Projektbeschreibung Wahlfach autonomes Fahren

David Gaede, Tom Freudenmann, Jonas Weihing

17. März 2024

1 Projektbeschreibung

Dieses Projekt setzt eine modulare Pipeline für ein autonom fahrendes Auto in der *Gymnasium Car Racing Simulation*[1] um. Die Pipeline besteht aus den folgenden Modulen und baut auf den in der Vorlesung vermittelten Inhalten auf:

1. Fahrspurerkennung
2. Pfadplanung
3. Querregelung
4. Längsregelung

Für die Umsetzung muss das beigefügte Framework verwendet werden, das bereits mehrere Testdateien und eine Grundstruktur zur Verfügung stellt. Die `Readme.md` erläutert dabei, wie alle notwendigen Bibliotheken installiert werden können und wo sich welche Datei befindet.

2 Module

Im Folgenden werden die einzelnen Pipelinemodule erläutert. Die beigefügten Tipps dienen als Orientierung, müssen jedoch nicht umgesetzt werden. Zusätzlich kann jedes Modul unabhängig über einzelne Test-Skripte getestet werden. Abschließend werden Vorschläge für optionale Erweiterungen zur Verfügung gestellt, um einen Bonus zu erhalten (siehe 3. Bewertung).

2.1 Fahrspurerkennung

Im ersten Modul der Pipeline wird die Fahrspurerkennung durchgeführt. Ziel ist es, die linke und rechte Begrenzung der Fahrspur zu erkennen und zuzuordnen. Je robuster und zuverlässiger die Erkennung ist, desto besser können die nachfolgenden Module darauf aufbauen.

Ziel: Erkennen und Zuordnen der linken und rechten Fahrspurbegrenzungen.

Tipps:

- Erkannte Fahrspuren können als Pixelkoordinaten oder mathematische Funktion dargestellt werden (z. B. Polynom oder Spline).
- Empfohlenes Vorgehen:
  1. **Bildvorverarbeitung:** Zuschneiden des Bildes und Maskierung des Fahrzeugs, um Störungen zu minimieren.
  2. **Kantenerkennung:** Berechnen der Bildgradienten zur Hervorhebung der Übergänge zwischen Fahrspur und Grünfläche.
  3. **Zuordnung:** Zuordnung der erkannten Kanten zur linken und rechten Fahrspurbegrenzung.
  4. **Funktionsanpassung:** Optional kann eine Funktion auf die zugeordneten Kanten angepasst werden, um eine mathematische Repräsentation zu erhalten.
- Testen Sie die einzelnen Komponenten in mehreren Situationen, um Fehler in der Perzeption frühzeitig zu finden.

[1] https://gymnasium.farama.org/environments/box2d/car_racing/

Testing: Um die Fahrspurerkennung zu testen, steht Ihnen das Skript *test_lane_detection.py* zur Verfügung. Abhängig von der spezifischen Implementierung Ihrer Fahrspurerkennung kann es notwendig sein, die *run*-Methode entsprechend anzupassen. Das Auto kann mithilfe der Tastatur (WASD oder Pfeiltasten) in der Simulation gesteuert werden.

Optional:
- Spur auch für zufällige Farbwerte erkennen. Setzten Sie dafür den Parameter 'domain_randomize' beim Ausführen der Testdatei auf 'True':
  `python main.py --domain_randomize`
- Verbesserungen, die über den in der Vorlesung besprochenen Ansatz hinausgehen und die Stabilität und Robustheit erhöhen (Bitte ausführlich dokumentieren).

2.2 Pfadplanung

Im zweiten Modul wird mithilfe der im ersten Modul erkannten Fahrspurbegrenzungen der zu fahrende Pfad geplant. Dies kann im einfachsten Fall die Mitte der Fahrspur sein, für höhere Geschwindigkeiten und optimierte Kurvenfahrten kann diese jedoch in Abhängigkeit von der Krümmung der Kurve angepasst werden.

Ziel: Berechnung eines gültigen Pfads/Trajektorie.

Tipps:
- Verwenden Sie zunächst die Mitte der Fahrspur, das reicht für die Mindestanforderungen.
- Fahrspuren können entweder als Pfad (Menge an Punkten) oder Trajektorie (Funktion) beschrieben werden.
- Überlegen Sie sich, welches Koordinatensystem verwendet werden soll.
- Das Format der Fahrspur hat einen direkten Einfluss auf die Regelung.
- Die Krümmung der Kurve kann anhand des Kurvenradius oder der zweiten Ableitung berechnet werden.

Testing: Um die Pfadplanung zu testen, steht Ihnen das Skript *test_path_planning.py* zur Verfügung. Die Fahrspurbegrenzungen werden Ihnen aus der Simulation in Form von Pixelkoordinaten zur Verfügung gestellt, sodass Sie die Pfadplanung unabhängig von der Fahrspurerkennung entwickeln und testen können. Abhängig von der spezifischen Implementierung Ihrer Pfadplanung kann es jedoch notwendig sein, die *run*-Methode entsprechend anzupassen. Das Auto kann mithilfe der Tastatur (WASD oder Pfeiltasten) in der Simulation gesteuert werden.

Optional:
- Optimierung der Trajektorie anhand der Kurvenkrümmung.
- Weitere Verbesserungen, die die Performance maßgeblich beeinflussen (Bitte ausführlich dokumentieren).

2.3 Querregelung

Das dritte Modul der Pipeline widmet sich der Querregelung, also der Bestimmung des Lenkwinkels. Hierfür kann der bewährte Stanley-Regler verwendet werden, aber auch die Implementation von anderen Reglern wie der Pure Pursuit Regler ist möglich.

Ziel: Funktionierender Stanley oder Pure Pursuit Regler, der das Fahrzeug zuverlässig in der Spur hält.

Tipps:
- Achten Sie auf das Format aus der Pfadplanung.
- Implementieren Sie zunächst den Stanley Regler mit den Standardwerten.
- Die Parameter des Reglers lassen sich iterativ anpassen (am besten das Test-Skript verwenden).

Testing: Um die Querregelung zu testen, steht Ihnen das Skript *test_lateral_control.py* zur Verfügung. Der zu fahrende Pfad wird Ihnen aus der Simulation in Form von diskreten Wegpunkten zur Verfügung gestellt, sodass Sie die Querregelung unabhängig von der Fahrspurerkennung und Pfadplanung entwickeln und testen können. Abhängig von der spezifischen Implementierung Ihrer Querregelung kann es jedoch notwendig sein, die *run*-Methode entsprechend anzupassen. Die Geschwindigkeit des Autos kann mithilfe der Tastatur (W, S oder Pfeiltasten) gesteuert werden.

Optional:
- Implementieren und Vergleichen beider Regelungsansätze. Wichtig: Beide müssen lauffähig sein und über einen Parameter aktiviert werden können.
- Entwickelung einer Dämpfung, die dem Differenzialteil des PID-Reglers ähnelt.
- Weitere Verbesserungen, die die Performance maßgeblich beeinflussen (Bitte ausführlich dokumentieren).

2.4 Längsregelung

Das letzte Modul der Pipeline implementiert die Längsregelung, die für die Steuerung der Fahrzeuggeschwindigkeit zuständig ist. Dazu wird die aktuelle Geschwindigkeit von der Simulation bereitgestellt. Die Längsregelung besteht aus zwei Teilen. Zum einen muss die Zielgeschwindigkeit bestimmt werden, die sich idealerweise nach der Krümmung der Straße richtet. Zum anderen muss mithilfe des PID-Reglers die Geschwindigkeit des Fahrzeugs durch gezieltes Beschleunigen oder Bremsen geregelt werden.

Ziel: Dynamische PID-Längsregelung, die die Geschwindigkeit des Farhzeugs an die Fahrsituation anpasst.

Tipps:
- Bremse und Gas müssen an die Simulation separat übertragen werden.
- Fangen Sie langsam an, es handelt sich um ein Rennauto.
- Nicht Bremsen bzw. Gas geben und gleichzeitig stark lenken, Sie werden sehen warum.
- Ermitteln Sie die Zielgeschwindigkeit anhand der Kurvenkrümmung.

Testing: Um die Längsregelung zu testen, steht Ihnen das Skript *test_longitudinal_control.py* zur Verfügung. Die Lenkung des Fahrzeugs muss dabei händisch gesteuert werden, sodass die Querregelung unabhängig von der Fahrspurerkennung, Pfadplanung und Querregelung entwickelt und getestet werden kann. Abhängig von der spezifischen Implementierung Ihrer Längsregelung kann es jedoch notwendig sein, die *run*-Methode entsprechend anzupassen. Die Lenkung des Autos kann mithilfe der Tastatur (A, D oder Pfeiltasten) gesteuert werden.

Optional:
- Verbesserte Berechnung der Zielgeschwindigkeit, z. B. durch weitere Heuristiken.
- Weitere Verbesserungen, die die Performance maßgeblich beeinflussen (Bitte ausführlich dokumentieren).

3 Bewertungskriterien und Richtlinien

Die Projekte werden mit Ubuntu 20.04 getestet. Stellen Sie dazu in Ihrem Interesse sicher, dass Ihre Abgabe auch unter Ubuntu lauffähig ist. Für die Basisanforderung muss die Pipeline lauffähig sein und das Auto sollte mit einer moderaten Geschwindigkeit auf der Strecke fahren und dabei einen Score von mindestens 300 erreichen. In unserer Minimallösung erreichen wir um die 400 Punkte und in der stark optimierten Lösung bis zu 950 Punkte.
Für die Umsetzung dürfen Python-Standard-Bibliotheken wie `numpy` und `scipy` verwendet werden. `OpenCV` kann für die Darstellung und das Debugging verwendet werden. Verwendet die Pipeline `OpenCV` für die Bildverarbeitung oder andere Methoden, die **nicht** zum Debugging gehören, **werden Punkte abgezogen**. Denn Sie sollen die Bildverarbeitung selbst mit simplen `numpy` Methoden implementieren. Abschließend muss der Code in einer Readme und mit Kommentaren dokumentiert werden. Die Dokumentation sollte dabei ihre Ansätze und Gleichungen erläutern und Verbesserungen hervorheben.

Bonuspunkte können für besondere Leistungen vergeben werden, wie zum Beispiel Verbesserung bzw. Erweiterung der Regler über die Standardregler hinaus, Erreichen eines besonders hohen Scores oder Entwicklung einer besonders robusten Fahrspurerkennung. Tipps für optionale Verbesserungen wurden in den Beschreibungen der Module bereits genannt. Achten Sie dennoch auf eine ausführliche Dokumentation der Bonus Features!

Punktabzug kann erfolgen, wenn der Code nicht lauffähig ist, ein schlechter Score erreicht wird, die Laufzeit des Codes besonders schlecht ist, `OpenCV` verwendet wird, oder der Code unübersichtlich und schlecht dokumentiert ist.

Falls eine individuelle Bewertung gewünscht wird, muss dies im Vorfeld mitgeteilt werden. Ohne entsprechende Ankündigung wird eine Gruppenbewertung vorgenommen. Für eine individuelle Bewertung muss im Code klar gekennzeichnet sein, was von wem entwickelt wurde.