

Yield Sign Detection

Felix Hamann

June 3, 2017

Contents

1	Introduction	3
1.1	Filter pipeline overview	4
1.2	Binarization	5
1.3	Morphological operations	5
1.3.1	Erosion and Dilation	6
1.3.2	Opening and closing	7
1.3.3	Application to the system	7

Chapter 1

Introduction

This document describes the implementation of a system to detect yield signs from images.

1. *Notation definitions*

- (a) *Always height x width*

- (b) *For every step in the pipeline there is 'source' and 'target'*

2. *Image test set*

3. *Requirements*

- (a) *Changing light (night, day, dawn, dusk)*

- (b) *Image sizes*

- (c) *Performance*

1.1 Filter pipeline overview

...

1. Binarization

(a) Red Amplification

(b) Binarization

1.2 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:

$$\begin{aligned} f : \Gamma \times \Gamma \times \Gamma &\rightarrow \mathbb{Z} \\ f(r, g, b) &= \alpha r - (g + b) \\ target(y, x) &= \begin{cases} 1 & \text{if } f(source(y, x)) > \delta \\ 0 & \text{else} \end{cases} \end{aligned} \tag{1.1}$$

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 into account. An example is depicted in Table 1.1. 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

1.3 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.








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 1.1: Example results of function f in 1.1 with $\alpha = 1.8$, $\delta = 200$. Note that the result is equivalent for light red, orange and magenta.

The system implements two optional steps with four morphological operations that are separately adjustable. The first step consists of dilation and erosion, which is called “closing”, the second step of erosion and dilation, called “opening”. This also means it is possible to first erode, then dilate (and optionally erode again). The purpose and functioning of these algorithms are described below.

1.3.1 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 1.2 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 1.3. Commonly speaking, the target pixel is only set if all surrounding pixels masked by S are also set.

$$S = \begin{bmatrix} - & source(y-1, x) & - \\ source(y, x-1) & source(y, x) & source(y, x+1) \\ - & source(y+1, x) & - \end{bmatrix} \quad (1.2)$$

$$target(y, x) = 1 \iff \bigwedge_{s \in S} s \neq 0 \quad (1.3)$$

The functioning of the algorithm is depicted in 1.4. 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.

$$source = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mapsto target = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1.4)$$

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 1.5. An example is given in 1.6. 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.

$$target(y, x) = 1 \iff \bigvee_{s \in S} s \neq 0 \quad (1.5)$$

$$source = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \mapsto target = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (1.6)$$

1.3.2 Opening and closing

...

1.3.3 Application to the system

Iterations as free parameters.