it can be a smoother image. Then, by using

connected components analysis, we can identify regions of objects, which can be further analyzed via features that are transition, scale, and rotation invariant.

This project was an amazing learning experience for me, and it allowed me to see the endless applications that computer vision has in this specific field of object recognition. Even with this project alone, I see so many ways I can improve it.

Anjith

This project actually helped me understand a lot of pre-processing and computer vision-related tasks. I realized that whenever you have a CV problem, you should not throw it directly at a Neural Network; instead, work on it like this. I had a really fun time working with Sam, as well as setting up the lighting and making sure the system is all set for training. As we built thresholding and cleanup from scratch, this opened us up to a lot of understanding of what actually happens behind the libraries. Towards the deadline, I got many ideas that could actually improve the results and user interaction. I will be trying out those ideas in future assignments.

Acknowledgement

- Referencing Professor Maxwell's code demonstrated in class – DNN Example, Grassfire Transform, and Eigenspace Example
- Referencing past project code – to re-use some code
- Stack Overflow – to check all the errors caused while building the code.
- Cplusplus.com – for c++ documentation.
- OpenCV Documentation – to find the syntax for implementation.
- TAs – to fix some errors in code