A Novel Approach to Image Segmentation for Traffic Sign Recognition
Jon "Jay Hack and Sidd Jagadish

Introduction/Motivation: As autonomous vehicles, such as Google's 'self-driving car,' have recently become more prominent, the ability to detect and recognize informational road signs is of elevated importance. Road sign detection faces the same challenges as many other detection tasks, namely that of shadows and outdoor lighting conditions obscuring the signs; road sign detection is easier than many other detection tasks, however, as one can assume regular shapes, nearly uniform colors, single angle of presentation and non-deformable nature. A majority of existing approaches to traffic sign recognition separate the task into two phases designed to capitalize on these advantages[1]: The first phase, known as the "segmentation phase," determines which regions of an image are likely to yield traffic signs, and the second phase, known as the "classification phase," determines what kind of sign (if any) is contained in this region. Here, we describe a new approach to the "segmentation" phase.

We trained and tested our algorithm on a publicly-available[2] database of images of three different types of road signs (bike crossing, pedestrian crossing, and intersection signs). These road signs were of three different shapes, (square, circle and equilateral triangle), multiple colors and pictured in a diverse set of lighting and weather conditions.

Initial Approach – Center-Surround Saliency: Traffic signs are designed to be salient to the human visual system and, more often than not, to 'pop out' from their surroundings. Therefore, having extensive experience and understanding of the human perception system, our first approach to creating a novel segmentation algorithm drew inspiration from the human visual system's implementation of center-surround and bottom-up saliency.

---

[1] Shadeed, W. G., D. I. Abu-Al-Nadi, and M. J. Mismar. "Road traffic sign detection in color images." *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*. Vol. 2. IEEE, 2003.

[2] http://www.cs.rug.nl/~imaging/databases/traffic_sign_database/traffic_sign_database.html

In the human visual system, center-surround color opponency describes the behavior of photoreceptors in the eye that react most prominently to stimuli that stand out from their surroundings. A red-green color-opponent photoreceptor fires most rapidly if the center of its visual field is very red, and the surrounding areas are very green (and least rapidly if the opposite is true). Blue yellow color-opponent cells, intensity center-surround cells, and orientation center-surround cells behave similarly (although they react to blue-yellow color opponency and differences between orientations of edges in the center and surrounding areas of their visual fields respectively).

Figure 1

We implemented the model of the human visual system's bottom-up saliency described by Itti et al[1], which used blue/yellow, red/green and intensity center-surround, as well as oriented Gabor features, in order to discern which regions of the image were of greatest interest. We found that high center-surround value points part of exceptionally salient signs such as that in Figure 1, but failed to identify less prominent signs, such as that in Figure 2.

Figure 2

New Approach – Color Region Segmentation: As previously mentioned, empirical analysis revealed that our saliency-based approach to road sign detection did not provide adequate performance. We therefore hypothesized that features of *regions* of the image would be more informative than saliency values for individual points within an image. Our next approach consisted of segmenting the image into clusters of pixels of similar colors and

[1] Itti, Laurent, Christof Koch, and Ernst Niebur. "A Model of Saliency-based Visual Attention for Rapid Scene Analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.11 (1998): 1254-259. Print

analyzing properties of these clusters in order to make a prediction of whether they contained road signs or not.

In order to determine which pixels were of a similar color, we implemented the the k-means clustering algorithm to compress the input image such that every pixel was one of a small number of values. We experimentally determined that k = eight clusters provided optimal performance, as any fewer often caused the pixels of road-signs to be grouped along with other visual elements of the image, such as the sky. We then implemented the union-find algorithm to group neighboring pixels of the same cluster into 'regions.' These regions were the objects of classification: having computed statistics to describe each region, we input these as features to machine learning algorithms. (logistic regression, SVM)

MTA Ratio	#Pixels Inside Region/#Pixels Inside TBB (Mass to area)
Aspect Ratio	Width/Height
Centroid x	Location of Mass Centroid Relative to TBB
Centroid y	Location of Mass Centroid Relative to TBB
Avg. Hues	Average RGB components Inside TBB/Outside TBB

For each region of neighboring pixels belonging to the same k-means cluster, (henceforth referred to as merely 'regions') we constructed a tangent bounding-box (TBB) and computed the following statistics to use as input features to our machine learning algorithms:

Figure 3

[1]

[1] Figure 3 illustrates our recognition process. The first image shows a grey-scale version of our k-means image compression. The second image shows the region (created by union-find) that corresponds to the sign. The third image shows the tangent bounding box of the region in the second image, and the fourth shows the tangent bounding box around the original image, after classifying it as a positive example

Adjusting Parameters: We evaluate our success based on our true positive rate, positive ` value (*TP/(TP + FP*), and accuracy. Note that for this task, since it is meant to quickly process an image before running a more sophisticated object recognition algorithm on a small region, our true positive rate is extremely important (a low true positive rate would mean missed traffic signs, even at this early stage). We also want to maximize *PPV = P(Traffic Sign|Classified Positive),* to ensure we are sufficiently narrowing down the image. As such, for our logistic regression classifier, we altered the threshold *t* at which we classified an example as positive, finding that, for all $t > 0.3$, $TPR > 0.95$. As the number of false positives decreases dramatically with increases in *t,* we ultimately chose a threshold $t = 0.90$, testament to the degree of separation logistic regression provides between positive and negative examples.

For a task performed in moving vehicles, such as traffic sign recognition, algorithmic efficiency is of obvious importance. Unfortunately, unaltered k-means and union-find are fairly inefficient. We therefore took measures to optimize both algorithms for this task. We first limited k-means on our test data to a mere 50 iterations. To increase our performance with such a small number of k-means iterations, we initialized our clusters for our test data to "mean centroids" for the training-images.[1] With very little change in performance after limiting k-means to 50 iterations, we opted to completely forgo running the k-means algorithm on our test images, instead compressing each test image by replacing each pixel in the test image with the closest "mean centroid," taking the same amount of time as one k-means iteration (the time to

[1] To do this, for each k-means centroid in the first training image, we find the closest k-means centroid (by Euclidean distance) in each of the other training images (think nearest one neighbor by Euclidean distance for each of the , and take the mean of these n centroids (where n is the number of training images). We repeat for each of the other centroids in the first training image, so we are left with eight "mean centroids" – one for each k-means cluster. Doing so increases the degree of convergence through 50 iterations.

find the nearest centroid). We found that this quick substitution takes roughly 10.28% of the time k-means takes to run.[1]

Although k-means is a fairly slow algorithm, union-find accounts for the majority of the time it takes our algorithm to process a test image. Our original implementation involved finding regions corresponding to all eight k-means centroids. Instead, we opted to find the average color within the traffic sign, define a "feasible color range", and only find regions corresponding to colors within the "feasible color range," both when training and testing. This both decreased our number of false positives (our models are now more sensitive to small changes in color-related features) by 21.2% (on average) and decreased our average computation time for a test image to 41.3% of the former computation time.

Results/Conclusion: We performed 16-fold cross-validation for both the logistic regression and SVM variants of our algorithm, presented below. As we had hoped, our TPR is very high, at near 1 for each kind of sign, so very few signs were missed. Our PPV values are not high enough for this method alone to detect a sign, but are sufficiently high that it is clear we have succeeded in narrowing down the area of the original image that could potentially yield a sign. Our accuracy values are extremely high, but these are somewhat inflated due to an abundance of examples (regions) too small to hold a sign. Ultimately, requiring a few seconds per image, this method of image segmentation may not be ideally suited to a time-sensitive task like traffic sign detection. However, the high true positive rate and sufficiently high PPV imply that this may be a useful method to screen images for many other object detection tasks.

k=8, t=0.9, n+=20, n=45000	Type of Sign	TPR	PPV	Accuracy
Logistic Regression	Bike	1	0.197	0.996
	Crossing	1	0.244	0.998
	Pedestrian	0.944	0.288	0.999
SVM	Bike	0.944	0.144	0.998
	Crossing	1	0.267	0.998
	Pedestrian	1	0.308	999

[1] Theoretically, going from 50 k-means iterations to one iteration should take only 2% of the time, but we implemented the quick substitution ourselves in MATLAB, whereas with 50 k-means iterations, we used MATLAB's built-in k-means implementation, which is optimized for MATLAB.