UDACITY

NANODGREE PROGRAM SYLLABUS

Sensor Fusion Engineer

Overview

Learn to detect obstacles in lidar point clouds through clustering and segmentation, apply thresholds and filters to radar data in order to accurately track objects, and augment your perception by projecting camera images into three dimensions and fusing these projections with other sensor data. Combine this sensor data with Kalman filters to perceive the world around a vehicle and track objects over time.

IN COLLABORATION WITH

Mercedes-Benz

Estimated Time: 4 Months at 10-15hrs/week

Prerequisites: Object-oriented programming

Flexible Learning: Self-paced, so you can learn on the schedule that works best for you

Need Help? udacity.com/advisor Discuss this program with an enrollment advisor.

Course 1: Lidar

Process raw lidar data with filtering, segmentation, and clustering to detect other vehicles on the road. Understand how to implement Ransac with planar model fitting to segment point clouds. Also implement Euclidean clustering with a KD-Tree to cluster and distinguish vehicles and obstacles.

Course Project
Lidar Obstacle Detection

Filter, segment, and cluster real point cloud data to detect obstacles in a driving environment.

	LEARNING OUTCOMES	
LESSON ONE	**Introduction to Lidar & Point Clouds**	• Lidar data representation • Working with a simulator to create PCD • Visualizing Lidar data
LESSON TWO	**Point Cloud Segmentation**	• Using PCL to segment point clouds • The RANSAC algorithm for planar model fitting
LESSON THREE	**Clustering Obstacles**	• Using PCL to cluster obstacles • Using a KD-Tree to store point cloud data • Implementing Euclidean Clustering to find clusters • Applying bounding boxes around clusters
LESSON FOUR	**Working with Real Point Cloud Data (PCD)**	• Working with real self-driving car PCD data • Filtering PCD data • Playing back multiple PCD files • Applying point cloud processing to detect obstacles

Course 2: Radar

Analyze radar signatures to detect and track objects. Calculate velocity and orientation by correcting for radial velocity distortions, noise, and occlusions. Apply thresholds to identify and eliminate false positives. Filter data to track moving objects over time.

Course Project
Radar Obstacle Detection

Calibrate, threshold, and filter radar data to detect obstacles in real radar data.

		LEARNING OUTCOMES
LESSON ONE	**Introduction to Radar**	• Handling real radar data • Calculating object headings and velocities • Determining the appropriate sensor specifications for a task
LESSON TWO	**Radar Calibration**	• Correcting radar data to account for radial velocity • Filtering noise from real radar sensors
LESSON THREE	**Radar Detection**	• Thresholding radar signatures to eliminate false positives • Predicting the location of occluded objects.

Course 3: Camera

Fuse camera images together with lidar point cloud data. Extract object features from camera images in order to estimate object motion and orientation. Classify objects from camera images in order to apply a motion model. Project the camera image into three dimensions. Fuse this projection into three dimensions to fuse with lidar data

Course Project
Camera and Lidar Fusion

Detect and track objects in 3D space from the benchmark KITTI dataset based on camera and lidar measurements. Compute time-to-collision based on both sensors and compare the results. Identify the best combination of keypoint detectors and descriptors for object tracking.

	LEARNING OUTCOMES	
LESSON ONE	**Sensor Fusion & Autonomous Driving**	• Understanding the SAE levels of autonomy • Comparing typical autonomous vehicle sensor sets including Tesla, Uber and Mercedes • Comparing camera, lidar and radar using a set of industry-grade performance criteria
LESSON TWO	**Camera Technology and Collision Detection**	• Understanding how light forms digital images and which properties of the camera (e.g. aperture, focal length) affect this formation • Manipulation images using the OpenCV computer vision library • Designing a collision detection system based on motion models, lidar and camera measurements
LESSON THREE	**Feature Tracking**	• Detecting features from objects in a camera image using state-of-the-art detectors and standard methods • Matching features between images to track objects over time using state-of-the-art binary descriptors
LESSON FOUR	**Camera and Lidar Fusion**	• Projecting 3D lidar points into a camera sensor • Using deep-learning to detect vehicles (and other objects) in camera images • Creating a three-dimensional object from lidar and camera data

Course 4: Kalman Filters

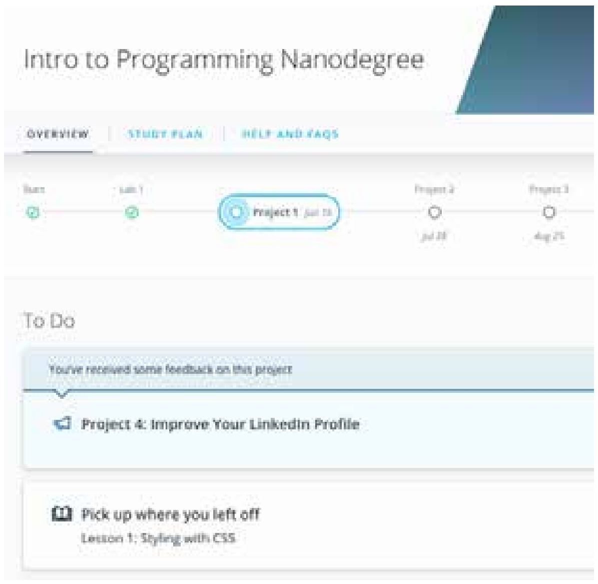
Fuse data from multiple sources using Kalman filters. Merge data together using the prediction-update cycle of Kalman filters, which accurately track object moving along straight lines. Then build extended and unscented Kalman filters for tracking nonlinear movement.

Course Project
Unscented Kalman Filters Project

Put your skills to the test! Code an Unscented Kalman Filter in C++ in order to track highly non-linear pedestrian and bicycle motion.

	LEARNING OUTCOMES	
LESSON ONE	**Kalman Filters**	• Constructing Kalman filters • Merging data from multiple sources • Improving tracking accuracy • Reducing sensor noise
LESSON TWO	**Lidar and Radar Fusion with Kalman Filters**	• Building a Kalman Filter in C++ • Handling both radar and lidar data.
LESSON THREE	**Extended Kalman Filters**	• Predicting when non-linear motion will cause errors in a Kalman filter • Programming an extended Kalman filter to cope with non-linear motion • Constructing Jacobian matrices to support EKFs
LESSON FOUR	**Unscented Kalman Filters**	• Estimating when highly nonlinear motion might break even an extended Kalman Filter • Creating an unscented Kalman Filter to accurately track non-linear motion.

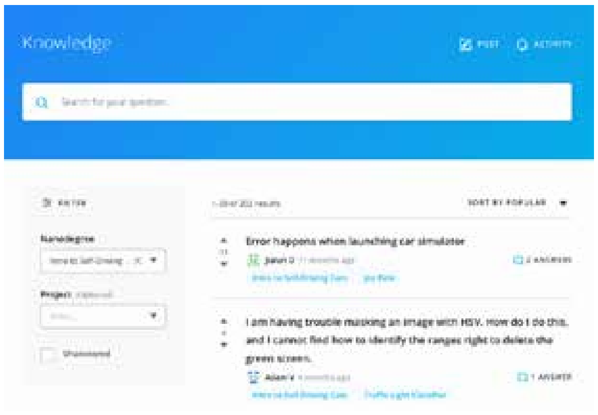
Our Classroom Experience



You've received some feedback on this project

Project 4: Improve Your LinkedIn Profile

Pick up where you left off
Lesson 1: Styling with CSS



Error happens when launching car simulator

I am having trouble masking an image with HSV. How do I do this, and I cannot find how to identify the ranges right to delete the green screen.

```
import turtle
amy = turtle.Turtle()
amy.color(green)
for side in [1, 2, 3, 4]:
    amy.forward(100)
    amy.right(90)
```

REAL-WORLD PROJECTS
Build your skills through industry-relevant projects. Get personalized feedback from our network of 900+ project reviewers. Our simple interface makes it easy to submit your projects as often as you need and receive unlimited feedback on your work.

KNOWLEDGE
Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students and discover in real-time how to solve the challenges that you encounter.

STUDENT HUB
Leverage the power of community through a simple, yet powerful chat interface built within the classroom. Use Student Hub to connect with your technical mentor and fellow students in your Nanodegree program.

WORKSPACES
See your code in action. Check the output and quality of your code by running them on workspaces that are a part of our classroom.

QUIZZES
Check your understanding of concepts learned in the program by answering simple and auto-graded quizzes. Easily go back to the lessons to brush up on concepts anytime you get an answer wrong.

CUSTOM STUDY PLANS
Work with a mentor to create a custom study plan to suit your personal needs. Use this plan to keep track of your progress toward your goal.

PROGRESS TRACKER
Stay on track to complete your Nanodegree program with useful milestone reminders.

Learn with the Best

David Silver

HEAD OF CURRICULUM

David Silver leads the Udacity Curriculum Team. Before Udacity, David was a research engineer on the autonomous vehicle team at Ford. He has an MBA from Stanford, and a BSE in Computer Science from Princeton.

Stephen Welch

INSTRUCTOR

Stephen is a Content Developer at Udacity and has worked on the C++ and Self-Driving Car Engineer Nanodegree programs. He started teaching and coding while completing a Ph.D. in mathematics, and has been passionate about engineering education ever since.

Andreas Haja

INSTRUCTOR

Andreas Haja is an engineer, educator and autonomous vehicle enthusiast with a PhD in computer science. Andreas now works as a professor, where he focuses on project-based learning in engineering. During his career with Volkswagen and Bosch he developed camera technology and autonomous vehicle prototypes.

Stephen Welch

INSTRUCTOR

Stephen is a Content Developer at Udacity and has worked on the C++ and Self-Driving Car Engineer Nanodegree programs. He started teaching and coding while completing a Ph.D. in mathematics, and has been passionate about engineering education ever since.

UDACITY

All Our Nanodegree Programs Include:

EXPERIENCED PROJECT REVIEWERS

REVIEWER SERVICES

- Personalized feedback & line by line code reviews
- 1600+ Reviewers with a 4.85/5 average rating
- 3 hour average project review turnaround time
- Unlimited submissions and feedback loops
- Practical tips and industry best practices
- Additional suggested resources to improve

TECHNICAL MENTOR SUPPORT

MENTORSHIP SERVICES

- Questions answered quickly by our team of technical mentors
- 1000+ Mentors with a 4.7/5 average rating
- Support for all your technical questions

PERSONAL CAREER SERVICES

CAREER SUPPORT

- Resume support
- Github portfolio review
- LinkedIn profile optimization

Frequently Asked Questions

PROGRAM OVERVIEW

WHY SHOULD I ENROLL?

Sensor fusion engineering is one of the most important and exciting areas of robotics.

Sensors like cameras, radar, and lidar help self-driving cars, drones, and all types of robots perceive their environment. Analyzing and fusing this data is fundamental to building an autonomous system.

In this Nanodegree program, you will work with camera images, radar signatures, and lidar point clouds to detect and track vehicles and pedestrians. By graduation, you will have an impressive portfolio of projects to demonstrate your skills to employers.

WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

As a Sensor Fusion Engineer, you'll be equipped to bring value to a wide array of industries and be eligible for many roles.

Your opportunities might include roles such as an:

- Imaging Engineer.
- Sensor Fusion Engineer.
- Perception Engineer.
- Automated Vehicle Engineer.
- Research Engineer.
- Self-Driving Car Engineer.
- Object Tracking Engineer.
- Sensor Engineer.
- System Integration Engineer.
- Depth Engineer.

HOW DO I KNOW IF THIS PROGRAM IS RIGHT FOR ME?

If you're interested in learning about lidar, radar, and camera data and how to fuse it together, this program is right for you.

Sensors and sensor data are used in a wide array of applications -- from cell phones to robots and self-driving cars -- giving you a wide array of fields you could enter or advance a career in after this program.

ENROLLMENT AND ADMISSION

DO I NEED TO APPLY? WHAT ARE THE ADMISSION CRITERIA?

There is no application. This Nanodegree program accepts everyone,

FAQs Continued

regardless of experience and specific background.

WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

To optimize your chances of success in the Sensor Fusion Engineer Nanodegree program, we've created a list of prerequisites and recommendations to help prepare you for the program curriculum. You should have the following knowledge:

- Advanced knowledge in any object-oriented programming language,
- preferably C++
- Intermediate Probability
- Intermediate Calculus
- Intermediate Linear Algebra
- Basic Linux Command Lines

IF I DO NOT MEET THE REQUIREMENTS TO ENROLL, WHAT SHOULD I DO?

For aspiring sensor fusion engineers who currently have a limited background in programming or math, we've created the **Intro to Self-Driving Cars Nanodegree** program to help you prepare. This program teaches C++, linear algebra, calculus, and statistics.

If you have a limited background in programming, we've created the **C++ Nanodegree** program to help you prepare for the coding in this program.

TUITION AND TERM OF PROGRAM

HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Sensor Fusion Engineer Nanodegree program is comprised of content and curriculum to support four (4) projects. We estimate that students can complete the program in four (4) months, working 10 hours per week.

Each project will be reviewed by the Udacity reviewer network. Feedback will be provided and if you do not pass the project, you will be asked to resubmit the project until it passes.

HOW LONG IS THIS NANODEGREE PROGRAM?

Access to this Nanodegree program runs for the length of time specified in the payment card above. If you do not graduate within that time period, you will continue learning with month to month payments. See the **Terms of Use** and **FAQs** for other policies regarding the terms of access to our Nanodegree programs.

FAQs Continued

SOFTWARE AND HARDWARE

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?
We have few requirements since you will be coding on our virtual environments ("Workspaces") in the browser. This means you can complete all coursework within our platform, and do not need to install anything on your own machine.

If you choose to complete projects on your local machine, you should install:

- C++ Version 11
- Point Cloud Library 1.7

Hardware Requirements:

- None