DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Android-based camera-based traffic sign detection in driving simulator data with OpenCV

Tobias Bauer

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Android-based camera-based traffic sign detection in driving simulator data with OpenCV

Android-basierte Kamera-gestützte Verkehrschilder-Erkennung in Fahrsimulator-Daten mit OpenCV

Author:	Tobias Bauer
Supervisor:	Prof. Dr. Uwe Baumgarten
Advisor:	Sebastian Eckl, M.Sc.
Submission Date:	15.02.2018

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.02.2018 Tobias Bauer

Acknowledgments

Abstract

Contents

1 Introduction

1.1 Section

1.1.1 Subsection

2 Motivation

2.1 Section

Citation test [**latex**].

2.1.1 Subsection

SubSub

3 Related Work

3.1 Section

Citation test [**latex**].

3.1.1 Subsection

4 Neural Netwokrs

4.1 Fundamentals

Citation test [**latex**].

4.2 Common Types

5 Concept

The task of looking at a picture and telling what is in it, is quite trivial to humans and would usually not pose a challenging task to perform, whereas it is not that simple to Computers. In order to Classify images, or more exactly the objects comprised, machines ordinarily need a certain amount of Preprocessing to extract characteristic Features. As described in chapter 4 it is common practice to partition the many small steps of the process in two bigger, more generalized, steps of Feature Extraction and Classification. Our targeted devices are going to be any smartphones running Android as operating system, basically the App on the device should be able to handle live input from the Camera, detecting and classifying in real time. In the following subsections I am going to break down in detail, what methods are used and how they work.
Komponentendiagramm

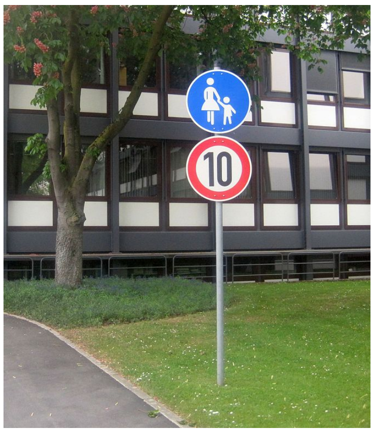


Figure 5.1: Original image

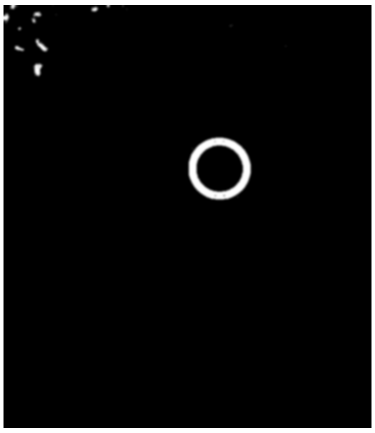


Figure 5.2: Filtered Image

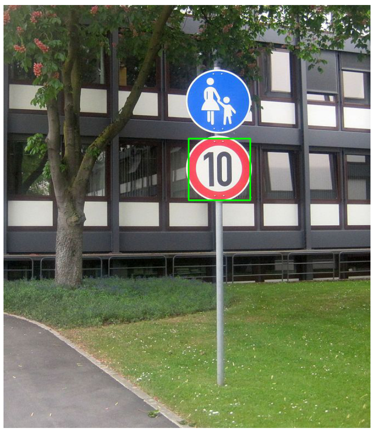


Figure 5.3: Detected image

5.1 Feature Extraction

Before we pass data through to the Classification phase, it is a reasonable step in our Program, to specify a Region Of Interest, with regard to saving some processing power. In order to specify ROIs, I took advantage of the red circle around European Speed signs, the main steps of this part are to filter every red pixel out of the original image (Figure: 5.2), subsequently an algorithm checks the filtered image for circles (Figure: 5.3). Filtering red pixels from a picture is not that big of a sorcery, so i am going to skip that part for now and come back later to this in the implementation section. The real magic happens when you try to search an image for circles, therefore a common method is the Circle Hough Transform.

5.1.1 Circle Hough Transform

The Circle Hough Transform is a special case of the Hough transform, which determines lines in an image through Edge Detection Algorithms. There are two slightly different Circle Hough Transforms, the first is only able to find circles of fixed radial size, the second is generally capable of finding Circles of any size. At first I am just going to start with the fixed radius form of the algorithm, subsequently we are going to extend that approach to a more universal algorithm.

Fixed Radius

First of all our image is being Converted to a binary image (Figure: 5.4), further it is being processed via edge detection to determine every visible edge contained in the photo (Figure: 5.5). After this, the actual core of the Algorithm is being executed. Therefore an Accumulator Matrix is initialized with zeros. As the radius of the circle is considered constant, every ring in the picture can solely be described by Above-mentioned Matrix can be pictured as a grid over the original image, where single pixels can be combined to bigger baskets. Now every basket or pixel of a detected edge (Figure: 5.5) becomes the center of a new circle, with given fixed radius. Now each pixel that got "cut" by the circle gets incremented or up-voted in corresponding Accumulator Matrix (Figure 5.6). The baskets or pixels with the highest votes can be considered centers of circles.

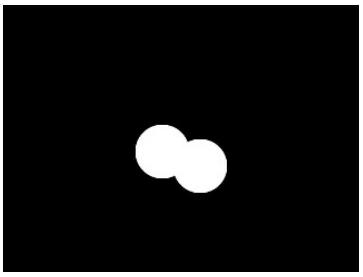


Figure 5.4: Binary image

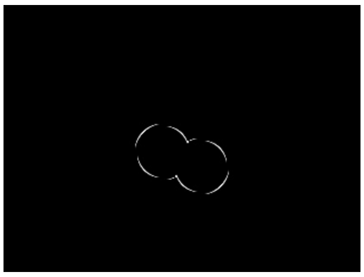


Figure 5.5: Edge detected

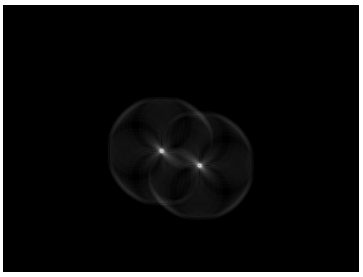


Figure 5.6: Voted image

Variable Radius

A circle on a two-dimensional plane can be described by following formula:

$$(x-a)^2 + (y-b)^2 = r^2$$

, meaning the circle can be represented by the parameters (a, b, r), where "a" and "b" are the "x" respectively "y" value representing the center of the circle, whereas "r" stores the radius of given ring.

5.2 Classification

Preceding stage delivers an image of fixed size, that needs to be categorized into one out of 10 groups, these are Speed limits "10km/h", "20km/h",..., "90km/h" and "no sign". To gain these information about the sub-image of the original image, I decided to apply a Feed Forward Deep Neural Network. Said Neural Network is fed each individual pixel, provided by the ROI to its input neurons, the trained Model then outputs a prediction on what class the input picture might belong to.
wieso nur bis 90?
Neuronales netz model

6 Integration into existing App

6.1 Overview and existing functions of App

The App I had to build upon, already had some features related to the open source car driving simulator "Speed Dreams", but before we go deeper into theses functionalities, I am going to start with the present structure and Appearance. First of all, a dark theme has been chosen for the design of the App. As can be seen in Figure 6.1, the main screen contains different tiles for each task. It is possible to choose the source of intended input, therefore the user decides between loading an existing image (Figure 6.2), or hitting the camera icon to call a life-video feed of the back-facing camera. Depending on which input source has been chosen, the functionalities are applied either one single time or multiple times per second, both ways the outputs are printed onto the original images. Both features, that are named "Vehicle Detection" and "Lane Detection" are pretty much self explaining, but it should be mentioned that both methods are very likely trained by "Speed Dreams" data, meaning that the Lane detection is good for detecting lanes in the game, but might not be that brilliant in detecting real lanes, furthermore the vehicle detection recognizes only the back of the red car in the virtual rally course.

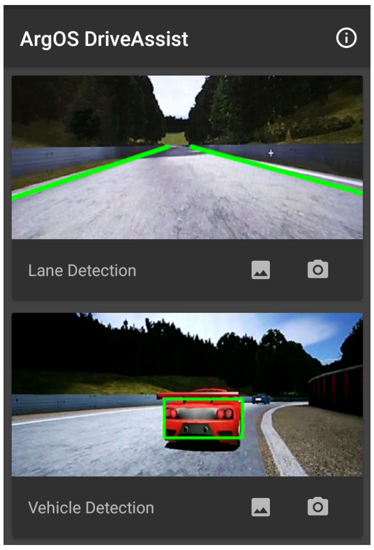


Figure 6.1: Main screen

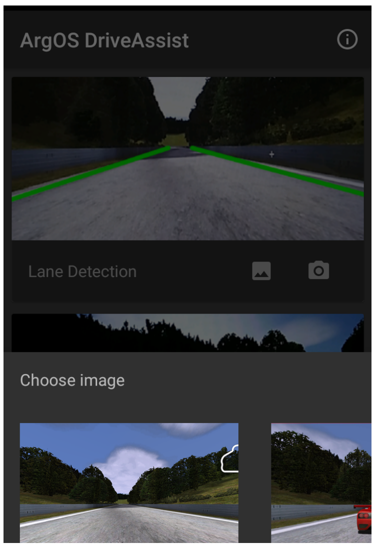
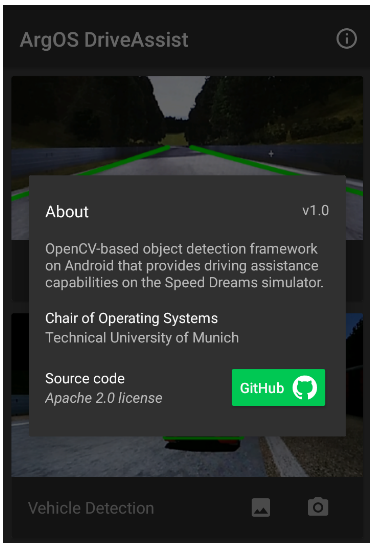


Figure 6.2: Load Image



OpenCV-based object detection framework on Android that provides driving assistance capabilities on the Speed Dreams simulator.

Chair of Operating Systems
Technical University of Munich

Source code
Apache 2.0 license

Figure 6.3: About

7 Implementation

7.1 Android Studio

7.2 Native Development Kit

7.3 Image processing (Opencv)

7.4 Neural Net Classification

7.4.1 Building Neural Network

8 Evaluation

8.1 Section

Citation test [**latex**].

8.1.1 Subsection

List of Figures

List of Tables