STREET SIGN RECOGNITION

Devin Dang, Curtis Wilson, Yiran Zheng

COSI 177, Brandeis University 2016

ABSTRACT

This project investigates street sign-recognition methods in the field of computer vision. The goal is to take in a roadside image and to identify street signs within the image based on known definitions of signs. Shape-based detection using the fast radial symmetry transform is used to identify symmetrical points of interest in the attempt to segment street signs from input images. Using a database of known signs, template matching was determined to be moderately accurate but inefficient. Color-detection employs known color ranges for street signs and creates a two-dimensional area targeting potential signs. The segmented areas created through these methods are then textually processed using optical character recognition augmented by using maximally stable extremal regions.

1. INTRODUCTION

Street sign recognition has substantial applications within the field of computer vision. Information on signs can range from vital, such as alerting drivers to stop before entering an intersection, to trivial, such as street names. As a result, street sign recognition must be highly accurate to be useful.

To address varying detection conditions and requirements, we have divided street signs into two categories: standardized and non-standardized signs. Standardized signs in the United States have definitions set that allow us to assume proportion, color, text, and other standardized characteristics. Examples of standardized signs are stop, crosswalk or do-not-enter signs. Non-standardized signs also have similar standards set but will have varying text that needs to be independently detected and read. Examples of non-standardized signs are street name or highway city signs. Both categories have different variables that need to be addressed before detection is possible and so, our project splits signs into these two sign categories to address recognition.

2. STANDARDIZED SIGNS

Since standardized signs have predictable features, particularly shape, we attempted to recognize signs through shape detection. Street signs typically consist of triangular, rectangular, or octagonal shapes and so identifying the shape of a potential sign will allow a narrower classification. For instance, octagonal shapes detected should guarantee a stop sign. In addition to shape detection, template matching can be used to directly identify a standardized sign. For example, since all stop signs are red, octagonal and contain the text "STOP" in the center, using an existing template image of a stop sign may provide an accurate recognition of stop signs in the target image. Compared to non-standardized signage, standardized signs allow us to make assumptions that narrow down classification attempts.

2.1. Fast Radial Symmetry Transform

Specifically for shape-based sign recognition, the fast radial symmetry transform (FRST) is a feature extraction technique that has been proven to be effective at detecting street signs due to their symmetrical, polygonal shape. At the basic level, the FRST technique detects points of interest by determining pixels who exhibit the greatest amount of radial symmetry. Using a gradient magnitude image, edge pixels are first detected to narrow down the pool of pixels being tested. A polling line is then established based off of the predicted radius length of the target shape. The polling line runs through what is expected to be the center of the object (using predicted radius length). Each edge pixel votes on pixels on the polling line representing where it thinks the "center" is. The pixel(s) on the polling line with the highest number of votes is determined as a "center." The resulting transform image will highlight pixels that have been determined as features[1, 2].

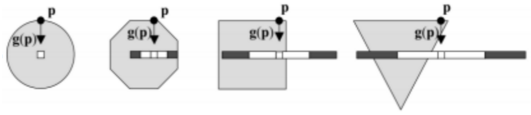


Fig. 1. Varying Polling Lines

The goal of FRST in street sign recognition is to isolate out polygonal shapes in the input image. Segmenting potential sign candidates by shape would allow narrowing down classification. However, FRST requires dynamically changing the parameters that produce images that vary in usefulness and clarity, specifically the radius parameter. The radius parameter determines how many discrete radii (length) are tested against the edge pixels. The issue that arose was

determining what radius parameter to set; signs that appear closer to the camera are larger and quality transform images are produced only using a higher value radius parameter while the opposite is true for signs farther from the camera. This problem is compounded when attempting to consider varying image resolutions. Higher resolution images will require a higher value radius parameter since there are more pixels overall in the image. While the apparent size of the sign in an input image is the same, varying resolutions of the same sign will require different parameter settings. This problem severely limits the range of images that can be accurately tested against.

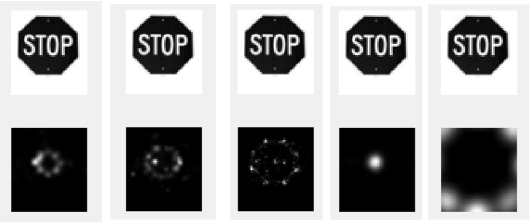


Fig. 2. Radii Parameters: 5, 10, 13, 20, and 50 respectively

In addition, interpretation and using the resulting transform image itself is a complex problem. Resulting transform images may identify features that have some radially symmetrical features but are not polygonal shapes. Not all points of interests detected in the transform image will be a sign and so we cannot directly link all detected points of interests to signs. This is a fundamental level problem with using FRST to segment shapes out of images. Being able to interpret the transform image is a dynamic challenge not possible to be addressed without strict hard coding. Potential directions for using FRST in shape-based detection and image segmentation is to employ a neural network and to train it to address the issues of setting parameters for useful transform images and to interpret the transform image itself. Furthermore, it was determined both through our own investigation and research that FRST is most successful at detecting circular shapes[3]. Other feature detection techniques will need to be investigated for other shapes.

2.2. Template Matching

The other method used for standardized sign recognition is template matching. The strategy behind template matching is given a template image, we position the template image over every possible position in the input image, calculating the normalized cross-correlation using equation 1. The position that returns the highest correlation value is the area with the closest match to the template.

$$NCC(Im1, Im2) = \frac{1}{N\sigma_1\sigma_2}\sum_{x,y}(Im1(x,y) - \overline{Im1}) * (Im2(x,y) - \overline{Im2}) \quad (1)$$

As shown in equation 1, under the assumption that all input images have the same size and are in grayscale, normalized cross-correlation calculates the similarity between two images. The results are invariant to global brightness change since it subtracts the mean image brightness from each pixel value [4]. Equation 1 returns a normalized cross-correlation value that is between -1 and 1. The ideal value is 1, representing exact correlation between the template and input images.

Since the formula calculates the normalized cross-correlation between two images by pixel, template matching would be less accurate if the size of the template image is not close to the size of the street sign in the input image. Therefore, a huge database of different standardized signs with different resolutions is required to find a match between our template image and input image.

As such, we attempted to address this problem by creating a database of template images that vary in size using the MATLAB imresize() function. We then iterate through each of them and pick the one that gives us the highest normalized cross-correlation.

Fig. 3. Result of Template Matching

As shown in figure 3, the input image has a size of 300*335 and the matched template image is 100*100, which has the normalized cross-correlation of 0.6547 that is higher than the other results from other template images[5].

Though template matching provides a good result, it has two major drawbacks. First, it has huge running time. Positioning the template image on each possible location in the input image and iterating through each template image makes the running time exponential. Second, a huge database is required to match the template image with the input image. Though a huge space can be saved if all the input images have the same size, a certain amount of template images is still required to detect the template image in the input image. Therefore, although we can locate a specific sign in an image using template matching, it is both time consuming and space consuming.

3. NON-STANDARDIZED SIGNS

Different from standardized signs such as do-not-enter signs and stop signs, non-standardized street signs have characters on them to indicate the name of a road. Though the characters on the signs are different, we assume all the signs have rectangular shapes and the background color of the signs is always green. Based on these two features of the non-standardized signs, given an image of street sign, color detection methods using RGB and HSV color models will be used to segment the green color area from an image. Then, Optical Character Recognition(OCR) and Maximally Stable Extremal Regions(MSER) in MATLAB are able to recognize text in the segmented area and then present the text on the original image.

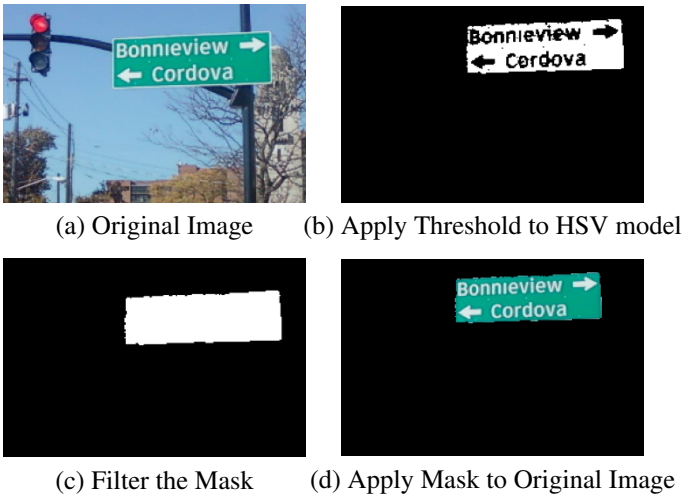
3.1. Color Detection

Two color detection methods were used to recognize the green color in an image. One is manually picking some threshold value from an RGB color model to detect the green color and the other is using a pre-determined threshold value in the HSV color model to detect the green color in the image automatically. The second method was chosen for character recognition since it requires less data feeding from the user.

The first approach for color detection is very simple. Once the image is read, the user would manually pick some number of pixels in the image that appear to have green color on the screen [6]. Using these pixels' RGB color values, the program will generate three ranges for red, green, and blue color levels and then use these ranges to threshold out all the other RGB color levels that are not in these ranges. Hence, the green area in the image will be segmented out from the original image so that it is easier for the text recognition method to detect the text in the image.

As this method gets the range of the RGB color levels from the user, if we could have predetermined this range, then the green area in images would be detected automatically. However, it is hard to use RGB color model to construct these ranges since it is an additive color model, which means different combinations of red, green, and blue color levels could possibly result in the same green color on the screen [7]. This means each non-standardized street sign image would have different ranges in the RGB color model for green and it is not possible to pre-determine the ranges for all the images. Therefore, instead of using the RGB color model, we use the HSV color model for this method of color detection, which will provide a more general predetermined range for green.

In a HSV color model, H stands for Hue, which is the pure color of an image. S stands for saturation, which is the colorfulness of an image. V stands for value, which is the whiteness of an image [8]. By setting up the hue range of an image, green color in an image is easier to detect. Here are the steps for color detection using HSV color model [9]:



(a) Original Image (b) Apply Threshold to HSV model

(c) Filter the Mask (d) Apply Mask to Original Image

Fig. 4. Color Detection Steps

As shown in figure 4, once we have the original image, we first apply the the thresholds to the HSV model of the image to get a basic mask and then filter the mask so that it would include the text area in the image. Finally, we apply the mask to the original image and get the result. Here are more results using this method:

Fig. 5. Color Detection Results

However, as shown in the figure 5, the problem with this method is that it also detects other green color area besides the street sign. However, once we use this pre-processed result image for OCR, the OCR process is able to eliminate the areas where there is no text.

3.2. OCR/MSER

The next step in the process is identifying the text on the signs. The first part of this is isolating the text in the signs from the rest of the image. The final goal of this part is placing each word on the sign in its own box. Then the built-in func-

tion OCR (Optical Character Recognition)[10] can be performed on each box, returning objects containing each word as a string.

Isolating the words is done using MSER, or Maximally Stable Extremal Regions. The MATLAB function detectMSERFeatures[11] detects MSER regions in an image. These regions are "blobs" that very similarly colored throughout. Characters on street signs are always very homogenous, so they are almost always detected by the function. However, we cannot yet perform OCR. While MSER usually detects all of the characters on the sign, it also usually detects other blobs in the image that we don?t want. First the blobs must be narrowed down to just the letters. This is done by filtering them out based on several features.[12]

The first feature used to eliminate regions is the aspect ratio, or the width divided by the height. Letters don?t tend to be overly long. Next is the eccentricity, a metric similar to the aspect ratio. Long and thin blobs have higher eccentricity. The next parameter is the solidity, a number measuring how many pixels inside the polygon traced by the blob are actually part of the blob (i.e. blobs with indents or holes have lower solidity). Extent is the next parameter. Similar (but not the same) to solidity, extent is the ratio of pixels in the region with the total number of pixels in the bounding box. Finally regions with Euler number less than -3 are discarded. Euler number indirectly measures the number of holes in a region. Characters tend to have only one, maybe two holes so regions with any more are removed.

We also use stroke width to narrow down the results. Since characters are made to be written, they tend to have consistent stroke width. Regions with inconsistent stroke width are removed.

Then the bounding boxes of the letters are merged into words. This is done by expanding the boxes slightly, and then merging boxes that overlap. A final step of filtering is done here as boxes that are not merged at all are eliminated, since characters tend to be with other characters.

Now that each word is found, OCR is performed on each box alone. Since the words are prominent within the boxes and the backgrounds are mostly consistent, OCR performs quite well. The biggest problem OCR has is with stuff on the backgrounds of signs, like highway shields.

4. CONCLUSION

We have had good success with our project, and a few problems that we can solve. We can detect a sign with template matching accurately, and we can isolate a green street sign and identify its words with only a few issues. We also have ideas for solving the problems we do have, such as time-consuming template matching or inaccurate ocr, and improving our program. First we can standardize our inputs for template matching, making the search easier and faster. Color detection can be improved with a linear regression model of the HSV color space with training images, allowing us to more accurately choose a color threshold. We also think it is possible to train a neural network to detect standard signs, which could be faster than template matching. OCR can be improved by completely isolating the final regions by removing all other pixels. It is also possible to eliminate results from OCR that contain characters not seen on street signs, such as &, @. Or #, because they are then obviously wrong.

5. REFERENCES

[1] G. Loy and A. Zelinsky, "Fast radial symmetry for detecting points of interest," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 959–973, Aug 2003.

[2] Sandro, ," http://www.mathworks.com/matlabcentral/fileexchange/45961-fast-radial-symmetry-transform.

[3] N. Barnes and A. Zelinsky, "Real-time radial symmetry for speed sign detection," in *Intelligent Vehicles Symposium, 2004 IEEE*, June 2004, pp. 566–571.

[4] "Template matching," http://docs.adaptive-vision.com/current/studio/machine_vision_guide/templatematching.html.

[5] "Documentation," http://www.mathworks.com/help/images/ref/normxcorr2.html.

[6] Gaurav Kumar, "Color detection using matlab," .

[7] "Rgb color model," *Wikipedia, the free encyclopedia*.

[8] "Hsl and hsv," *Wikipedia, the free encyclopedia*.

[9] "Simplecolordetectionbyhue() - file exchange - matlab central.," http://www.mathworks.com/matlabcentral/fileexchange/28512-simplecolordetectionbyhue--.

[10] MathWorks, ," http://www.mathworks.com/help/vision/ref/ocr.html?searchHighlight=ocr.

[11] MathWorks, ," http://www.mathworks.com/help/vision/ref/detectmserfeatures.html?s_tid=gn_loc_drop.

[12] MathWorks, ," http://www.mathworks.com/help/vision/examples/automatically-detect-and-recognize-text-in-nat
html.