Predict Foreground Object in Still Image
Stanford Univeresity
CS229 Fall 2013

Amer Hammudi
ahammudi@stanford.edu

Darren Koh
dtkoh@stanford.edu

Abstract

In this paper, we are interested in the detection of foreground objects in a single still image. Using the collections of outdoor scene RGB images, we applied supervised and unsupervised machine learning techniques to predict the foreground objects. The main challenge of foreground object detection, is that single pixel from a given image is not sufficient to distinguish between foreground or background, and thus, combination of features needs to be considered.

1 Introduction

Foreground object detection has been successful using multi-color background model per pixel (Gaussian Mixture Model) proposed by Grimson and Stauffer [1, 2, 3]. However it required sequence of same scene images, we are interested in detecting foreground object with single still image.
The ability of finding foreground objects has many useful real life applications. For example, an autonomous self driving vehicle can utilize foreground object detection and pairing depth map information (such as stat of the art technique done by Andrew Ng and his team [4, 5, 6, 7]), and pre-apply braking as object getting closer to the vehicle. Alternatively, the autonomous vehicle can turn on warning brake lights as object is approaching and passing over safety distance from the rear. This can replace the expensive and bulky equipment such as Radar or Lidar system being adapted in the auto industry.

2 Approach

Our approach is to collect outdoor scene images as our training set, then extract features that we consider relevant for foreground object classification. We will use K-mean to cluster scenes, train Logistic Regression hypothesis and employ Neural Network to get the final hypothesis that allows us to predict foreground pixel within a single still image.

3 Data Set

We use the Stanford Background Dataset [8] (715 images) and Semantically-Augmented Make3D Dataset [4, 5, 9] (534 images) as our training data. This dataset contains labeled regions, surfaces and layers such as foreground object, sky, tree, road, sky, building, etc. We mainly focused on the labeled foreground object and treat other labeled regions as background. We filtered out 349 images (retained 900 images) that do not contains foreground object label. Furthermore, to make our data easier to work with, we resized all images to 160x120. Figure 1 is the Raw Image and Figure 2 is the labeled foreground pixels.

Our approach in using the dataset is to leave out 10% of the data (90 images) for testing and the remaining 90% (810 images) for training.

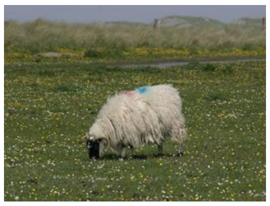
Figure 1: *Raw Image*

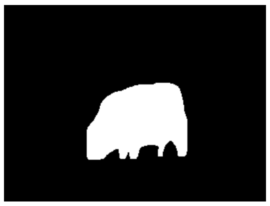
Figure 2: *Labeled foreground (White), background (Black)*

4 Feature Extractions

The features that we think are relevant to foreground object detection are **Pixel Location**, **RGB Color**, **Texture Segmentation**, and **Pixel Intensity**.

4.1 Pixels Location

We use the X and Y coordinate of a pixel as the pixel location feature. The intuition of using pixel location is, we believe with-in an outdoor scene image, a foreground object is most likely distributed around the horizontal axis.

4.2 RGB Color

We think the RGB color of an individual pixel is a relevant feature to distinguish foreground objects in an outdoor scene image, for example, the green color (such as tree, grass) and the blue color (such as sky) are more likely distributed in the background than foreground in an outdoor scene.

4.3 Texture Segmentation

Using each pixel entropy value of the 9-by-9 neighborhood around the corresponding pixel in an image as texture. We want to see if it provides any relevancy to the foreground information. It is generated with the Matlab function **"entropyfilt"**.

4.4 Pixel Intensity

We believe object that is closer should have brighter color intensity in the outdoor scene image. To extract the pixel's intensity feature, we convert each pixel to gray scale color (0-255) using the Matlab function **"rgb2gray"**.

5 Evaluation Result

To evaluate the accuracy of our foreground object detection algorithm, we use the equation

$$Accuratecy = \frac{1}{2}\left(\frac{LF}{TF} + \frac{LB}{TB}\right) \tag{1}$$

where LF is Labeled Foreground, TF is True Foreground, LB is Labeled Background and TB is True Background.

6 Logistic Regression

In our first step to classify pixels as foreground we will train four different sigmoid functions $\frac{1}{1+\exp^{-\theta^T x}}$ and to classify each pixel as foreground based on the features mentioned above. Since the data is not linearly separable we define the feature vectors

- $\vec{V}_{loc} = (1, i, j, i^2, j^2, i * j)$
- $\vec{V}_{rgb} = (1, r, g, b, r^2, g^2, b^2, r * g, r * b, g * b)$
- $\vec{V}_{txt} = (1, r, g, b, r^2, g^2, b^2, r * g, r * b, g * b)$
- $\vec{V}_{int} = (1, s, s^2)$

where (i, j) in $\vec{V}_{loc}$ represent the location of the pixel in the image. (r, g, b) in the $\vec{V}_{rgb}$ and $\vec{V}_{txt}$ represent the RGB color and Texture. Finally s in $\vec{V}_{int}$ represent the Intensity.

In this classification we are assuming that the foreground pixels are generated through randomized process Bernoulli(Φ) where

$$\Phi = \frac{\sum \text{Foreground pixel}}{\text{Total number of pixel}} \tag{2}$$

7 Training Phase - Initialization Step

In the first step of training phase, in order to reduce the training time with 810 images that each contain 19,200 pixels, we are utilizing the Parallel and GPU Computing Toolbox in Matlab and as a result, we cut down the time exponentially in comparison. In particular, we employed **"parfor"** and **"gpuarray"**.

The resulting test error of each of the hypothesis function are

- Location Error $= 31.25\%$
- RGB Error $= 37.74\%$
- Texture Error $= 50.38\%$
- Intensity Error $= 41.60\%$

Figure 4 below is the general area where Location hyphothesis function classifies pixel as foreground.

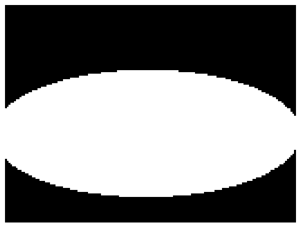

Figure 3: *Location - Predicted foreground*

Figures 4-18 below are the original images, actual foreground pixels, and their corresponding predicted foreground pixels resulting from the trained RGB, Texture, and Intensity hypothesis.

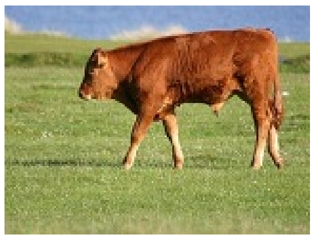

Figure 4: *Raw Image*

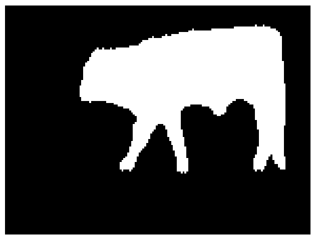

Figure 5: *Actual foreground pixels*

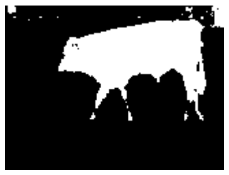

Figure 6: *RGB - Predicted foreground*

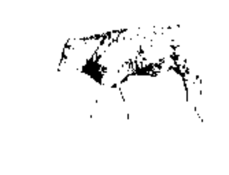

Figure 7: *Texture - Predicted foreground*

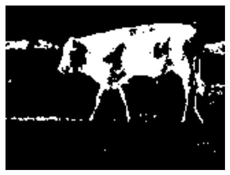

Figure 8: *Intensity - Predicted foreground*

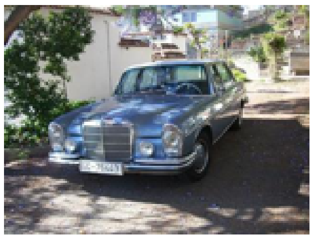

Figure 9: *Raw Image*

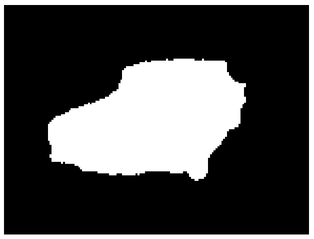

Figure 10: *Actual foreground pixels*

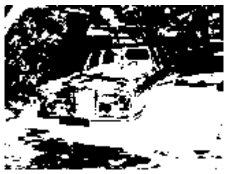

Figure 11: *RGB - Predicted foreground*

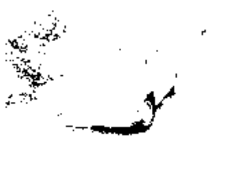

Figure 12: *Texture - Predicted foreground*

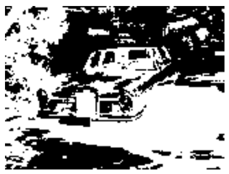

Figure 13: *Intensity - Predicted foreground*

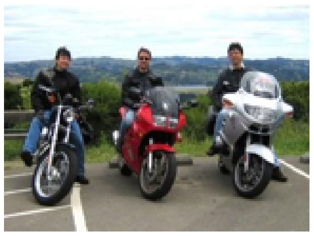

Figure 14: *Raw Image*

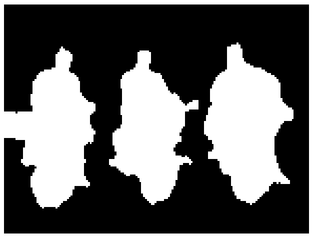

Figure 15: *Actual foreground pixels*

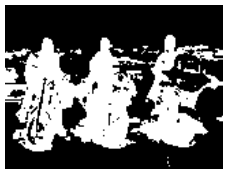

Figure 16: *RGB - Predicted foreground*

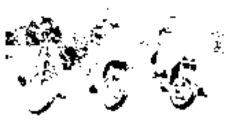

Figure 17: *Texture - Predicted foreground*

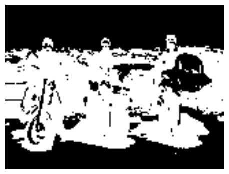

Figure 18: *Intensity - Predicted foreground*

8 Neural Network Model

Our next step is to integrate the information produced by the four hypothesis (Location, RGB, Texture, Intensity) and produce our final output. To do that, we created a neural network shown in Figure 19.

The activation for the final neuron is the result of the four logistic functions mentioned above. In addition, we included the quadratic of the results from the four hypothesis due to the linear separability issue.

The test error from the output hypothesis is 23.79% and the images produced on the test images are shown in Figure 20-25.

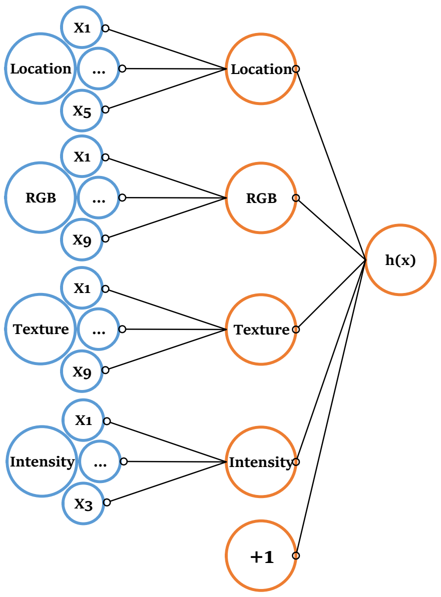

Figure 19: *Neural Network with 23 input units and 4 hidden units*

Clearly we have improved our result significantly from the independent hypothesis, Location, RGB, Texture, and Intensity logistic functions.

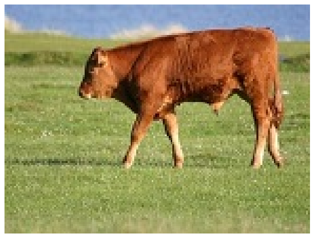

Figure 20: *Raw Image*

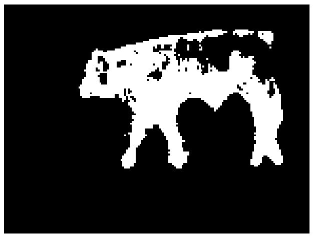

Figure 21: *Predicted Foreground using Neural Network*

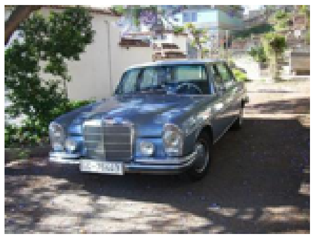

Figure 22: *Raw Image*

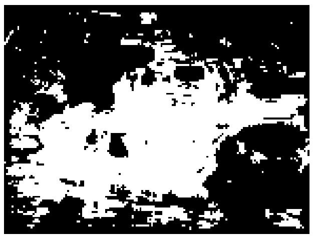

Figure 23: *Predicted Foreground using Neural Network*

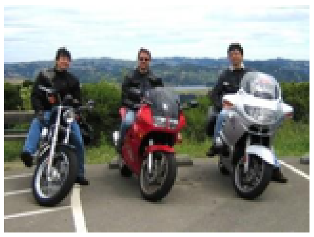
Figure 24: *Raw Image*

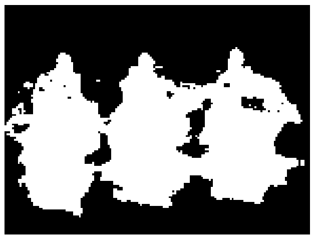
Figure 25: *Predicted Foreground using Neural Network*

9 K-mean Scene Classifier

To further improve our hypothesis of image foreground pixels classification, we believe clustering the outdoor images in the training set and then create the hypothesis function for each cluster should improve our prediction. The intuition is that, similar scenes should have similar foreground objects distribution.

We used the Scale Invariant Feature Transform (SIFT) to produce a feature vector for each image and then use K-mean to create clusters from the training data set. We then used SIFT method to produce a feature vector for the test image to decide on the relevant hypothesis function in order to classify the test image pixels.

We selected SIFT for clustering because SIFT is invariant to image scaling and rotation and partially invariant to illumination and viewpoint changes. Furthermore, the SIFT method has been widely used and popular in scene recognition [10]. We tested different values for $K \in \{10, 5, 3\}$ for the K-mean algorithms. For $K = 10$, the number of images in certain clusters is much less than others. For our purposes we selected $K = 3$ due to the density of the three clusters and is sufficient to show that clustering produces better results, however we think that larger K would improve our result and further research is needed to decide on the number of clusters. We believe that the number of clusters should be a function of the number of different objects identified by the SIFT method, and the hypothesis function for the test images would then be selected based on the cluster with the max number of objects matches.

After applying scene classification and train each hypothesis and neural network, the test error has improved slightly by 0.53% from 23.79% to 23.26%. Figure 26-31 shows the same images with improved predicted foreground pixels after applying scene classification.

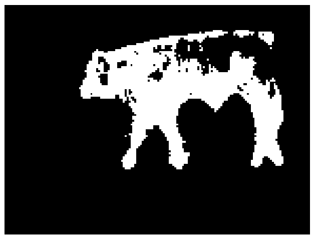
Figure 26: *Before scene classification*

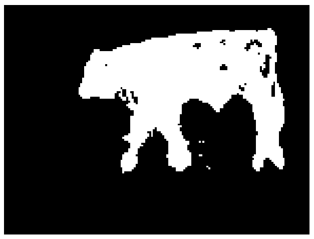
Figure 27: *After scene classification*

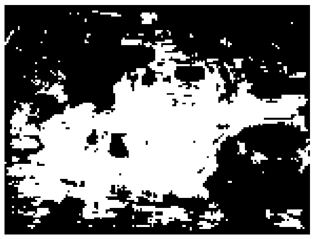
Figure 28: *Before scene classification*

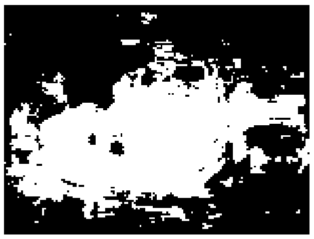
Figure 29: *After scene classification*

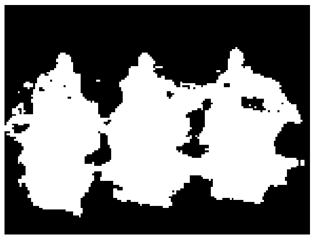
Figure 30: *Before scene classification*

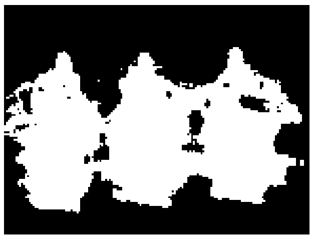
Figure 31: *After scene classification*

10 Cross Validation Check

To verify that we do not have high variance due to the number of features that we are using, we ran 10 rounds of **Hold-out Cross Validation** (HOCV) on the neural network, each round split the data in $S_{train} = 70\%$ and $S_{cv} = 30\%$ randomly. The result shows, our generalization error is very close to our training error, therefore we believe we do not have high variance in our algorithm.

Figure 32 below is the result of 10 rounds HOCV and their corresponding test error (the average test error of 3 clusters). The doted line is the test error on cross validation (S_{cv}), while the solid line is the test error on the 90 images (10% of 900 images that were left out for testing, as mentioned in last part of **Section 3 - Data Set** above).

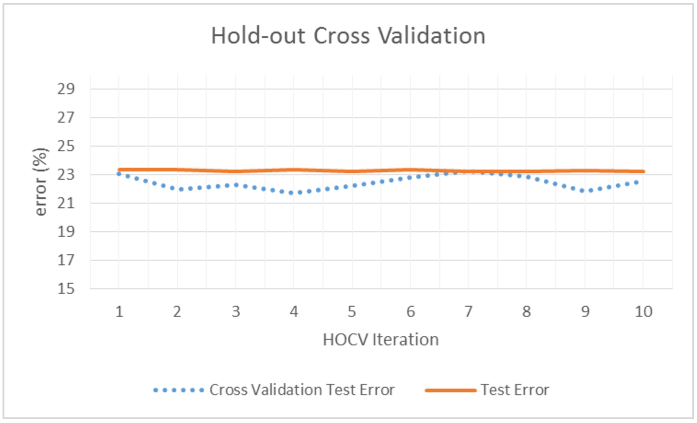

Figure 32: *Before to scene classification*

11 Future Work

Figure 33 below is the Learning Curve and it does seems to suggest that as we increase the data set, the test error is stabilizing and no longer dropping. Hence, we have a bias problem with our algorithms. Accordingly we think adding other image features is necessary to improve our result, specifically we think that integrating Histogram Oriented Gradient (HOG) in our Neuron network should improve our result. To integrate HOG, we think it is best to train an SVM and use the output as another input to the last Neuron or add one more layer to the Neuron network. Furthermore, we have shown that using only three clusters in scene classification, we believe having more clusters should improve the result.

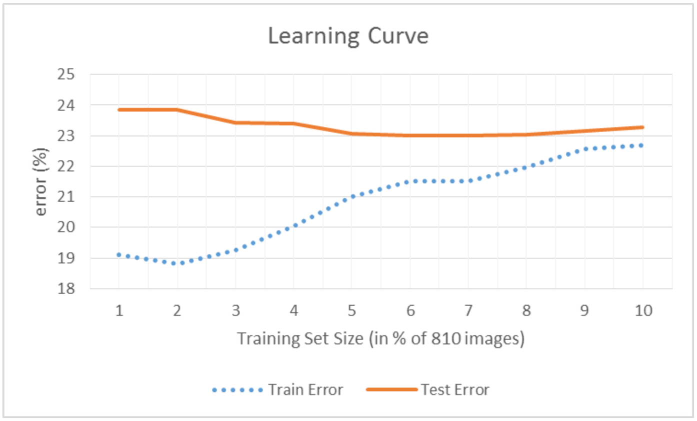

Figure 33: *Before to scene classification*

12 ACKNOWLEDGEMENTS

We would like to thank Brody Huval (CS229 TA) for his advice and input on computer vision.

References

[1] Chris Stauffer and W. E L Grimson. Adaptive background mixture models for real-time tracking, 1999.

[2] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site, 1998.

[3] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, August 2000.

[4] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2005.

[5] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. of Pattern Analysis and Machine Intelligence (PAMI)*, 2009.

[6] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision (IJCV)*, 2007.

[7] A. Saxena, J. Schullte, and A. Y. Ng. Depth estimation using monicular and stereo cues. *In International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[8] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. *Proceedings of International Conference on Computer Vision (ICCV), 2009.*

[9] B. Liu, S.Gould, and D. Koller. Single image depth estimation from predicted semantic labels. *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[10] Guoshen Yu and Jean-Michel Morel. Asift: An algorithm for fully affine invariant comparison. *Image Processing Online*, 2011.