

Yield Sign Detection

Felix Hamann

June 4, 2017

Contents

1	Introduction	3
1.1	Requirements	3
1.2	Notation	3
2	The pipeline	5
2.1	Segmentation	6
2.2	Morphological operations	7
2.2.1	Erosion and Dilation	8
2.2.2	Opening and closing	9
2.3	Component labeling	9
2.4	Edge Detection	11
2.5	Line Detection	11
2.6	Yield Sign Detection	11
2.6.1	Finding points of intersection	11
2.6.2	Determining Triangles and Yield Signs	12
3	Evaluation and Conclusion	14
3.1	Performance	14
3.2	Application to Videos	14
3.3	Prospect	14

Chapter 1

Introduction

This document describes the implementation of a system to detect yield signs from images. **To be written:**
Some general introduction (problem description, scope, constraints, reason)

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 lane¹ 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 $\{0, 1\}$.

¹Except 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.

To do: Nice picture of the final result

Chapter 2

The pipeline

For yield sign detection a modular system of components is used. Each step of the pipeline takes at least the result of the previous step as input for processing. Simply put, color images are fed into the pipeline and a set of yield sign descriptions is returned. Each element of this set contains all three vertices and the center of the sign.

The pipeline itself consists of the following components: First the red color component of the image is extracted. A threshold is introduced to create a binary image masking all red areas. This is described in section 2.1. The next component applies morphological operations to enhance the quality of the extracted areas. A description of this can be found in section 2.2. The following section 2.3 describes how the amount of relevant red areas can be greatly reduced by filling the “background” of the image utilizing component labeling. As this proved quite useful but hindered detection completely for some corner cases, this step can be enabled optionally. The now remaining relevant areas are transformed in such way that their shapes are defined by their edges (either dilation + xor or sobel). Section 2.4 describes this and the succeeding section 2.5 explains how this information can be used to detect and extract lines from the

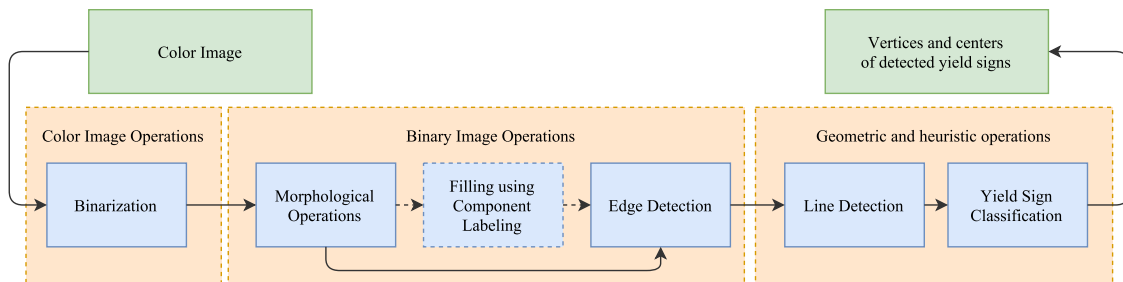


Figure 2.1: A high level overview of the processing pipeline

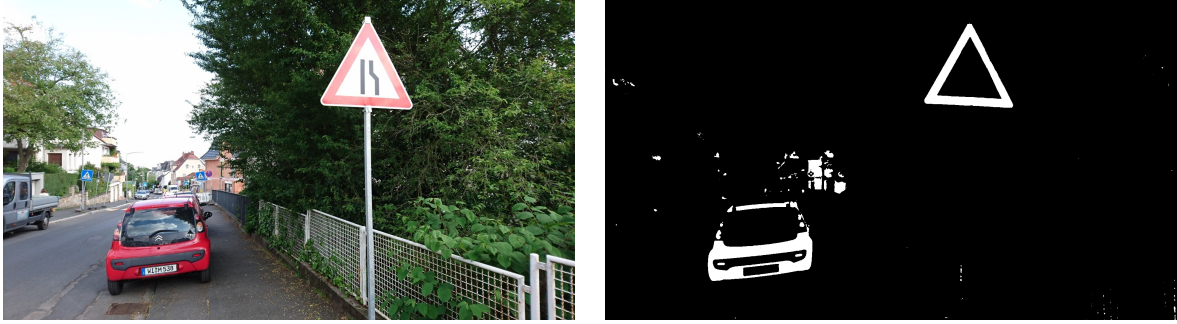


Figure 2.2: Color segmentation to create a binary image with red areas marked true. The sign in the source image reflects a lot of sunlight which lessens the saturation. Used parameters are $\alpha = 1.8$, $\delta = 0.3 * 255$.

image. After all lines are extracted section 2.6 discusses how some heuristics are applied to determine all relevant triangles and classify whether those triangles could be yield signs. A graphical depiction of this process is given in 2.1.

To do: Apply a median filter before or after segmentation?

2.1 Segmentation

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:

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

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








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.

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.

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.

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

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

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. The number of iterations are free parameters of the system and may be adjusted. Introducing another erosion step to remove white noise mostly resulted in eroding the yield sign itself and is therefore no longer part of the pipeline.

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.

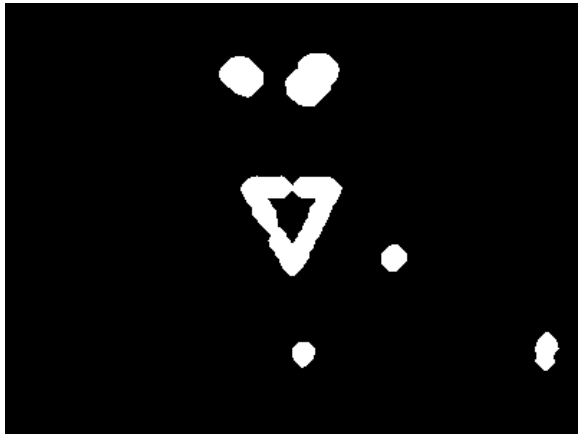
The algorithm works by iterating the image from top-left, selecting all white pixels and checking whether a new label needs to be introduced or - if the neighbors are already labeled adopt that label for the current pixel. Formally, given a mask M and a set $L = \{1, 2, \dots\}$ of unassigned labels,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



(a) Original image



(b) Segmented image



(c) After dilation



(d) After erosion

Figure 2.3: Closing applied to an image with a vandalized yield sign. Note how the form is closed in 2.3d on the left side. However, the number of iterations were not sufficient to close the form completely. All noise is removed from the red lights.

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

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

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.

To do: Example pictures

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 Δ 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. An example of why this a useful step is given in figure 2.4 where a triangle is falsely classified as a yield sign although there is no sign remotely close. 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 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



Figure 2.4: An example where a triangle with the shape of a yield sign is found if no prior filtering of intersections based on detected red value is applied.

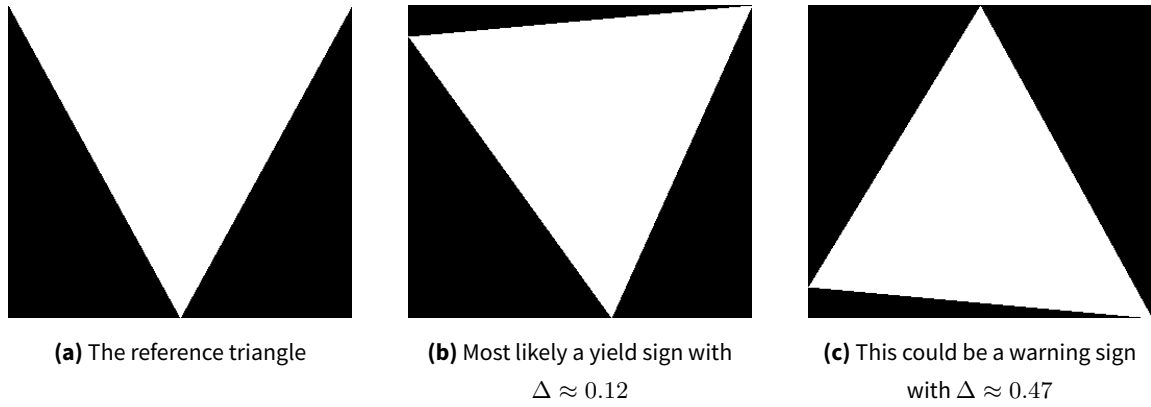


Figure 2.5: Example results of equation 2.11

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 reference images' height and w the reference images' width, then the difference between the two is defined in 2.11. The resulting real number $\Delta \in [0, 1]$ 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

3.1 Performance

To be written

3.2 Application to Videos

To be written (ROI, Frame-Skipping)

3.3 Prospect

To be written (Other approaches: k-means clustering for segmentation, Otsu? ...)