KI Praktikum 3: Aufgabe „Kalman Filter“

Einführung

In diesem Praktikum ist es Ihre Aufgabe, die Position eines Fahrzeugs mit das Kalman Filter zu bestimmen. Das Fahrzeug nutzt die Schätzung des Kalman Filters, um zu entscheiden, in welche Richtung es fahren muss. Wir nutzen für diese Aufgabe den von Microsoft entwickelten Simulator Airsim. Der Simulator wurde in der Unreal Engine entwickelt und ist Open Source. Airsim ist zum Entwickeln und Testen von autonomen Systemen entwickelt worden. Es gibt die Möglichkeit, ein Fahrzeug oder eine Drohne zu steuern. In diesem Praktikum lassen wir das Fahrzeug eine Runde im Kreis fahren. In den folgenden Bildern sehen sie die Umgebung und Strecken von unserem Fahrzeug.

Installation

1. Downloaden Sie die Umgebung AirSimNH der Version 1.8.1. Die Datei finden Sie hier: https://github.com/Microsoft/AirSim/releases.
2. Verschieben Sie die Datei in einen beliebigen Ordner und entpacken Sie diese.
3. Zum Testen können Sie einmal den Simulator starten. Bei Windows mit der Datei AirSimNH/WindowsNoEditor/AirSimNH.exe.
4. Sie sollten die Auswahl bekommen, mit der Drohne oder mit dem Fahrzeug zu starten. Danach sollten Sie die Möglichkeit haben, sich mit dem Fahrzeug in der Umgebung zu bewegen.
5. Falls Performanceprobleme auftreten sollten: Verknüpfung auf AirSimNH.exe erstellen und Parameter "-ResX=640 -ResY=480 -windowed" hinzufügen (um die Auflösung zu reduzieren, das sollte helfen).
6. Als nächstes verschieben Sie die settings.json Datei (diese liegt in der *.zip Datei, die Sie aus ilias heruntergeladen haben) in den Ordner, in dem die AirSimNH.exe Datei liegt.
7. Danach müssen Sie Python mit allen notwendigen Bibliotheken installieren. In der Datei environment.yml aus der ilias zip-Datei finden Sie alle benutzten Bibliotheken. Wir empfehlen die Entwicklungsumgebung mit Anaconda zu installieren. Anaconda installiert für Sie Python mit allen notwendigen Bibliotheken in einer separaten Entwicklungsumgebung. https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html#anaconda-or-miniconda.

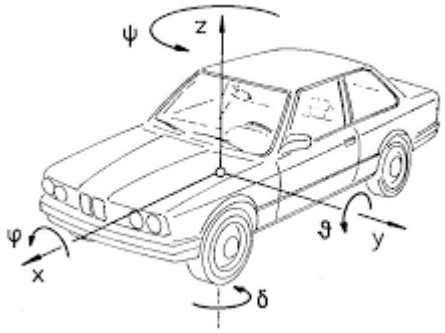
Python mit Anaconda oder Miniconda installieren:

1. Anaconda oder Miniconda herunterladen und installieren.
2. Die "Anaconda Powershell Prompt (anaconda3)" öffnen.
3. In der Konsole zu dem Pfad wechseln in der die Datei environment.yml liegt.
4. Eine virtuelle Umgebung für das Praktikum erstellen: "conda env create -f environment.yml". Dieser Befehl erstellt eine virtuelle Umgebung und installiert in dieser Python und alle benötigten Packages. Der Vorteil einer virtuellen Umgebung ist, dass Sie mehrere Python Versionen auf Ihrem PC installieren können, die bspw. konfliktäre Python Libraries haben können. Bspw. verträgt sich airsim nicht so gut mit Jupyter Notebooks, deshalb wird die Installation von airsim in einer virtuellen Umgebung (wie oben gezeigt) empfohlen.
5. Die virtuelle Umgebung starten: "conda activate kipraktikum3"

Zum Schluss AirSimNH.exe starten. Danach in der Anaconda Powershell zu der Datei student_task.py navigieren und mit dem Befehl „python student_task.py“ ausführen. Die Datei befindet sich in dem Projekt, welches Sie von Ilias heruntergeladen haben. Das Fahrzeug in der Simulation sollte jetzt automatisch fahren.

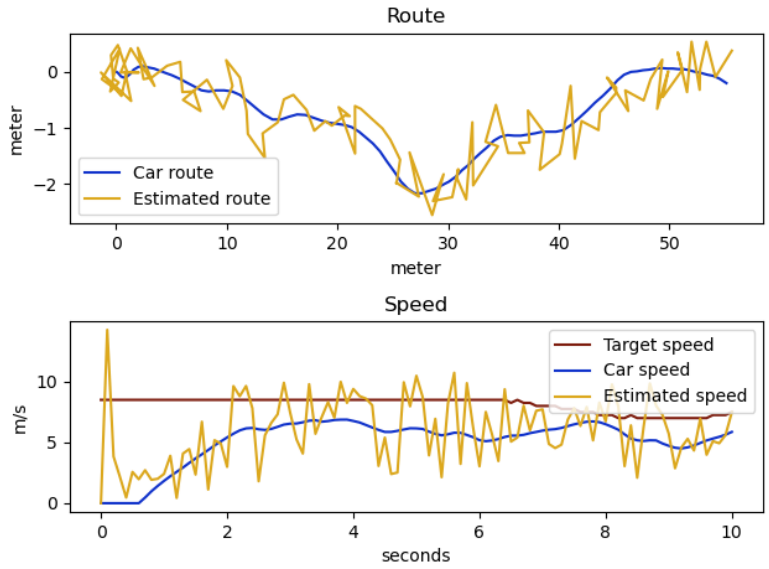
Sensoren

Das Fahrzeug ist mit einem **GPS**-Sensor und einem **IMU**-Sensor ausgerüstet. Der GPS-Sensor (global positioning system) misst die Latitude (Breitengrad), Longitude (Längengrad) und Altitude (Höhe) von unserem Fahrzeug. In diesem Projekt wird die GPS-Messung direkt in die X, Y und Z Position umgeformt. Der IMU-Sensor (inertial measurement unit) misst die Beschleunigung des Fahrzeugs in die X, Y und Z Richtung. Zudem misst die IMU auch noch die Rotationsgeschwindigkeit um die X, Y und Z-Achse. Die IMU misst die Beschleunigung im Koordinatensystem von dem Fahrzeug, jedoch wollen wir wissen, in welche Richtung das Fahrzeug in der Umgebung beschleunigt. Deswegen wird die gemessene Beschleunigung direkt in das Koordinatensystem der Umgebung umgeformt wird. Hinweis: Diese Koordinatentransformationen haben wir für Sie umgesetzt, darum müssen Sie sich nicht kümmern.

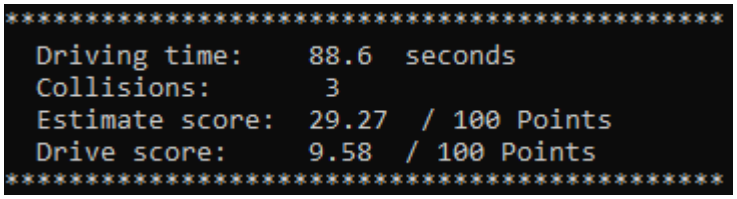


Visualisierung

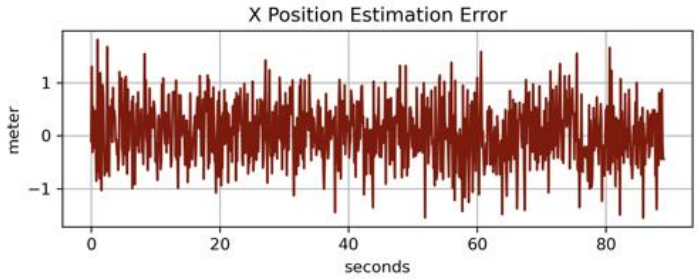
1. Während das Fahrzeug fährt, wird die gefahrene Route und Geschwindigkeit in zwei Graphen angezeigt.

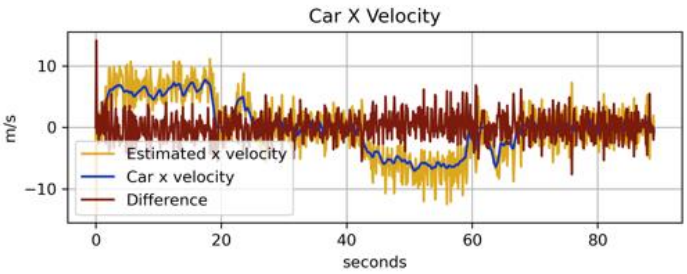


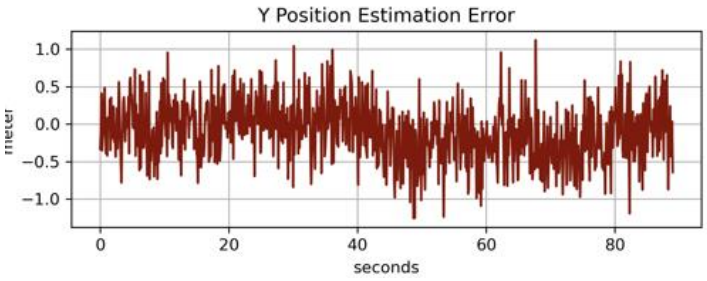
2. Wenn das Fahrzeug das Ende der Route erreicht hat, wird die Performance der Fahrt berechnet. Je besser die Schätzung der Position des Fahrzeugs ist, desto größer ist der „Estimate score“. Der „Drive score“ gibt an, wie gut das Fahrzeug der Route gefolgt ist.

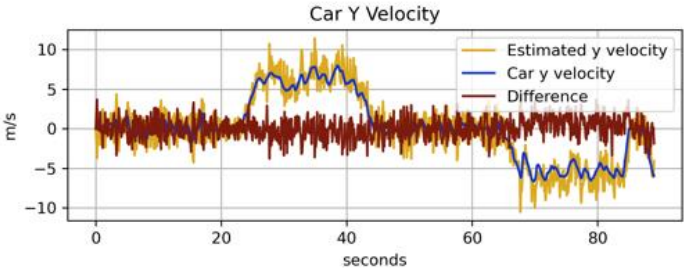


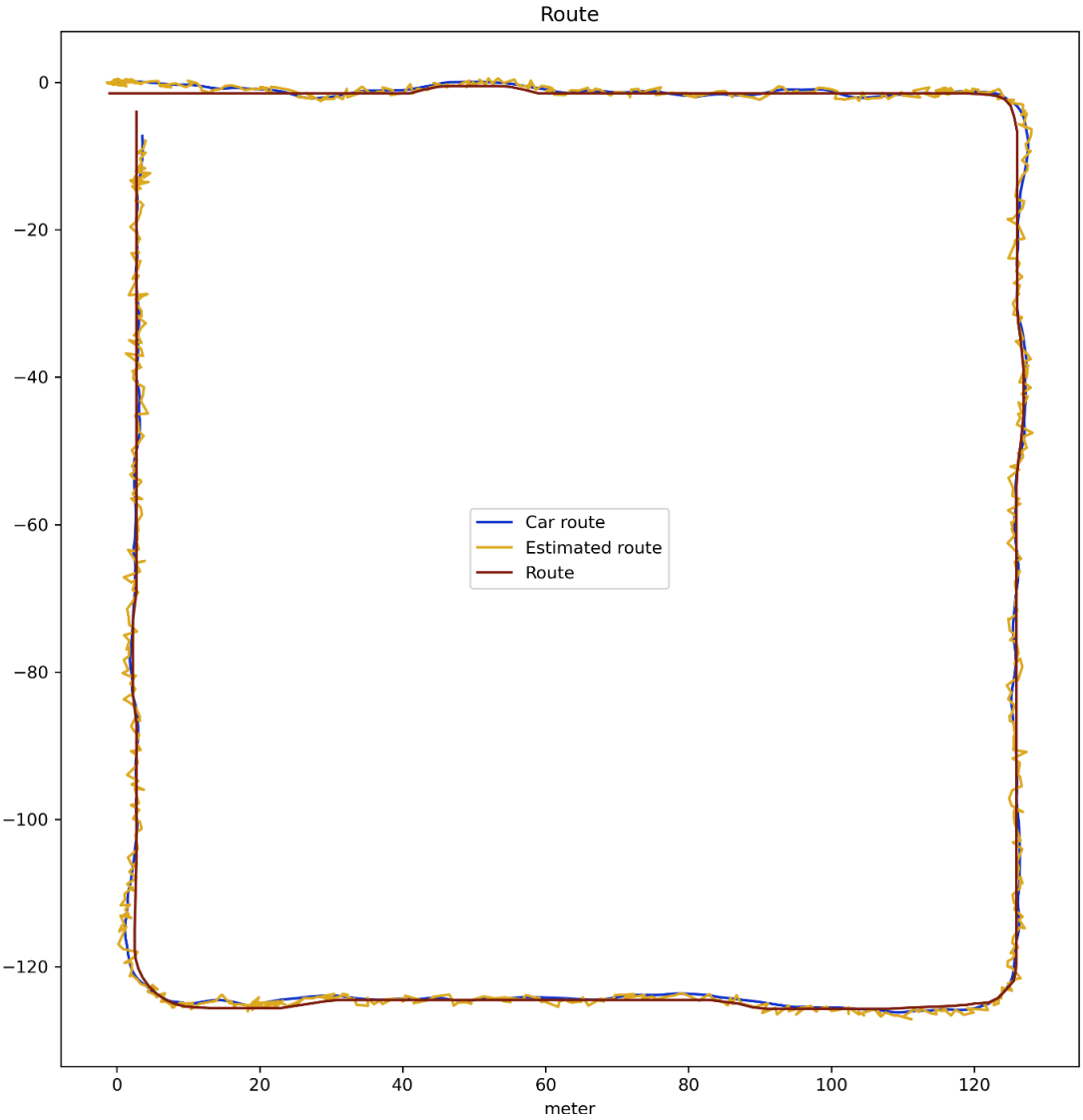
3. Am Ende werden noch zusätzliche Graphen gezeigt. Erstens: Die Entfernung der Schätzung zu der echten Position des Fahrzeugs. Zweitens: Die echte Geschwindigkeit mit der geschätzten Geschwindigkeit. Zum Schluss die gefahrene Route.


X Position Estimation Error
meter
seconds


Car X Velocity
m/s
seconds
Estimated x velocity
Car x velocity
Difference


Y Position Estimation Error
meter
seconds


Car Y Velocity
m/s
seconds
Estimated y velocity
Car y velocity
Difference


Route
meter
Car route
Estimated route
Route

Python Dateien

Für das Praktikum sind die Dateien manager.py und student_task.py wichtig. In der student_task.py Datei sind alle Aufgaben für das Praktikum beschrieben. Die Datei beinhaltet schon die Klasse KalmanFilter, welche Sie vervollständigen müssen. Die manager.py Datei bietet mit der Klasse Manager eine Schnittstelle zum Fahrzeug und den Sensoren. Sie können mit dem Manager die aktuellen Messungen bekommen und mit der „update“-Methode dem Fahrzeug die geschätzte Position und Geschwindigkeit mitteilen. Das Fahrzeug beschleunigt und lenkt aufgrund Ihrer Schätzung. Sie sollen nur die Datei student_task.py ändern!

Dateien, die Sie bearbeiten werden:	
student_task.py	Hier wird das Kalman Filter implementiert.
Dateien, die Sie sich ansehen sollten:	
manager.py	Die Schnittstelle zu dem Fahrzeug und den Sensoren. Verbindet sich mit dem Fahrzeug in der Simulation. Der Manager leitet mit der „update“ Methode die geschätzte Position und Geschwindigkeit an das Fahrzeug.
Unterstützende Dateien, die Sie ignorieren können:	
car.py	Die Schnittstelle zum Airsim Fahrzeug
control.py	Steuert das Fahrzeug.
gps.py	Die Schnittstelle zum Airsim GPS-Sensor.
imu.py	Die Schnittstelle zum Airsim IMU-Sensor.
pid_controller.py	Eine Implementierung des PID-Controllers
path.csv	Eine Liste mit den Punkten des Weges.
plot_manager.py	Stellt die gemessenen Daten in Graphen dar.
state_manager.py	Verwaltet den Zustand des Fahrzeugs.
waypoints.py	Verwaltet und liest die Wegpunkte von der path.csv Datei.
utils.py	Hilfreiche Funktionen, um das Fahrzeug zu steuern.

Aufgaben

1. Definieren Sie den Zustandsvektor des Fahrzeugs und die Systemmatrix. Welche Zustände sind wichtig für das Projekt und welche Zustandskomponenten können gemessen werden?
2. Implementieren Sie das Kalman Filter in der Datei student_task.py.
3. Testen Sie das Kalman Filter und versuchen sie einen „driving score“ von mindestens 60 zu erreichen.
4. Vergleichen Sie Ihre Ergebnisse, wenn Sie
    a. GPS als einzigen Sensor in das Kalman-Filter nutzen
    b. Beschleunigungssensor als einzigen Sensor in das Kalman-Filter nutzen
    c. GPS und Beschleunigungssensor als Sensoren in das Kalman-Filter nutzen
    d. Sie können die Ergebnisse mit einer Bildschirmaufnahme aufzeichnen. Mit der Tastenkombination „Windows Taste“ + G öffnet sich bei Windows ein Menü mit der Option, den Bildschirm aufzuzeichnen. Bei Ubuntu können sie den Bildschirm mit der Tastenkombination Strg + Umschalttaste + Alt + R aufzeichnen und am Ende auch stoppen.

Hinweis: Die Simulation ist recht rechenintensiv. Falls Sie Ihre Lösung auf einem etwas langsameren Rechner ausführen, kann es sein, dass das Fahrzeug nicht so perfekt fährt wie Sie sich das vorstellen (driving score unter 60), obwohl Ihre Lösung korrekt ist. Falls Sie in Ihrer Gruppe keinen Rechner haben, auf dem das Fahrzeug, bei einer korrekten Lösung, einen driving score von 60 erreicht, kontaktieren Sie uns bitte. Für diesen Fall können wir Ihre Lösung auf einen von unseren Rechnern testen.