**Kristine Umeh & Elizabeth Witten**

**Project 3**

**Real time 2D object Recognition**

2D real-time object recognition is a computer vision task that involves identifying objects in real-time images or video frames. The objective is to detect the presence and location of objects in the images and classify them into different categories. For this project we explored pre-processing which included thresholding a greyscale image, cleanup, segmentation, feature extraction and matching / classification.

Erosion and dilation functions are implemented by computing the Grassfire transform at each pixel, to computes the minimum distance to a non-background pixel in the neighboring pixels in the forward pass, and the minimum distance to a non-background pixel in the neighboring pixels in the backward pass. These minimum distances are then combined to produce the final distance value for the pixel which is then compares the threshold against the distance transform using the specified steps argument. The algorithm also determines whether to use a 4-connected or 8-connected depending on the required accuracy and was used with combined region growing algorithm and counting the number of components to get the feature vectors to store in a database.

For classification and matching Nearest Neighbor is the simplest method of cluster classification. It requires a single example from each class and an unknown feature vector will be matched to the most similar example from a training data set while K Nearest Neighbor uses k examples of each class and calculates the distance to a class, which is done by measuring the sum of distances to the k closest. For each class we found the k closest and sum their distances.

The function takes a vector of features for a target example and an unordered map of labeled training examples. The training examples are stored in the map as a vector of vectors, where each inner vector represents the features of a single example. The keys of the map are strings that represent the labels of the examples.

Our function iterates through each category in the map and computes the sum of the k smallest distances between the target example and the training examples in the category. The distance metric used is a mean-scaled Euclidean distance. The k smallest distances are summed and stored in an unordered map where the keys are the category labels.

After computing the sums of the distances for each category, the function finds the label with the smallest sum of distances. If the minimum distance is too large, indicating that the target example is far from all labeled examples, the function returns an "unknown" label. Otherwise, it returns the label with the smallest sum of distances.

It extracts features from the images, and optionally performs classification on the test features based on information from our training set. When the program is in classification mode, it loads a feature database from a CSV file and adds metadata. If the program is in SVM classification mode, it trains an SVM model on the feature database.

The performance of our different function will be summarized using the confusion / error matrix of some items.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SVM\_Test Set** | Bottle | Hair tie | Remote | Glasses |
| Bottle | 4 | 0 | 0 | 0 |
| Hair tie | 0 | 4 | 0 | 0 |
| Remote | 0 | 0 | 4 | 0 |
| Glasses | 0 | 0 | 0 | 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NN\_Test Set** | Bottle | Hair tie | Remote | Glasses |
| Bottle | 4 | 0 | 0 | 0 |
| Hair tie | 0 | 4 | 0 | 0 |
| Remote | 0 | 0 | 4 | 0 |
| Glasses | 0 | 0 | 0 | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **KNN\_Test Set** | Bottle | Hair tie | Remote | Glasses | Unknown |
| Bottle | 4 | 0 | 0 | 0 | 0 |
| Hair tie | 0 | 4 | 0 | 0 | 0 |
| Remote | 0 | 0 | 4 | 0 | 0 |
| Glasses | 0 | 0 | 0 | 3 | 1 |

In summary, when our program captures video from a camera device, the plain video feed and the feature view are displayed simultaneously, and frames processed and passed through one of the 4 modes: train, classify with nearest neighbor, classify with k-nearest neighbors, and classify with SVM.

In training mode, hitting the spacebar will capture the current frame to be labeled. The label is entered on the command line, and the label and feature vector are appended to the database csv file.

Graphical user interface

Description automatically generated

In the classification modes, hitting the spacebar will capture the current frame to be classified. The predicted label is output on the video output and the command line. The video output waits for any key press to resume live video.

The feature vectors were collected based on various metrics such as Hu moments and other distance metrics, morphological operations, and algorithms. We created a confusion matrix based off some items from the generated csv and used that to evaluate the model’s performance. The matrix showed no difference in performance between the SVM set and the Nearest Neighbors set. However, we see a misclassification of one of the glasses while using K Nearest Neighbors which puts it at a worse performance level than the Nearest Neighbors model and the SVM model.

**Submission Requirements**

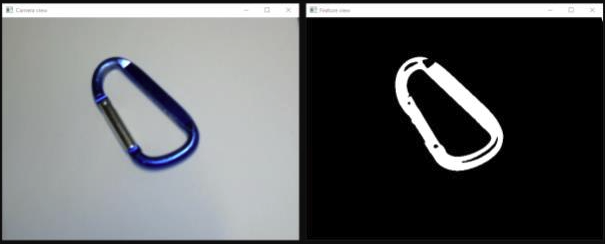
*Task 1*

· Preprocessing: 5x5 Gaussian blur

· Convert to grayscale

· Threshold at intensity = 127

· Invert so that foreground pixels are white

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

*Task 2*

· First dilate with 7 passes and 8-connectedness

· Then erode with 10 passes and 4-connectedness

A picture containing paper clip

Description automatically generated Graphical user interface

Description automatically generated with low confidence

A pair of glasses

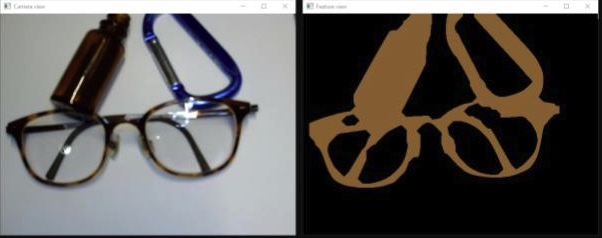
Description automatically generated with medium confidence

*Task 3*

· Use cv::connectedComponents

· Filter out regions that have an (region area) < 0.05 \* (image area)

Graphical user interface

Description automatically generated with medium confidence

*Task 4*

· Choose the largest region as the object of interest

· Compute Hu moments using cv::moments and cv::HuMoments

· Compute and visualize oriented bounding box and centroid

· Compute aspect ratio as (longest side) / (shortest side) of the oriented bounding box

· Compute percentage filled as (# foreground pixels) / (area of oriented bounding box)

Graphical user interface, application

Description automatically generatedA picture containing text, indoor

Description automatically generated

Graphical user interface

Description automatically generated

*Task 6*

· Nearest neighbors

· Distance metric is mean scaled Euclidean (with square root omitted):

Diagram

Description automatically generated

· Mean scaled Euclidean produces a distance measured in units of standard deviations and each feature’s standard deviation is computed across all training samples.

· If the best distance is more than 3 standard deviations, then the object is given the label UNKNOWN.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface, application

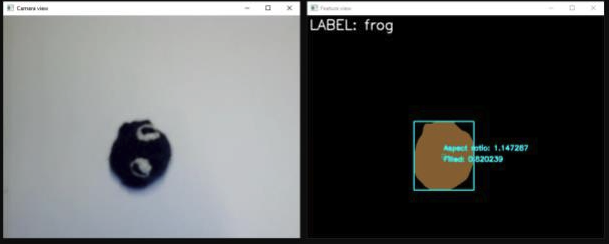
Description automatically generatedGraphical user interface

Description automatically generated

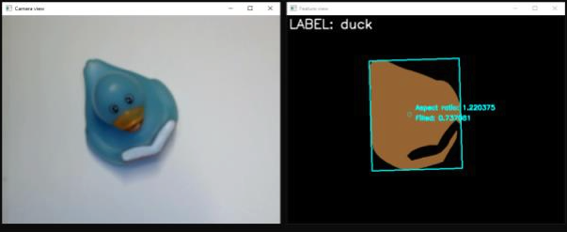
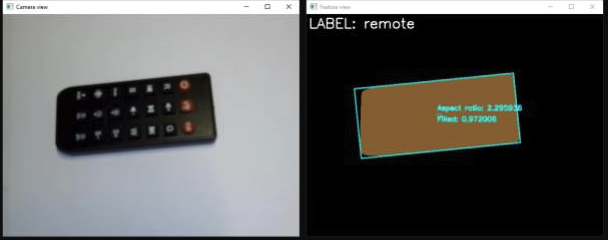
Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

Graphical user interface

Description automatically generated



Graphical user interface

Description automatically generated

A Video Demo to the process can be found here: <https://youtu.be/ns6hLb7FKow>

Real-time object recognition has numerous applications, including robotics, self-driving cars, surveillance systems, and augmented reality. For example, in self-driving cars, object recognition is used to detect and classify different types of objects, such as pedestrians, other vehicles, and traffic signals, to enable safe and reliable navigation.

In conclusion, 2D real-time object recognition is a crucial aspect of computer vision that has numerous real-world applications. Its success relies on the development of efficient algorithms and hardware that enable accurate and fast processing of images and videos.

Author links open overlay panelAlejandro Blumenkrans, et al. “Two-Dimensional Object Recognition Using a Two-Dimensional Polar Transform.” *Pattern Recognition*, Pergamon, 19 May 2003, https://www.sciencedirect.com/science/article/abs/pii/003132039190007R.

“Grassfire Transform.” *Semantic Scholar*, https://www.semanticscholar.org/topic/Grassfire-transform/2215069.

Li, Runya, and Shenglian Li. “Multimedia Image Data Analysis Based on KNN Algorithm.” *Computational Intelligence and Neuroscience*, U.S. National Library of Medicine, 12 Apr. 2022, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9018202/.

Nasrudin1, Mohd Wafi, et al. “IOPscience.” *Journal of Physics: Conference Series*, IOP Publishing, 1 July 2021, https://iopscience.iop.org/article/10.1088/1742-6596/1962/1/012028/meta.

“Confusion Matrix - Model Building and Validation.” *YouTube*, YouTube, 23 Feb. 2015, https://www.youtube.com/watch?v=AOIkPnKu0YA&ab\_channel=Udacity.