FLYING CAR NANODEGREE SYLLABUS

In this program, you'll learn the core concepts required to design and develop robots that fly. Working with the quadrotor test platform and our custom flight simulator, you will implement planning, control, and estimation solutions in Python and C++.

Course 1: Introduction

In this course, you will get an introduction to flight history, challenges, and vehicles. You will learn about our quadrotor test platform, work in our custom simulator, and build your first project—getting a quadrotor to take-off and fly around a backyard!

Lesson 1: Welcome
Lesson 2: Autonomous Flight
Lesson 3: Project: Backyard Flyer
Lesson 4: Drone Integration

Project 1: Backyard Flyer
In this project, you will write event-driven code in Python to get your drone to takeoff, fly a predetermined path, and land in a simulated backyard environment.

Course 2: 3D Motion Planning

Flying robots must traverse complex, dynamic environments. Wind, obstacles, unreliable sensor data, and other randomness all present significant challenges. In this course, you will learn the fundamentals of aerial path planning. You will begin with 2D problems, optimize your solutions using waypoints, and then scale your solutions to three dimensions. You will apply these skills in your second project—autonomously navigating your drone through a dense urban environment.

Lesson 1: Planning as Search
Lesson 2: Flying Car Representation

Lesson 3: From Grids to Graphs
Lesson 4: Moving into 3D
Lesson 5: Real World Planning
Lesson 6: Project: 3D Motion Planning

Project 2: 3D Motion Planning
In this project, you will move beyond the backyard test grounds and fly a drone around a complex urban simulated environment. To do so, you will load a map of a real city, plan a collision-free path between buildings, and watch your drone fly above city streets.

Course 3: Controls

In the previous course, we implemented 3D path planning but assumed a solution for actually following paths. In reality, moving a flying vehicle requires determining appropriate low-level motor controls. In this course, you will build a nonlinear cascaded controller and incorporate it into your software in the project.

Lesson 1: Vehicle Dynamics
Lesson 2: Introduction to Vehicle Control
Lesson 3: Control Architecture
Lesson 4: Full 3D Control
Lesson 5: Project: Building a Controller

Project 3: Building a Controller
In this project, you will no longer assume vehicle actuation but rather implement your very own cascaded controller in C++. You will attempt different motions (slow, fast, slalom, etc.) and analyze performance under different conditions.

Course 4: Estimation

In this course, we will finish peeling back the layers of your autonomous flight solution. Instead of assuming perfect sensor readings, you will utilize sensor fusion and filtering. You will design an Extended Kalman Filter (EKF) to estimate attitude and position from IMU and GPS data of a flying robot.

Lesson 1: Introduction to Estimation
Lesson 2: Introduction to Sensors
Lesson 3: Extended Kalman Filters

Lesson 4: The 3D EKF and UKF
Lesson 5: Project: Estimation
Lesson 6: GPS Denied Navigation

Project 4: Estimation
In this project, you will implement an EKF to estimate attitude and position from IMU and GPS data of a flying robot. After doing so, you will have implemented the full-stack for a single aerial robot!

[Optional] Course 5: Fixed Wing Aircraft

While quadrotors are the ideal test platform for aerial robotics, flying cars and other long-range aircrafts leverage the aerodynamic efficiencies of fixed-wing flight. In this course, you will learn how to adapt the concepts you've learned so far and successfully fly a fixed-wing aircraft in simulation.

Lesson 1: Introduction to Fixed-Wing Flight
Lesson 2: Lift and Drag
Lesson 3: Longitudinal Model
Lesson 4: Lateral-Directional Model
Lesson 5: Fixed-Wing Autopilot
Lesson 6: Project: Fixed-Wing Control

[Optional] Project 5: Fixed-Wing Control
In this project you will code a fixed-wing aircraft, and then implement solutions to a significantly more challenging control problem.