**DIPLOMARBEIT**

Gesamtprojekt

**Camera Controlled Swarm Robots**

**Positionserkennung**

Mottl Mario 5BHEL Betreuer: Dipl.-Ing Josef Reisinger

**Swarm Controll**

Clemens Pruggmayer 5BHEL Betreuer: Dipl.-Ing Josef Reisinger

**Visualisierung und Simulation**

Michael reim 5BHEL Betreuer/in: Dipl.-Ing Josef Reisinger

Schuljahr 2020/21

Abgabevermerk:

Datum: TT.MM.JJJJ übernommen von:

****

**Höhere Technische Bundeslehranstalt Hollabrunn**

**Höhere Lehranstalt für Elektronik und Technische Informatik**

**EIDESSTATTLICHE ERKLÄRUNG**

**Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.**

**Mario Mottl**

**Clemens Pruggmayer**

**Michael Reim**

Hollabrunn, am 5. April 2019

**DIPLOMARBEIT**

**DOKUMENTATION**

|  |  |
| --- | --- |
| Namen der  Verfasser/innen | Michael Reim, Clemens Pruggmayer, Mario Mottl |
| Jahrgang  Schuljahr | 5BHEL |
| Thema der Diplomarbeit | Camera Controlled Swarm Robots |
| Kooperationspartner | - |

|  |  |
| --- | --- |
| Aufgabenstellung | Es sollen mehrere 1 bis n viele autonome Fahrzeuge (STM32F107RB + MDDS Board) über eine 1,5m erhöhten Kamera erfasst werden. Positionen der Fahrzeuge sollen in (x/y) Koordinaten verwandelt werden. Diese Daten sollen an die Visualisierung und Simulation geschickt. Diese erzeugt über selbst gezeichnete Bilder einen Weg für die Fahrzeuge. Der Weg soll an das Swarm Controll weitergeschickt werden welche die Positionsdaten mithilfe der Kamera in Fahrkommandos umwandelt. |

|  |  |
| --- | --- |
| Realisierung | Zur Erfassung der Autos wurde eine „DFK 33UX273“ von ImagingSource verwendet. Diese wird per USB an einen Laptop/Computer angeschlossen. Für die Visualisierung und Simulation wurde eine 3D Visualisierungssoftware in C++ geschrieben. Welche die Autos in Echtzeit am Bildschirm anzeigt.  Swarm Controll wurde mithilfe des SvVis Protokoll realisiert. Auf den Autos(Cortex M4) läuft eine selbstgeschrieben RTOS-Software, die die Kommandos des SVIS Protokoll in Bewegungen umwandelt. |

|  |  |  |
| --- | --- | --- |
| Ergebnisse | Eine alte Version des SvVis Protokolls wurde abgeändert und verbessert. Eine Teststrecke wurde aufgebaut und mit einer Halterung für die Kamera erweitert. Positionserkennung wurde mithilfe von Python realisiert. Die Autosteuerungssoftware wurde selbständig mit RTOS realisiert. Visualisierung wurde von Grund auf in C++ + OpenGL geschrieben für den speziellen Anwendungsfall. |  |

|  |  |
| --- | --- |
| Typische Grafik, Foto etc.  (mit Erläuterung) | Die obige Grafik stellt den Aufbau des Gesamtsystems dar. Darin sieht man die 3 Hauptkomponenten Kamera + Kamerasoftware, Visualisierung und Swarm Controll. Daten zwischen den einzelnen Blöcken wir über TCP bereitgestellt. Verbindung zu den Autos kann auf zwei Wege bereitgestellt werden Bluetooth, Wlan. |

|  |  |
| --- | --- |
| Teilnahme an Wettbewerben,  Auszeichnungen |  |

|  |  |
| --- | --- |
| Möglichkeiten der Einsichtnahme in die Arbeit | HTL Hollabrunn  Anton Ehrenfriedstraße 10  2020 Hollabrunn |

|  |  |  |
| --- | --- | --- |
| Approbation  (Datum / Unterschrift) | Prüfer/Prüferin | Direktor/Direktorin  Abteilungsvorstand/Abteilungsvorständin |

**DIPLOMA THESIS**

**Documentation**

|  |  |
| --- | --- |
| Author(s) | Michael Reim, Clemens Pruggmayer, Mario Mottl |
| Form  Academic year | 5BHEL |
| Topic | Camera Controlled Swarm Robots |
| Co-operation partners | - |

|  |  |
| --- | --- |
| Assignment of tasks | One or more autonomous vehicle (powered by an STM32F107RB + MDDS Board) should get detected by a camera that is attached 1,5 metres above the table. The positions of the vehicles should be converted into an x-y coordinate grid. The produced data should then be sent to our visualisation and simulation for correction purposes. A picture of a path should be drawn. The generated path should then be sent of to the Swarm Controll. Where the path will be transformed into vehicle commandos. |

|  |  |  |
| --- | --- | --- |
| Realisation | For proper detection an “DFK 33UX273” from ImagingSource was used. It’s connected via USB to a Laptop. The visualisation and simulation was programmed in “C++” + “OpenGl” which draws the cars onto the screen in Realtime. Swarm Controll uses a technology called “SvVis”. The vehicles run on an self-implemented RTOS-Software (Real Time Operating System) which transforms the vehicle commandos into engine movement. |  |

|  |  |
| --- | --- |
| Results | An older version of the “SvVis” was used and altered to fit our purposes. A test track with a bracket for the camera was built. The tracking software was programmed in python. Vehicle Controll software was written with RTOS. The visualisation and simulation was programmed in “C++” + “OpenGl” for out specific purpose. |

|  |  |
| --- | --- |
| Illustrative graph, photo  (incl. explanation) | The graph above pictures the whole System. There you can see the three main components. Tracking, Swarm Controll and Visualisation and Simulation. The communication between the individual blocks is realised in TCP. The communication between laptop and cars can be established in two ways WLAN, Bluetooth. |

|  |  |
| --- | --- |
| Participation in competitions  Awards | - |

|  |  |
| --- | --- |
| Accessibility of  final project thesis | HTL Hollabrunn  Anton Ehrenfriedstraße 10  2020 Hollabrunn |

|  |  |  |
| --- | --- | --- |
| Approval  (Date / Signature) | Examiner/s | Head of Department / College |

DA Antrag und unterschriebene Erklärung aus der Diplomarbeitsdatenbank einfügen

Inhaltsverzeichnis

[1 Camera Controlled Swarm Robots 12](#_Toc64022130)

[1.1 Blockschaltbild 12](#_Toc64022131)

[1.2 Funktionsbeschreibung 12](#_Toc64022132)

[2 Positionserkennung 13](#_Toc64022133)

[2.1 Übersicht 13](#_Toc64022134)

[2.2 Aufbau 13](#_Toc64022135)

[2.3 Kamera Ansteuerung 13](#_Toc64022136)

[2.4 Erkennung Positions LEDs 13](#_Toc64022137)

[2.4.1 Algorithmus zur Erkennung 14](#_Toc64022138)

[2.4.2 Berechnung der Position 15](#_Toc64022139)

[2.5 Auswertung der Bilddaten 15](#_Toc64022140)

[2.6 Kommunikation mit Simulation / Visualisierung 15](#_Toc64022141)

[3 Visualisierung und Simulation 17](#_Toc64022142)

[3.1 Übersicht Softwarearchitektur 17](#_Toc64022143)

[3.2 Aufbau der Engine 17](#_Toc64022144)

[3.3 Erstellung von 3D-Modellen 17](#_Toc64022145)

[3.4 Bewegungen im Dreidimensionalen Raum 17](#_Toc64022146)

[3.5 Auswertung von erhaltenen Positionsdaten 17](#_Toc64022147)

[3.6 Übertragung zu Swarm Controll 17](#_Toc64022148)

[4 Swarm Controll 18](#_Toc64022149)

[4.1 Übersicht Softwarearchitektur 18](#_Toc64022150)

[4.2 Berechnung des Weges 18](#_Toc64022151)

[4.3 Konvertierung von Positionen auf Steuerbefehle (WIP) 18](#_Toc64022152)

[4.4 Steuerbefehle 18](#_Toc64022153)

[4.4.1 Aufbau von Steuerbefehlen (Protokollaufbau) 18](#_Toc64022154)

[4.4.2 Senden von Steuerbefehlen (API) 18](#_Toc64022155)

[5 Fahrzeug Software 19](#_Toc64022156)

[5.1 Übersicht der Architektur 19](#_Toc64022157)

[5.2 Kommunikation mit Swarm Controll 19](#_Toc64022158)

[5.2.1 Kommunikationsprotokoll 19](#_Toc64022159)

[5.3 Hardware Ansteuerung 19](#_Toc64022160)

[5.3.1 LED-Ansteuerung 19](#_Toc64022161)

[5.3.2 Kommunikationsmodule 19](#_Toc64022162)

[5.3.2.1 Bluetooth 19](#_Toc64022163)

[5.3.2.2 WLAN 19](#_Toc64022164)

[5.3.3 Motor Ansteuerung 19](#_Toc64022165)

[6 Ergebnisse 20](#_Toc64022166)

[6.1 Funktionalität Positionserkennung 20](#_Toc64022167)

[6.2 Steuerung der Fahrzeuge 20](#_Toc64022168)

[6.3 Simulationstest mit Pseudodaten 20](#_Toc64022169)

[6.4 Steuersoftwarte Funktionalitätstest 20](#_Toc64022170)

[6.5 Schwarmbewegung 20](#_Toc64022171)

[6.5.1 Kreis 20](#_Toc64022172)

[6.5.2 Zick-Zack 20](#_Toc64022173)

[7 Anhang 21](#_Toc64022174)

[7.1 Inbetriebnahme (f. 4Klasse TdoT) 21](#_Toc64022175)

[7.2 Projektmanagement 21](#_Toc64022176)

[7.2.1 Projektplan 21](#_Toc64022177)

[7.2.2 Projekttagebuch 21](#_Toc64022178)

[7.2.3 Projektkosten 21](#_Toc64022179)

[8 Quellenverzeichnis 22](#_Toc64022180)

[8.1 Bücher 22](#_Toc64022181)

[8.2 Onlinemedien 22](#_Toc64022182)

[8.3 Zeitschriften 22](#_Toc64022183)

Wenn Sie in Ihrer Dokumentation ausschließlich mit den „Formatvorlagen“ dieses Dokuments gearbeitet haben können Sie hier sehr einfach das Inhaltsverzeichnis automatisch aktualisieren lassen.

Klicken Sie dazu mit der rechten Maustaste in das vorhandene Inhaltsverzeichnis und wählen Sie im Kontextmenü „Felder aktualisieren“ und dann „gesamtes Verzeichnis aktualisieren“ auswählen – fertig! … diesen Text löschen!

# Camera Controlled Swarm Robots

## Blockschaltbild

## Funktionsbeschreibung

# Positionserkennung

## Übersicht

## Aufbau

## Kamera Ansteuerung

## Erkennung Positions LEDs

Für die Erkennung der Positions LEDs wird ein simpler Algorithmus verwendet. Durch die spezielle Anordnung der LEDs in einem gleichschenkeligen Dreieck kann man die ungefähre Position der anderen LEDs mit dem Satz des Pythagoras berechnen.



Um sicherzustellen, dass die Positions LEDs zu einem Auto und nicht zu einem anderen gehören. Werden immer nur zwei LEDs mit der Kamera erkannt und die Position der letzten LED wird berechnet.

### Algorithmus zur Erkennung

Für den Algorithmus wird der in OpenCV eingebaute Filter + Detektor verwendet. Die Parameter werden wie folgt gesetzt.

params = cv2.SimpleBlobDetector\_Params()

# Change thresholds

params.minThreshold = 0

params.maxThreshold = 256

# Filter by Area.

params.filterByArea = True

params.minArea = 30

# Filter by Circularity

params.filterByCircularity = True

params.minCircularity = 0.1

# Filter by Convexity

params.filterByConvexity = True

params.minConvexity = 0.5

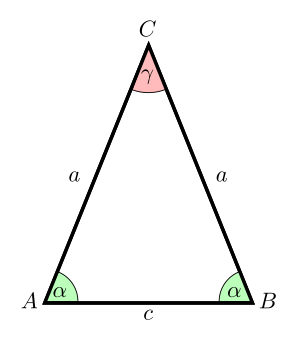
# Filter by Inertia

params.filterByInertia = True

params.minInertiaRatio = 0.5

detector = cv2.SimpleBlobDetector\_create(params)

### Berechnung der Position



Um die Seitenlängen zu bestimmen wurde einmal als Referenz a & c in Pixel gemessen.

Es wird weiteres als Kontrolle der Winkel α & γ berechnet.

Darüber hinaus wird in regelmäßigen Abständen die Seitenlänge neu berechnet. Um sicherzustellen, dass es sich noch immer um dasselbe Auto handelt.

## Auswertung der Bilddaten

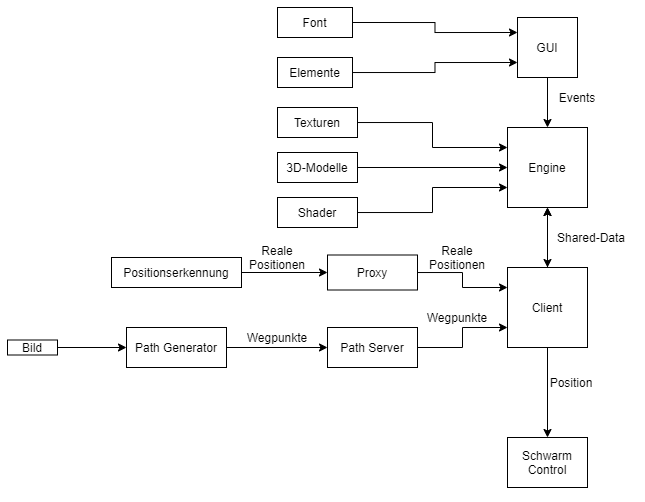
Um aus den vielen Bilddaten eine vernünftige Position zu generieren wird der Mittelpunkt des Autos berechnet. Jede Koordinate einer Positions LED wird in einem Tupel (x, y) (posLed) abgespeichert. Jedes Auto (cararray) besteht aus einem Array aus drei Tupel (posLed1, posLed2, posLed3) + Fahrtrichtung. Über diese drei Tupel wird gemittelt, um den Mittelpunkt des Fahrzeugs zu erhalten.

## Kommunikation mit Simulation / Visualisierung

# Visualisierung und Simulation

## Übersicht Softwarearchitektur

### Blockschaltbild



### GUI

Das GUI ist die graphische Oberfläche der Visualisierung. Es enthält Text und verschiedene Elemente wie zum Beispiel eine Eingabezeile. Die Schriftart, die verwendet wird, wird über ein File in das Programm eingelesen. Mehr zu diesem Thema gibt es im Punkt 3.2 (Graphical User Interface).

### Engine

Die Engine ist das Rückardt der Software. Sie beinhaltet das graphische Rendering, steuert Bewegungen und den Ablauf der Simulation und reagiert auf Events. Dazu wird im Punkt 3.3 (Aufbau der Engine) näher eingegangen.

### Client

Der Client ist für das Empfangen und Senden von Daten über das Netzwerk zuständig. Da die Engine mit dem Client kommunizieren muss, wird ein Shared-memory verwendet. Auf diesen Speicher haben nur die Engine und der Client Zugriff.

### Modelle, Texturen, Shader

Modelle beschreiben die Form der Objekte in einer Rendering-Szene. Sie beinhalten Raumpunkte, genannt Vertices, Normalvektoren und Texturkoordinaten.  
Texturen werden verwendet, um Modellen Farben zu geben. Diese sind Bild-Files, welche in den Speicher geladen werden, und anschließend am Modell angezeigt werden. Welcher Bereich einer Textur auf einem Modell angezeigt wird, wird über die Texturkoordinaten gesteuert.

Shader sind Programme, die auf der Graphikkarte ausgeführt werden. Sie berechnen mithilfe der Vertices, Texturkoordinaten und Normalvektoren die endgültige Farbe eines Pixels auf dem Bildschirm. Dieser Prozess wird „Shading“ genannt. Genauer erläutert wird das im Punkt 3.3 (Aufbau der Engine) und 3.4 (Erstellung von 3D-Modellen).

## Graphical User Interface

## Aufbau der Engine

## Erstellung von 3D-Modellen

## Bewegungen im Dreidimensionalen Raum

## Auswertung von erhaltenen Positionsdaten

## Übertragung zu Swarm Controll

# Swarm Controll

## Übersicht Softwarearchitektur

## Berechnung des Weges

## Konvertierung von Positionen auf Steuerbefehle (WIP)

## Steuerbefehle

### Aufbau von Steuerbefehlen (Protokollaufbau)

### Senden von Steuerbefehlen (API)

# Fahrzeug Software

## Übersicht der Architektur

## Kommunikation mit Swarm Controll

### Kommunikationsprotokoll

## Hardware Ansteuerung

### LED-Ansteuerung

### Kommunikationsmodule

#### Bluetooth

#### WLAN

### Motor Ansteuerung

# Ergebnisse

## Funktionalität Positionserkennung

## Steuerung der Fahrzeuge

## Simulationstest mit Pseudodaten

## Steuersoftwarte Funktionalitätstest

## Schwarmbewegung

### Kreis

### Zick-Zack

# Anhang

## Inbetriebnahme (f. 4Klasse TdoT)

## Projektmanagement

### Projektplan

### Projekttagebuch

### Projektkosten

# Quellenverzeichnis

## Bücher

## Onlinemedien

## Zeitschriften