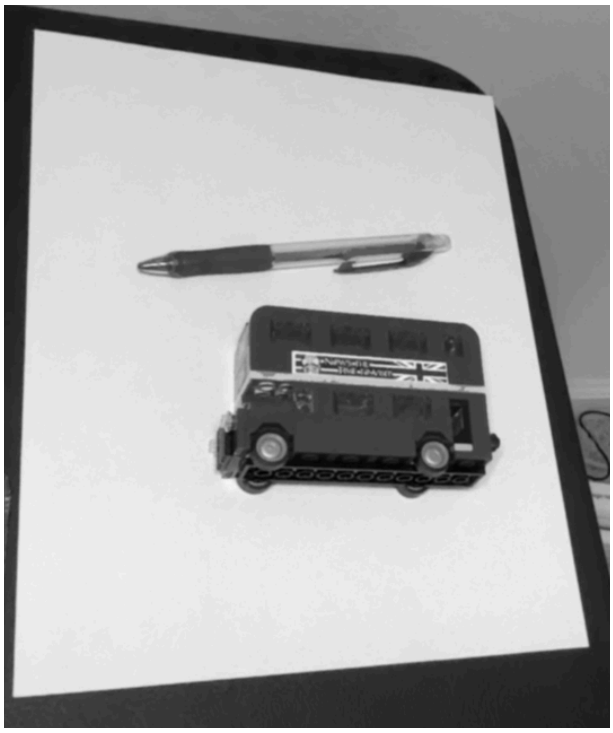
Spring CS5330
Project 3
Feb 16 2025
Yunyu Guo

1. **Threshold the input video**
   Thresholding in object recognition is an image segmentation technique that separates objects from their background by converting an image into a binary (black and white) image. If pixel_value > threshold, pixel becomes WHITE, otherwise Black.

   Required: Include 2-3 examples of the thresholded objects in your report and an explanation of your method.
   First method: pre-process the image using blurring and making strongly colored pixels (pixels with a high saturation value) darker. Then dynamically set threshold by analyzing the pixel values. Running a k-means algorithm on a random sample of pixel values with K=2. The average of two cluster centers is used as the threshold value because it represents the midpoint between the two dominant groups in the image. Pixels closer to one cluster centre will be in one group.

   Original

Preprocessed



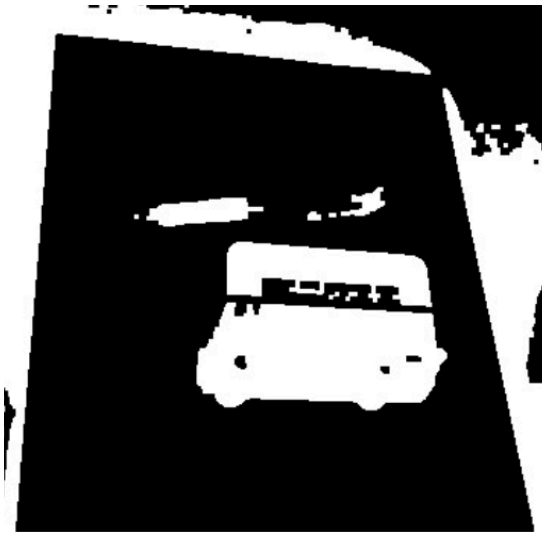
Thresholded

Extension1: use both RGB and depth information to do thresholding, drag the trackbar to adjust the RGB weight and depth weight. Then use morphological operations: first use Opening to remove small noise, then use closing to fill small holes. The morphological operation creates cleaner, more consistent object boundaries.

RGB

Depth

Combined Threshold
Images saved with weights: RGB 65%, Depth 35%



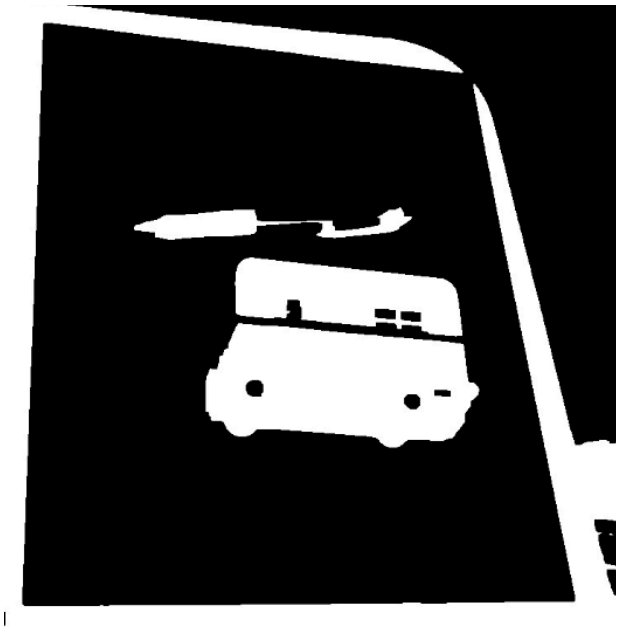
2. **Clean up the binary image**
   Required: Include 2-3 examples of cleaned up images in your report and an explanation of your method

   Thresholded
   The details: pen top and a cross symbol on the bus are missing in the thresholded image.

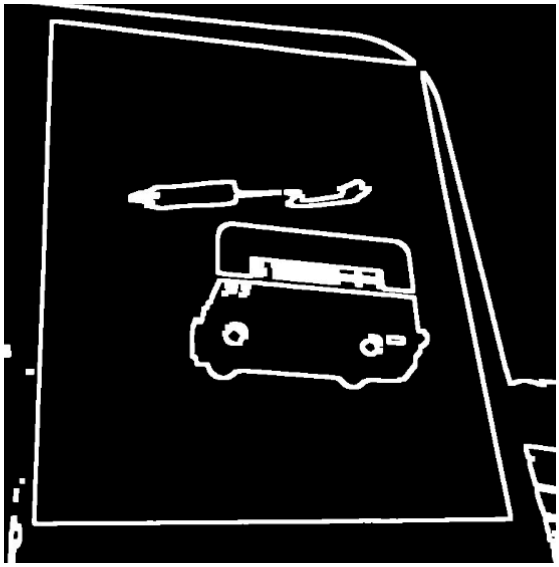
Open Morphological operation (remove noise)

Close Morphological operation(fill holes)

The pen top and the cross symbol are showing



Gradient Morphological operation



Use Open Morphological first to remove noise, then use Close Morphological operation to fill holes. Use Gradient Morphological operation to find object boundaries.

3. **Segment the image into regions**
   Connected components analysis on the thresholded and cleaned image to get regions. Focus on the region that is larger than a certain size, is most central to the image, and does not touch the image boundary. Compute the region size, region centroid, and region axis-oriented bounding box from a region map.

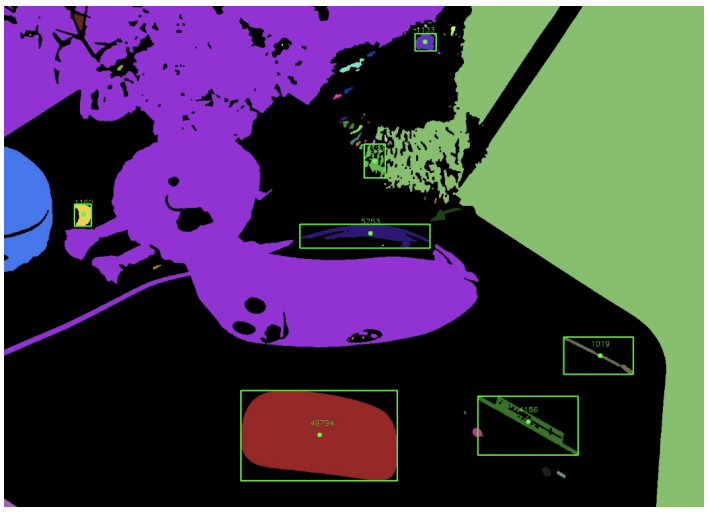
   Required images: Include 2-3 examples of region maps, ideally with each region shown in a different color.

   Origin

Region map
The number inside is the area (number of pixels) of the region

Origin

Region map



4. **Compute features for each major region**

Use OpenCV moment function: cv::moments(mask, true), I use these moments for features:

1. Area using zero-order moment

```
features.area = moments.m00;  // Total pixel count
```

2. Center of mass using first-order moments.

```
features.center = cv::Point2f(moments.m10/moments.m00,  // x coordinate
                              moments.m01/moments.m00);   // y coordinate
```

3. Orientation using second-order central moments

```
double mu11 = moments.mu11;  // Mixed second-order central moment
double mu20 = moments.mu20;  // Second-order central moment about x
double mu02 = moments.mu02;  // Second-order central moment about y
features.orientation = 0.5 * atan2(2*mu11, mu20 - mu02);  // Angle of least central moment
```

The feature vector components:

```
double area;          // 1. Total number of pixels (m00)
double perimeter;     // 2. Length of region boundary
double circularity;   // 3. roundness measure
double aspectRatio;   // 4. min(width,height)/max(width,height) of bounding box
double percentFilled; // 5. area/bounding_box_area (density measure)
double orientation;   // 6. Angle of least central moment
cv::Point2f center;   // 7,8. (x,y) center of mass from m10/m00, m01/m00.
```

Area (m00):

Translation, scale, rotation invariant. Measures object size in pixels. Useful for filtering out noise or small regions.

Perimeter:

Scale dependent. Length of object boundary. Helps characterize shape complexity. Longer perimeter = more complex shape

Circularity:
Translation, scale, rotation invariant. Measures how circular the object is. Helps distinguish round vs elongated objects. Range: 0-1 (1 = perfect circle).

Aspect Ratio:
Translation, scale, rotation invariant. Range: 0 to 1 (1 = square/circle). Measures elongation of shape. Helps distinguish tall vs wide objects.

Percent Filled:
Translation, rotation invariant.Range: 0 to 1 (1 = completely filled). Measures density of object in its bounding box. Helps distinguish solid vs hollow objects.

Orientation:
Translation invariant. Range: -π/2 to π/2. Principal axis angle. Helps determine object rotation. Important for alignment/matching.

Center (x,y):
Translation dependent. Object's center of mass. Important for tracking object position. Used as reference point for other measurements. This is not used for object recognition, only for visual center point and orientation line.

Displays:
Green dot for center
Green line for orientation
Red box for oriented bounding box
Complete feature vector in top-left

Required: Include 2-3 examples of regions showing the axis of least central moment and the oriented bounding box and explain your methods. Use the same images as for the prior tasks. Include the computed feature vectors for each object.
See video example

5. **Collect training data**

   This training system works as follows:

   1. Region Detection:

   Find connected components

   Sort and limit regions to 5 largest regions

   2. Region Selection:

   Press 1-5 to select region

   3. Compute features when a region is selected:

   RegionFeatures features = computeFeatures(mask, label);

   Features computed: area, perimeter, circularity, aspect ratio, percent filled, orientation

   4. Label Assignment and Saving:

   Press “n” to enter new label, then save feature vector to CSV

6. **Classify new images**

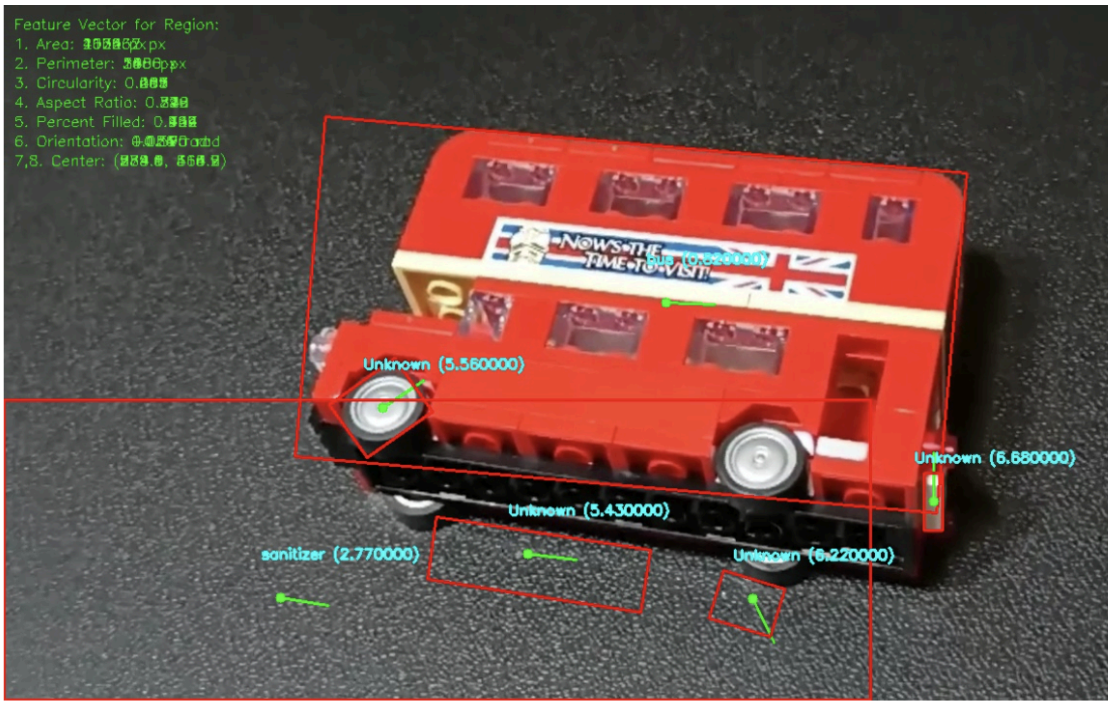
   Required images: include a result image for each category of object in your report with the assigned label clearly shown and explain your distance metric.

Distance calculation:

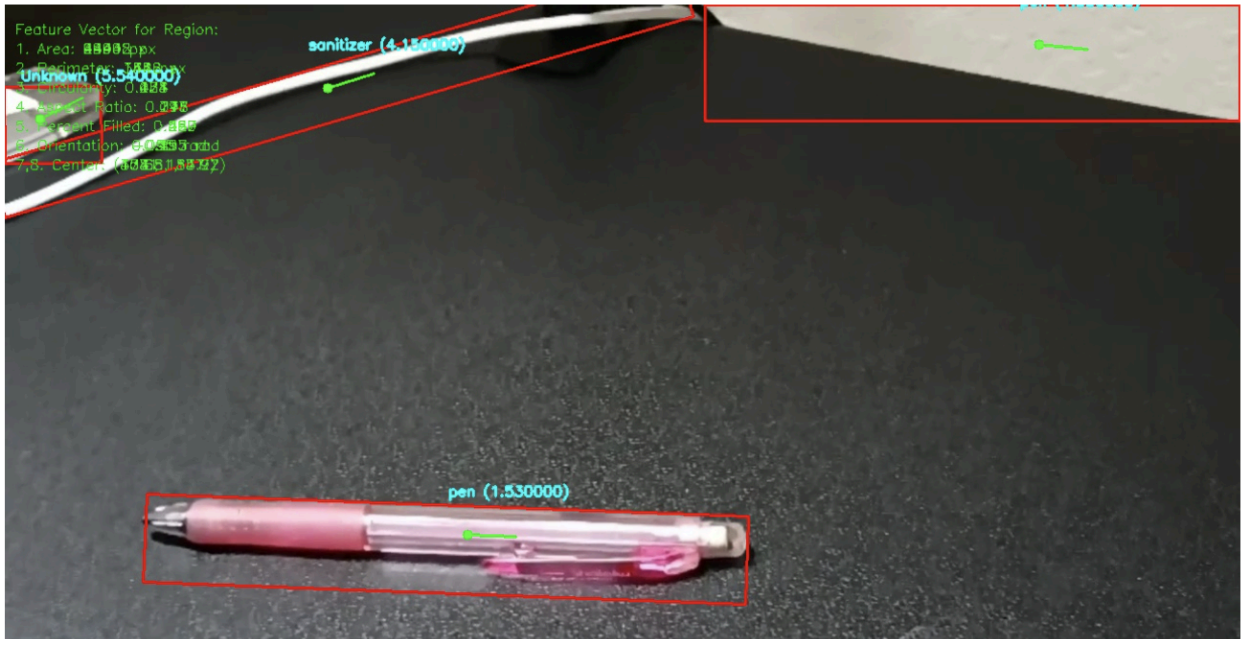
1. For each feature (area, perimeter, circularity, etc.):
   normalizedDiff = (current_feature - database_feature) / standard_deviation
   Normalization by standard deviation makes each feature contribute proportionally to its typical variation in the training data.
2. Total distance calculation:
   distance = sqrt(sum(normalizedDiff²))
3. Classification decision:
   If distance < 5.0: Use the label of closest match
   If distance > 5.0: Label as "Unknown"

   Object database: bus, pen, mouse, duck, sanitizer

Was able to detect: bus

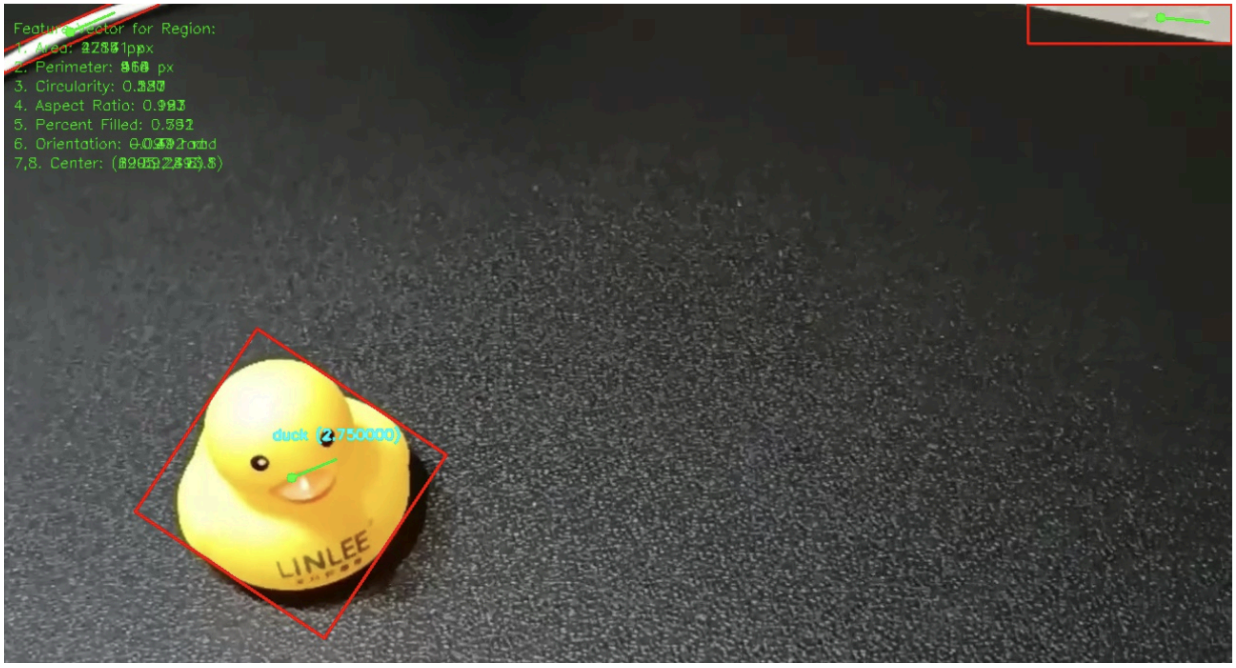


Was able to detect: pen

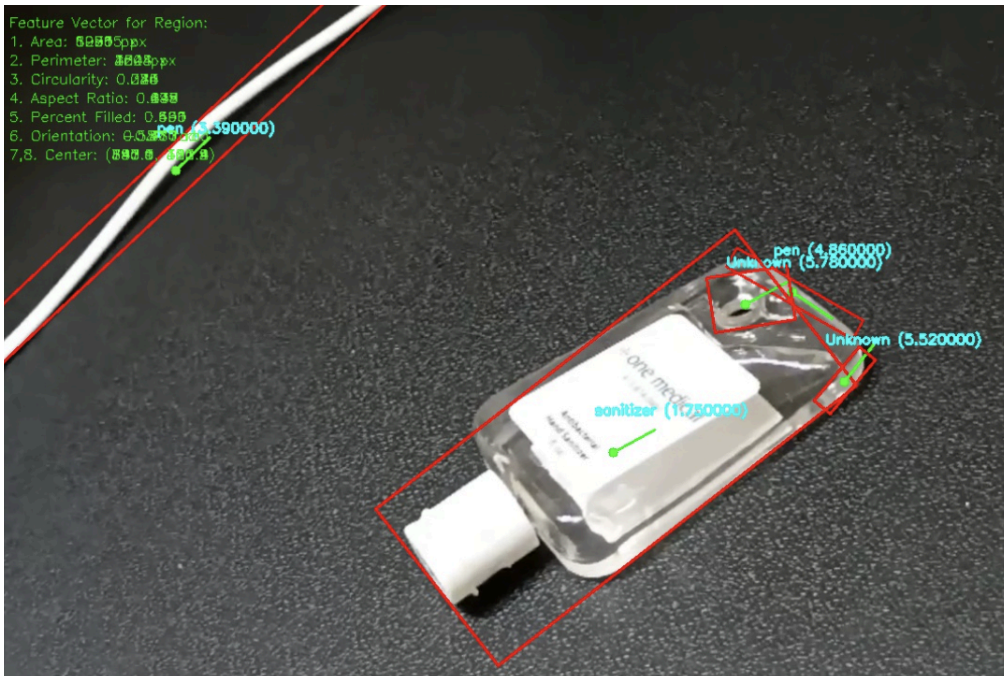


Was able to detect: mouse

Was able to detect: duck



Was able to detect: sanitizer



Extension2: An extension is to detect when an unknown object (something not in the object DB) is in the video stream or provided as a single image.

7. Evaluate the performance of your system

Evaluate the performance of the system on 3 different images of each object in different positions and orientations. Build a 5x5 confusion matrix of the results showing true labels versus classified labels.

Required: a 5x5 confusion matrix in your report.

A different dataset was used: mouse, orange, pen, book, cup

Confusion Matrix:

True\Pred	mouse	orange	pen	book	cup
mouse	0	3	0	0	0
orange	0	3	0	0	0
pen	0	0	3	0	0
book	0	0	0	3	0
cup	0	0	0	0	3

mouse: 0/3 (0% accuracy) - all misclassified as "orage"
orange: 3/3 (100% accuracy)
pen: 3/3 (100% accuracy)
book: 3/3 (100% accuracy)
cup: 3/3 (100% accuracy)

8. Capture a demo of your system working

Take a video of your system running and classifying objects.

See this video for system classifying objects with L1 (Euclidean) and L2 (Manhattan) showing on screen.

Check if object is unknown in both metrics

Show prompt to learn when unknown object detected

Press 'l' to learn new object

Save features to database with new label

5. Reload database to include new object

9. Implement a second classification method

Use nearest neighbour matching, compare different distance metrics: Euclidean distance with scaling by stddevs and Manhattan distance with scaling by stddevs (scaled L-1).

Explain your choice and how you implemented it. Also, compare the performance with your baseline system.

Use different thresholds for different distance:

```
const double L2_THRESHOLD = 3.0;  // Euclidean threshold
const double L1_THRESHOLD = 5.0; // Manhattan threshold (typically larger)
```

Euclidean:

```
double normalizedDiff = (features[i] - entry.features[i]) / stddevs[i];
    distance += normalizedDiff * normalizedDiff;
distance = sqrt(distance);
```

Manhattan:

```
for(size_t i = 0; i < features.size(); i++) {
    double normalizedDiff = (features[i] - entry.features[i]) / stddevs[i];
    distance += std::abs(normalizedDiff);
```

```
L2 (Euclidean) Confusion Matrix:

Confusion Matrix:
True\Pred mouse     pen       orange    book      cup
mouse     1         0         0         0         0
pen       0         1         0         0         0
orange    0         0         1         0         0
book      0         0         0         1         0
cup       0         0         0         0         1

L1 (Manhattan) Confusion Matrix:

Confusion Matrix:
True\Pred mouse     pen       orange    book      cup
mouse     1         0         0         0         0
pen       0         1         0         0         0
orange    0         0         1         0         0
book      0         0         0         1         0
cup       0         0         0         0         1
```

Using both distances has 100% accuracy.

Extension3: *Enable your system to learn new objects automatically by first detecting whether an object is known or unknown, and then collecting statistics for it if the object is unknown. Demonstrate how you can quickly develop a DB using this feature.*

In the classification step, it implements the following:

1. Check if object is unknown in both distance metrics
2. Show prompt to learn when unknown object detected: Press 'l' to learn new object
3. Save features to database with new label
4. Reload database to include new object

This is demonstrated in the video.