Lastenheft der Projektphase II

Robert Heise und Florian Edenhofner –

Education4Industry GmbH

UNIVERSITY
4 INDUSTRY

Zuletzt aktualisiert: 24.02.2023

Inhaltsverzeichnis

1 Projektauftrag

Titel: Camp 2 Code Projektphase II

Thema: Raspberry Pi Car – ein kleines autonomes Auto

1.1 Beschreibung des Ziels der Projektphase II / Überblick Projektphase II

In der Projektphase I wurde Software erstellt, welche ein Modellauto mittels Sensoren steuert. In der Projektphase II soll nun in dieses Modellauto eine Kamera integriert werden, um beidseitige Fahrbahnbegrenzungen zu erkennen und einer Fahrbahn folgen zu können. Dazu steht neben der Kamera eine zugehörige Basisklasse Camera zur Ansteuerung der Kamera zur Verfügung. Die Softwareentwicklung baut auf der in der Projektphase I entwickelten Software auf. In der ersten Woche soll eine Fahrbahnverfolgung mittels "klassischer" Methoden der Bildverarbeitung unter Verwendung von OpenCV entwickelt werden. In der zweiten Woche erfolgt die Entwicklung einer Fahrbahnverfolgung mittels eines Deep-Learning-Ansatzes basierend auf TensorFlow. Parallel dazu sollen ausgewählte Daten des Autos mittels einer App visualisiert werden. Für die Verwendung der Software soll ein User Interface entwickel werden, welches dem Anwender z.B. den Start der Software ermöglicht.

1.2 Materialien/Dateien

Folgende Materialien stehen zur Verfügung:

- Modellauto
- Kamera
- Python-Files:
  - basisklassen.py (Basisklassen zur Ansteuerung der einzelnen Komponenten für die Verwendung in basecar.py)
  - basisklassen_cam.py (Basisklasse zur Ansteuerung der Kamera)
  - basecar.py (Basisklasse für Steuerung des Modellautos, alternativ zur Eigenentwicklung aus Projektphase I)

2 Projektziele

1. **Planung**

- Projektzieldefinition und -abgrenzung (schriftlich festhalten)
- Projektzeitplan (Kanban-Board)
- Teamzusammensetzung
- Aufgabenverteilung

2. **Versionierungsverwaltung** mittels einer passenden Ordnerstruktur (z.B. über Dateinamen und Archivordner). Alternativ ist auch die Verwendung von Git möglich.
3. **Installation notwendiger Software und Download bereitgestellter Dateien.** Die entsprechenden Links zu den unterschiedlichen Repositorys werden gesondert zur Verfügung gestellt. Die notwendige Software (siehe Anleitung) sollte installiert werden.
4. **Einbau und Test der Kamera.** Es muss sowohl die Hardware als auch die Basisklasse Camera getestet werden.
5. **Entwickeln und Testen einer Klasse CamCar**. Durch die Klasse soll eine Fahrspurverfolgung mittels "klassischer Methoden" mit OpenCV in Python ermöglicht werden. Die Klasse CamCar soll von der Klasse BaseCar abgeleitet werden. Mittels dieser Klasse sollen folgende Anforderungen umgesetzt werden:

    - Verfolgen einer Spur mit je 2 Linien in ca. 15 cm Abstand
    - Aufnehmen und Speichern von Bilddaten mit passenden Lenkwinkel für das Training (siehe 6.)

    Die Gestaltung der Klasse Camcar, mittels Attributen und Methoden, obliegt Ihnen.

    Anmerkung: Sie können Ihre eigene Klasse **BaseCar** aus Projektphase I oder die neu zur Verfügung gestellte Klasse nutzen.

6. **Entwickeln und Testen einer Klasse DeepCar**. Entwicklung einer Fahrspurverfolgung mittels eines Neuronalen Netzes unter Verwendung von Tensorflow/Keras. Eine Klasse DeepCar soll ebenfalls von der Klasse BaseCar abgeleitet werden. Es sollen folgende Anforderungen erfüllt werden:

    - Entwicklung und Trainieren eines Neuronalen Netzes, das ein Bild als Input erhält und den Lenkwinkel als Output zurückgibt

- Verfolgen einer Spur mit je 2 Linien in ca 15 cm Abstand ausschließlich unter Verwendung des Neuronalen Netzes

Anmerkungen: Für das Trainieren des Neuronalen Netztes ist notwendig Trainigsmaterial zu erstellen. Hier ist es hilfreich auf die Fahrspurverfolgung mit CamCar zurückzugreifen. Alternative können Sie, falls, diese Fahrspurverfolgung nicht ausreichend gut funktioniert, auf eine Remotesteuerung zurückgreifen. Diese wird Ihnen zur Verfügung gestellt. Image Augmentation kann hilfreich sein, einen Trainingsdatensatz zu erweitern.

7. **Visualiserung der Fahrdaten mit Dash**. Interessante Fahrdaten und das aktuelle Bild der Kamera sollen mittels einer Dash-App visualisiert werden.
8. **Entwicklung eines User-Interfaces** Basierend auf den Klassen CamCar und DeepCar soll einem Nutzer erlaubt werden die jeweilige Fahrspurverfolgung zu starten. Dies kann mittels der Kommandozeile oder mittels der entwickelten App für die Visualisierung der Fahrdaten geschehen.
9. **Dokumentation** des Vorgehens und des erstellten Quellcodes
10. **Eigene Ideen** Je nach Projektfortschitt können Sie optional eigene Ideen bzw. Features in die Software integrieren.
11. **Präsentation** der Ergebnisse

3 Projektaufbau

Die Teams haben die Möglichkeit die Softwarearchitektur flexibel selbst festzulegen. Es soll auf bereitgestellten Basisklassen zurückgegriffen werden.

3.1 Arbeitsweise

Das Projekt sollte als Teamarbeit in einer Gruppe von fünf Personen unter Anwendung einer agilen Methode durchgeführt werden. Arbeitspakete sollten als kleine Einheiten (User-Stories) beschrieben werden, die in einem Kanban- oder Scrum-Board organisiert sind. So lassen sich Aufgaben innerhalb des Teams leicht zuweisen. Das Team wählt seine Arbeitsweise selbst: Es kann zusammen, zu Pairs oder allein gearbeitet werden. Wichtig ist, dass die Arbeitsschritte aufeinander abgestimmt sind und agile Rituale wie das “Daily Scrum” durchgeführt werden.

Es werden Projektteams zu je 5 bis 6 Personen gebildet. In jedem Team wird die Rolle des Scrum-Masters. Das Team ist für alle Arbeiten am Produkt verantwortlich: Analyse, Entwurf, Entwicklung, Tests und Dokumentation.

3.2 Dokumentation

Während der Projektarbeit soll begleitend eine Dokumentation angefertigt werden. Das soll zum einen dabei helfen das gelernte festzuhalten und besser nachvollziehen zu können. Andererseits soll damit auch eine Grundlage für den Projektabschluss vorbereitet werden, damit Schnittstellen, Bedienungsweisen, erreichte Ziele, aber auch Fehler oder Einschränkungen dokumentiert werden.

3.3 Vorstellung der Ergebnisse

Am letzten Tag der Projektphase soll jedes Team das Ergebnis kurz präsentieren. Die Präsentation sollte Anwendung der Software demonstrieren, bekannte Stärken und Schwächen darstellen und die grundlegende Softwarearchitektur erläutern.