Zipline Simulation Challenge

Autopilot system for Zipline Challenge.

Table of Contents

About

This package is an autopilot system implementation created to go along with the Zipline Simulation Challenge in order to provide customers with safe and reliable deliveries. To ensure a robust delivery system, this autopilot controller contains a few attributes that uphold safety as its top priority. The controller's design object detection contains a lidar sample cache which contains prior scan iterations, reinforcing various objects' locations in the world. Redundant checks are performed throughout the controller with flags identifying if the vehicle is actively looking for a delivery site, avoiding collision, or approaching its recovery. The delivery controls operate independently of the speed controller, ensuring that our system only prepares a package drop when in the vicinity of a site. All of these features seamlessly combine to ensure the highest customer satisfaction.

Our autopilot system is also proven to have a high performance in terms of ROI. For Zipline engineer's development can be completed without having to look through crowded source code, wasting precious hours trying to locate where specific function of the autopilot system is being driven. With proper abstraction and modularity, the piloting controller's systems can be easily built upon or even replaced with new ones as they become available. For example, as a new object detection system is implemented, engineers can go in and create a derived controller which makes use of those detection processes independent of the deployed autopilot. This structure is sure to please both engineers and customers as they both benefit from having streamlined optimizations!

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites

What things you need to install the software and how to install them.

The commands below will assume the following version of python 3:

```
Python 3.9
```

Package Installer

```
pip
```

Alternative to pip installation(Optional):

```
pipenv
```

Documentation for pipenv installation https://pypi.org/project/pipenv/

Installing

A step by step series of examples that tell you how to get a development env running.

Setting up a working environment:

Using pip

```
python -m venv .test-env
```

On Mac/Linux

```
source ./test-env/bin/activate
```

On Windows:

```
source ./test-env/Scrips/activate
```

Install dependencies:

```
pip install -r -requirements.txt
```

Using pipenv (Optional):

Create a new project using Python 3.9, specifically:

```
pipenv --python 3.9
```

Install all dependencies needed to run project:

```
pipenv install
```

Check your installed dependencies for security vulnerabilities:

```
pipenv check
```

Activate virtual environment:

```
pipenv shell
```

Usage

To run the Zip Simulator with this autopilot:

```
python zip_sim.py python test_pilot.py
```

Authors

- @cedryc-midy - conception and develoment of piloting system

Acknowledgements

- Thank you to Zipline for giving me the opportunity to work on this project! I had a lot of fun building this
- Thank you numpy for all you do