Hack&Slash: Gesture Recognition for Dashboards

Seminar Thesis

Nico Rausch
Richard Tran
Giorgio Groß
1856484 - Richard - 1938927

ToDo

At the Department of Economics and Management
Institute of Information Systems and Marketing (IISM)
Information & Market Engineering

Reviewer: Prof. Dr. Alexander Mädche
Advisor: Eric Pescera (TECO)

20th of July 2017

Contents

1. Introduction

Nowadays gesture recognition has achieved a high usage and meaning, especially for medical purposes and industrial production processes. These new technologies are available not only for industrial purposes, but also for the mass market, for which devices like the Xbox Kinect or the Wii Remote set the basis a few years ago. Until now, many research groups (Zitation) have already worked on technologies which enable the user to control computers with gestures. Most of these devices work vision-based, i.e. with cameras which recognize the visual changes caused by body movement and translate them into computer commands. These applications have the disadvantage that the user cannot move freely in the room, since he is bounded to the space captured by the camera in order to transmit the commands properly. This can be impractical for specific situations, e.g. presentations, when flexibility is needed considering the movement in the room. **ToDo**

Furthermore, there are also several technologies which track the physical data of the movement with the help of sensors (accelerometer, gyroscope, etc.) to recognize a gesture. These can be used independently from the location of the user in the room, but often demands expensive equipment, for instance a data glove. In order to develop a cheap and accurate sensor based gesture recognition device, different approaches use data from an accelerometer to classify gestures. These show a high accuracy, but are limited in the number of different identifiable gestures due to the less data.

In our work, we used the "Thunderboard React", which is a development platform with several sensors including an accelerometer and a gyroscope. We examined, how well such a sensor platform is suited for gesture recognition and if we could get even better results than previous works in terms of accuracy and number of recognizable gestures. We did this by calculating "features" like the maximal acceleration from the collected data and classified these with different machine learning approaches. Overall, we could recognize 4 different gestures with an average accuracy of 95 %. Furthermore, we created a dashboard to test the usability of our implementation. **ToDo** **ToDo**

2. Dashboard

The main goal of our seminar work was to implement a gesture recognition program with which we can reliably control applications like dashboards on the computer. Therefore, we created a test dashboard with which we examined the usability of our program. It consists of several diagrams, which can be modified by steering elements like a dropdown menu and radio buttons. The main characteristic of the dashboard is that it can be controlled only with the keyboard

ToDo

Of course, this is just a simplified example of a dashboard with only a few features compared to what can be found in actual corporate dashboards, which allow more sophisticated commands. Nevertheless, our dashboard can be easily extended with further features by implementing more steering elements like slide bars or check buttons and by adding more recognizable gestures with which these elements can be controlled.

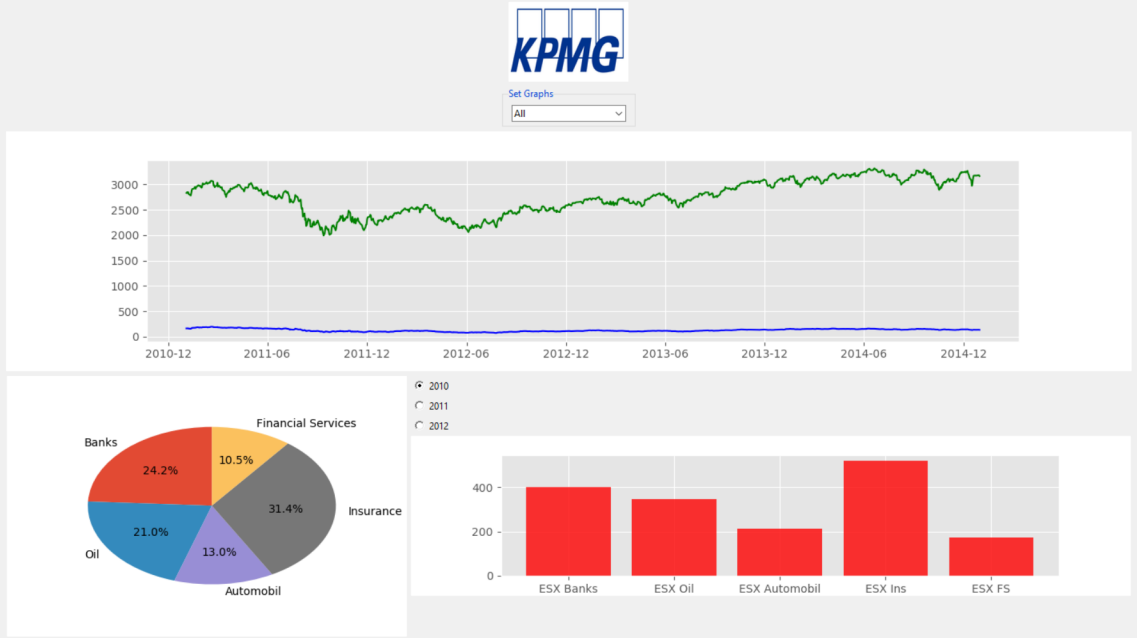

Figure 2.1.: Our demo application front end

3. System Design

...

4. Data Collection & Manipulation

Working with machine learning algorithms requires certain actions in order to choose, prepare and classify data streams. Choosing the right data is the key decision in order to be able to train the algorithm properly. In addition to this step, preparing data and transforming it into "features" is equally important.

4.1. Data Collection

To define which data will be collected we had to think about which kind of gestures we will want to recognize. Attaching the sensors to the back of our hand enables us to observe acceleration data for x, y, and z axis as well as tilt angles alpha, beta and gamma. We thought about attaching a magnet to our thumb so that we can use the hall-sensor as well, but we decided to leave this for further research. The collected acceleration and tilt angle data can later be used to distinguish gestures from each other. To avoid recording and scanning data observed during normal gesticulation we require the acceleration vector length of each gesture to exceed a certain threshold. In our code we set this treshold to 1.2 G.

4.2. Feature Design

Before we can train our algorithm, we have to transform the recorded data into a set of features. Choosing and designing those features affects the performance of the gesture recognition algorithm heavily.

4.2.1. Preprocessing

Standardization, scaling, normalization and binarization are common preprocessing techniques. Though, as the acceleration vector and tilt angel sizes are key characteristics to distinguish gestures scaling, normalization and binarization are not applicable for our use case.

ToDo

4.2.2. Features

Add images of feature selection We first thought about using the variance as a feature, but early tests already revealed that the acceleration data often has a variance near zero and thus gives us almost no information about the executed gesture.

Instead, maximum and minimum acceleration per axis differ widely across gestures and are worth adding to our feature set. To give our classifier a idea of which minimum/maximum was the first, we put them in the right order, so if we have a maximum first, this will be the first feature and the miniumum will become the second feature. For instance, swiping left first shows a negative minimum acceleration, e.g. -1.7G, and then a positive maximum acceleration, e.g. 1.2G, on the x axis. We now pick those values out of our buffer and

figure out which one was earlier, so we can put them in our feature vector in the right order. In this example, we end up with the tuple (-1.7, 1.2).

To include the tilt angles into our feature set, we had to come up with a different approach. Not only was their variance near zero, but also their minimum and maximum values are often equally great across different gestures. Summing up all angles led to non-satisfying results, so we collect now two features per angle: One sums up the absolute values of negative angle differences and the other one sums up the absolute values of the positive angle differences. Comparing both approaches the latter one increased our average precision score by 6%. Include stats "First Approach" vs "Final Solution" here **ToDo**

4.3. Data Classification

...

Appendix

A. First Appendix Section

...