Probabilistic Color Based Segmentation and Detection

ECE 276A: Sensing and Estimation in Robotics

Inderjot Singh Saggu
Electrical and Computer Engineering
University of California, San Diego[1]

***Abstract*— A pixel-wise segmentation algorithm is implemented that uses logistic regression to classify each pixel as blue or not-blue . The blue regions are then grouped together and shape statistics is used to identify possible candidates for "Barrel" class. Finally, a list of bounding box coordinates are generated corresponding to each detected blue barrel.**

I. INTRODUCTION

Object based segmentation and localization are fundamental problems in computer vision that have tremendous applications in robotics, autonomous vehicles and more complex tasks like motion planning, scene understanding and object tracking. Most state of the art algorithms today use deep learning based techniques that require large amount of annotated data which becomes a major bottleneck. These algorithms rely on building a hierarchy of feature transforms where all the features are learned by the algorithm itself. This works well but is an overkill for implementing a simple and robust single object class segmentation and localization. The goal of this project is to segment and detect instances of "blue barrels" in an image. The classifier uses pixel values as features and implements logistic regression for classifying each pixel as blue barrel or not. The learned model is robust to illumination changes, scale, and minor occlusion. It can also detect multiple instances of blue barrels and returns the bounding box coordinates for each one of them. The "barrelness" for each region is determined using shape statistics that are based on the properties of the bounding box and fitted ellipsoid. The properties of interest mainly are the eccentricity and the major/minor axis lengths. The accompanying code has two methods, "segment_image" and "get_bounding_box" that return a mask image for blue barrels and a list of bounding box coordinates for the same respectively.

II. PROBLEM FORMULATION

We consider each pixel in an image as a vector $\mathbf{x_{ij}}$ where i,j represent the pixel location. Each pixel vector, $\mathbf{x_{ij}} \in \mathcal{R}^3$ for RGB images, and specifically in the range [0, 255] for uint8 type images. Also, each vector has an associated label y_{ij} that is either $\{1, 0\}$ depending upon whether the pixel belongs to blue barrel or not respectively.

This problem is modelled as a classic discriminative learning problem where we want to learn $\mathbf{p(Y|X,}\ \theta)$, here Y are the labels associated with the set of vectors X and the probability function is parameterized by θ. We use logistic regression for converting the continuous estimates into discrete distributions for classification. Particularly, we use the sigmoid function, $\sigma(z)$.

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{1}$$

What makes the sigmoid function interesting is that it's derivative can be evaluated easily.

$$\sigma^{'}(z) = \sigma(z) * (1 - \sigma(z)) \tag{2}$$

For logistic regression the parameter θ are the weights $\mathbf{w}$. The bias term can be included by augmenting each pixel vector to $\mathcal{R}^4$ with value 1 in the extra dimension.

$$p(y_i = 1|\mathbf{x_i}, w) = \sigma(\mathbf{x_i^T}w) = \frac{1}{1 + \exp(-\mathbf{x_i^T}w)} \tag{3}$$

$$p(y_i = 0|\mathbf{x_i}, w) = 1 - \sigma(\mathbf{x_i^T}w) \tag{4}$$

A. Training

We use this to model the probability distribution of a pixel belonging to class blue-barrel or not blue-barrel. We can measure the performance of our model using cross-entropy loss which quantifies the difference between two probability distributions.

$$L = -\frac{1}{N}\sum_{i=1}^{n} y_i * log(p(x_i) + (1 - y_i) * log(1 - p(x_i)) \tag{5}$$

We use gradient descent algorithm to iteratively estimate $\mathbf{w}$. Here α is the learning rate.

$$w^{(t+1)} = w^{(t)} + \alpha * \frac{1}{N}\sum_{i=1}^{n} x_i(y_i - \sigma(\mathbf{x_i^T}w^{(t)})) \tag{6}$$

B. Testing

Given a test image we compute $\sigma(\mathbf{x_{ij}}^T w)$ for all i,j belonging to the image. This gives us a single channel matrix with values ranging between 0 and 1, and the same size as the original image. We then use the heuristic that, if the value at each pixel in the computed matrix lies above mean + 2* (standard deviation) it is classified as blue region belonging to a barrel, else not-blue barrel. The next part involves detecting contours, defining a "barrelness" criteria and subsequently generating bounding box coordinates.

[1]

III. Technical Approach

The segmentation and localization pipeline for this project can be broken down into broadly into five parts

1) Manual annotation of images using Region-of-Interest Polygon method
2) Training a Logistic Regression Classifier for generating pixel-wise scores for segmentation.
3) Testing the model resulting in a binary mask.
4) Combining common regions together and using Region Properties to determine "barrelness"
5) Returning bounding box coordinates for blue-barrel regions.

A. Manual Annotation of Images

We use an open-source python function roipoly (Region of Interest Polygon) for manual annotation. It allows the user to draw a polygon bounding box around region of interest, generating a binary mask where everything inside the region has a value of one and the rest zero. It also allows us to draw multiple polygons.

Fig. 1. Manual Annotation Results

B. Training Logistic Regression Classifier

We first normalize the images we are dealing with so that they lie within [-0.5, 0.5] for ease of learning. We have already discussed the math involved in implementing a generic binary logistic regression classifier. For this particular problem, each image had a size of 800*1200 and we had a total of 46 images. Hence, training the classifier on the entire batch would have been computationally intensive. Instead we implement mini-batch learning that treats each image as one mini-batch, computes the gradients based on the error between the output of the model and ground-truth labels and updates the weights. For training set of 'n' images we have 'n' updates per epoch. The model is trained on multiple epochs and the performance is evaluated using cross-entropy loss on both training set and a validation set. We split the complete dataset into 36 images for training and 10 images for testing. We train the model on 30 images and compute the cross-entropy loss on both the training and validation set to check overfitting.

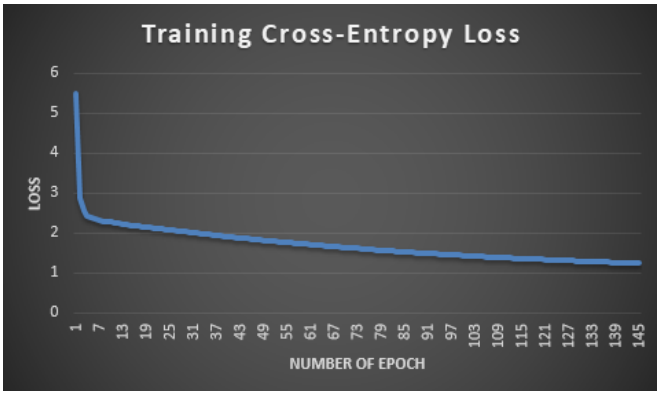

Fig. 2. Training Loss per Epoch

C. Testing the Trained Model

For testing the model we load the trained weights and pass the image as input to our model, where the computation is carried out pixel-wise. We then used the heuristic that blue regions would have higher predicted probabilities and hence values above mean + 2* (standard variance) are classified as blue. We can also use a simple threshold of greater than or less than 0.5 but observed that the proposed heuristic works better especially when there are several moderately blue objects present in the scene. The thresholding method classifies these pixels as blue-regions too whereas the method used is able to neglect these semi-blue regions for strong and definitive blue of the barrel.

D. Combining Region and Defining "Barrelness" using Region Properties

Once we have the binary mask we can use contour detection to give boundary to select regions that belong to a particular class (blue-barrel). This makes the model robust to occlusion to some extent. We can also use other OpenCV functions like dilute that gives surrounding pixels the same value, so regions hidden also get considered to be classified as blue-barrel if they are close enough to the barrel originally. Based on the resulting mask we use Scimage.measure's label function that finds different connected regions and assigns the same value to all pixels belonging to a particular region. This can be viewed as a method for generating proposals for possible barrels based on the mask image.

Defining Barrelness: Based on the different generated regions and the actual ground-truth we attempt to find out the conditions that define a barrel. For this region properties method is used from Skimage.measure that fits a ellipse to the region and gives us all the fields related to the ellipse along with many other fields. The ones we are particularly interested are eccentricity, and major/minor axis length. For this project the following constraints were used

1) **Eccentricity**: Based on region properties of different false positives and true positives, a range was determined for eccentricity values (¿ 0.7)
2) **Orientation**: Orientation values between $\frac{-\pi}{4}$ and $\frac{\pi}{4}$ were determined to be best suited to the given dataset

3) **Ratio of region-area to Bounding-box area**: Since the barrel is approximately a rectangle this ratio is close to 1 for valid barrel-regions (¿ 0.75)
4) **Area**: We want to make our model robust to scale hence it might seem counter intuitive to have .area as a constraint but in practice it works well to remove small noisy regions.

E. Result: Bounding-Box Coordinates and Segmented Image

We use the final detected regions to modify the initial mask image and ignore the segmented regions which are not bounding boxes. This gives us the final segmented image. We then use Matplotlib's in built method for generating boxes for each of the final detected barrel regions.

IV. Results

The first part in this project was manual annotation of images which gives us a decent approximation to ground-truth. There are miss-classifications that are bound to creep in but the model is robust enough to ignore this noise. Also the problem that we are looking at involves just two classes so binary labels are enough. Fig. 1 shows two of the resulting images. The training loss per epoch is plotted in Fig.2. We can now look at the results of the model on training images and what worked and what didn't. The train/test split was selected a random with approximately 80:20 split between them.

Fig. 3. Results on Training-Set Images

The model gives reasonable results on trained images, and is robust to occlusion, and blue non-barrel objects that are not close to the barrel. It also successfully segments multiple instances of barrels.

Fig. 4. Results on Test-Set Images

From the test results we observe the robustness of the model to slight occlusion and blue non-barrel objects. The model also performs well under low-light conditions.

Fig. 5. Results on Low-Light Images

Till now we have only looked at cases that worked well but for some cases we get many false positives and enforcing a stronger barrelness condition ends up reducing the number of true-positives which we want to avoid.

Fig. 6. Types of Failed Cases: (Top-to-bottom) (a) Segmentation failure; (b) Multiple false-positives; (c) Occlusion; (d) False region detected

For the first image the blue structure possibly distorts the perceived geometry of the barrel leading to a failed blue-barrel segmentation and subsequently bounding-box. The second image is probably a result of weights trying to accommodate blue-barrels in low-light conditions. The false-positive regions appear as if there is a barrel that is in the dark (darker shade of blue). This is a fundamental constraint of a model which is not complex, like logistic regression. For the third image parameters chosen for dilate method are unable to combine the smaller detected regions together to accurately segment the barrel. The region segmented doesn't satisfy the barrel constraints and is rejected. The final image has a barrel surrounded by a light blue background. The model detects a barrel like shape in the corner of the region and ignores the rest. It is likely that the blue region detected by logistic regression, contained the barrel along with a part of the surrounding region, since it's blue. The resulting region would not have satisfied the shape constraints imposed for barrels.

V. Conclusion and Future Work

Color based segmentation and detection of a particular object, like a barrel for our case, can achieve reasonable results using logistic regression based pixel-wise classification algorithm. We achieved a final test-set score of 80/90 and the model performed well on our test set as well under different illumination conditions, occlusion, scale and multi-instance segmentation and detection. The model can be made more robust by improving each step of the pipeline starting from manual annotation.

- Current methods is accurate but cannot be scaled. If we have more data the performance of the model will definitely improve but we would require automated ways of labeling images, e.g., by unsupervised image segmentation, or an adaptive region flooding algorithm.
- Adding multiple classes would make it easier for the model to distinguish different types of blue regions, and we can also experiment with different color spaces.
- Training more complex models like SVM or neural-network, given large enough data, can improve results considerably.
- The size of the mini-batch sized for this problem is too big in hindsight which limits the number of updates per epoch. Thereby we need to run over the images for many epochs for the model to train.
- The number of not-barrel-blue data points far exceed the number of blue-barrel data points and can add unnecessary bias to our model. We can solve this by training on a subset so that the size of two (or n) classes is balanced.

APPENDIX

Link to github page (project would be up once the deadline has passed):

https://github.com/ijssaggu

This would include code for manual annotation of images, training the logistic regression model, plotting cross-entropy loss, learned weights and biases and any future work.