DRONE PROJECT

I. Project overview

Build a Pixhawk drone with autonomous navigation ability.

II. Resources

Development team: 2 developers

- ❏ C++
- ❏ Computer vision

III. Technology

- Firmware: PX4 - Flight control solution for drones
- ROS: The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms and powerful developer tools, ROS has what you need for your next robotics project. And it's all open source
- We will use Gazebo to build an environment that resembles a real-life environment including models (trees, wires, buildings, etc) for testing.
- Flow:
    1. Flash PX4 firmware to Flight Controller (pixhawk mini)
    2. Flash Ubuntu to companion computer (Raspberry Pi, Jetson TX2) and install ROS
    3. Realsense camera will stream data to some ROS topics
    4. Companion computer will handle that data then build a 3D map
    5. Companion computer will find a new path to avoid obstacles then send new path to FC

IV. Issues and challenges

Theoretically, there are many options/devices to integrate a drone. For example, we can mix:

- Qualcomm Snapdragon Flight + Odroid + Realsense
- Pixhawk series + Raspberry pi + Realsense

- Pixhawk series + Jetson TX2 + Realsense, etc.

For our project, we will test 2 mixes

- Pixhawk 4 mini + Raspberry pi 3 model B + Realsense D345i
- Pixhawk 4 mini + Jetson TX2 + Realsense D345i

➔ Since they are different items, they will not always be compatible with each other. Consequently, there will always be issues that we need to fix => which is our challenge. The drone will need to test frequently to detect bugs/errors to help the real flying experiment would happen as expected.

➔ Flying on a real device is also a challenge unlike in a simulator. We may encounter unforeseen situations and to prepare for that, it demands a lot of time for research, preparation and work on algorithms.

Note

Possible hardwares for autopilot development:

- Qualcomm Snapdragon Flight
- Raspberry Pi 2/3 Navio2
- BeagleBone Blue
- Pixhawk Series

Hardwares can be used as companion computer

- Raspberry Pi,
- Odroid,
- Tegra K1
- Nvidia Jetson TX2
- Others that can run Ubuntu

Deep Camera:

- Realsense series, etc.

V. Milestones

PHASE 1: Adding avoidance ability to the drone

<u>Milestone 1</u>: Prepare drone in a simulator and build the drone

Estimation: 20 days (01/06/20 - 01/22/20) (Holiday break: 01/23 - 01/29 not working) (01/30 - 02/07/20)

Prepare in the simulator

- Test using Realsense camera with Jetson TX2
- Develop object avoidance ability in a simulator

Build the drone

<u>Milestone 2</u>: Drone development

Estimation: 1 month (from 02/10/20 - 03/11/20)

Step 1: Load PX4 Firmware into flight controller (3 days)

Step 2: Set up an environment for companion computers (5 days)

Step 3: Set up ROS on companions computers (5 days)

Step 4: Run Avoidance ROS Node (10 days)

<u>Milestone 3</u>: Testing in the real environment

Estimation: 15 days (from 03/12 - 04/01)

- Test Avoidance feature

VI. Result/ Target

As for phase I, the drone will have the abilities to avoid obstacles and mission-mode flying.

VII. Notes and discussion

1) If we are using Realsense D345i from Intel, why not use Intel Edison?

- From our evaluation, Edison's configuration is weak to support our training.

2) what obstacles will you try to avoid? wires, trees, etc. How will you test with them in the simulator and IRL (in real life)?

- We will try to build the drone to avoid mostly everything in her sight.
- We will use Gazebo to build an environment that resembles a real-life environment including models (trees, wires, buildings, etc) for testing.

3) what simulator will you use?

- Also using Gazebo

https://www.youtube.com/watch?time_continue=32&v=qfFF9-0k4KA&feature=emb_logo

4) how will you train the drone to detect obstacles?

- Realsense camera supports us detecting obstacles, Realsense will send data to our companion computer. Companion computer then extracts the data to get information about the obstacles.

5) how will you train the drone to avoid obstacles?

- Gazebo builds environments that resemble real-life environments. Then the drone will fly according to given missions to train the ability.

6) Will you train using ML or something else?

- For now, we going to use computer vision algorithms
    + local_planner is a local VFH+* based planner that plans (including some history) in a vector field histogram
    + global_planner is a global, graph-based planner that plans in a traditional octomap occupancy grid

=> We will user VFH+* for object avoidance

http://ceur-ws.org/Vol-1319/morse14_paper_08.pdf

7) Will you do training on the drone, or only inference on the drone, but training on a PC/Mac with Gazebo?

- First: We will use Software in the loop (SITL) to develop only on PC with Gazebo (not related to our real drone)
    + SITL runs on a development PC in a simulated environment and uses firmware specifically generated for that environment. Other than simulation drivers to provide fake environmental data from the simulator the system behaves normally.

- Second: We will use Hardware in the Loop Simulation (HITL):
    + HITL is a simulation mode in which normal PX4 firmware is run on real flight controller hardware (our Pixhawk item). This approach has a benefit of testing most of the actual flight codes on the real hardware.

    - Third: Transfer everything in development to our real drone then testing in the real environment

8) What is the link for Gazebo software?
- Gazebo is a powerful 3D simulation environment for autonomous robots that is particularly suitable for testing object-avoidance and computer vision. This page describes its use with SITL and a single vehicle. Gazebo can also be used with HITL and for multi-vehicle simulation.

link: http://gazebosim.org
link demo:
https://www.youtube.com/watch?time_continue=47&v=qfFF9-0k4KA&feature=emb_logo