Pixel Classification and Bin Detection Using Multinomial Logistic Regression

Eskil Berg Ould-Saada
University of California San Diego
La Jolla, CA92093
eskilbo@stud.ntnu.no

1 Introduction

In the latter years, color classification and object detection has been an important part of different vision-based systems wanting to become more and more autonomous, like Tesla's autonomous driving camera seen in Figure 1. The importance of a good classifier and detector especially in the field of autonomous vehicles and traffic is immense. As objects are subject to different lighting, shading and different factors from the eye of the car's cameras, the classification and detection model should be robust to the different forms and colors of objects.

In this project, a classifier made to distinguish the three different colors red, green and blue has been implemented using supervised learning for the training of the model. Furthermore, after hand labeling recycling bin pixels, a bin detector was implemented using the same training model as the pixel classifier. Different techniques using the detected areas' geometry were used to create a bounding box surrounding the objects classified as recycling bins. A recycling bin detector like this could be used to make the process of collecting trash autonomous.

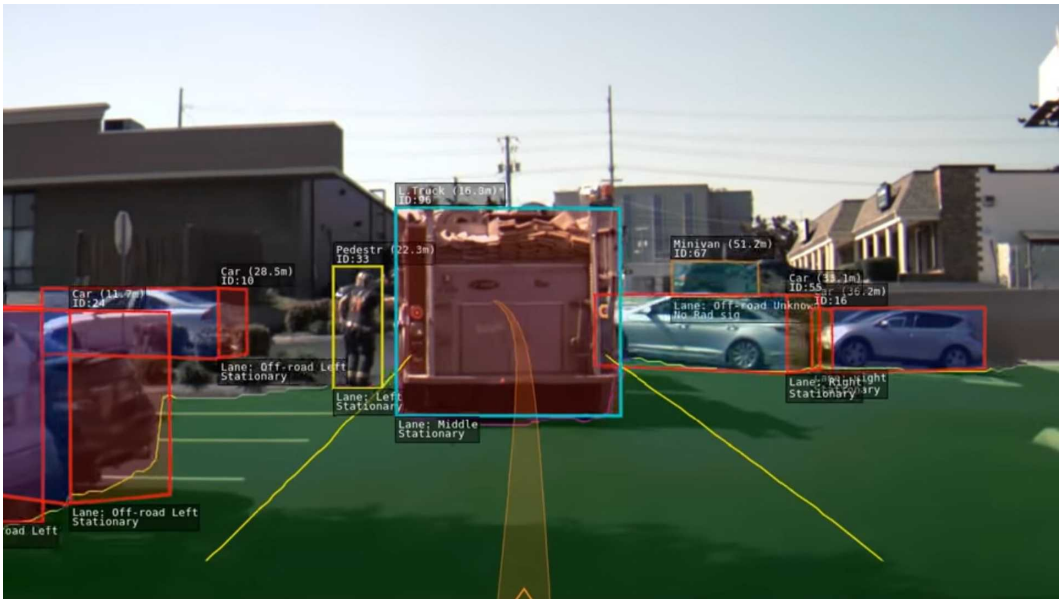

Figure 1: What the Tesla cameras see.

2 Problem Formulation

2.1 Pixel Classification

The pixel classifier takes a set of pixels with corresponding labels indicating which class the pixel belongs to of the different colors in $y = red, green, blue$. The classifier will predict which of these three classes a pixel is most likely of being in. To measure this performance, the accuracy will be computed as

$$Accuracy = \frac{\sum_{i=1}^{N} I(y_i^{'}, y_i)}{N} \tag{1}$$

where I is an indicator function with output set to 1 if the prediction and the true label is the same. The input to the classifier will be an array with dimensions Nx3 which signify the n pixels each with three values (RGB).

2.2 Bin Detection

The bin detector consists of two parts; segmenting the image by making a mask with the colors detected as recycle bin blue as 1 and the rest as 0, and the part creating a bounding box around the objects identified as being recycle bins.

For the image segmentation, the program has an RGB image of size HxWx3 and outputs a mask of the image with ones where the pixel classifier detects a recycle bin blue color. After this classification has been done, the mask will be fed into the bin detector which creates a bounding box around the recycle bins.

The bounding box creator segments the different areas of ones as a region of interest, and uses the region's area and geometry to decide if an area is likely to be a recycle bin. If it is the case, it will draw a bounding box with corners $[x_{min}, y_{min}, x_{max}, y_{max}]$ and compare this to the true bounding boxes in the image using the following function:

$$IoU = \frac{B' \cap B}{B' \cup B} \tag{2}$$

which yields true if $IoU \geq 0.5$, where B and B' are the bounding boxes for the estimate and the true box. In that case, the bounding box is said to be correctly predicted.

3 Technical Approach

3.1 Pixel Classification

To classify the pixels in the best way, a logistic regression model was implemented in order to train the model on the training data. For this part a K-ary logistic regression model was implemented to classify one of the three classes red, blue or green.

The weights for the model were initialized with random values, and a bias term was added to the weight matrix, as well as to the input images, instead of treating the bias as a separate part for simplification.

One-hot encoding of the labels is being used on the true labels of the dataset as K-ary logistic regression uses the Softmax function in Equation 3 for prediction of the k classes, in this case three.

$$y_k^n = \frac{e^{a_k^n}}{\sum_{k'} e^{a_{k'}^n}} \tag{3}$$

The weights are updated using stochastic gradient descent on mini-batches of the data. What this means is that it runs a forward pass of a subset of data points and computes the gradient and updates the weights using the following update rule:

$$w_{t+1} = w_t + \alpha(\sum_{i_1}^{N}(\mathbf{y_{onehot}} - \mathbf{s}(\mathbf{w^T x_i}))x_i^T) \tag{4}$$

where $\mathbf{s}$ is the Softmax function predicting the labels.

To evaluate the model, a forward pass was used by matrix multiplying the weights with the test images $\mathbf{X}$, and do `argmax` of the resulting prediction, and then comparing the output to the true labels, resulting in an accuracy out of 100%.

3.2 Bin Detection

3.2.1 Hand labeling data

The first thing that was done for the bin detection part of the project was to use `roipoly` as in Figure 2 to hand label regions of recycle bin blue pixels and other pixels to create the two classes $y = binblue, notbinblue$ to be used in the training of the model. The important part was to label different shades and variations of the blue color to make the classifier as accurate as possible, as well as labeling different colors like green, brown, etc. as not recycle bin blue color.

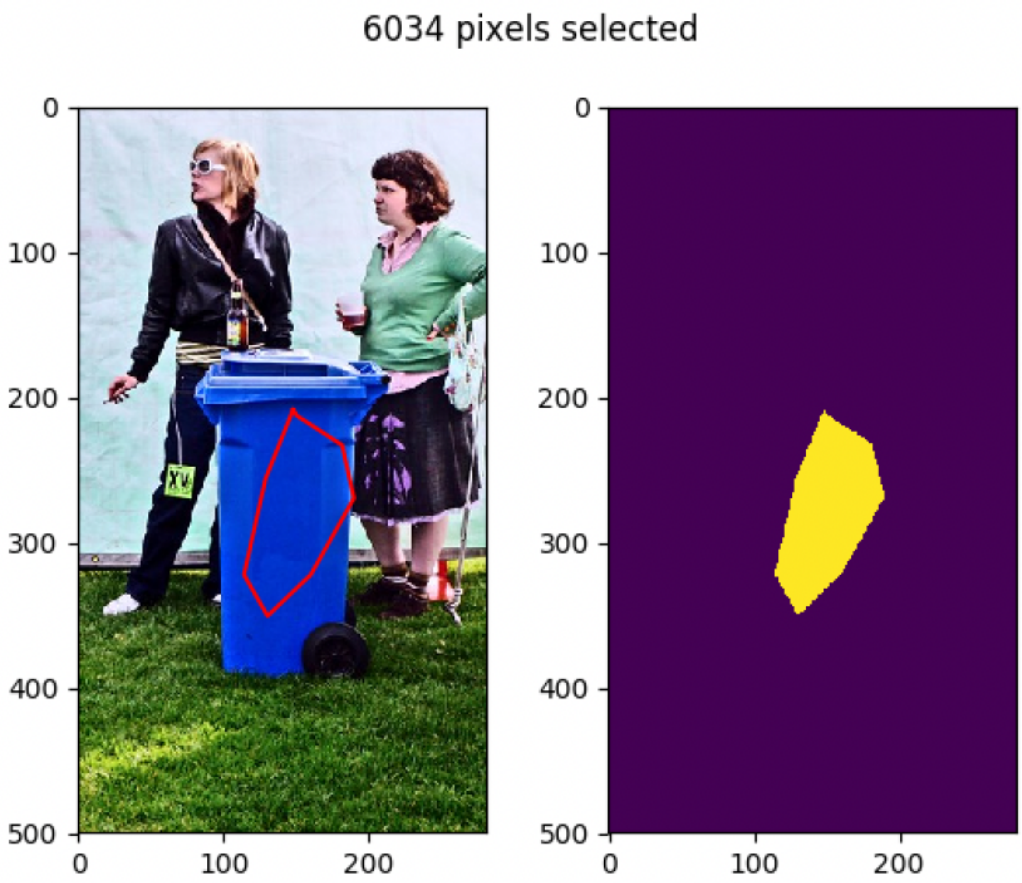

Figure 2: Use of `roipoly` for hand labeling of pixels.

3.2.2 Training the model

The model in this part uses the same training function as for the pixel classifier discussed in subsection 3.1. The difference is the size of the weight vector, as there are only the two classes blue and not blue for the model to predict. It would have been possible to use the Sigmoid function for this classification, but as the Softmax function is a generalization of Sigmoid for multiple classes, it works for two classes as well. The rest of the training procedure was identical to the pixel classifier.

3.2.3 Bounding boxes

A mask of the input images to the bin detector was created by sending the image through the classifier, and setting the pixel values where the classifier classified as recycle bin blue to 1 and the rest of the image to zero. By doing this, regions of interest where a set of pixels have the value 1 can be identified. This was done after the two techniques `erosion` and `dilation` were implemented on the masked image to create clearer areas of connected 1 values. `Label` and `Regionprops` from the `skimage.measure` library were used to create different regions in the image with different characteristics such as area and bounding boxes for the different detected regions.

The next part was to distinguish which of the detected regions that were recycle bins and which were not. The different region properties used in this part was the different region's area and the ratio between the width and height of the suggested bounding boxes. The different thresholds were in the end set to

$$0.5 > \frac{area}{numpixels} > 0.005 \tag{5}$$

for the acceptance of the size of the region. Furthermore, a check for the ratio between the width and the height of the region was implemented to remove the really long and slim regions and other regions that doesn't look like recycle bins. The threshold for this was set to:

$$1.5 > \frac{x_2 - x_1}{y_2 - y_1} > 0.33 \tag{6}$$

An example of a mask and its computed bounding box can be found in Figure 3.

Figure 3: Example of mask and bounding boxes

4 Results

4.1 Pixel Classification Results

After training of the model, the final weights are found in Equation 7.

$$\mathbf{w} = \begin{bmatrix} 31.5568539 & -16.22354567 & -15.78354851 & -0.08625269 \\ -16.33206885 & 31.31506731 & -15.69272559 & 0.11991397 \\ -16.218694 & -15.92775108 & 31.38069252 & 0.31214527 \end{bmatrix} \tag{7}$$

These weights resulted in an accuracy of 100% on the training and validation sets, as well as a 10/10 score on Gradescope, hence a 100% accuracy there as well. This is not very surprising as training a model to distinguish the three different colors is a really easy problem which converges to the optimal solution really fast. In stead of training a model to predict the color, a simple `argmax` function to choose the largest of the three RGB values would have done equally as good, showing that the actual problem of RGB pixel classification is relatively easy.

Different choices of values for maximal number of iterations, batch size and tolerance for early stopping were experimented with, but all yielded really good results, so the choice was made with the computing time in mind, as well as keeping the 100% accuracy on the validation set.

4.2 Bin Detection Results

The weight parameters after training of the model to classify recycle bin blue color are found in Equation 8.

$$\mathbf{w} = \begin{bmatrix} 1.49678923 & -40.96462496 & 36.36870051 & -4.89983946 \\ -1.26248478 & 41.05639961 & -36.45069465 & 4.07212496 \end{bmatrix} \tag{8}$$

These weights resulted in the masks and bounding boxes for the ten images in the validation set found at the end of the report (Figure 4-13). In the images, the actual bounding box is the red square and the calculated bounding box is the green square. The techniques of `erosion` and `dilation` have parameters for how many iterations it should do the operation, and the different number of iterations bettered and worsened the precision of the bounding boxes. If the erosion was too high, examples where two recycle bins are close to each other resulted in a single bounding box surrounding both bins instead of two separate bounding boxes.

One can observe that the classifier gave significantly larger weights for the blue color for the first line, which means that it will be good at recognizing blue colors and distinguish them from the other colors. The result on the bounding boxes show that the different thresholds expressed in the technical approach parts could do so that water, the sky and different blue objects not resembling recycling bins were not detected as recycling bins. In the end, the detector achieved 100% accuracy on the validation set, and got a score of 9.5/10 on Gradescope, only missing by half a point on Bin 3. One can observe that the detector manages to not detect the sky in Figure 7, the blue part of the door in Figure 10 or the blue thing in Figure 13. It shows that the detector works really well.

The coordinates for the calculated bounding boxes for the different images can be found in Table 1, and most of the bounding boxes are really close to the true bounding boxes, as seen in the figures at the end of the report.

Img	min x	min y	max x	max y
61	184	139	313	286
62	25	347	134	498
63	171	67	291	236
64	349	104	466	265
65	764	416	925	623
66	-	-	-	-
67	578	305	707	505
67	711	305	831	510
68	-	-	-	-
69	-	-	-	-
70	-	-	-	-

Table 1: Coordinates of bounding boxes in validation set

Figure 4: Mask and bounding box img 1

Figure 5: Mask and bounding box img 2

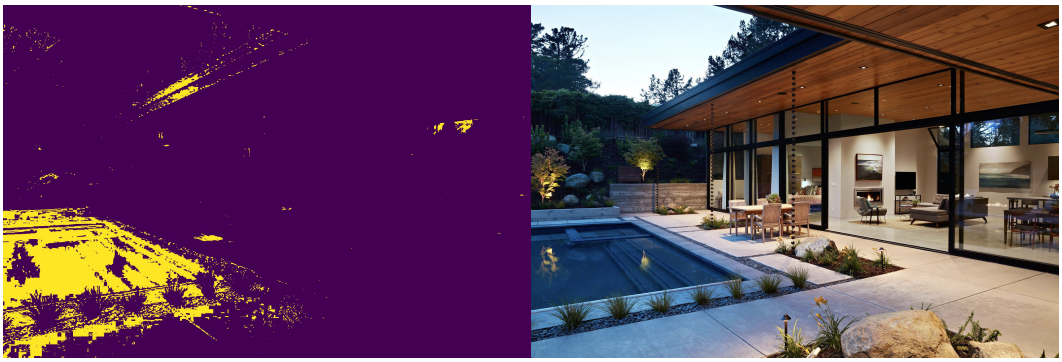

Figure 6: Mask and bounding box img 3

Figure 7: Mask and bounding box img 4

Figure 8: Mask and bounding box img 5

Figure 9: Mask and bounding box img 6

Figure 10: Mask and bounding box img 7

Figure 11: Mask and bounding box img 8

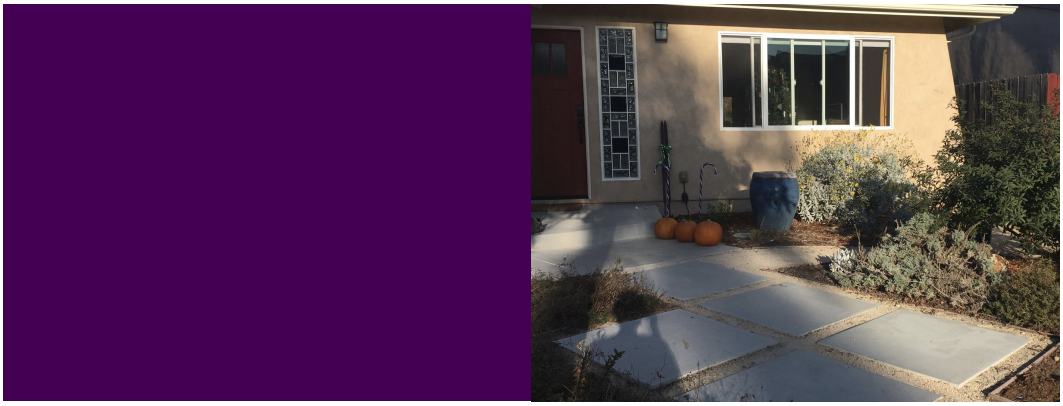

Figure 12: Mask and bounding box img 9

Figure 13: Mask and bounding box img 10