UDACITY

NANODEGREE PROGRAM SYLLABUS

Self-Driving Car Engineer

Overview

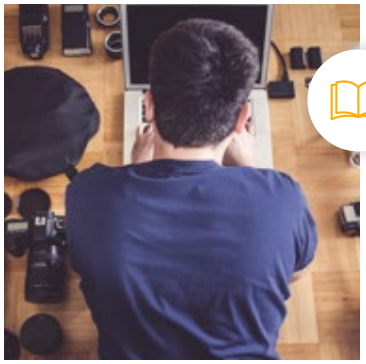
This program will cover the techniques that power self-driving cars across the full stack of a vehicle's autonomous capabilities. To begin, you'll learn how to apply computer vision and deep learning toward perception problems like lane finding and classifying traffic signs, as well as a full end-to-end algorithm for driving with behavioral cloning. You will also learn how to track objects from radar and lidar data with sensor fusion. From there, you'll learn and implement the concepts behind localization, path planning and control, making sure your vehicle knows where it is in the environment and how to navigate through it.

IN COLLABORATION WITH

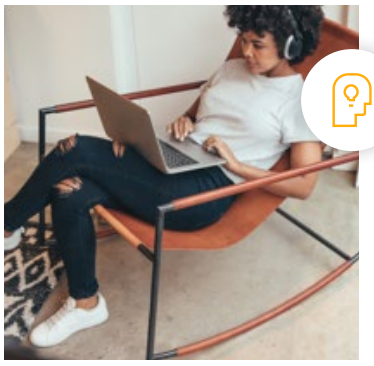
Mercedes-Benz

WAYMO

Estimated Time:
5 Months at
10 hrs/week



Prerequisites:
Python, C++,
Linear Algebra
and Calculus



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

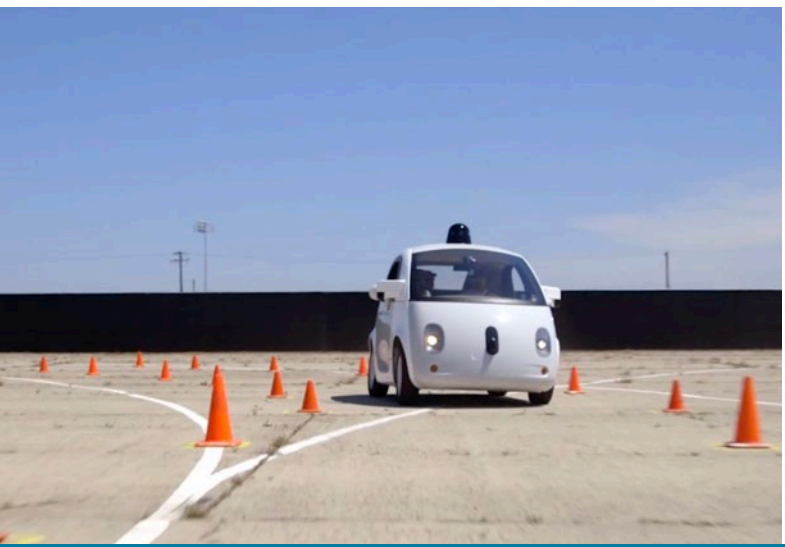
Need Help?
udacity.com/advisor
Discuss this program
with an enrollment
advisor.

Course 1: Computer Vision

In this course, you will develop critical machine learning skills commonly leveraged in autonomous vehicle engineering. You will learn about the life cycle of a machine learning project, from framing the problem and choosing metrics to training and improving models. This course will focus on the camera sensor, and you will learn how to process raw digital images before feeding them into different algorithms, such as neural networks. You will build convolutional neural networks using TensorFlow and learn how to classify and detect objects in images. With this course, you will be exposed to the entire machine learning workflow to have a good understanding of the work of a Machine Learning Engineer and how it translates to autonomous vehicle engineering.

Course Project
Object Detection in an Urban Environment

In this project, students will create a convolutional neural network to detect and classify objects using data from the Waymo Open Dataset. Students are provided a dataset containing images of urban environments with annotated cyclists, pedestrians and vehicles. First, they will perform an extensive data analysis, including the computation of label distributions, displaying sample images and checking for object occlusions. This analysis will inform students to decide what augmentations are meaningful for the project. Next, they will train a neural network to detect and classify objects. Then, students will monitor the training with TensorBoard and decide when to end it. Finally, they will experiment with different hyperparameters to improve performance.



		LEARNING OUTCOMES
LESSON ONE	**The Machine Learning Workflow**	• Identify the key stakeholders in a ML problem • Frame the ML problem • Perform exploratory data analysis on an image dataset • Pick the most adequate model for a particular ML task • Choose the correct metric • Select and visualize the data
LESSON TWO	**Sensor and Camera Calibration**	• Manipulate image data • Calibrate an image using checkerboard images • Perform geometric transformation of an image • Perform pixel level transformation of an image
LESSON THREE	**From Linear Regression to Feedforward Neural Networks**	• Implement a logistic regression model in TensorFlow • Implement backpropagation • Implement gradient descent • Build a custom neural network for a classification task
LESSON FOUR	**Image Classification with Convolutional Neural Networks**	• Write a custom classification architecture using TensorFlow • Choose the right augmentations to increase a dataset variability • Use regularization techniques to prevent overfitting • Calculate the output shape of a convolutional layer • Count the number of parameters in a convolutional network
LESSON FIVE	**Object Detection in Images**	• Use the TensorFlow object detection API • Choose the best object detection model for a given problem • Optimize training processes to maximize resource usage • Implement non-maximum suppression • Calculate mean average precision • Choose hyperparameters to optimize a neural network

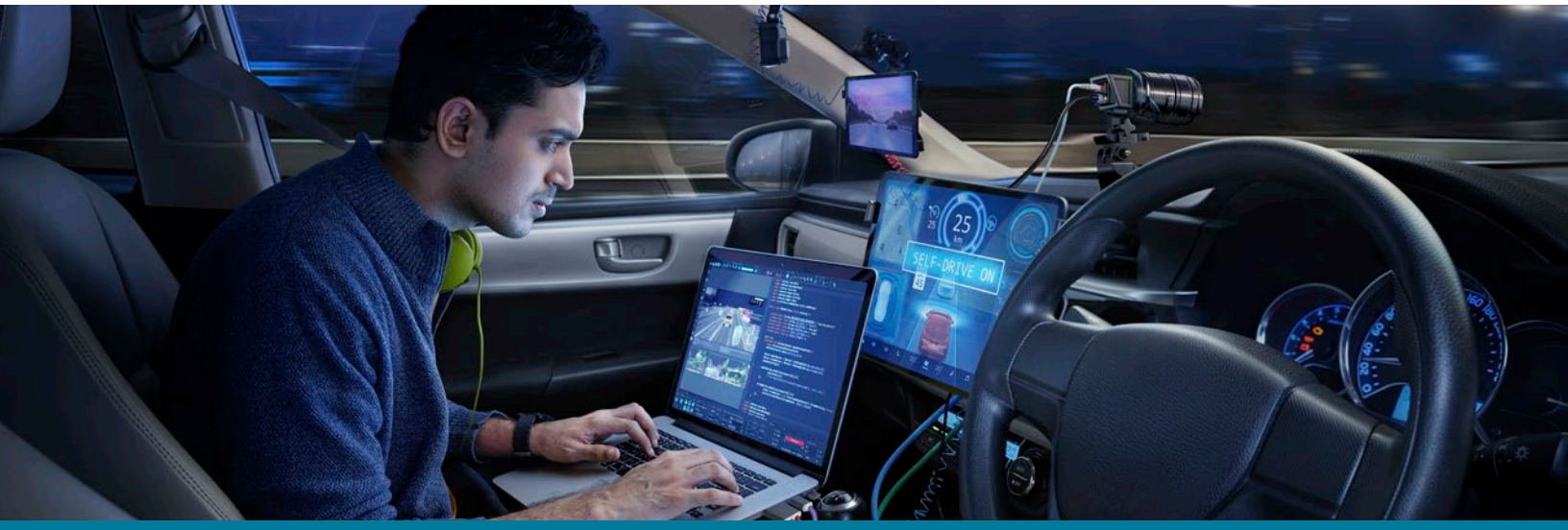
Course 2: Sensor Fusion

In this course, you will learn about a key enabler for self-driving cars: sensor fusion. Besides cameras, self-driving cars rely on other sensors with complementary measurement principles to improve robustness and reliability. Therefore, you will learn about the lidar sensor and its role in the autonomous vehicle sensor suite. You will learn about the lidar working principle, get an overview of currently available lidar types and their differences, and look at relevant criteria for sensor selection. Also, you will learn how to detect objects such as vehicles in a 3D lidar point cloud using a deep learning approach and evaluating detection performance using a set of state-of-the-art metrics.

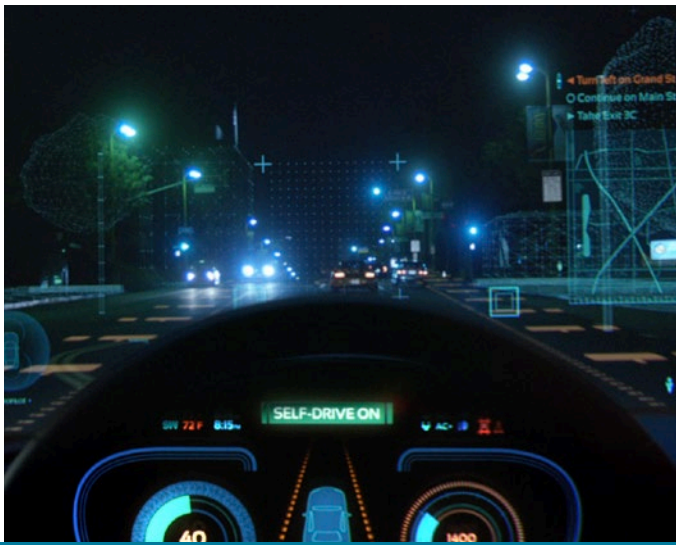
In the second half of the course, you will learn how to fuse camera and lidar detections and track objects over time with an Extended Kalman Filter. You will get hands-on experience with multi-target tracking, where you will learn how to initialize, update and delete tracks, assign measurements to tracks with data association techniques, managing several simultaneously. After completing the course, you will have a solid foundation to work as a sensor fusion engineer on self-driving cars.

Course Project
3D Object Detection

Students will first load and preprocess 3D lidar point clouds and then use a deep learning approach to detect and classify objects (e.g., vehicles, pedestrians). They will then evaluate and visualize the objects, including calculating key performance metrics. This project combines with the Sensor Fusion project to form an entire detection pipeline.



		LEARNING OUTCOMES
LESSON ONE	**Introduction to Sensor Fusion and Perception**	• Distinguish strengths and weaknesses of each sensor
LESSON TWO	**The Lidar Sensor**	• Explain the role of lidar in autonomous driving • Extract lidar data from the Waymo dataset • Extract lidar technical properties such as coordinates • Visualize lidar data
LESSON THREE	**Detecting Objects in Lidar**	• Describe the state-of-the-art in 3D object detection • Transform a point cloud into a birds-eye view (BEV) • Perform model inference using BEV images • Visualize detection results • Evaluate object detection performance with metrics • Evaluate object detection performance between models



Course Project
Sensor Fusion

In this project, students will solve a challenging multi-target tracking task by fusing camera and lidar detections. They will implement an Extended Kalman filter to track several vehicles over time, including the different measurement models for camera and lidar. This task also requires a track management module for track initialization and deletion and a data association module to decide which measurement originated from which track. Finally, students will evaluate and visualize the tracked objects. To complete this project, students will use a real-world dataset, exposing them to the everyday challenges of a sensor fusion engineer.

	LEARNING OUTCOMES	
LESSON ONE	**Kalman Filters**	• Track objects over time with a linear Kalman filter
LESSON TWO	**Extended Kalman Filters**	• Track objects over time with an extended Kalman filter • Implement motion and measurement models • Derive a Jacobian for nonlinear models • Apply appropriate coordinate transforms (e.g. sensor, vehicle coordinates) • Fuse lidar measurements with camera detections with appropriate camera models
LESSON THREE	**Multi-Tracking Tracking**	• Initialize, update and delete tracks • Define and implement a track score and track state • Calculate a simple detection probability / visibility reasoning • Associate measurements to tracks for multi-target tracking • Reduce association complexity through a gating method • Evaluate tracking performance through RMSE

Course 3: Localization

In this course, you will learn all about robotic localization, from one-dimensional motion models up to three-dimensional point cloud maps obtained from lidar sensors. You'll begin by learning about the bicycle motion model, an approach to use simple motion to estimate location at the next time step, before gathering sensor data. Next, you'll move on using Markov localization to perform 1D object tracking, as well as further leveraging motion models. From there, you will learn how to implement two different scan matching algorithms, Iterative Closest Point (ICP) and Normal Distributions Transform (NDP), working with 2D and 3D data. Finally, utilizing these scan matching algorithms in the Point Cloud Library (PCL), you will localize a simulated car with lidar sensing, using a 3D point cloud map obtained from the CARLA simulator.

Course Project
Scan Matching Localization

In this project, students will recover the position of a simulated car using lidar with either ICP or NDT, two scan matching algorithms, aligning point cloud scans from the CARLA simulator. Students will need to achieve sufficient accuracy for the entirety of a drive within the simulated environment, updating the vehicle's location appropriately as it moves and obtains new lidar data.

		LEARNING OUTCOMES
LESSON ONE	**Introduction to Localization**	• Explain how a self-driving car might use GPS or detected objects to localize itself in an environment • Predict motion to estimate location in a future time step using the bicycle motion model
LESSON TWO	**Markov Localization**	• Apply the law of total probability to robotic motion • Derive the general Bayes/Markov filter • Implement 1D localization in C++
LESSON THREE	**Creating Scan Matching Algorithms**	• Explain ICP for localization • Explain NDT for localization • Implement ICP and NDT for 2D localization in C++
LESSON FOUR	**Utilizing Scan Matching in 3D**	• Align 3D point cloud maps with ICP • Align 3D point cloud maps with NDT • Create point cloud maps in the CARLA simulator

Course 4: Planning

Path planning routes a vehicle from one point to another, handling reactions when emergencies arise. The Mercedes-Benz Vehicle Intelligence team will take you through the three stages of path planning. First, you'll apply model-driven and data-driven approaches to predict how other vehicles on the road will behave. Then you'll construct a finite state machine to decide which one of several different maneuvers your vehicle should perform. Finally, you'll generate a safe and comfortable trajectory to execute the maneuver.

Course Project
Motion Planning and Decision Making for Autonomous Vehicles

In this project, you will implement two of the main components of a traditional hierarchical planner: the behavior planner and the motion planner. Both will work in unison to avoid static objects parked on the side of the road, preventing collision with these objects by executing either a "nudge" or a "lane change" maneuver, navigate intersections, and track the centerline on the traveling lane.

		LEARNING OUTCOMES
LESSON ONE	**Behavior Planning**	• Learn how to think about high level behavior planning in a self-driving car
LESSON TWO	**Trajectory Generation**	• Use C++ and the Eigen linear algebra library to build candidate trajectories for the vehicle to follow
LESSON THREE	**Motion Planning**	• Program a decision making framework to plan a vehicle's motion in an urban environment • Incorporate environmental information into the motion planning algorithm • Generate an "optimal", feasible and collision free path • Navigate the vehicle through an urban driving scenario in simulation following the rules of the road, in a human-like fashion

Course 5: Control

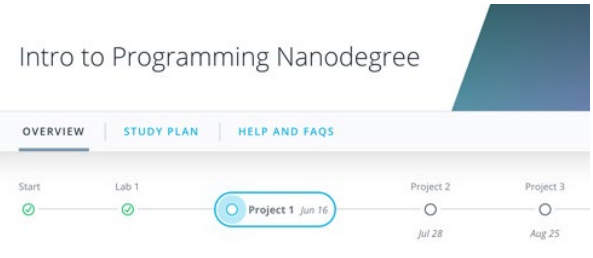
This course will teach you how to control a car once you have the desired trajectory. In other words, how to activate the throttle and the steering wheel of the car to move it following a trajectory described by coordinates. The course will cover the most basic and most common controller: the Proportional Integral Derivative or PID controller. You will understand the basic principle of feedback controls and how they apply to autonomous driving techniques.

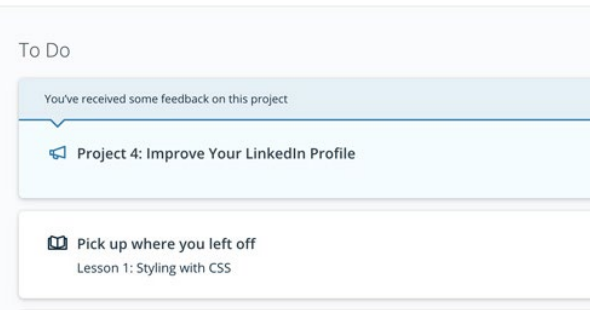
Course Project
Control and Trajectory Tracking for Autonomous Vehicles

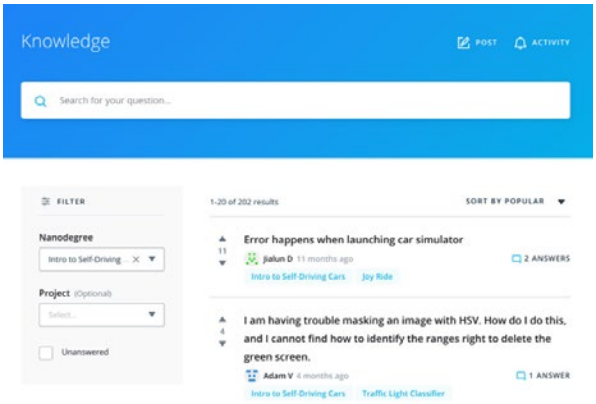
In this project, applying the skills acquired in previous projects, you will design a PID controller to perform vehicle trajectory tracking. Given a trajectory as an array of locations and a simulation environment, you will design and code a PID controller and test its efficiency on the CARLA simulator. This project will help you understand the power and the limitations of the PID controller while utilizing feedback control. This project is good training for C++ coding, which is the standard language used in the industry.

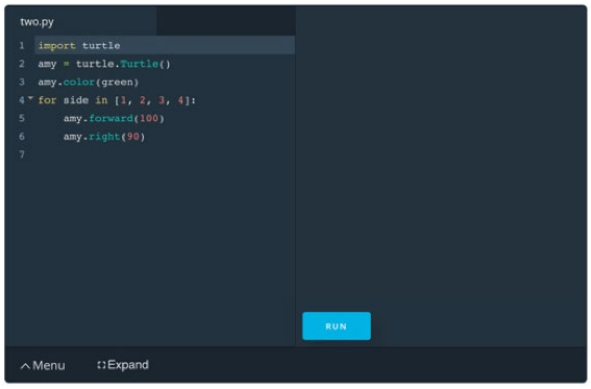
		LEARNING OUTCOMES
LESSON ONE	**PID Control**	• Recognize the observation of the state of the vehicle (position, velocity), the action (steering, accelerator, brake) and the possible perturbations • Design and code the feedback controller (PID and MPC) for trajectory tracking, using he PID controller, and understanding how to choose the parameters to guarantee stability, and then with the MPC, a more general controller with non-linear dynamics • Test the controllers and evaluate their robustness to real-world perturbations • Analyze the differences between the two controllers

Our Classroom Experience









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

Thomas Hossler

SR DEEP LEARNING ENGINEER

Thomas is originally a geophysicist but his passion for Computer Vision led him to become a Deep Learning engineer at various startups. By creating online courses, he is hoping to make education more accessible. When he is not coding, Thomas can be found in the mountains skiing or climbing.

Antje Muntzinger

SELF-DRIVING CAR ENGINEER

Antje is a self-driving car engineer and a technical lead for sensor fusion at Mercedes-Benz. She wrote her PhD about sensor fusion for advanced driver assistance systems. By educating more self-driving car engineers, she hopes to realize the dream of fully autonomous driving together in the future.

Andreas Haja

PROFESSOR

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.

Aaron Brown

SENIOR AV SOFTWARE ENGINEER

Aaron Brown has a background in electrical engineering, robotics and deep learning. Currently working with Mercedes-Benz Research & Development as a Senior Autonomous Vehicle Software Engineer, Aaron has worked as a Content Developer and Simulation Engineer at Udacity focusing on developing projects for self-driving cars.

Learn with the Best

Munir Jojo Verge

LEAD AUTONOMOUS & AI SYSTEMS DEVELOPER AT MITRE

Previously, Munir was a Motion Planning & Decision-Making Manager at Amazon. He also worked for a 2 Self-driving car companies and for WaltDisney Shanghai building TronLightcycle. Munir holds a B.Eng. in Aerospace, a M.S. in Physics and a M.S. in Space Studies.

Mathilde Badoual

FIFTH YEAR PHD STUDENT AT UC BERKELEY

Mathilde has a strong background in optimization and control, including reinforcement learning and has an engineering diploma from the electrical engineering school Supelec, in France. Previously she worked at Tesla in the energy and optimization team.

David Silver

CURRICULUM LEAD

David Silver leads the School of Autonomous Systems at Udacity. 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.

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

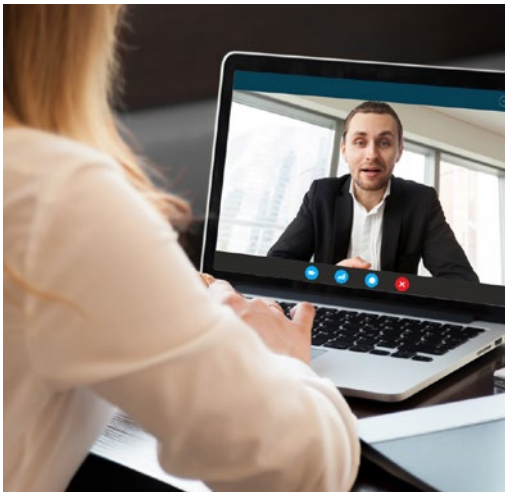
PERSONAL CAREER SERVICES

CAREER SUPPORT

- Resume support
- Github portfolio review
- LinkedIn profile optimization

Frequently Asked Questions

PROGRAM OVERVIEW



WHY SHOULD I ENROLL?

The Self-Driving Car Engineer Nanodegree program is one of the only programs in the world to both teach students how to become a self-driving car engineer and support students in obtaining a job within the field of autonomous systems. The program's projects equip students with invaluable skills across a wide array of critical topics, including computer vision, sensor fusion, localization, motion control and more. As part of their capstone project, students have the opportunity to run their code on the open-source simulator CARLA.

WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

Our wide-ranging curriculum will prepare you for a variety of roles in the autonomous vehicle industry, including: System Software Engineer, Deep Learning Engineer, Vehicle Software Engineer, Localization and Mapping Engineer and many others. If you elect to work outside of automotive engineering, your foundation in deep learning and robotics will enable you to fill any number of related roles in artificial intelligence, computer vision, machine learning and more.

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

This advanced Nanodegree program is ideal for anyone with a programming, technical, or quantitative background who is interested in obtaining a job within the field of autonomous systems, or refreshing or developing their skills within the realm of machine and deep learning, systems integration, sensor fusion and many others.

WHAT IS THE DIFFERENCE BETWEEN THE INTRO TO SELF-DRIVING CARS NANODEGREE PROGRAM AND THE SELF-DRIVING CAR ENGINEER NANODEGREE PROGRAM?

The Intro to Self-Driving Cars Nanodegree program is an intermediate program open to anyone with an interest in autonomous systems, who has some programming experience, and/or a quantitative background. The Self-Driving Car Engineer Nanodegree program is an advanced program focusing on in-depth knowledge of autonomous systems. The program is designed for those with moderate to high programming, technical, and/or quantitative skills.

ENROLLMENT AND ADMISSION

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

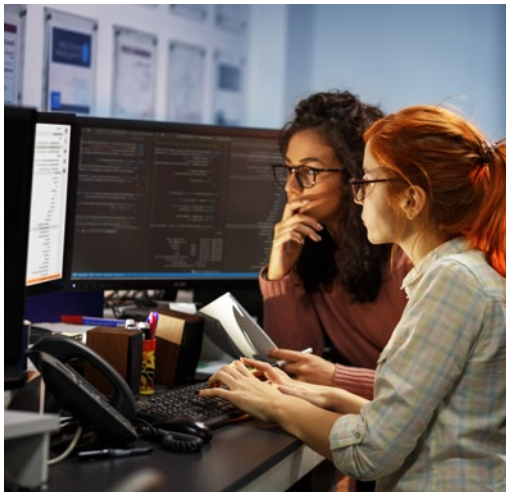
There is no application. This Nanodegree program accepts everyone, regardless of experience and specific background.

FAQs Continued

WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

A well-prepared student will be able to:

- Build object-oriented programs in any language (ideally Python or C++)
- Compute integrals and derivatives of polynomial functions
- Multiply matrices and understand related aspects of linear algebra
- Calculate mean, median, and standard deviation of a dataset
- Model the effects of forces on point masses



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

We have a number of Nanodegree programs and free courses that can help you prepare, including: **Intro to Self-Driving Cars Nanodegree** program, **Robotics Software Engineer Nanodegree program** and **AI for Robotics Course**.

TUITION AND TERM OF PROGRAM

HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Self-Driving Car Engineer Nanodegree program is comprised of content and curriculum to support six (6) projects. We estimate that students can complete the program in five (5) 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 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.

CAN I SWITCH MY START DATE? CAN I GET A REFUND?

Please see the Udacity Nanodegree program **FAQs** for policies on enrollment in our programs.

I HAVE GRADUATED FROM THE SELF-DRIVING CAR ENGINEER NANODEGREE PROGRAM BUT I WANT TO KEEP LEARNING. WHERE SHOULD I GO FROM HERE?

Once you have completed the Self-Driving Car Engineer Nanodegree program, the **Robotics Software Engineer Nanodegree** program and the **Flying Car Nanodegree** program are ideal for continuing your learning.

FAQs Continued

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

For this Nanodegree program, you will need to the minimum equipment requirements outlined here: **https://www.udacity.com/tech-requirements**.

WHICH LIBRARIES OR LANGUAGES ARE USED IN THIS PROGRAM?

The following versions are used in this program (subject to update):

- Tensorflow version 2+
- Python version 3
- C++ version 14