Probabilistic Color Segmentation and Depth Estimation

Alif Ilham Madani

February 13, 2026

1 Introduction

Autonomous systems such as robots require robust perception systems to identify obstacles and path boundaries. This report details the implementation of a vision-based system designed to segment orange traffic cones from images and estimate their distance relative to the camera. The system utilizes a probabilistic color segmentation model based on multivariate Gaussian distributions and a polynomial regression model for distance estimation.

2 Problem Statement

The objective is to detect a specific traffic cone in a series of test images and estimate its physical distance. The input consists of RGB images and annotation files containing ground truth polygons. The system must:

1. Segment the cone pixels from the background.
2. Compute a bounding box for the detected cone.
3. Estimate the physical distance based on the bounding box dimensions.

3 Approach

3.1 Color Space Selection

We utilized the **YCrCb** color space for segmentation. Unlike RGB, where luminance and chromaticity are coupled, YCrCb separates luminance (Y) from chrominance (Cr, Cb). Since traffic cones are defined by their distinct orange hue rather than their brightness, this separation allows the model to generalize better across varying lighting conditions (e.g., shadows vs. bright sunlight).

3.2 Data Annotation

To create the ground truth required for supervised learning, we utilized the Computer Vision Annotation Tool (CVAT). For each image in the training set, we manually drew

polygonal annotations around the visible traffic cones. These annotations were exported in XML format, which were then parsed to generate binary masks. These masks served as the pixel-wise labels ($y = 1$ for cone, $y = 0$ for background) used to train the Gaussian models.

3.3 Probabilistic Color Segmentation

We modeled the color distribution of the pixels using a multivariate Gaussian model. Since pixel-wise ground truth labels were available from the CVAT annotations, we employed a supervised learning approach using maximum likelihood estimation (MLE) [1].

3.3.1 Mathematical Formulation

We assume the color vector $\mathbf{x} \in \mathbb{R}^3$ (representing Y, Cr, Cb components) for a given class k is drawn from a multivariate normal distribution:

$$P(\mathbf{x}|C_k) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^3|\boldsymbol{\Sigma}_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right) \tag{1}$$

where $k \in \{\text{cone}, \text{background}\}$. We utilized a single Gaussian component per class.

3.3.2 Parameter Estimation (Training)

We estimated the model parameters directly from the training data using MLE. For a dataset with N_{total} pixels, where N_k pixels belong to class k:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_i \tag{2}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \tag{3}$$

The class prior $P(C_k)$ is estimated as the fraction of pixels belonging to class k:

$$P(C_k) = \frac{N_k}{N_{total}} \tag{4}$$

We assumed a full covariance matrix structure (non-diagonal), allowing the model to capture correlations between color channels (e.g., the relationship between Cr and Cb in orange pigments).

3.3.3 Inference (Classification)

To segment a new image, we assign each pixel to the class k that maximizes the posterior probability $P(C_k|\mathbf{x})$. Applying Bayes' rule and taking the logarithm (to avoid numerical underflow), the decision rule becomes:

$$\hat{y} = \underset{k}{\operatorname{argmax}} \left(\ln P(\mathbf{x}|C_k) + \ln P(C_k)\right) \tag{5}$$

Substituting the Gaussian PDF, the discriminant function $g_k(\mathbf{x})$ for class k is:

$$g_k(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln P(C_k) \tag{6}$$

To optimize inference speed, we quantized the color space to 6-bits per channel and pre-computed a lookup table mapping every possible color triplet to the class label $\hat{y}$.

3.4 Distance Estimation Model

We modeled the relationship between the cone's bounding box height (h, in pixels) and the physical distance (d) using a polynomial regression on the logarithm of the distance. The pinhole camera model suggests an inverse relationship ($d \propto 1/h$) [2], which is well-approximated by an exponential decay in our feature space.

The model is defined as:

$$\ln(d) = \sum_{j=0}^{M} w_j h^{M-j} \quad \Longrightarrow \quad d = \exp\left(\sum_{j=0}^{M} w_j h^{M-j}\right) \tag{7}$$

We selected a polynomial degree of $M = 4$ to capture non-linear lens distortions, solving for coefficients w_j via least squares.

3.5 Implementation Overview

The algorithm pipeline is summarized below:

Algorithm 1 Training and Inference Pipeline

procedure TRAIN($Images, Masks, Distances$)
    Convert all $Images$ to YCrCb space.
    for each class $k \in \{Cone, Background\}$ **do**
        Extract pixels $\mathbf{X}_k$ where $Mask == k$.
        Compute $\boldsymbol{\mu}_k$ (Eq. 2) and $\boldsymbol{\Sigma}_k$ (Eq. 3).
        Compute prior $P(C_k) = \frac{N_k}{N_{total}}$.
    end for
    Pre-computation: Generate Look-Up Table (LUT) for all 2^{18} colors using Eq. 5.
    Regression: Fit polynomial coefficients $\mathbf{w}$ for Height $\rightarrow$ ln(Distance).
end procedure

procedure INFERENCE($Image$)
    Convert $Image$ to YCrCb.
    $Mask \leftarrow$ Apply LUT to $Image$ (Pixel-wise classification).
    $Regions \leftarrow$ ConnectedComponents($Mask$).
    for each $region \in Regions$ **do**
        if $region.area >$ MinArea **and** $region.ratio \approx$ ExpectedRatio **then**
            $h \leftarrow$ BoundingBoxHeight($region$).
            $d_{pred} \leftarrow \exp(\text{PolyVal}(\mathbf{w}, h))$.
            Output Bounding Box and d_{pred}.
        end if
    end for
end procedure

The thresholds for minimum area and expected ratio margin is based on trial and error on the detection using the training dataset. We calculate the aspect ratio based on the given data on the actual height and width of the cone (43.18 cm / 19.05 cm). The minimum area is set to 200 pixel area, and the aspect ratio margin is set at 0.4, meaning we consider regions with 1 ± 0.4 times of the aspect ratio as valid cones. The core image processing routines and connected component analyses were implemented utilizing standard computer vision libraries [3, 4].

4 Results and Discussion

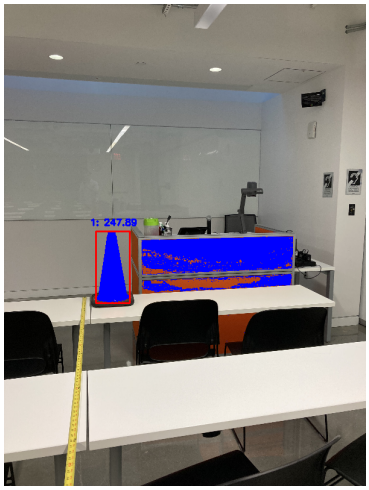
4.1 Qualitative Analysis on Training Set

The proposed color segmentation and bounding box estimation pipeline was evaluated on the training dataset. As shown in Figure 1, the system demonstrated robustness in identifying traffic cones across various environments, including cluttered indoor scenes and outdoor settings.

A key observation from the results is the importance of the post-processing stage. As seen in Figure 1a, the Gaussian color model produced false positives for the red chairs in the foreground due to the lighting conditions, which make the chairs' color look similar to the cone's. However, the aspect ratio and area filtering steps successfully rejected these artifacts, placing the bounding box correctly only on the cone. Similarly, in Figure 1c, despite significant noise on the back wall, the system isolated the cone on the desk.

(a) Train Image #6

(b) Train Image #15



(c) Train Image #20

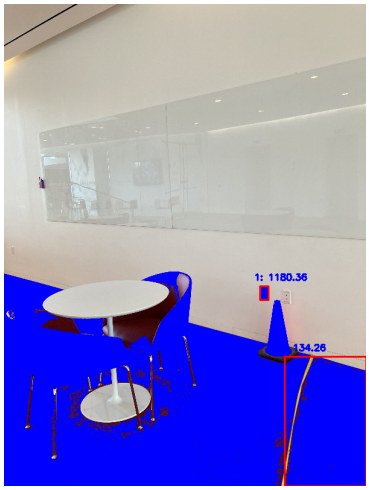
Figure 1: Successful results overlaid with color segmentation and the final bounding box.

4.1.1 Failure Modes

Figure 2 highlights the limitations of a purely color-based approach. In Figure 2a, the cone is obstructed by a metal rod, which makes the model detects the cone partially. The area and aspect ratio filter only successfuly captures the top part of the cone. In Figure 2b, due to the lighting condition, the red carpet color looks very similar to the orange cone in the YCrCb space. The segmentation mask (shown in blue) covered the entire floor, creating a connected component too large and geometrically distinct to be filtered

correctly. In Figure 2c, the desk's color looks very similar to the cone's, which makes the model detect part of the desk as being "cone" class pixels. The post-processing method also wrongly detects some part of the desk as a cone.

(a) Train Image #2



(b) Train Image #7

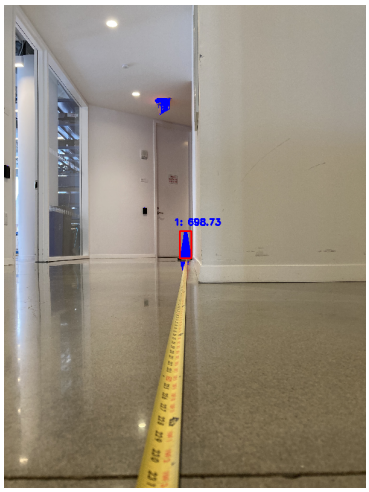
(c) Train Image #19

Figure 2: Unsuccessful color segmentation on difficult scenes.

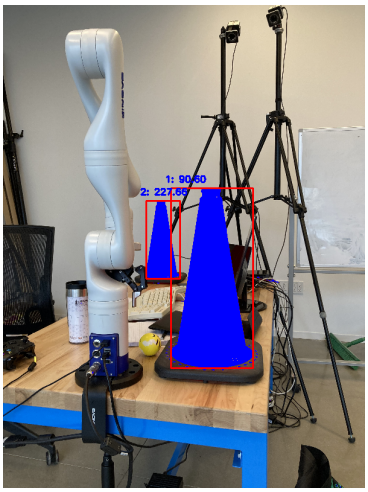
4.2 Performance on Test Dataset

The model was subsequently applied to the unseen test dataset. Figure 3 illustrates the system's ability to generalize to new environments. Figure 3a demonstrates successful color segmentation in an outdoor environment with high percentage of occlusion. However, the post-processing part did not capture this since the aspect ratio in this image is larger than the threshold. The system successfully detected cones at varying distances. In Figure 3b, the cone is positioned far down a hallway (estimated distance ≈ 699 cm), yet the segmentation remained coherent enough to generate a valid bounding box. In Figure 3c, the system managed to detect multiple cones by successfully segmenting the colors and analyzing the regions.

(a) Test Image #1



(b) Test Image #4



(c) Test Image #5

Figure 3: Performance on the test dataset.

4.3 Distance Estimation Accuracy

The 4th-degree polynomial regression on the log-distance yielded a low root mean square error (RMSE) of **12.78** cm on the training dataset. The use of $\ln(d)$ linearized the exponential relationship inherent in perspective projection, stabilizing the least-squares solution. However, as the object gets farther, the error grows, while closer objects have lower error.

4.4 Future Improvements

While the single-Gaussian model works for this controlled dataset, a Gaussian mixture model (GMM) with $K > 1$ components per class (trained via EM) [1] would be necessary to handle more complex environments where cones are occluded and the surrounding objects have similar color. The additional regression model can be applied to learn the appropriate aspect ratio and area instead of manually tuning the thresholds. The distance estimation can also be improved by using stereo vision instead of a monovision camera.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

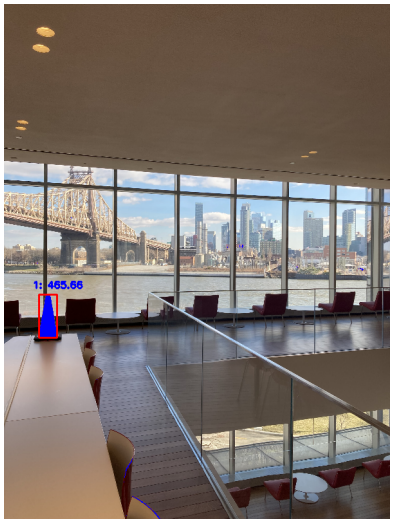
[3] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[4] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

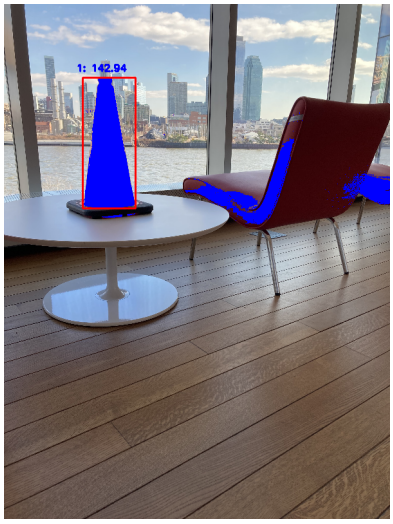
A Appendix: Complete Training Data Results

(a) Train Image #1

(b) Train Image #2



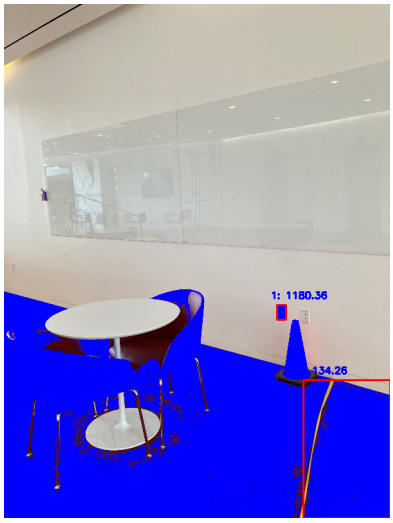
(c) Train Image #3



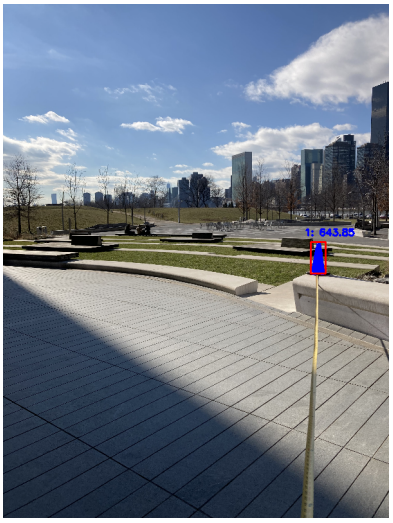
(d) Train Image #4

(e) Train Image #5

(f) Train Image #6



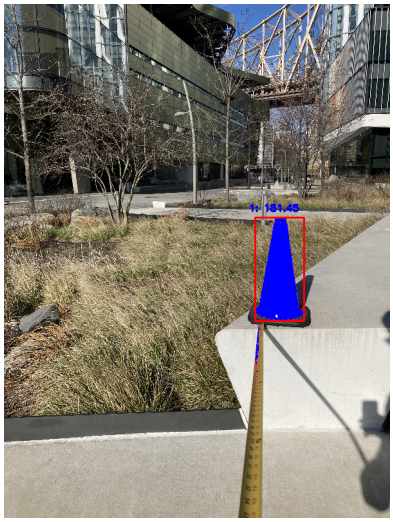
(g) Train Image #7



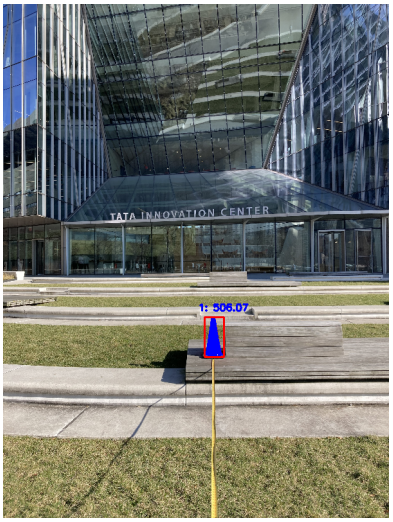
(h) Train Image #10

(i) Train Image #12

Figure 4: Training results (Part 1).

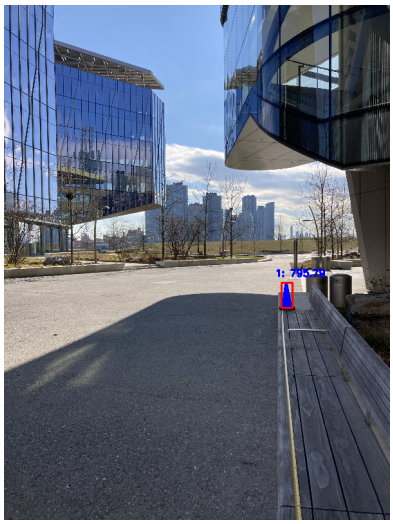


(a) Train Image #13

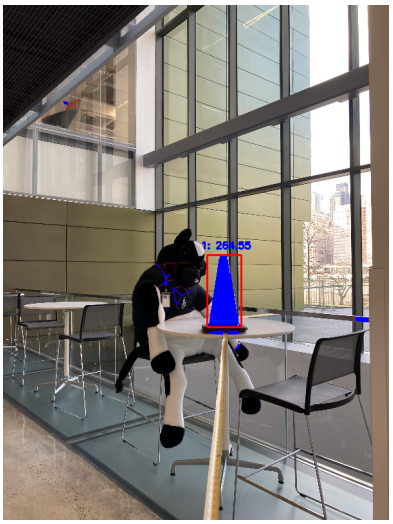


(b) Train Image #14

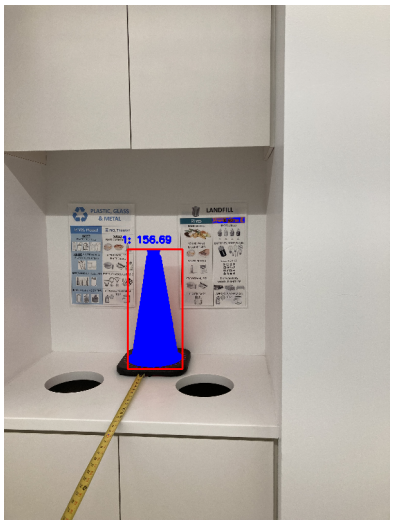
(c) Train Image #15



(d) Train Image #16

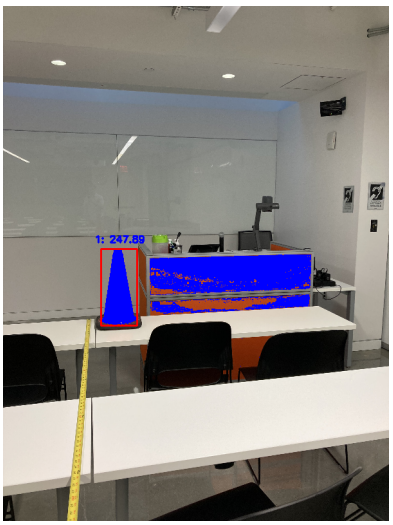


(e) Train Image #17

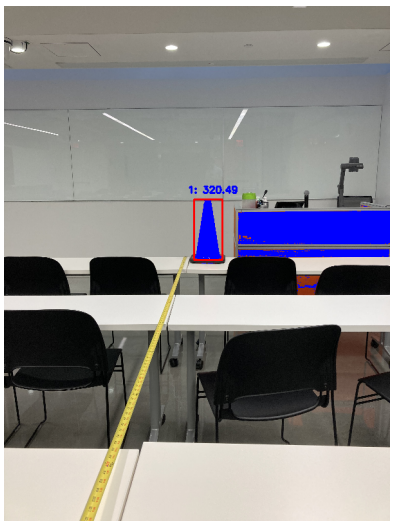


(f) Train Image #18

(g) Train Image #19



(h) Train Image #20

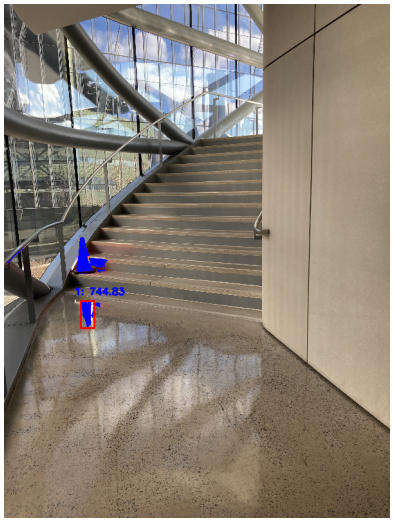


(i) Train Image #21

Figure 5: Training results (Part 2).

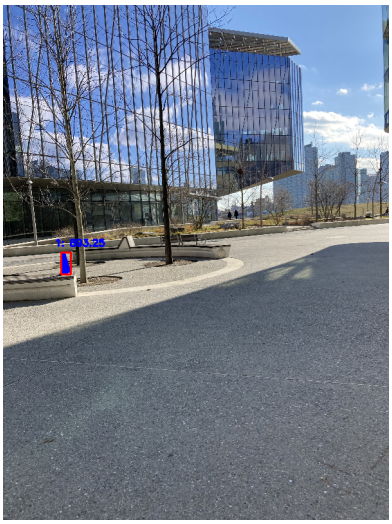
(a) Train Image #22

(b) Train Image #24



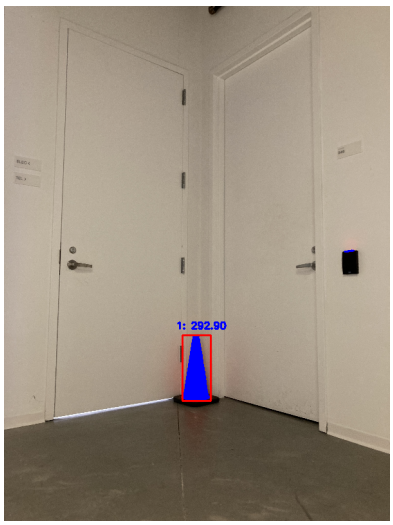
(c) Train Image #25

(d) Train Image #26



(e) Train Image #27

(f) Train Image #29



(g) Train Image #31

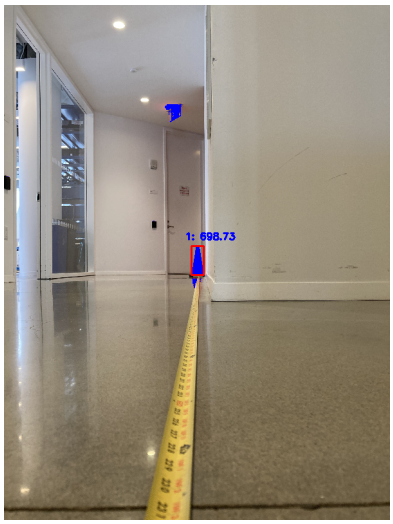
Figure 6: Training results (Part 3).

B Appendix: Complete Test Data Results

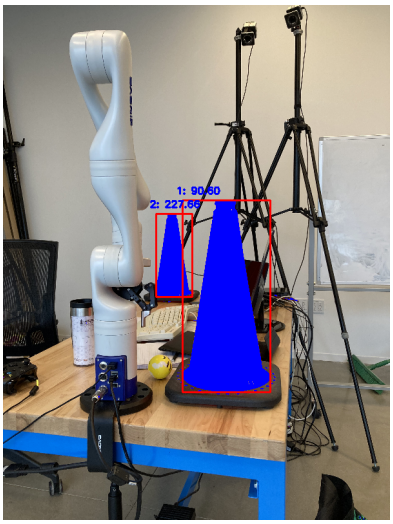
(a) Test Image #1

(b) Test Image #2

(c) Test Image #3



(d) Test Image #4



(e) Test Image #5

Figure 7: Qualitative results on the test dataset.