# Fall 2017 Project Proposal (Shuang)

**General goal at the end of this quarter:**

The final objective at the end of fall 2017 is to design, test and build a single motor flying robot with sensing and controlling implemented on board.

**Background / related work / references**

Crazyflie [1] is an open source small quadrocopter project. From Crazyflie 2.0 technical report [2], Crazyflie used a cascade control loop to control the system. The outer loop of crazyflie controller controls the position of the system and the inner loop controls the attitude. Both loops in the controller are PID controllers. The control states were readed from sensors.

Paper [3] proposed a method to control a quadrocopter losing one, two and three propellers. The control strategy controls the quadrocopter spinning about an fixed axis with respect to the vehicle,and the axis is tilted for translational control. Although successfully implement the control algorithm on quadrocopter losing one and two propellers, losing three propellers case cannot provide enough thrust to lift the quadrocopter. Therefore this case was not tested on real quadcopters but only validated using simulation. The control states were readed from positioning system.

Using similar control algorithm as [3], paper [4] proposed a controllable flying vehicle with a single moving part. This system used a cascade control loop, with a outer PID controller controlling the position and an inner LQR controller controlling reduced attitude.

**Approaches for achieving the goal:**

Using crazyflie as a platform, we want to remove 3 propellers and motors from the system and modify the control algorithm to build a single motor helicopter. Through the thrust test, we found out that under the current configuration, removing 3 propellers and motors from the system will cause the lack of thrust force, similar as the losing 3 propellers case in paper [3]. Therefore making changes on physical structures are also necessary. Since current propeller and motor on crazyflie do not provide enough thrust, using larger propeller and more powerful motors will be one solution. Also, since the system only use one motor, the battery size can be less to reduce system's weight. Tests will be designed and performed by Carol to determine the choice of motor, propeller and battery.

In order to get better understanding of how to develop and test the control systems and system-level performance of a single motor helicopter, we want to start with a building a Crazyflie 2.0 model with four motors. Therefore, the primary goal of simulation is to build the model with control systems so that the simulated Crazyflie 2.0 will have the same performance as the Crazyflie 2.0 in real world.

Then, the next step is to design and simulate the control algorithm for the single motor helicopter. The control method from paper [3] and [4] are good reference to develop the control algorithm for a smaller helicopter. Therefore, computer simulation for this control algorithm will be used as a guide. Modification of the algorithm and simulations will be performed along with the design of single motor helicopter's physical body.

For paper [3] and [4], the state estimations feeding into the controller are not come from onboard sensors. For our goal, using onboard sensors to get state information is necessary. Since crazyflie already have onboard sensors, getting datas from these sensors and convert to the states we need is the next task.
The implementation of the control algorithm on Crazyflie is the last step. This task is mostly converting the algorithm into C and adjust parameters in the controller through tests.

I**ndividual task breakdown:**

My primary role in this project is to design and implement the control algorithm on the single rotor system and simulate the system-level performance using computer simulation tools.

**Past works:**

As stated previously, to have better understanding of the design process, we started building a Crazyflie 2.0 model with four motors instead of one single motor. So far, I have built a simulation model for Crazyflie 2.0 using Matlab Simulink and Simscape Multibody. To build this model, I first use simple 3D solids to model Crazyflie 2.0's battery, frame, motors and propellers. The size, mass and inertia for each parts were assigned based on the measurement of crazyflie 2.0 and data from paper [2]. Then, all the parts were assembled by adding joints, constraints and transforms between each solid blocks. The physical model then can be placed in a gravitational environment and visualized through 3D animation in Simscape Multibody by creating a simulink model. The complete 3D model is shown in Figure 1.

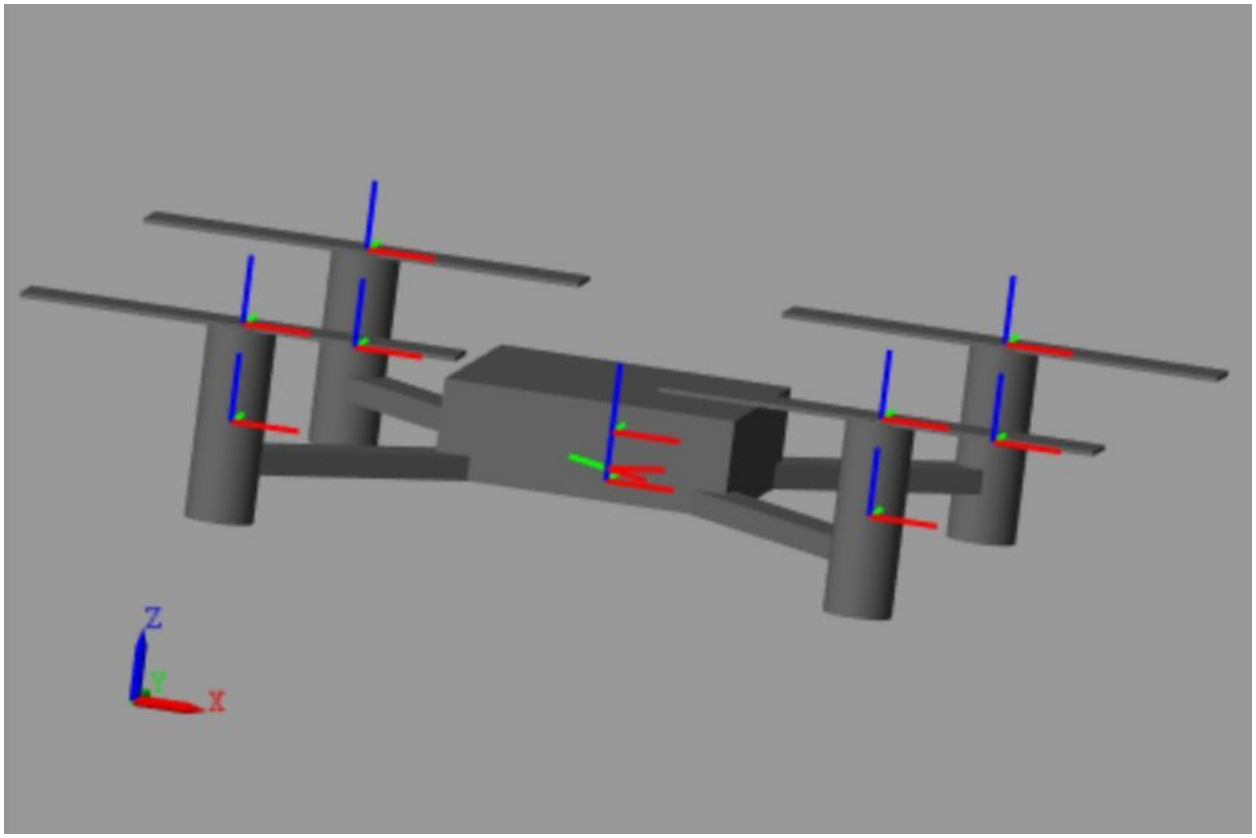


**Figure 1 - Crazyflie 3D Model**

After building the physical model, force and torque due to the rotation of propeller were added onto the system according to the equations from paper [5].

To help future development of control systems and test system-level performance, a sensing subsystem is added to the system to get simulation results of all 12 states of Crazyflie's dynamic model.

Base on this Crazyflie 2.0 model, the control algorithms proposed in paper [2] can be simulated on the system by adding a control subsystem to the current simulink model, and the performance of these algorithms can be test by analysis the output states from the sensing blocks.

**Quarter Tasks:**

The first step is to verify that the performance of Crazyflie simulation matches the behavior of real Crazyflie 2.0. To do this, we need to vary the rotation speed for each motor on Crazyflie, compare the readings of states from Crazyflie client with the simulation results from simulink sensing block.

Next, the control algorithms for Crazyflie 2.0 proposed in paper [2] can be implemented using simulink blocks and then added onto the Crazyflie simulation. Tuning parameters will then be done through the simulation tests. Then, the control algorithm will be transformed into C code and applied onto crazyflie.

After the verification of system performance on Crazyflie 2.0 model, the next step is to begin to remove propellers from Crazyflie. This process will be produced gradually by removing one propeller at each step. We want to build models for Crazyflie with one and two motor loss and implement the control algorithm proposed from paper [3].

After these process, the final step is to remove three propellers and implement the control algorithm proposed from paper [4]. Since we already know the current Crazyflie configuration cannot lift the entire system with one motor and one propeller, the structure of the single rotor system will be modified based on the tests for motor, battery and propeller performances.

**Weekly Plan:**

| Month | Week | Goals | Deliverables | Tasks |
|---|---|---|---|---|
| Oct. | 1 | Optimize simulation of Crazyflie in Matlab Simulink. | **1.**Reduce the simulation generation time to less than 10 minutes.<br><br>**2.** Ensure the Feedback from output states to propeller speed control block do not significantly increase the simulation generation time. | **1.** Use [7] as guidance to simplify model.<br><br>**2.** Use less multibody library components in the sensing blocks. |
| Oct. | 2 | Test Crazyflie model by simulate Crazyflie system behavior of hover and roll, pitch, yaw motions. | **1.** Crazyflie system behavior of hover can be simulated and control by the change of propeller speeds.<br><br>**2.** Crazyflie simulation can be put into roll, pitch and yaw motions with change of propeller speeds.<br><br>**3.** The Crazyflie simulation has similar performance as quadcopter simulation results from [6]. | **1.** Set the sum of thrust equal to gravity, and Crazyflie can maintain hovering in the simulation. Change the four propellers' speed at a same time, Crazyflie can do ascending and descending motion along z direction.<br><br>**2.1** Increase the velocity of the fourth rotor and decrease the velocity of the second rotor, see if roll motion observed.<br>**2.2** Increase the velocity of the third rotor and decrease the |

| | | | | |
|---|---|---|---|---|
| | | | | velocity of the first rotor, see if pitch motion observed.<br><br>**2.3** Increase the velocities of the first and the third rotors, decrease the velocities of the second and the fourth rotors, see if yaw motion can be observed.<br><br>**3.** Perform task 2.1-2.3 and generate position and angle plots, see if they match the results from paper [6]. |
| Oct. | 3 | Implement four motor control algorithm in Crazyflie 2.0 simulation. | **1.** Simulation results matches plots from paper [2]. | **1.** Follow paper [2], build control blocks in simulink.<br><br>**2.** Run test cases for each individual blocks to verify performances.<br><br>**3.** Plot simulation results for Crazyflie system. |
| Oct. | 4 | Implement four motor control algorithm on Crazyflie 2.0. | **1.** Generate C code that can successfully compiled.<br><br>**2.** Successfully upload the code onto Crazyflie. | **1.** Convert simulation block to C code.<br><br>**2.** Upload and replace control algorithm on crazyflie. |

| | | | | |
|---|---|---|---|---|
| Nov. | 1 | Implement four motor control algorithm on Crazyflie 2.0. | **1.** Crazyflie has similar behavior as simulated. | **1.** Test Crazyflie behavior and compare with simulation. |
| Nov. | 2 | Simulate system-level performance of Crazyflie when losing one and two propellers. | **1.** Observe one and two motors stop rotating in the simulation. | **1.** Disable one and two propellers in the simulation. |
| Nov. | 3-4 | Develop and simulate the control algorithm for losing one and two propellers. | **1.** Simulation results matches results from paper [3]. | **1.** Follow paper [3], build control blocks in simulink.<br><br>**2.** Run test cases for each individual blocks to verify performances.<br><br>**3.** Plot simulation results for Crazyflie system. |
| Dec. | 1-2 | Modify Crazyflie model for one motor case, develop and simulate the control algorithm for single motor system. | 1. Thrust from propeller is larger than gravity. | **1.** Modify the parts in the simulation. |

**Reference:**

[1] https://www.bitcraze.io/crazyflie-2/
[2] Carlos Luis, Jérôme Le Ny. (2016) Design of a Trajectory Tracking Controller for a Nanoquadcopter.
[3] Mueller, M. W., & Dandrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. 2014 IEEE International Conference on Robotics and Automation (ICRA).
[4] Zhang, W., Mueller, M. W., & Dandrea, R. (2016). A controllable flying vehicle with a single moving part. 2016 IEEE International Conference on Robotics and Automation (ICRA).

[5] Subramanian, Giri Prashanth. (2015) “Nonlinear Control Strategies for Quadrotors and Cubesats”
[6] T Luukkonen. “Modelling and control of quadcopter”
[7] https://www.mathworks.com/help/physmod/sm/mech/ug/improving-performance.html