Image Processing

Mars Rover Tech Team

Vedant Bhageria

2026-01-09

1 Introduction

OpenCV (Open Source Computer Vision Library) is a widely used library for image and video processing. Image processing treats an image as numerical data, essentially a matrix of pixel values, and applies mathematical operations to analyze and modify it. Since computers don't understand objects directly, they rely on properties like color intensity, contrast, edges, and shape. OpenCV provides tools to work with these properties, such as color space transformations (RGB, HSV, LAB), filtering to reduce noise, thresholding to isolate regions of interest, morphological operations to clean masks, and contour detection for object localization.

In our assignments, we focused on building end-to-end classical computer vision pipelines. One task involved detecting and segmenting a colored mallet from an image. This required carefully choosing an appropriate color space, isolating the relevant channel, applying thresholding to separate the mallet from the background, and refining the result using morphological operations to remove noise and fill the object. The emphasis was on understanding how low-level pixel decisions affect the final segmentation.

The second task extended these ideas to a harder problem: detecting and localizing cones across multiple images with varying lighting conditions, backgrounds, scales, and even multiple cones in a single frame. Here, the challenge was to design a single robust pipeline that generalizes well, moving from pixel-level processing to contour extraction and geometric reasoning to identify valid cone candidates. No machine learning was used; the solution relied entirely on classical image processing concepts and careful parameter design.

2 MALLET DETECTION

2.0.1 Problem Context

The objective of this task was to detect and segment a colored mallet from video frames using **classical computer vision techniques only**. The main difficulty arose from varying lighting conditions, background colors that partially overlapped with the mallet's color, and the mallet appearing at different scales and contrasts across videos. Multiple pipelines were explored to understand how different preprocessing, color spaces, and thresholding strategies affect robustness.

2.1 Pipeline 1: Contrast Enhancement → HSV → Adaptive Thresholding

2.1.1 Method

1. Applied global contrast enhancement using `alpha = 1.75` and `beta = -200`.
2. Converted the image to the HSV color space.
3. Used adaptive thresholding to segment the mallet.
4. Applied morphological operations: close followed by open, to clean noise.

2.1.2 Outcome

The resulting binary mask primarily captured **edges of the mallet** instead of a filled region.

2.1.3 Reason for Failure

- Aggressive contrast enhancement amplified local intensity differences, especially along edges.
- Adaptive thresholding responds to local contrast, so it emphasized boundaries rather than homogeneous regions.
- HSV channels, particularly V, became unstable after contrast stretching, causing interior regions of the mallet to be split across thresholds.

- Morphological operations cannot recover a solid object if the core region is never segmented in the first place.

2.2 Pipeline 2: LAB Color Space Thresholding with Fixed Ranges

2.2.1 Method

1. Converted the image to LAB color space.
2. Applied a median blur to reduce salt-and-pepper noise.
3. Thresholded each channel independently using:
   - `L [200, 255]`
   - `A [130, 180]`
   - `B [145, 190]`
4. Combined masks.
5. Applied a long morphological chain: erode → open → dilate → open → dilate → close → open.

2.2.2 Outcome

- This pipeline **successfully detected the mallet in many cases**.
- However, in some videos, the mallet partially blended into the background.

2.2.3 Reason for Partial Failure

- LAB separates luminance (L) from chromaticity (A and B), which makes color-based segmentation more stable.
- However, when background objects shared similar A and B values (especially under warm lighting), the fixed thresholds were no longer discriminative.
- The L channel threshold was sensitive to illumination changes, causing either loss of mallet regions or inclusion of background highlights.

Figure 1: Original Image

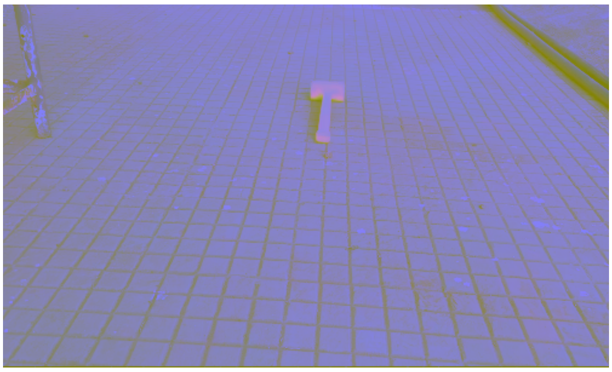

Figure 2: LAB image after Pre-Procesing

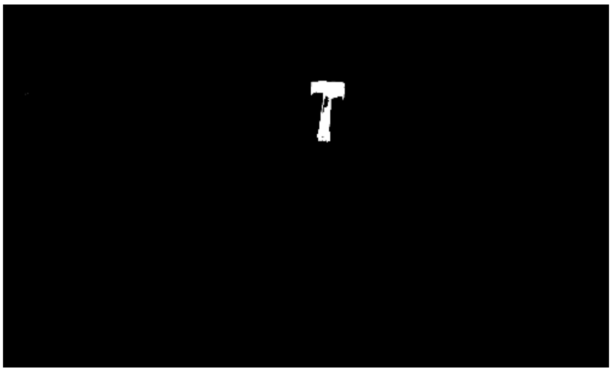

Figure 3: Thresholded Image

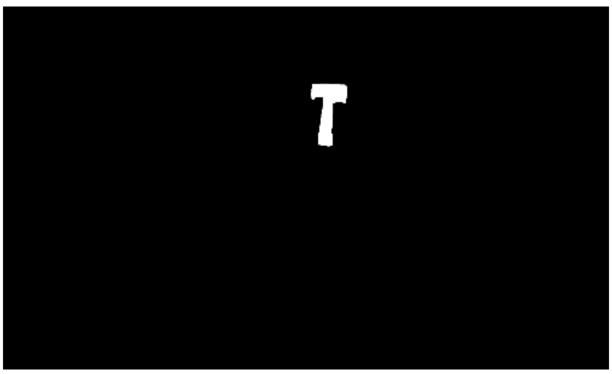

Figure 4: After Refinement

2.3 Pipeline 3: Combined LAB and HSV Thresholding

2.3.1 Method

1. Converted the image to both LAB and HSV.
2. Used LAB thresholding as the primary mask.
3. Refined it using the most suitable HSV channel to suppress background leakage.
4. Applied the same morphological operations as Pipeline 2.

2.3.2 Outcome

- Reduced background blending compared to Pipeline 2.
- Often failed to capture the **entire mallet**.

2.3.3 Reason for Failure

- Combining multiple color spaces made the pipeline overly restrictive.
- Regions of the mallet that slightly deviated in either LAB or HSV were removed.
- The pipeline became **too specific to certain lighting conditions**, reducing generalization.
- Intersection of masks trades false positives for false negatives, which caused incomplete segmentation.

2.4 Pipeline 4: Histogram Equalization on L + Adaptive Thresholding

2.4.1 Method

1. Converted the image to LAB.
2. Applied histogram equalization on the L channel.
3. Used adaptive thresholding on the equalized L channel.
4. Followed by morphological cleaning.

2.4.2 Outcome

Significant bleeding of the mallet region into the background.

2.4.3 Reason for Failure

- Histogram equalization globally redistributes intensities, boosting contrast in both foreground and background.
- Background textures gained similar luminance profiles to the mallet.
- Adaptive thresholding then interpreted background regions as part of the object.
- This pipeline over-amplified illumination variations instead of suppressing them.

2.5 Pipeline 5: Nonlinear Channel Scaling Based on Mallet Color Statistics

2.5.1 Observation

Across experiments, a **consistent range of L, A, and B values** was observed where the mallet appeared in all videos. This suggested that instead of tighter thresholding, the problem could be reframed as **contrast manipulation**: compress background values and expand mallet values.

2.5.2 Method

1. Quadratic Expansion Around the Mean l = np.clip((l-155)/abs(l-155)*(l-155)**2 + 155, 0, 255).astype(np.uint8)

a = np.clip((a-167)/abs(a-167)*(a-167)**2 + 167, 0, 255).astype(np.uint8)

b = np.clip((b-227)/abs(b-227)*(b-227)**2 + 227, 0, 255).astype(np.uint8)

Issue: This aggressively amplifies deviations from the mean. While it separates the mallet strongly, it also explodes noise and small background variations, making the transformation numerically unstable near the mean and highly sensitive to lighting changes.

2. Exponential Scaling (Most Effective in Practice) l = np.clip(1.007**(l-227) * l, 0, 255).astype(np.uint8)

a = np.clip(1.007**(a-155) * a, 0, 255).astype(np.uint8)

b = np.clip(1.007**(b-167) * b, 0, 255).astype(np.uint8)

Outcome: This method provided strong but controlled contrast expansion around the mallet's average color. In several videos, the mallet became clearly separable at the preprocessing stage itself, sometimes requiring little to no additional thresholding.

Limitations: The method is sensitive to the chosen base (1.007) and mean values. Poor tuning can still lead to saturation, and generalization beyond tested videos has not yet been fully validated.

3. Linear Scaling Away from the Mean l = np.clip(l + (l-227)*3, 0, 255).astype(np.uint8)

a = np.clip(a + (a-155)*3, 0, 255).astype(np.uint8)

b = np.clip(b + (b-167)*3, 0, 255).astype(np.uint8)

Outcome:

Outcome: This approach produced moderate contrast expansion and behaved predictably, but the separation between mallet and background was weaker compared to the exponential method. In difficult scenes, additional threshold tuning was still required.

(Alternative quadratic and linear expansion functions were also tested.)

2.5.3 Outcome

- **In several videos, the mallet became almost separable at the preprocessing stage itself.**
- **Thresholding became trivial or unnecessary in some cases.**

2.5.4 Limitations

- **The method has not yet been tested extensively outside the preprocessing images used for tuning.**
- **Strong dependence on assumed average mallet color values.**
- **Risk of overfitting to specific lighting conditions if not carefully regularized.**

Figure 5: orginal image

Figure 6: preprocessed image

2.6 Key Takeaways

- **Color space choice matters, but contrast behavior matters more.**
- **Fixed thresholds fail under illumination changes.**
- **Adaptive methods can overreact to texture.**
- **LAB is effective, but only when luminance is controlled.**
- **Manipulating the data distribution itself (instead of just thresholding) showed the most promise for robust segmentation.**