Pattern Recognition and Computer Vision

CS 5330 – Spring 2026

Project 2

“Real-time 2-D Object Recognition”

Name: Harsh Vijay Mamania

NUID: 002036770

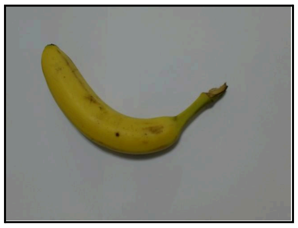
Overall Project Description:

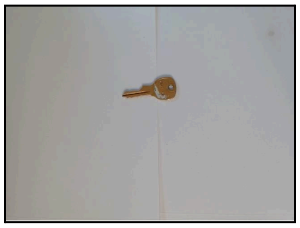
This project implements a real-time 2D object recognition system capable of identifying objects placed on a uniform white background using a downward-facing webcam. The system processes live video through a multi-stage pipeline: dynamic thresholding to separate objects from the background, morphological filtering to clean the binary image, connected components analysis to segment distinct regions, and moment-based feature extraction to characterize each region. A nearest-neighbor classifier using scaled Euclidean distance matches extracted features against a trained object database stored as a CSV file. The system supports recognition of up to 10 distinct objects and can detect up to 3 objects simultaneously in a single frame. In addition to the hand-crafted feature pipeline, the system integrates a pre-trained ResNet18 network to compute 512-dimensional CNN embeddings for one-shot classification, with a PCA-based 2D scatter plot for visualizing the embedding space. Both the thresholding algorithm and the morphological filtering operations were implemented from scratch in C++ using OpenCV for supporting utilities.

Objects used:

Ten everyday objects were selected for this experiment, chosen for their distinctive 2D shapes when viewed from above. The objects vary in shape complexity, aspect ratio, and silhouette - making them well-suited for shape-based recognition.

The objects used are: *banana, sleep mask, key, scissor, Swiss knife, lens case, sticker, toothpaste, allen key, and comb*.





Note: For each task, only 2-3 representative images are included. Images for all remaining objects are available in the accompanying image folder submitted with this report.

Extensions:

1. Multi-Object Detection:
   The system supports simultaneous detection and classification of up to 3 objects in a single frame, with each region displayed in a distinct color and labeled independently.

2. Embedding Separability Plot:
   After collecting CNN embeddings for multiple training examples, all embeddings are projected onto their top 2 principal components and displayed as a color-coded scatter plot, allowing visual inspection of how well the embedding space separates different object categories.

3. Additional From-Scratch Implementations:
   Both ***thresholding*** and ***morphological*** filtering were written entirely from scratch, exceeding the single required from-scratch task for solo students.

4. Expanded Object Set:
   The system was trained and evaluated on 10 objects (double the required minimum of 5).

The above is a brief overview of the extensions implemented. Each extension will be discussed in greater depth in its corresponding section throughout the report.

1. Thresholding the input video.

Summary:

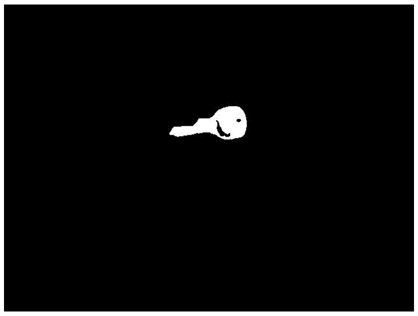
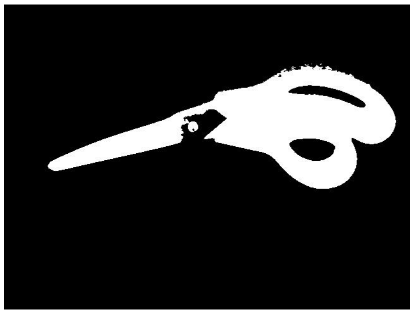
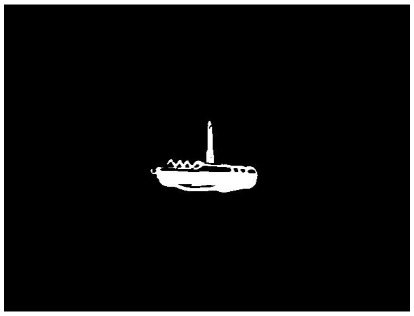
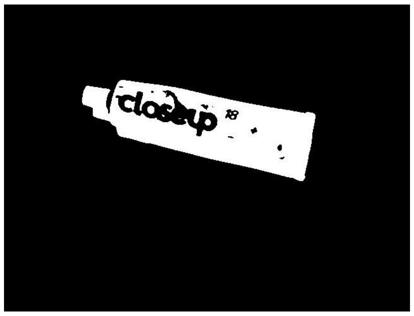
The thresholding pipeline separates the object from the background in each video frame, producing a binary image where foreground pixels are white and background pixels are black. The pipeline was implemented entirely from scratch in three stages.

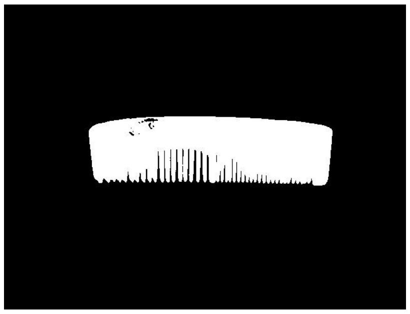
- First, each frame is preprocessed by applying a Gaussian blur, converting to HSV color space, and computing a saturation-weighted brightness value per pixel using modified_V = V x (1 - S/255). This darkens colorful pixels alongside dark ones, ensuring both are captured as foreground while the white unsaturated background remains bright.

- Second, a dynamic threshold is computed using the ISODATA algorithm (k-means with K=2) on a random sample of 1/16 of all pixels. The threshold is set as the midpoint between the two converged cluster means, allowing the system to adapt to varying lighting conditions automatically.

- Third, pixels with modified brightness below the threshold are labeled foreground (255) and the rest background (0).

Demo Images:



Observations:

- The thresholding pipeline performs well across all objects, cleanly separating foreground from the white background without any manual tuning, thanks to the saturation-weighted preprocessing and dynamic ISODATA thresholding.

- Key (Image 1): The key silhouette is captured cleanly. The hollow region near the key head appears as a small dark gap, which will be addressed in the morphological cleaning step.

- Scissor (Image 2): The scissor body is well-captured, but some edge noise and internal holes from the open handles are visible. A small isolated speck is present near the pivot point. These will be resolved during morphological filtering.

- Swiss Knife (Image 3): The main body is well-defined. The open blade creates thin disconnected regions above the body, and internal tool details introduce small holes, both of which will be filled during cleanup.

- Toothpaste (Image 4): The most challenging case. The printed branding on the tube thresholds as internal holes due to the saturation-weighted preprocessing responding to the colored text. The overall tube outline is still captured.

- Comb (Image 5): The comb body is cleanly captured. The fine teeth produce a jagged bottom edge with narrow gaps, and a small noise cluster is visible near the top-left corner, both expected to be resolved during morphological filtering.

---

2. Clean up the binary image

Summary:

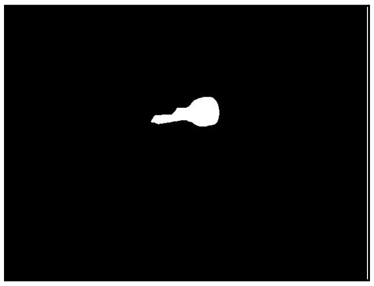
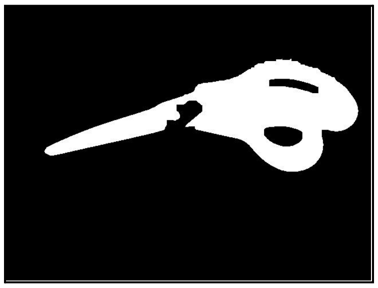
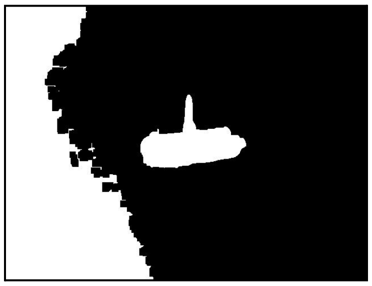
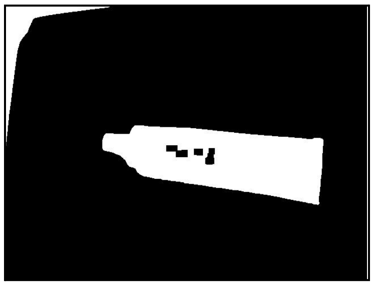
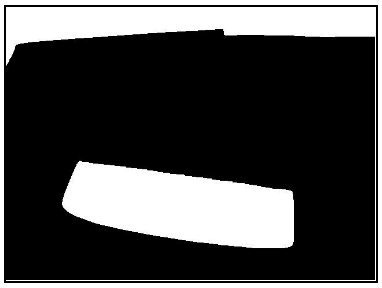
After thresholding, the binary image contains several artifacts including small noise clusters, internal holes within object regions, and jagged edges along fine structural details such as comb teeth. Morphological filtering was applied to clean up these artifacts, and both the dilation and erosion operations were implemented entirely from scratch using a 3x3 kernel.

The chosen strategy is **morphological closing**, which applies dilation followed by erosion.

- Dilation expands foreground regions outward, merging nearby noise clusters and bridging small gaps, while the subsequent erosion shrinks the regions back to approximately their original size.
- This order was chosen specifically because the dominant artifacts observed after thresholding were internal holes and edge noise rather than unwanted background regions.
- Closing fills holes and smooths boundaries while largely preserving the overall shape and size of the object silhouette.

To handle larger holes such as those seen in the scissor handles, toothpaste branding, and swiss knife internals, **5 passes of dilation** followed by **5 passes of erosion** were applied rather than a single pass. Multiple passes with a small kernel fill larger holes more effectively without the shape distortion that would result from using a single large kernel.

Demo Images:

Observations:

- Morphological closing successfully removes most noise and fills internal holes across the majority of objects, producing cleaner and more solid silhouettes compared to the raw thresholded output.
- Comb (Image 1): The comb body is significantly cleaner, with the jagged tooth gaps and top edge noise largely resolved. The overall silhouette is more solid compared to the thresholded result.
- Key (Image 2): The key is cleaned very well. The small hole in the key head is fully filled, noise is eliminated, and the silhouette is smooth and compact. This is one of the cleanest results across all objects.
- Scissor (Image 3): The scissor body is largely cleaned up with edge noise reduced. However, the large open handle regions remain as holes due to their size, and the isolated speck near the pivot point also persists.
- Swiss Knife (Image 4): The swiss knife object itself is cleaned well, with the internal tool details largely filled and the main body solid.
- Toothpaste (Image 5): Good improvement over the thresholded result. Most of the internal branding artifacts are filled, with only a few small holes remaining near the center of the tube. The overall tube silhouette is now largely solid and well-defined.

---

3. Segment the image into regions

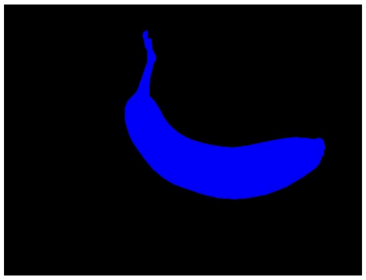
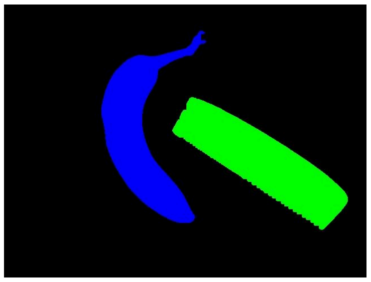
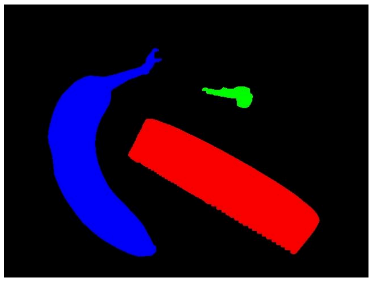
Summary:

Connected components analysis is applied to the cleaned binary image to identify and label distinct foreground regions. OpenCV's ***connectedComponentsWithStats*** function with 8-connectivity is used, which returns a label map, per-region statistics (bounding box, area), and centroids for each region.

To focus on meaningful object regions, two filtering criteria are applied: regions smaller than 500 pixels are discarded as noise, and regions whose bounding box touches any image boundary are rejected to avoid partial or merged edge regions. From the remaining valid regions, the top 3 largest regions are selected for further processing, enabling multi-object detection simultaneously.

For visualization, each valid region is assigned a distinct color from a fixed palette and displayed on a black canvas, making it easy to distinguish multiple regions at a glance.

Demo Images:

Observations:

- The connected components analysis successfully segments and color-codes distinct foreground regions. The multi-object detection extension is clearly demonstrated, with up to 3 simultaneous objects detected and labeled in distinct colors. Small noise regions are correctly filtered by the minimum area threshold.

- Banana alone (Image 1): A single region is detected and colored blue, cleanly capturing the full banana silhouette including the stem.

- Banana and Comb (Image 2): Two objects are correctly detected and assigned distinct colors. The comb's jagged bottom edge from the teeth is preserved in the region map.

- Banana, Comb, and Key (Image 3): Three objects are simultaneously detected, demonstrating the multi-object extension at full capacity. The key region is noticeably smaller but still passes the minimum area threshold and is correctly segmented.

---

4. Feature Computation

Summary:

For each detected region, a feature vector is computed that is invariant to translation, scale, and rotation. The pipeline computes the orientation angle of the region using second-order central moments via the formula **theta = 0.5 x atan2(2 x mu11, mu20 - mu02)**, and the oriented bounding box using OpenCV's *minAreaRect* on the region's pixel coordinates.

The feature vector consists of four features.

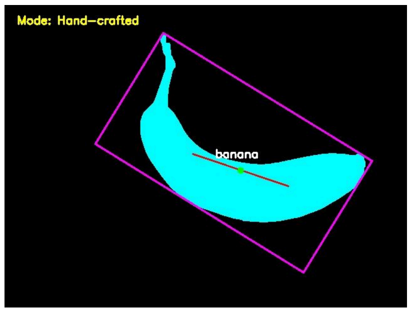
- "**Percent filled**" is the ratio of the region's pixel area to its oriented bounding box area, capturing how compactly the object fills its bounding box.
- "**Aspect ratio**" is the ratio of the longer to shorter side of the oriented bounding box, always greater than or equal to 1 regardless of orientation.
- "**Hu moment 1**" is derived from the second-order normalized central moments and captures the overall spread or mass distribution of the region. It is roughly equivalent to the normalized sum of variances along both axes, making it sensitive to how elongated or compact a shape is.
- "**Hu moment 2**" is derived from both second-order and mixed central moments and captures the difference in spread between the two principal axes. It is sensitive to how asymmetric or irregular the mass distribution is across the two axes.

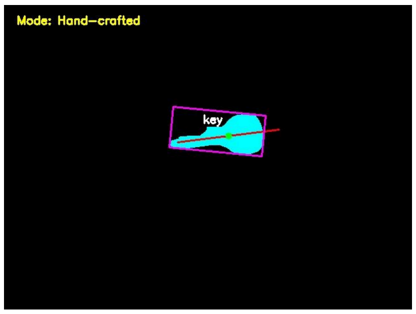
The primary axis and oriented bounding box are drawn overlaid on each region in the output display, rotating naturally as the object is moved or rotated in frame.

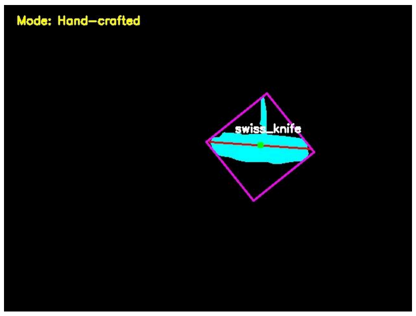
Feature Vectors:

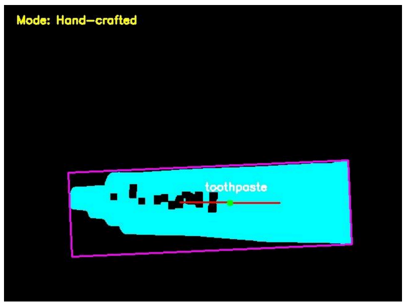
swiss_knife	*0.428937166*	*1.098651275*	*0.301183493*	*0.041178441*
toothpaste	*0.836712448*	*3.579226732*	*0.332765622*	*0.082504114*
allen_key	*0.300801442*	*3.323226427*	*1.044242962*	*0.889455657*
banana	*0.398201292*	*1.965023965*	*0.403791446*	*0.074697791*
key	*0.517336208*	*2.225746909*	*0.31478089*	*0.055879994*

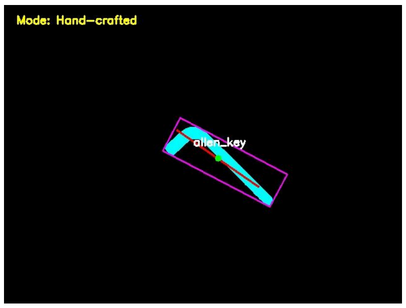
Demo Images:











Observations:

- The primary axis and oriented bounding box track each object accurately across different positions and orientations. The centroid is correctly placed at the visual center of mass for all objects, and the bounding box tightly fits the object silhouette in each case.

- Swiss Knife (Image 1): Feature vector: [0.429, 1.099, 0.301, 0.041]. The bounding box is rotated at roughly 45 degrees, correctly aligned with the knife's orientation. The low percent filled (0.429) reflects the open blade adding empty space to the bounding box, and the near-unity aspect ratio (1.099) indicates a relatively compact shape.

- Toothpaste (Image 2): Feature vector: [0.837, 3.579, 0.333, 0.083]. The bounding box fits the tube tightly and is well-aligned with its elongated shape. The high percent filled (0.837) confirms the tube is a solid, well-filled region, and the high aspect ratio (3.579) correctly captures its elongated form.

- Allen Key (Image 3): Feature vector: [0.301, 3.323, 1.044, 0.889]. The thin L-shaped profile results in the lowest percent filled (0.301) among all objects. The high Hu moment 2 (0.889) reflects the irregular, asymmetric mass distribution of the L-shape, distinguishing it clearly from other elongated objects.

- Banana (Image 4): Feature vector: [0.398, 1.965, 0.404, 0.075]. The curved shape results in a moderately low percent filled (0.398), as the bounding box encloses significant empty space around the curve. The aspect ratio (1.965) captures the moderate elongation of the fruit.

- Key (Image 5): Feature vector: [0.517, 2.226, 0.315, 0.056]. The bounding box fits compactly, with a moderate percent filled (0.517) and aspect ratio (2.226) reflecting the elongated head-to-shank proportion.

---

5. Collecting Training Data

Summary:

The training system operates in real time alongside the main recognition pipeline.

- When the user presses the '**n**' key while an object is in frame, the system prompts the user to enter a label for the current object via the console.
- The feature vector computed for the largest detected region at that moment is then appended to a CSV file called '***object_db.csv***' along with the entered label.
- Each row in the file contains the label followed by the four feature values: percent filled, aspect ratio, Hu moment 1, and Hu moment 2, stored with 9 decimal places of precision.
- Multiple training examples can be collected per object by placing the object in different positions and orientations and pressing 'n' each time. This builds a richer database that better captures the natural variation in each object's feature vector.
- The database is loaded into memory at startup, and per-feature standard deviations are computed immediately from the loaded examples to prepare for scaled Euclidean distance classification.

The training system is lightweight: no retraining or model rebuilding is required after adding new examples, as the nearest-neighbor classifier operates directly on the stored feature vectors at classification time.

Note: The complete training database is included as '***object_db.csv***' in the submission. It contains all training examples collected across all 10 objects used in this project.

6. Classification

Summary:

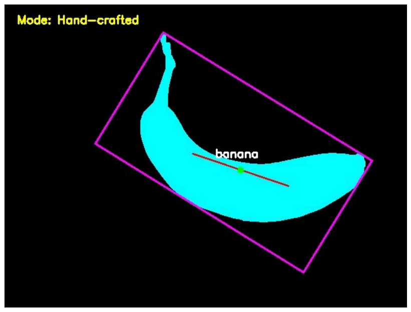
- The classification system uses a nearest-neighbor approach with a scaled Euclidean distance metric. For each detected region, the computed feature vector is compared against every entry in the training database, and the label of the closest matching entry is assigned to the object.

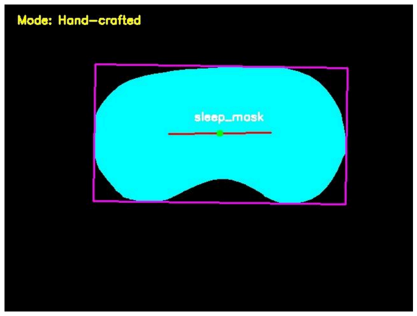
- The distance metric used is the scaled Euclidean distance, where each feature difference is divided by the per-feature standard deviation computed across all training examples. This normalization ensures that features with larger absolute ranges do not dominate the distance calculation over features with smaller ranges. The formula computes the square root of the sum of squared scaled differences across all four features.

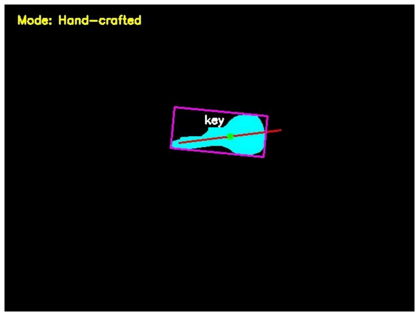
- Per-feature standard deviations are computed once at startup after loading the database and reused for every classification call. If a feature has zero variance across all training examples, its standard deviation is floored at 1e-6 to avoid division by zero.

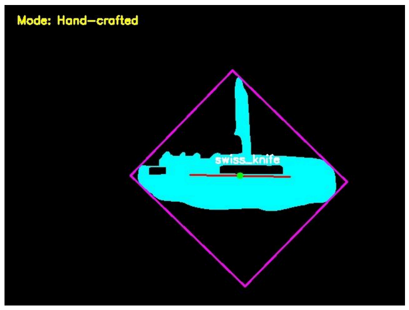
- The system supports two classification modes togglable at runtime using the 'x' key: hand-crafted feature classification using the nearest-neighbor approach described above, and CNN embedding classification described in Task 8. The active mode is displayed on the output window at all times.

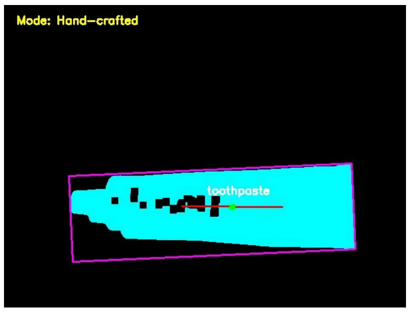
Demo Images:

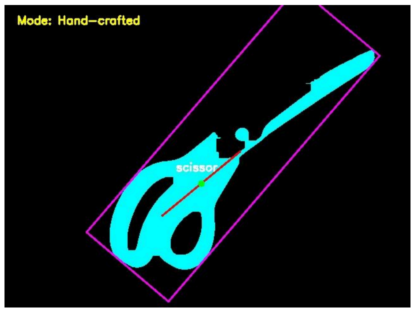

Mode: Hand—crafted
banana

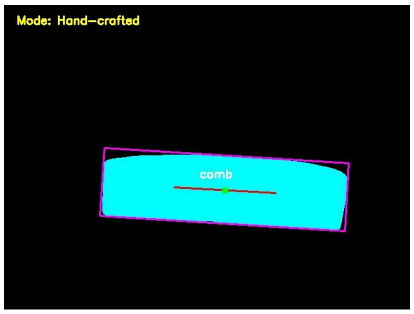

Mode: Hand—crafted
sleep_mask

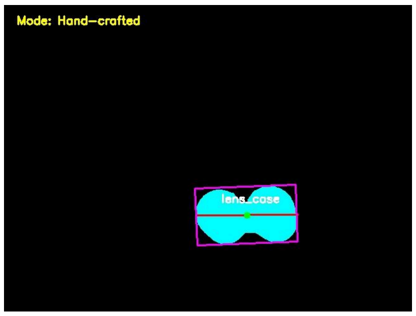

Mode: Hand—crafted
key

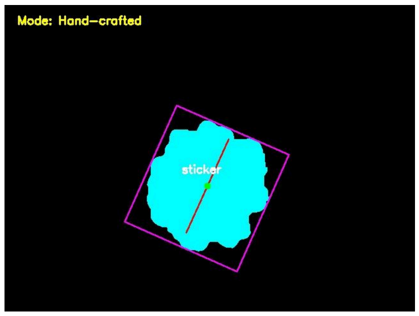

Mode: Hand—crafted
swiss_knife

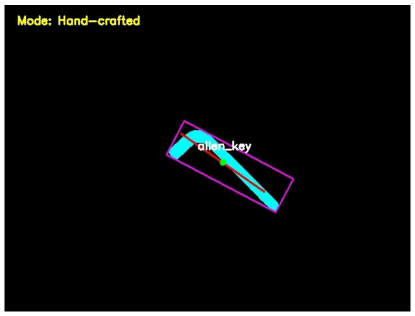

Mode: Hand—crafted
toothpaste


Mode: Hand—crafted
scissor


Mode: Hand—crafted
comb


Mode: Hand—crafted
lens_case


Mode: Hand—crafted
sticker


Mode: Hand—crafted
allen_key

Observations:

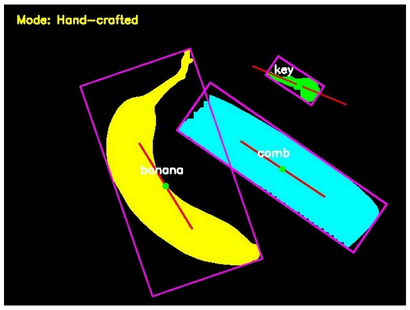
- The hand-crafted nearest-neighbor classifier correctly identifies all 10 objects in single-object mode. The predicted label is displayed at the centroid of each detected region, and the oriented bounding box and primary axis rotate correctly with the object across different positions and orientations.
- Banana (Image 1): Correctly classified. The curved silhouette and moderate aspect ratio produce a distinctive feature vector that separates it well from other objects.
- Sleep Mask (Image 2): Correctly classified. The wide, compact, and nearly symmetric silhouette with a low aspect ratio and high percent filled makes it one of the more distinctive objects in the set.
- Key (Image 3): Correctly classified. The small elongated shape with a rounded head produces a compact and consistent feature vector.
- Swiss Knife (Image 4): Correctly classified. The open blade creates a large bounding box relative to the object area, resulting in a distinctively low percent filled value that aids classification.
- Toothpaste (Image 5): Correctly classified despite the residual internal holes from the branding. The high percent filled and high aspect ratio together produce a sufficiently distinctive feature vector.
- Scissor (Image 6): Correctly classified. The open handle holes are visible within the region, but the overall shape features are distinctive enough for correct nearest-neighbor matching.
- Comb (Image 7): Correctly classified. The rectangular, high aspect ratio silhouette with high percent filled distinguishes it cleanly from other objects.
- Lens Case (Image 8): Correctly classified. The compact, nearly square double-lobed silhouette with low aspect ratio produces a distinctive feature vector.
- Sticker (Image 9): Correctly classified. The roughly circular, irregular-edged silhouette results in a near-unity aspect ratio and moderate percent filled, distinguishing it from more elongated objects.
- Allen Key (Image 10): Correctly classified. The thin L-shaped profile produces the lowest percent filled and highest Hu moment 2 values in the dataset, making it one of the most distinctively separable objects.

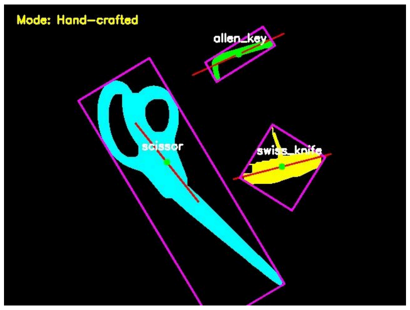
Extension: Multi-Object Classification

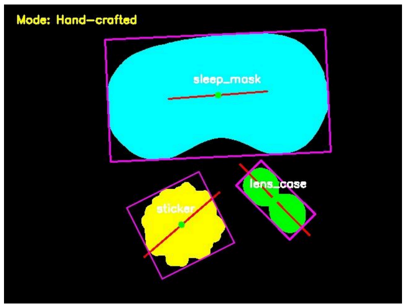
The system extends single-object classification to support simultaneous detection and classification of up to 3 objects in a single frame.

- After connected components analysis, the top 3 largest valid regions are selected and processed independently in a loop.
- For each region, moments, oriented bounding box, and feature vector are computed separately, and the nearest-neighbor classifier is run independently for each region.
- Each object is displayed in a distinct color with its predicted label overlaid at its centroid location.
- The primary axis and oriented bounding box are drawn for each region individually, rotating correctly with each object's orientation.
- This means the system can classify three entirely different objects simultaneously in real time, with each region going through the full feature extraction and classification pipeline independently.

Demo Images:







Observations:

- The multi-object extension successfully detects and classifies up to 3 objects simultaneously. Each region is processed independently through the full pipeline, assigned a distinct color, and labeled at its centroid with the correct predicted label.

- Banana, Comb, and Key (Image 1): All three objects are correctly classified. The distinct shapes and sizes of the three objects produce well-separated feature vectors, making simultaneous classification straightforward.

- Scissor, Allen Key, and Swiss Knife (Image 2): All three objects are correctly classified despite significant size differences between them. The bounding boxes and primary axes are individually correct for each object, including the rotated diamond-shaped box for the swiss knife.

- Sleep Mask, Sticker, and Lens Case (Image 3): All three objects are correctly classified. The sleep mask, sticker, and lens case each produce sufficiently distinct feature vectors, and the classifier correctly identifies all three simultaneously.

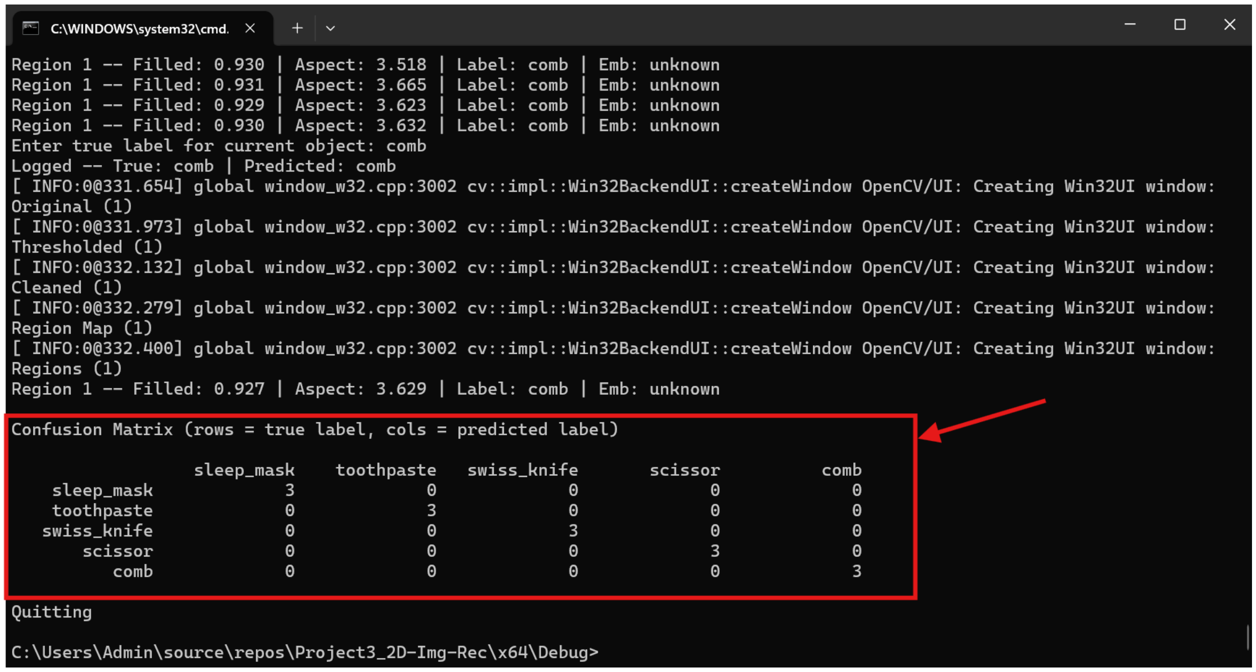
---

7. Performance Evaluation

Summary:

The system's classification performance was evaluated on at least 3 images of each object in different positions and orientations. The evaluation mode is triggered by pressing the '**e**' key during live operation, which prompts the user to enter the true label for the currently detected object and logs the true/predicted label pair. After all evaluations are collected, the confusion matrix is printed to the console by pressing the '**m**' key.
For this report, evaluation is presented for 5 objects, producing a 5x5 confusion matrix. The same methodology can be directly extended to all 10 objects.

Confusion Matrix:



```
Region 1 -- Filled: 0.930 | Aspect: 3.518 | Label: comb | Emb: unknown
Region 1 -- Filled: 0.931 | Aspect: 3.665 | Label: comb | Emb: unknown
Region 1 -- Filled: 0.929 | Aspect: 3.623 | Label: comb | Emb: unknown
Region 1 -- Filled: 0.930 | Aspect: 3.632 | Label: comb | Emb: unknown
Enter true label for current object: comb
Logged -- True: comb | Predicted: comb
[ INFO:0@331.654] global window_w32.cpp:3002 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Win32UI window:
Original (1)
[ INFO:0@331.973] global window_w32.cpp:3002 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Win32UI window:
Thresholded (1)
[ INFO:0@332.132] global window_w32.cpp:3002 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Win32UI window:
Cleaned (1)
[ INFO:0@332.279] global window_w32.cpp:3002 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Win32UI window:
Region Map (1)
[ INFO:0@332.400] global window_w32.cpp:3002 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Win32UI window:
Regions (1)
Region 1 -- Filled: 0.927 | Aspect: 3.629 | Label: comb | Emb: unknown

Confusion Matrix (rows = true label, cols = predicted label)

                  sleep_mask    toothpaste   swiss_knife       scissor          comb
    sleep_mask             3             0             0             0             0
    toothpaste             0             3             0             0             0
   swiss_knife             0             0             3             0             0
       scissor             0             0             0             3             0
          comb             0             0             0             0             3

Quitting

C:\Users\Admin\source\repos\Project3_2D-Img-Rec\x64\Debug>
```

Observations:

- The confusion matrix shows perfect classification performance across all 5 evaluated objects, with each object correctly classified in all 3 test images. The diagonal is entirely populated with 3s and all off-diagonal entries are 0, yielding a 100% accuracy across 15 total evaluations.

- Each of the 5 objects, sleep mask, toothpaste, swiss knife, scissor, and comb, produces a sufficiently distinct feature vector such that the nearest-neighbor classifier finds the correct match in every case. This strong performance is attributed to the diversity in shape, aspect ratio, and fill characteristics across these particular objects, which are well-separated in the 4-dimensional feature space.

- The overall system accuracy for this evaluation is 15/15 (100%). As noted, this evaluation can be extended to all 10 objects in the same manner to produce a full 10x10 confusion matrix.

---

8. Video demo of my system

Here's the link to the video demo of my 2D-Object-Recognition-System in action:

https://drive.google.com/file/d/1dA2q8kXkPgLNkyLNd-A5uUciM8vvJaxO/view?usp=sharing

9. One-Shot Classification system using CNN Embedding

Summary:

The one-shot classification system uses a pre-trained ResNet18 network to compute 512-dimensional embedding vectors for each detected object, replacing the hand-crafted feature vector with a deep learned representation. The implementation uses professor-provided utility functions (***prepEmbeddingImage*** and ***getEmbedding***).

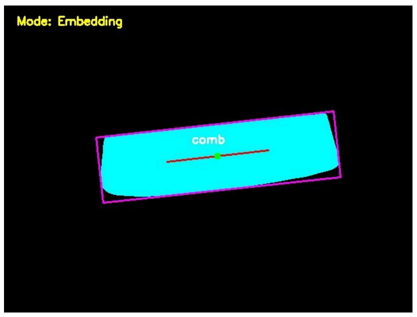
The pipeline for computing an embedding follows five steps.

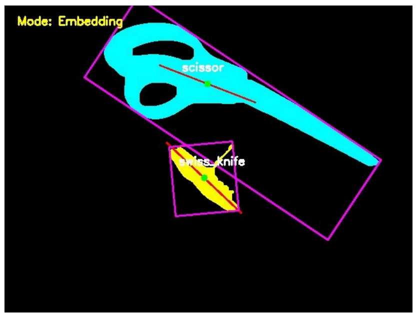
- First, the centroid, primary axis angle, and axis extents of the detected region are computed from the moment analysis performed in Task 4.
- Second, the original frame is rotated so the region's primary axis aligns with the X-axis using ***getRotationMatrix2D*** and ***warpAffine***.
- Third, the axis-aligned bounding box of the object is extracted as a region of interest from the rotated image using the computed axis extents.
- Fourth, the extracted ROI is resized to 224x224 pixels for ResNet18 input.
- Fifth, the image is passed through the pre-trained ResNet18 network and the 512-dimensional output from the penultimate layer is extracted as the embedding vector.

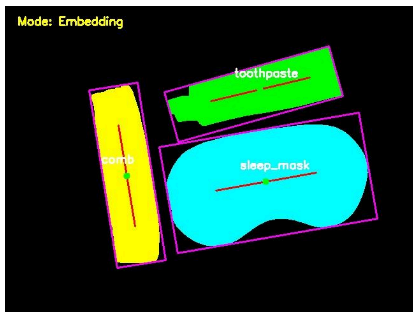
Training in embedding mode is triggered by pressing the **'t'** key, which saves the current embedding along with a user-entered label into an in-memory embedding database. Classification is performed using sum-squared difference between the unknown embedding and all stored embeddings, with the nearest match assigned as the predicted label. The active classification mode can be toggled between hand-crafted and embedding modes at runtime using the **'x'** key.

Note that the embedding database is stored in memory only and is not persisted to disk, meaning all collected embeddings are lost when the program exits and must be recollected at the start of each new session.

Demo Images:







Observations:

- The CNN embedding classifier correctly identifies all objects across all test images in both single and multi-object modes. The mode indicator in the top-left confirms the system is operating in embedding mode.

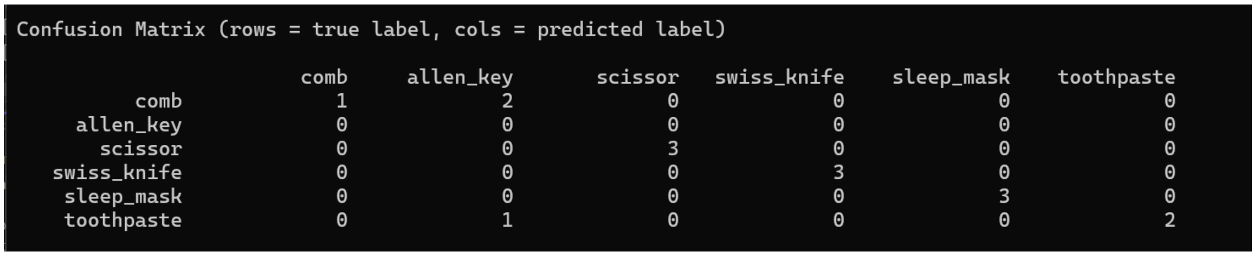
- Comb (Image 1): Correctly classified in single-object mode. The oriented bounding box and primary axis are well-aligned with the comb's elongated shape.

- Scissor and Swiss Knife (Image 2): Both objects correctly classified simultaneously. The scissor's large open-handle silhouette and the swiss knife's compact rotated shape produce sufficiently distinct embeddings for correct one-shot classification.

- Comb, Toothpaste, and Sleep Mask (Image 3): All three objects correctly classified simultaneously in embedding mode. The three objects are well-separated in the 512-dimensional embedding space despite their varying sizes and orientations.

Confusion Matrix:

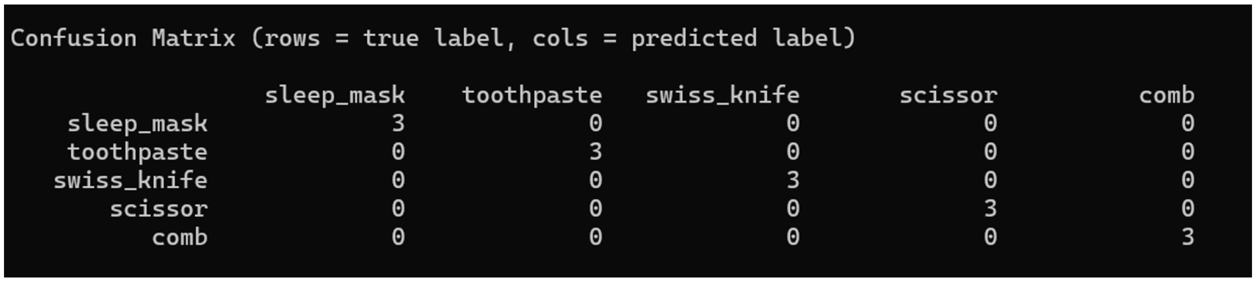
A) From CNN Embeddings:



```
Confusion Matrix (rows = true label, cols = predicted label)

                        comb     allen_key       scissor   swiss_knife    sleep_mask    toothpaste
         comb              1             2             0             0             0             0
    allen_key              0             0             0             0             0             0
      scissor              0             0             3             0             0             0
  swiss_knife              0             0             0             3             0             0
   sleep_mask              0             0             0             0             3             0
   toothpaste              0             1             0             0             0             2
```

B) From Hand-built features:

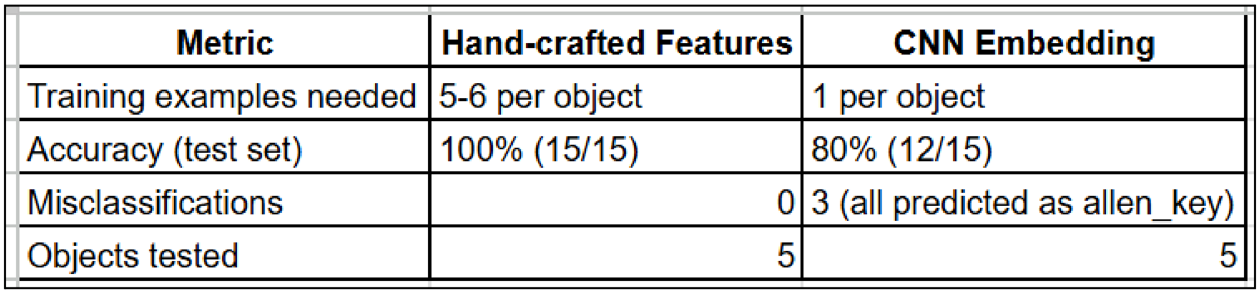


```
Confusion Matrix (rows = true label, cols = predicted label)

                  sleep_mask    toothpaste   swiss_knife       scissor          comb
   sleep_mask              3             0             0             0             0
   toothpaste              0             3             0             0             0
  swiss_knife              0             0             3             0             0
      scissor              0             0             0             3             0
         comb              0             0             0             0             3
```

Observations:

A) **Quantitative Comparison:**



Metric	Hand-crafted Features	CNN Embedding
Training examples needed	5-6 per object	1 per object
Accuracy (test set)	100% (15/15)	80% (12/15)
Misclassifications	0	3 (all predicted as allen_key)
Objects tested	5	5

Confusion Matrix Observations:

- Hand-crafted matrix showed perfect diagonal- every object correctly classified across all 3 test captures. No off-diagonal entries, meaning no object was confused with any other.

- CNN matrix showed two failure patterns.
  - (I remember, there was a pattern with the misclassification. All the misclassifications were labeled as 'allen-key' when they were tested in a specific orientation.)
  - Comb was misclassified as allen_key twice out of 3 captures, suggesting the comb's position at certain orientations resembled the single allen_key training example in embedding space.
  - Toothpaste was misclassified as allen_key once out of 3 captures, similarly suggesting orientation-dependent confusion.
  - Scissor, swiss_knife, and sleep_mask all achieved perfect 3/3 classification, showing CNN was reliable for visually distinctive objects.

Qualitative Comparison:

- Hand-crafted features required deliberate engineering- four features (percent filled, aspect ratio, Hu1, Hu2) were carefully designed to capture shape properties invariant to translation, scale, and rotation. Multiple training captures per object were needed to ensure robustness across orientations.
- The features are interpretable and computationally lightweight, and in this case cleanly separated all 10 objects.

- CNN embeddings required almost no engineering effort- one forward pass through ResNet18 per object with no feature design decisions.
- However, the one-shot nature means the single training capture must be representative of the full range of object appearances.
- The three misclassifications all predicted allen_key, likely because the training capture for allen_key shared orientation similarity with comb and toothpaste at certain angles, and the axis alignment in ***prepEmbeddingImage*** did not fully normalize these appearances.

The 2D PCA plots of the embeddings (mentioned in the next ‘Extension’ section) further support this- across two independent runs the spatial arrangement of objects shifted, with some objects clustering closely in one run and separating in another. Objects that appeared close in the plot corresponded directly to the misclassifications observed in the confusion matrix.

My finally thoughts:

Hand-crafted features outperformed CNN here not because they are inherently superior, but because they had more training examples and the four chosen features happen to cleanly fingerprint these specific objects by shape. With more CNN training examples the accuracy would likely match or exceed hand-crafted performance, as CNN embeddings capture richer appearance information including texture and fine structure that hand-crafted features cannot.

Extension: CNN Embedding Separability Visualization via PCA Projection

Summary:

What I did:

- After collecting one CNN embedding per object using the ***t*** key, all 10 embeddings (each a 512D vector from ResNet18's second-to-last layer) were stacked row-wise into a single matrix using ***cv::PCA*** with ***numComponents=2***.
- This finds the two directions of maximum variance across all 10 embeddings- the two eigenvectors that best spread the data apart in 2D.
- Each 512D embedding was then projected onto these two eigenvectors using ***pca.project()***, reducing it to a single 2D point.
- The resulting points were normalized to fit a 600x600 canvas with a 60-pixel margin, and each object was drawn as a filled circle with its label in a distinct color.
- The plot is triggered by pressing ***v*** and opens as a live OpenCV window called "Embedding Plot" alongside the existing pipeline windows.

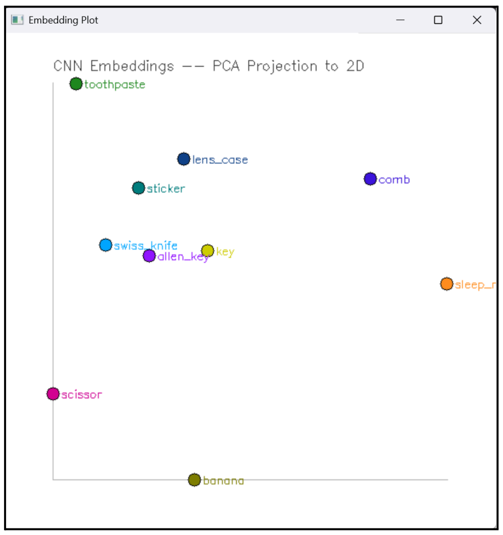
Why I chose PCA for visualization:

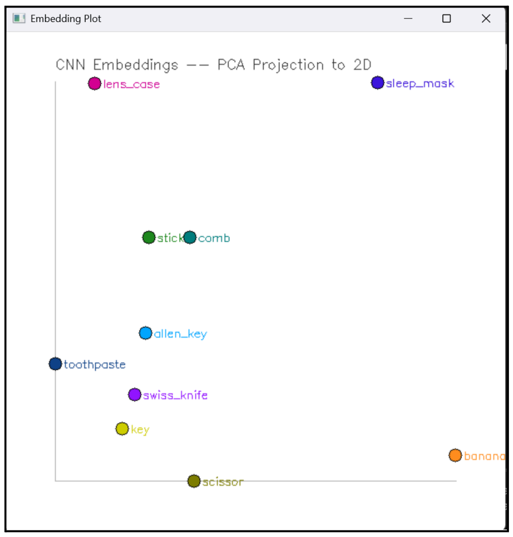
- The CNN embedding space is 512-dimensional- impossible to visualize directly.
- PCA finds the two axes of maximum variance, meaning the projection preserves as much of the original separation between objects as possible in 2D.
- It is not a perfect representation of the full 512D distances but gives the best possible 2D approximation.

What this shows:

- The spatial arrangement of objects in the plot reveals how well the CNN embedding space separates the 10 object categories.
- Objects that are far apart are easily distinguishable by the classifier.
- Objects that cluster together are likely to be confused with each other under nearest neighbor classification.

Demo Images:





Observations:

Run 1 (Graph #1):

- Allen_key, swiss_knife, and key form a tight cluster in the center. These three metallic elongated objects have visually similar embeddings.
- Toothpaste is isolated top left yet still misclassified as allen_key once, suggesting the tight tool cluster in the center was pulling nearby embeddings toward allen_key as the nearest neighbor.
- Comb is isolated far right yet still misclassified as allen_key twice, reinforcing that the 2D projection does not fully capture 512D distances and the misclassification was driven by the dense tool cluster.
- Sleep_mask is isolated far right, consistent with its perfect 3/3 classification.
- Scissor is isolated bottom left, consistent with its perfect classification.
- Banana sits at bottom center, consistent with correct classification.
- Lens_case and sticker appear close together in the upper center, both compact symmetric objects sharing appearance properties in embedding space.

Run 2 (Re-run from scratch, Graph #2):

- Sticker and comb are now nearly overlapping in the center, in contrast to Run 1 where they were well separated, showing how a single different training capture dramatically shifts embedding position.
- Allen_key now sits just below the sticker/comb cluster rather than forming a tight tool cluster, meaning the tool cluster from Run 1 has completely dissolved.
- Lens_case is now isolated top left and sleep_mask isolated top right, both remaining well separated from everything else across both runs.
- Toothpaste is now isolated far left, much better separated than in Run 1.
- Banana remains isolated far right, consistently well separated across both runs.
- Scissor remains isolated at the bottom, consistently distinctive across both runs.
- Key and swiss_knife are now spread out in the lower center rather than clustering with allen_key.

Key cross-run observation:

- Sleep_mask, scissor, and banana remained well isolated and distinctive across both runs. These are the most stable objects in CNN embedding space, likely because their appearance is unique enough that even slightly different training captures produce consistently separated embeddings.
- Everything else shifted significantly between runs, most notably the tool cluster of allen_key, swiss_knife, and key which was tightly grouped in Run 1 but dissolved in Run 2, and sticker/comb which were separated in Run 1 but nearly overlapping in Run 2.
- This instability directly confirms that one-shot embeddings are highly sensitive to the specific training capture orientation, and the 2D PCA plot serves as a reliable visual predictor of which objects the classifier will confuse.

Reflections:

- This project represented a significant step forward from my previous computer vision projects, moving into building a complete end-to-end object recognition system that combines classical computer vision techniques with modern deep learning.

- Implementing the thresholding and morphological filtering entirely from scratch deepened my understanding of how binary images are constructed and manipulated at the pixel level, and gave me genuine appreciation for why higher-level OpenCV functions are designed the way they are.

- Working through the full recognition pipeline, from raw pixels to a classified label, gave me a clear mental model of how each stage depends on the one before it, and how a failure or imperfection at any stage, such as thresholding artifacts propagating into feature computation, can affect the final result.

- Designing and computing hand-crafted features that are truly invariant to translation, scale, and rotation was one of the most intellectually satisfying parts of the project, as it required thinking carefully about what geometric properties of a shape remain stable under transformation.

- Integrating the ResNet18 CNN embedding pipeline alongside the hand-crafted system gave me hands-on experience with one-shot classification and a concrete understanding of how deep learned representations differ from manually engineered ones in terms of both expressiveness and setup cost.

- Implementing the PCA-based embedding visualization was a rewarding extension that connected ideas from linear algebra directly to a practical computer vision application, making the abstract concept of high-dimensional embedding spaces visually interpretable.

- Building the multi-object detection and classification extension pushed me to think about the system more generally, moving from a single-region pipeline to one that handles multiple independent regions simultaneously in real time, each going through the full feature extraction and classification process.

- This project as a whole reinforced my growing passion for computer vision and my drive to go beyond the minimum requirements, and it gave me confidence that I can design and implement a complete recognition system from the ground up.

Acknowledgement:

- I would like to express my sincere gratitude to **Professor Bruce Maxwell** for his exceptional teaching and patience in addressing my most fundamental questions, explaining concepts multiple times until they clicked.

- His consideration in granting me a few days extension given what I was going through, allowed me to explore the material thoroughly without rushing, which directly enabled the depth and creativity of the extensions presented in this report.

- I am grateful to the **course teaching assistants** for their consistent support and readiness to help whenever I encountered difficulties. Their guidance on debugging issues and clarifying implementation details was invaluable throughout the project development process.

- The **OpenCV documentation** served as an essential reference for understanding function parameters, image processing techniques, and best practices for efficient implementation.

- **YouTube tutorials** provided supplementary explanations and visual demonstrations that complemented course materials, particularly helpful for grasping complex concepts like frequency domain transformations and optimization techniques.

- Special thanks to **Claude** for being available 24/7 to help me ideate, refine half-formed concepts, explore implementation possibilities, understand new techniques, and provide subject matter expertise that accelerated my learning process throughout this project.
