

# **Yield Sign Detection**

Felix Hamann

June 4, 2017

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                             | <b>3</b>  |
| 1.1      | Requirements . . . . .                          | 3         |
| 1.2      | Notation . . . . .                              | 3         |
| <b>2</b> | <b>The filter pipeline</b>                      | <b>5</b>  |
| 2.1      | Binarization . . . . .                          | 5         |
| 2.2      | Morphological operations . . . . .              | 6         |
| 2.2.1    | Erosion and Dilation . . . . .                  | 7         |
| 2.2.2    | Opening and closing . . . . .                   | 8         |
| 2.2.3    | Application to the system . . . . .             | 8         |
| 2.3      | Component labeling . . . . .                    | 8         |
| 2.4      | Edge Detection . . . . .                        | 9         |
| 2.5      | Line Detection . . . . .                        | 9         |
| 2.6      | Yield Sign Detection . . . . .                  | 9         |
| 2.6.1    | Finding points of intersection . . . . .        | 9         |
| 2.6.2    | Determining Triangles and Yield Signs . . . . . | 10        |
| <b>3</b> | <b>Evaluation and Conclusion</b>                | <b>12</b> |

# Chapter 1

## Introduction

This document describes the implementation of a system to detect yield signs from images. **To be written: some introduction**

### 1.1 Requirements

The system should be able to detect yield signs from the perspective of a car participating in traffic. This means that the point of view is approximately between one and two meters from the ground on the right lane<sup>1</sup> of a street. There are no hard requirements regarding size and quality of the images. For testing purposes both small images with inferior quality and high resolution images should be tested. This is necessary to make a prediction whether the system is suitable for real-time use in videos. Videos may be cropped and shrunk to fit these requirements. The detection should work with changing ambient light. Without white balancing, the standardized red color value of yield signs changes when captured by camera based on the daytime. The color also varies greatly when the sunlight is reflected on the sign.

### 1.2 Notation

All images are either two or three dimensional. Two dimensional images are either grayscale or binary images, three dimensional images are color images with three color components: red, green and blue (in this order). Origin of each image is the top left corner and iteration is row-major. This is reflected when addressing single pixel values. Given some image  $I$ , then  $I(y, x)$  returns the pixel  $y$  in the images' vertical and  $x$  in the images' horizontal direction. Binary images are two dimensional and every value is in  $\{0, 1\}$ .

---

<sup>1</sup>Except for countries with left hand traffic that are not considered here.

Thus, boolean operations can be applied. Pixel values of 0 are called “black” or “false” and values of 1 are called “white” or “true”. Each step in the pipeline that is described in chapter 2 has a *source* image which is the result of the previous step of the pipeline and a *target* image that is written to.

## Chapter 2

# The filter pipeline

...

### 1. Binarization

- (a) Red Amplification
- (b) Binarization

## 2.1 Binarization

The first step of the whole pipeline consists of extracting the red color from the coloured input image *source*. Thus a mapping for each pixel from three dimensions to one dimension is needed. Simply returning the red color component does not suffice, as with larger values of the green and blue components the source color either shifts towards yellow, magenta or white. A very simple method that is used in this application takes the red component, optionally amplifies it by some factor and subtracts the green and blue components values.

Let  $\Gamma = \{0, 1, \dots, 255\}$  be all possible pixel values,  $\delta \in \Gamma$  the threshold and  $\alpha \in \mathbb{R}, \alpha \geq 1$  the factor for red amplification, then the pixel written to the binarized image *target* is:

$$\begin{aligned} f : \Gamma \times \Gamma \times \Gamma &\rightarrow \mathbb{Z} \\ f(r, g, b) &= \alpha r - (g + b) \\ target(y, x) &= \begin{cases} 1 & \text{if } f(source(y, x)) > \delta \\ 0 & \text{else} \end{cases} \end{aligned} \tag{2.1}$$








| Color   | Red | Green | Blue | $f(r, g, b)$ | $target(y, x)$ |
|---|-----|-------|------|--------------|----------------|
|  | 0   | 0     | 0    | 0            | 0              |
|   | 255 | 255   | 255  | -51          | 0              |
|  | 0   | 255   | 0    | -255         | 0              |
|  | 0   | 0     | 255  | -255         | 0              |
|  | 255 | 0     | 0    | 459          | 1              |
|  | 255 | 125   | 125  | 209          | 1              |
|  | 255 | 125   | 0    | 334          | 1              |
|  | 255 | 0     | 126  | 334          | 1              |

Table 2.1: Example results of function  $f$  in 2.1 with  $\alpha = 1.8$ ,  $\delta = 200$ . Note that the result is equivalent for light red, orange and magenta.

This method has one drawback however. There is no distinction between less saturated or darker shades of red and orange or magenta. This is a result of not taking the distance between green and blue into account. An example is depicted in Table 2.1. More elaborate variations of this calculation were tested (considering the ratio of red to all colors or weighting by green-blue distance) but this did not increase the overall quality of red detection. Actually, without constantly white balancing the camera used for capturing, narrowing the range of red would perform worse as the ambient light differs greatly over the course of a day. The drawback of this approach are more falsely marked areas of the image that have to be considered in all further steps.

*Performance, Pictures*

## 2.2 Morphological operations

At this point of the pipeline, the *source* image is a binary image with all areas considered red marked true. The quality of the image and found red areas depends greatly on a variety of factors. For one the camera itself might introduce pixel errors to the picture, resulting in falsely colored pixels (when the sensor is of inferior quality or getting old) or blacked out areas (e.g. when the lens is dirty). On the other hand the to-be-detected yield sign could be dirty, altered by vandalism or simply reflect light which would obscure its shape. As it is essential to expose the signs form for best detection, some morphological operations may be applied for noise removal.

The system implements two morphological operations that are separately adjustable. These operations are called dilation and erosion. This is called “closing” when combined. The purpose and functioning of these algorithms are described below.

### 2.2.1 Erosion and Dilation

Main purpose of erosion is to remove white noise from the binary image. For every pixel of the *source* image, some range around the pixel is considered. This range - as used the current implementation - can be seen in 2.2 and is commonly called “structuring element”  $S$ . Now, for every pixel in the *source* image, the mask is applied and the target pixel is set based on 2.3. Commonly speaking, the target pixel is only set if all surrounding pixels masked by  $S$  are also set.

$$S_{y,x} = \begin{bmatrix} - & source(y-1, x) & - \\ source(y, x-1) & source(y, x) & source(y, x+1) \\ - & source(y+1, x) & - \end{bmatrix} \quad (2.2)$$

$$target(y, x) = 1 \iff \bigwedge_{s \in S_{y,x}} s \neq 0 \quad (2.3)$$

The functioning of the algorithm is depicted in 2.4. Closed white components shrink by the factor determined by the size of  $S$ . Spurious grains of white are eliminated completely. Thus the erosion is used to remove white noise on black backgrounds such as the grain on the bottom right of the example.

$$source = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mapsto target = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.4)$$

The dilation operation alters the image in such way that closed white forms expand by the factor determined by  $S$ . Grains of black are removed from white backgrounds. The algorithm itself works analogous to the erosion algorithm but pixels in the target image are set if any of the masked source pixels is white. The equation changes accordingly and is defined in 2.5. An example is given in 2.6. It can be observed that the white form grows and the black grains inside that form vanish. Note that the form is not smoothed and the cut propagates to the target.

$$target(y, x) = 1 \iff \bigvee_{s \in S_{y,x}} s \neq 0 \quad (2.5)$$

$$source = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \mapsto target = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (2.6)$$

### 2.2.2 Opening and closing

When erosion and dilation are combined they form an operation called opening. This is due to the fact that if any forms are connected by thinner areas this connection vanishes through erosion and the original forms size is restored by dilating again afterwards. The opposite effect occurs when first dilating and then eroding the image. Here, thin connections are strengthened and connections between forms may be established. This proves useful if a closed yield sign shapes are required even if somebody took a pencil and drew over the red area or light reflections break the red surface.

### 2.2.3 Application to the system

As stated before, closing is embedded to the system. Erosion nearly never proved useful however. Most of the time the already small red areas spanned by the yield signs are lost. This leads to never detecting any yield signs that are far away. The number of iterations are free parameters of the system and may be adjusted. Using one or two iterations of dilation and one or zero iterations of erosion work fairly well.

Picture examples

## 2.3 Component labeling

This optional step tries to greatly reduce the impact of red image areas that are not yield signs. The basic idea is that yield signs enclose a white triangle and thus there are black areas enclosed by closed white borders in the binary image. If all black areas are labeled, the label with the most associated pixels must be the background - assuming the yield sign is not the most prominent object captured. All forms that do not enclose any area vanish that way, greatly reducing the amount of considered objects. Because the algorithm is defined for labeling white areas, the source image needs to be inverted first.

The algorithm works by iterating the image from top-left, selecting all white pixels and checking whether a new label needs to be introduced or - if the neighbors are already labeled adopt that label for the current pixel. Formally, given a mask  $M$  and a set  $L = \{1, 2, \dots\}$  of unassigned labels, the target pixels value



is given by formula 2.8. Because multiple values can be returned by  $M$ , another set  $E$  is needed to store all equivalents. After the first labeling step, the equivalents are resolved by assigning the smallest label of possible candidates. This algorithm is called two-pass algorithm and the form of  $L$  checks for 8-connectivity as all eight neighboring pixels are taken into account.

$$M_{y,x} = \begin{bmatrix} source(y-1, x-1) & source(y-1, x) & source(y-1, x+1) \\ source(y, x-1) & source(y, x) & - \end{bmatrix} \quad (2.7)$$

$$target(y, x) = \begin{cases} min(L) & L = L \setminus l & \text{if } \forall m \in M_{y,x} : m = 0 \\ any(M) & E = E \cup \{(m, n)\} \quad m, n \in M_{y,x} & \text{else} \end{cases} \quad (2.8)$$

After labeling all distinct components of the image, the component with the largest amount of associated pixels is set to white and the image is again reversed. Although this step proves to be very effective it has one big drawback: As soon as the detected yield sign shape is not closed the whole detection system fails immediately. This does not happen otherwise as even without a closed form, the other edges are prominent enough to allow detection. Thus this step can optionally be enabled or disabled.

Picture examples

## 2.4 Edge Detection

To be written: Either Dilation + XOR or Sobel for Fast Hough

## 2.5 Line Detection

To be written: Either Hough or Fast Hough

## 2.6 Yield Sign Detection

To be written

### 2.6.1 Finding points of intersection

The vectors returned by the hough transform contain the respective values describing each found line by their normal vector and distance from the images' origin. A simple method for detecting points of intersections is to create homogeneous vectors and use their cross product. If the third component is

zero, no point of intersection is found. Let  $A$  be a vector containing all radii and  $\Delta$  contain all distances for  $n$  found lines. Thus for any  $i \in 1, \dots, n$  the homogeneous vector  $h_i$  is defined as:

$$\begin{aligned} A &= (\alpha_1, \alpha_2, \dots, \alpha_n) \\ \Delta &= (\delta_1, \delta_2, \dots, \delta_n) \end{aligned} \quad (2.9)$$

$$v_i = \begin{pmatrix} \sin(\alpha_i) * \delta_i \\ \cos(\alpha_i) * \delta_i \\ 1 \end{pmatrix} \quad w_i = \begin{pmatrix} \cos(\alpha_i) \\ -\sin(\alpha_i) \\ 0 \end{pmatrix} + v_i \quad h_i = v_i \times w_i \quad (2.10)$$

Now all 2-permutations  $(i, j) \in \pi_2(n)$  are tried for calculating intersections  $p_{i,j} = h_i \times h_j$ . Thus the set of all intersections is  $\{(y, x)_{i,j} | y = \frac{p_{i,j,0}}{p_{i,j,2}}, x = \frac{p_{i,j,1}}{p_{i,j,2}} \forall p_{i,j,2} \neq 0\}$ . Also, to keep all further computation small even if a lot of lines are found, the points of intersection are filtered by whether there is some red to be found nearby. To do so, the result of the morphological computations (section 2.2) is consulted and the point of intersection is kept if any white pixel is found there. The points are further filtered by whether they are inside the image plus some boundary. Even for truncated signs the relevant point of intersection is near the image boundary. The parameters for controlling the size of both the lookup area and the image boundary are free parameters. The intersections are saved as an unambiguous mapping of  $T = \min(i, j) \rightarrow \max(i, j) \rightarrow (y, x)_{i,j}$ .

### 2.6.2 Determining Triangles and Yield Signs

All the previously detected candidates are now iterated by checking whether any  $i, j, k$  exists where a combination of intersections is transitive, meaning  $(i, j, p_0), (j, k, p_1), (i, k, p_2) \in T$ . If the condition holds, than the three points  $p_0, p_1, p_2$  are the vertices of a triangle and the tuple  $(i, \min(j, k), \max(j, k))$  unambiguously identifies it. Now they are be filtered further by discarding all triangles whose ratio does not fit (yield signs are nearly equilateral, even when taking perspective into account). Also signs can be discarded whose total size exceeds or undercuts some threshold. These thresholds can be calculated based on the images size.

The last step of detecting whether the found triangle actually corresponds to a yield sign consists of looking at the actual shape of the triangle itself. To do so, a reference triangle is created, mimicking the shape of a yield sign, and an equally sized image with a triangle having the shape of the detected one. Both images are binary images where the triangles are white. Let  $ref$  be the reference triangle image and  $tri$  the detected triangle,  $h$  the images' height and  $w$  the images' width, then the difference between the two is defined in 2.11. The resulting real number expresses the similarity between the two and the larger that value gets the more different the triangles are. Whether the triangle is classified as a yield sign

is controlled by a free parameter threshold. This threshold can be raised to detect yield signs that are distorted by perspective.

$$\Delta = \frac{1}{h * w} \left( \sum_{0 \leq y < h} \sum_{0 \leq x < w} ref(y, x) \oplus tri(y, x) \right) \quad (2.11)$$

## **Chapter 3**

# **Evaluation and Conclusion**

To be written