Assignment 4: Classical CV

Ram Singhal

January 12, 2026

1 Exercise 1: Spot the Mallet

1.1 Objective

The primary objective of this exercise was to design a robust computer vision pipeline capable of segmenting an orange mallet from video frames. The key challenge was ensuring consistent segmentation across varying lighting conditions (shadows vs. direct sunlight) and dynamic backgrounds (sand, rocks, and grass).

1.2 Pipeline

The solution was implemented using a processing pipeline consisting of four major stages: Preprocessing, Color Space Conversion, Thresholding, and Morphological Refinement.

1.2.1 Color Space Selection

We adopted the HSV (Hue, Saturation, Value) color space because:

- Hue (H): Isolates the color. The mallet consistently registered in the orange range regardless of illumination.
- Saturation (S): Critical for background separation. The sand (yellow/brown) has low saturation. The mallet has high saturation.
- Value (V): Allowed us to filter out noise without losing the mallet in shadowed regions.

1.2.2 Preprocessing

To reduce noise and smoothen the image, a Gaussian Blur with a 5×5 kernel was applied before segmentation. This smoothed the image, ensuring that the resulting mask don't consist disjoint pixels.

1.2.3 Thresholding Parameters

After multiple trials, the following HSV ranges were choosen:

Parameter	**Min**	**Max**
Hue	5	25
Saturation	90	255
Value	70	255

Table 1: HSV Threshold Values

1.2.4 Morphological Refinement

The raw binary mask contained salt noise (sand speckles) and pepper noise (holes in the mallet). We applied the following morphological operations:

1. **Opening (5×5 Kernel):** Erosion followed by Dilation. This successfully removed small white regions from the background.
2. **Closing (5×5 Kernel):** Dilation followed by Erosion. This filled small holes within the mallet caused by reflections on the mallet surface.

1.3 Observations & Failures

- The mallet was successfully segmented in most of the videos.
- **Limitations:** In some of the videos it segmented a large portion of floor also where there is some specific reflection or floor is dirty. Also it is not much good at the boundaries of mallet.

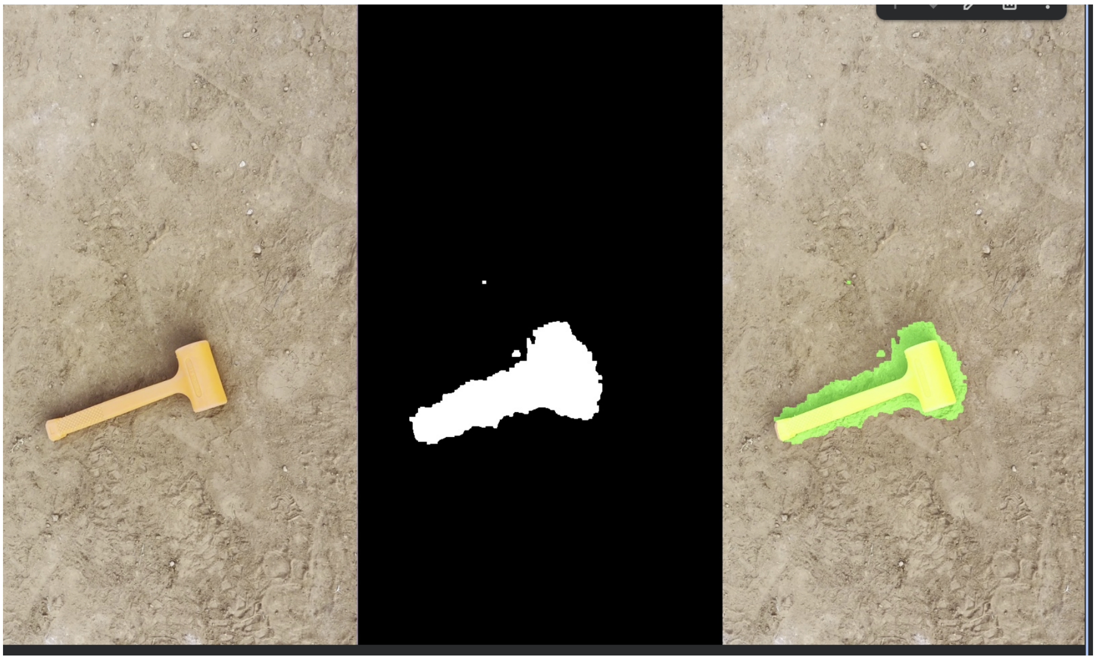

Figure 1: Final results

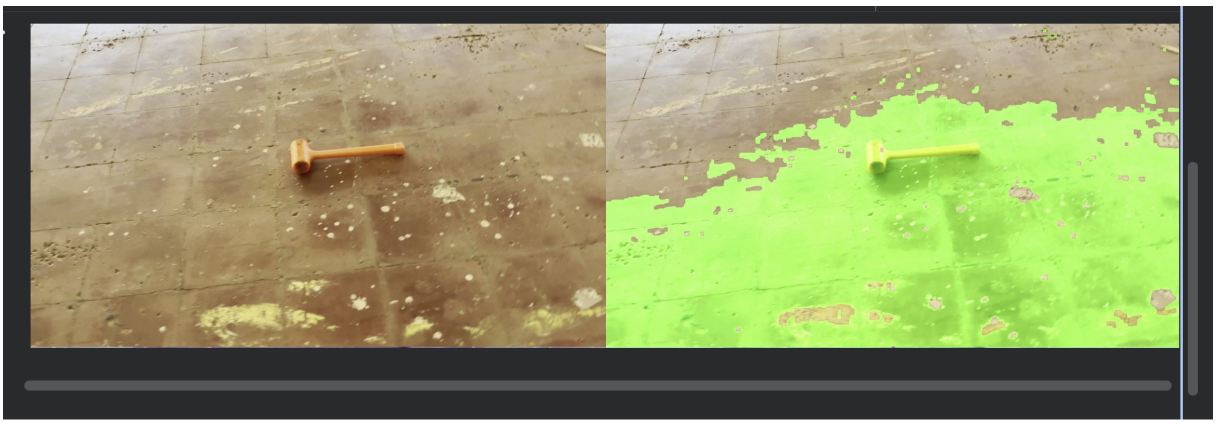

Figure 2: Failure case

2 Exercise 2: Cone Detection

2.1 Objective

The objective of this exercise was to design a robust, end-to-end pipeline to detect and localize traffic cones in diverse environments. The system needed to handle challenges such as multiple cones, varying backgrounds (grass, road), lighting variations (shadows, highlights), and partial occlusions.

2.2 Pipeline Architecture

Our approach evolved through two iterations. The initial attempt utilized edge detection and Hough transforms, but the final robust pipeline focused on color segmentation in the LAB space combined with a custom contour grouping algorithm.

The final pipeline consists of the following stages:

1. **Preprocessing:** The input image is smoothed using a Gaussian Blur to suppress high-frequency noise before color conversion.

2. **Color Space Conversion (LAB):** The image is converted from BGR to the LAB color space. Unlike RGB, LAB separates lightness (L) from color components (a and b), making the detection more robust to lighting changes/shadows compared to standard RGB thresholding.

3. **Color Segmentation:** We apply thresholding on the A channel. The specific range allows us to isolate the orange hue of the cones effectively while ignoring the background.

4. **Morphological Refinement:** To clean the binary mask, we apply:

   - **Opening:** Removes small noise speckles (false positives).
   - **Closing:** Fills small holes within the detected cone regions.

5. **Contour Grouping & Convex Hull:** A key challenge was that a single cone often appeared as multiple fragmented contours ("strip" problem). To solve this, we implemented a custom clustering algorithm:

   - Centroids of all valid contours are calculated.
   - Contours whose centroids are within a distance of 100 pixels are grouped together.
   - A single Convex Hull is computed for each group, creating a unified bounding box for the cone.

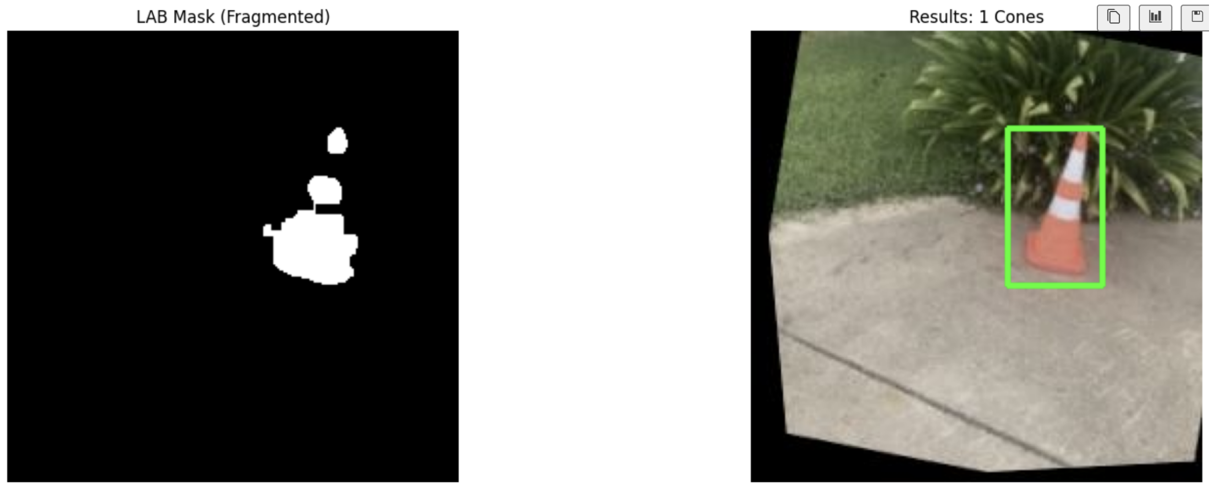

Figure 3: Final Pipeline Results: The left image shows the fragmented LAB mask (before grouping). The right image shows the final detection after morphological refinement and grouping.

2.3 Justification of Choices

2.3.1 Why LAB Color Space?

In our first version, we experimented with Gamma correction and CLAHE. However, we found that converting to LAB space provided better separability. The 'A' channel (for Orange and Red cones) and in some cases 'B' channel (for Yellow cones) are particularly effective for isolating the cones from background regardless of the lighting intensity (L-channel).

2.3.2 Edge vs. Region-Based Detection

Our initial implementation (v1) attempted to use Canny edge detection and Probabilistic Hough Lines to find the structural sides of the cones. While conceptually sound, this approach was prone to false positives from background clutter (e.g., fences, pavement cracks). The region-based approach (Color Segmentation) proved significantly more robust for this specific dataset.

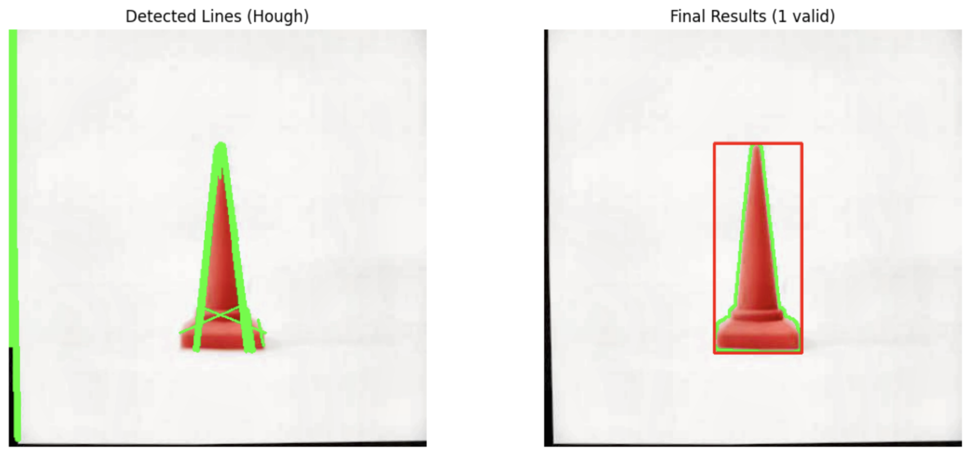

Figure 4: Initial Approach (v1): Edge detection - success case

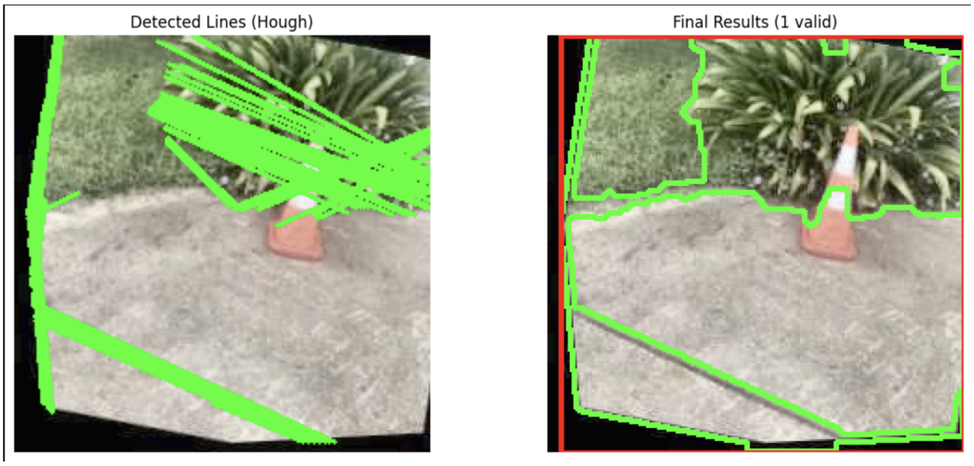

Figure 5: Initial Approach (v1): Edge detection - failure case

2.4 Observations and Failures

- **Strip failure:** The grouping algorithm handled many cases where the cone's strips caused the mask to break into top and bottom halves. By clustering centroids, these halves were correctly identified as a single object in many cases, but it still fails in few cases.
- **Color:** The current pipeline relies heavily on the specific orange color of the cones. It may fail if:
  - There are other orange objects in the scene (e.g., orange clothing).
  - The lighting is extremely tinted (e.g., strong sunset), significantly shifting the a, b values.
- **Multiple cones:** It is not very accurate in case of multiple cones. When cones are close, the grouping function considers them a single cone (see Figure 6).

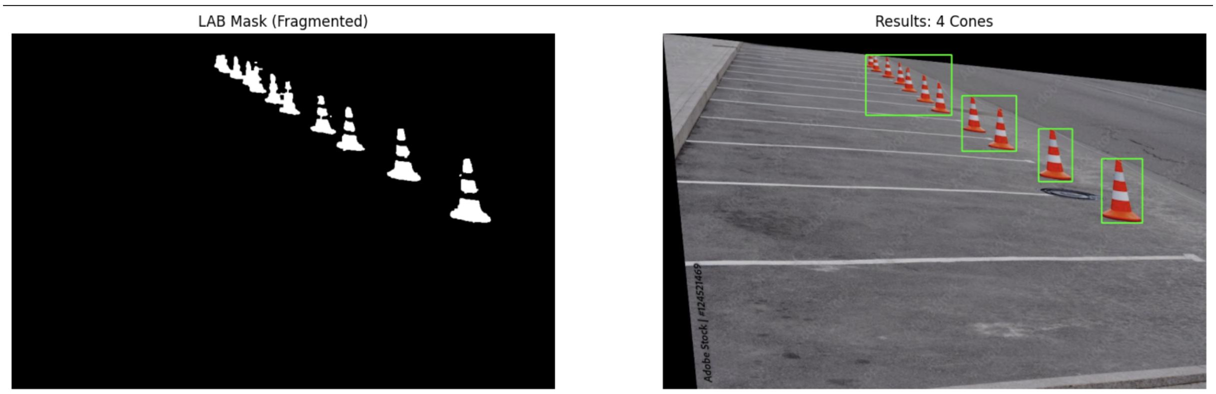

Figure 6: Failure Case: The grouping threshold (100px) causes two distinct cones in close proximity to be merged into a single bounding box.