Motion Control of Quadrotor

Rucheng Du, *rcdu@umich.edu* Cong Fu, *congfu@umich.edu* and Kai Jia, *kajia@umich.edu*

***Abstract*—In previous ArmLab and BotLab the robot arm and PID controller have been discussed. In this project, we design a flight controller for quadrotor, which enable different functions as hovering and waypoint transition based on feedback from Optitrack system. A gripper mounted on delta arm is designed on the quadrotor for grabbing and dropping blocks. To accomplish complex tasks, we implement state machine to transit between hovering, waypoint tranisition, grabbing, and dropping. We conduct many experiments to explore the flying stability with various PID parameters. Finaly, the quadrotor can hover and transit stably, and the state machine works well to grab and drop the blocks.**

***Index Terms*—Quadrotor, delta arm, OptiTrack, state machine, PID control, sensor filter.**

I. Introduction

Unmanned aerial vehicles (UAVs) especially quadrotors are commonly used in many fields these days, such as agriculture, military and entertainment industry. The quadrotor vehicle consists of four motors positioned in four corners in a square, with two pairs of counter-rotating, fixed-pitch blades, which is shown in Fig.3. Due to its specific flexibility and stability, autonomous quadrotors have been utilized for a variety of applications such as DJI's camera quadrotor Matrix 100, fire risk surveillance, search and rescue.[1] UAV with arm can achieve tasks like carrying objects in dangerous scenes and express delivery. Delta arm robot is commonly used in the production line and is famous for it's rapidity and flexibility.

In this project, we use a $SkySpecsc$ carbon fiber quadrotor platform. The system consists of four T-Motor MN-4010 370 kV motors. The system can carry payloads weighing up to 1 kg with a flight time of approximately 15 minutes [2]. A DJI Naza quadrotor controller is attached to the quadrotor to give an inner-loop controller to directly adjust the PWM duties of quadrotor motors. The pitch, yaw, roll and thrust PWM commands are given the the Naza controller. The optitrack motion capture system can give the position, orientation (pitch, roll, yaw) and speed information of the quadrotor by monitoring the center mass defined by several markers attached to the quadrotor. The delta arm is equiped with 4 Dynamixiel XL-320 motors, three are mounted in the shoulder and one installed with the gripper and they are driven by Dynamixel XL-320. All the sensor information processing and control logic is implemented in the Beaglebone Black board, which is mounted under the quadrotor.

The objective of the project is to control the quadrotor to fly stably and transit flexibly and during the flight. And the delta arm should do a pick-and-place task. We use other two robots as block holder and bucket holder. The quadrotor would transit to the adjacent position of the block and hover up in the position. Then the quadrotor reaches down to catch the block and flies to the drop area and drop the block. For the Quadrotor, we implement a speed controller with PD parameters. For pick and place task, we implement a state machine.

In the report, we first discuss the methodology of the project in Section II including the delta arm operation, delta arm design, definition of the reference frames, PID controller for the flying, sensor filter method and state machine. Then we show the performance of different tasks such as hovering, waypoint transition, state machine and grasping and dropping in Section III. At last, we discuss the results of different tasks specified for the assignment in Section IV.

II. Methodology

A. Delta arm

Delta arm robot is a kind of parallel robot. It consists of two platforms, the upper platform on which three motors are mounted, and the lower platform on which an end effector is mounted. The two platforms are connected by three arms with parallelograms. Due to the mechanical constraints of the parallelograms, the lower base is approximately parallel to the upper base. Therefore the upper base only has three translate degree of freedom. Compared with serial robot arms, delta arm with parallel structure has faster speed, owning to the reduction of payload. Delta arm is effective for picking up light objects quickly.

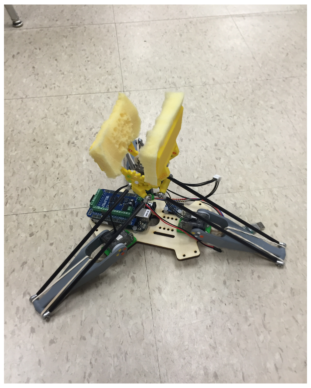
Fig. 1: delta arm.

Backlash of motors causes error of end effector in reaching target position. Due to the limited torque of XL-320 motor, the three motors mounted on the upper base are usually overloaded. To help picking up objects, we use rubber bands between upper arm and upper base to provide external force. The trick improves the performance of delta arm in terms of precision and speed.

Furthermore, to handle the space constraint, we limit the angle of each motor to avoid overload when grabbing. Below is the table of limited angle of each motor.

TABLE I: Limited Angle Table

Motor	$Maxangle$	$Minangle$
upper motor 1	78	0
upper motor 2	78	0
upper motor 3	78	0
End-effector	70	0

B. Gripper Design

Gripper is a very important part of design to grab and drop blocks efficiently. We notice that XL-320 motor has very limited torque and OLLO rivet and hole mate are generating relatively large friction force. In order to avoid overload when grasping the block, we reduce transmission mechanism for the gripper. We design a gripper driven directly by motor without any middle mechanism part except for gear mate(Fig 2). This compact gripper can make the most of the XL-320 torque to grasp objects more tightly.

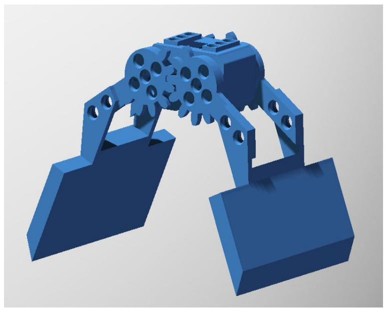
Fig. 2: Gripper Design.

In addition, we make the claw much wider to provide redundancy accommodate the tolerance caused by slightly instability and steady-state error of quadrotor when hovering. And we stick foam on the surface of claw to increase friction force when grabbing.

C. Reference Frames

The frames used in this projects are world frame from Optitrack system, quadrotor frame, and delta arm frame. Since yaw is not changed, the transformation between world frame and quadrotor frame only consists of translation when the quadrotor is in steady state, i.e., pitch and roll angle are zero. We estimate the offset between quadrotor frame and delta arm frame by calibration. The calibration process is discussed in Experiments. The frames used are shown in Fig.4.

$$T_W^D = T_Q^D T_W^Q \tag{1}$$

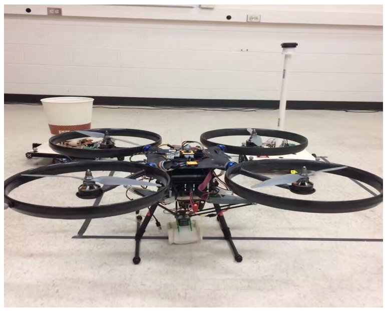
Fig. 3: Quadrotor Picture

- Optitrack Frame and Quadrotor Frame
  As mentioned before, the pitch, roll, and yaw angle are assumed to be zero in steady state. The degenerated form of transformation only has translation component.
  We try to use a frame to synthesize the quadrotor frame and the OpitiTrack frame. However, since the markers are not set symmetrically, the center of the quadrotor has an offset with the origin of the Opititrack frame and the world frame. So we add an offset to make them in the same origin.

$$T_W^Q = Tran_d \tag{2}$$

$$T_W^Q = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

- Delta Arm Frame
  The delta arm frame is fixed in quadrotor frame, as shown in Fig.3. The x axis is along the setting of the DJI Naza flight controller. The y axis is along the plane of the four rotors. And the axises are in a right-handed frame. The z axis is set downward. The relative yaw, pitch, and roll angel are obvious to observe. After calibration the offset in three dimensions, the transformation is illustrated in

$$T_Q^D = R_{x,-\pi} R_{z,\frac{\pi}{2}} Tran_{z,d} \tag{4}$$

$$T_Q^D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & x_{off} \\ 0 & 1 & 0 & y_{off} \\ 0 & 0 & 1 & z_{off} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

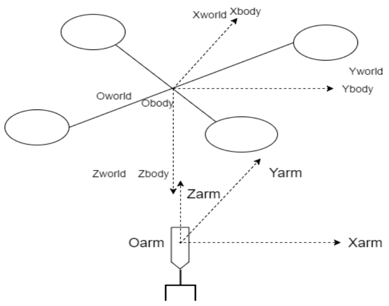

Fig. 4: Quadrotor Frame and Delta Arm Frame

D. PID controller

Provided with inner loop controller, we just need to implement outer loop controllers for roll, pitch and altitude respectively. The communication with DJI Naza flight controller is based on flight control block and transmitter. The PID controller is illustrated in Fig.5. The input is the current pose and target pose of the quadrotor; the output is the PWM signal in three channels. The PWM ranges are 1500 ± 120 for pitch, 1500 ± 120 for roll, and 1500 ± 75 for trust respectively.

In the continuous time domain, the PID control action can be written as:

$$P = K_p \cdot e(t) \tag{6}$$

$$I = K_p \cdot \frac{1}{T_i} \int^t e(\tau) d\tau \tag{7}$$

$$D = K_p \cdot T_d \frac{de(t)}{dt} \tag{8}$$

$$u = P + I + D \tag{9}$$

The outer loop controllers basiclly issue PWM commands to the DJI Naza flight controller which is same as the stick commands issued by pilot. In the control law, we set velocity as our control action. Where as distance is the objective value we need to control, so we map the distance error to a desired velocity(Fig.6). The desired velocity will increase along with increase of the distance error. The error fed into the controller is shown below.

$$V_{des} = K \cdot Error_{dist} \tag{10}$$

$$Error_v = V_{des} - V_{obs} \tag{11}$$

And we also set a saturation speed for desired velocity in order to avoid too high velocity during transition. The slope and saturation value for each transition direction is shown in Table.IV.

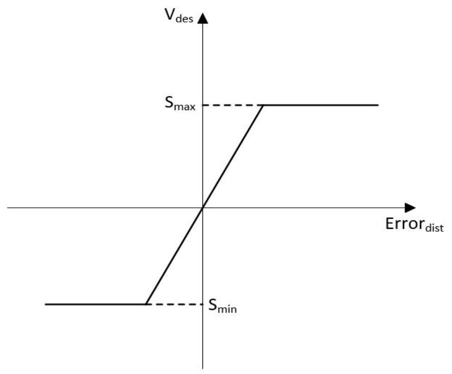

Fig. 6: Mapping from Distance Error to Desired Velocity.

TABLE II: Distance to Velocity Mapping Table

Mapping	K	S_{min}	S_{max}
D_x to V_x	3	-0.3	0.3
D_y to V_y	3	-0.3	0.3
D_z to V_z	6	-0.5	0.5

Fig.7 shows the desired velocity profile corresponding to distance. This kind of profile is reasonable because we want the quadrotor can have a smooth change of speed and a specific constant velocity towards the target position. In the beginning of transition, the speed should increase rapidly to make it robust to the instant change. And when the quadrotor gets close to the target position, the desired velocity can decrease down to zero fast.

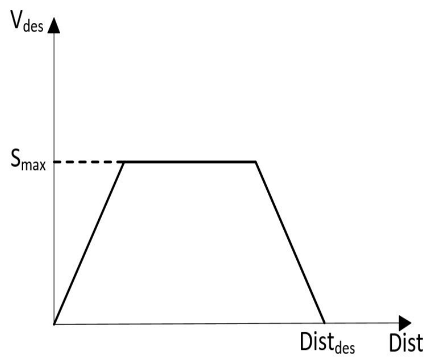

Fig. 7: Velocity Profile.

Fig.5 is the whole control block diagram. As mentioned above, we apply `proportional` term to the distance error to give a desired velocity profile. And then we use PD control to make the quadrotor to traverse the desired velocity. P term can reduce tracking error but will cause oscillation to the system. So we use large K_p value to increase response rapidity. And D term can reduce oscillation but too large K_d will make the system be sensitive to high frequency noise. So we use little K_d compared with K_p. Below is the PID parameter table which the final values we found is robust and fast-respond at the same time.

TABLE III: PID Parameter Table

Loop	K_p	K_d	K_i	e_{min}	e_{max}
$Velocity_x$	250	5	0	-0.3	0.3
$Velocity_y$	250	5	0	-0.3	0.3
$Velocity_z$	150	15	0	-0.5	0.5

E. Sensor filter

For a feedback loop control system, it is essential to measure the output of the system accurately. For this system, the output is come from optitrack motion capture system. The unstable vibration of the quadrotor and intrinsic error of optitrack system and environment disturbance will introduce noise to the actual output.

In order to avoid feeding this kind of noise to the controller, we use an exponential filter. Exponential smoothing is commonly applied to smoothen observed data. Like other low-pass filters, exponential filter is also used to remove high frequency noise. But the difference between exponential filter and simple moving average is that exponential filter decrease weights exponentially, where as simple moving average pro-

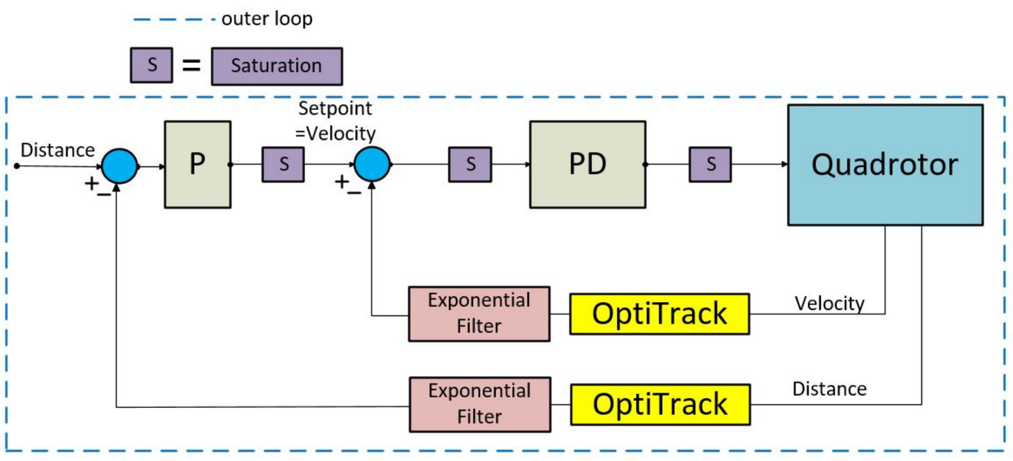

Fig. 5: PID control loop

cess data according equal weights. Below is the exponential filter formulas.

$$s_0 = x_0 \tag{12}$$

$$s_t = \alpha x_t + (1-\alpha)s_{t-1}, t > 0 \tag{13}$$

where α is the smoothing factor. We choose this factor as 0.5. And it can remove noise effectively, the result shown as below.

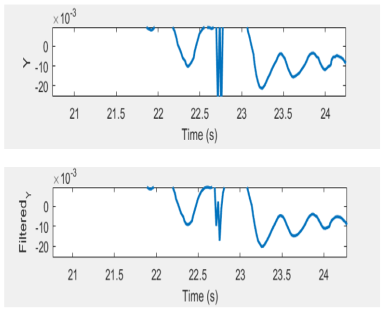

Fig. 8: Exponential filter for displacement along y axis.

F. State machine

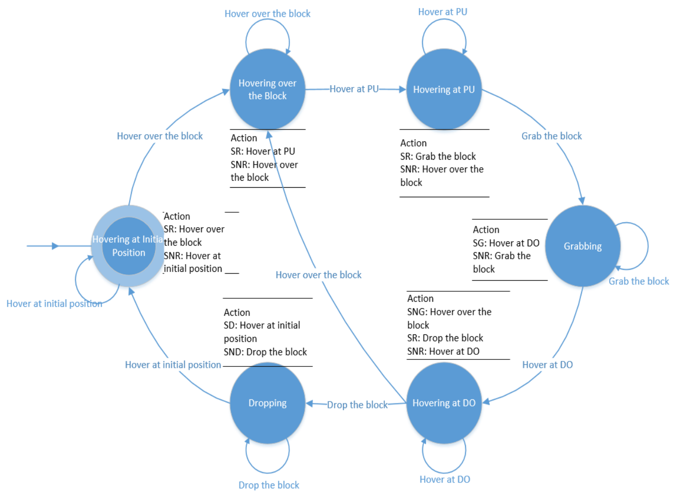

Fig. 9: The state machine for task 3.

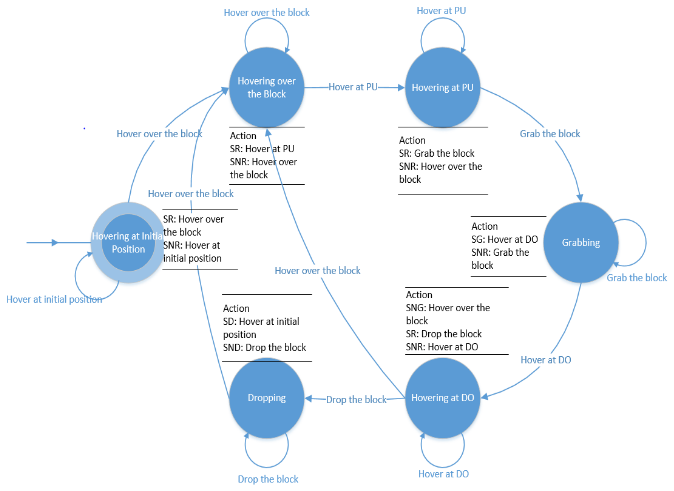

Fig. 10: The state machine for task 4.

The state machine is used to achieve the task-level control logic. In this project, quadrotor is designed to transit between waypoints, grab and drop blocks. The state machine receives feedback from Optitrack, indicating if the quadrotor reaches the waypoint, and motor, indicating if the block is grabbed or dropped. For task 2, we the states are a subset of states of task 3; without loss of generality, we only discuss the state machine for task 3 and 4 here.

1) Hovering at initial position: When the the quadroter enters autonomous mode, the target waypoint is set as the current position. The quadroter enters the initial state to hover at current position, with target velocity set to zero. Transition is based on feedback of Optitrack. If the $L1$ distance between current pose and target pose is less than threshold, the controller send a SR (sensor reached) signal to state machine:

$$\begin{aligned} E =& |x_c - x_t| + |y_c - y_t| + |z_c - z_t| + \\ & |\dot{x_c} - \dot{x_t}| + |\dot{y_c} - \dot{y_t}| + |\dot{z_c} - \dot{z_t}| < E_{thre} \end{aligned} \tag{14}$$

where (x_c, y_c, z_c) is the current position, $(\dot{x_c}, \dot{y_c}, \dot{z_c})$ is the current velocity, (x_t, y_t, z_t) is the target position, $(\dot{x_t}, \dot{y_t}, \dot{z_t})$ is the target velocity.

If the SR is received, the target waypoint is set to the position over the block; otherwise the quadrotor is still hovering at initial position until stable.

2) Hovering over the block: Instead of flying directly toward the PU position, the quadrotor is instructed flying over the position of the block first, because the oscillation in altitude may lead to collision. x_t and y_t are set to x_{PU} and y_{PU}, while $z_t = z_{PU} + z_{off}$, where z_{off} is a constant offset.

When the quadrotor is hovering over the block stably, the controller will send a SR signal following by the transition to next state: hovering at the pick up position.

3) Hovering at PU: In this state, the quadrotor is hovering at the block before grabbing. Like last state, the target pose $(x_t, y_t, z_t) = (x_{pu} + x_{off}^{pu}, y_{pu} + y_{off}^{pu}, z_{pu} + z_{off}^{pu})$. Offset $(x_{off}^{pu}, y_{off}^{pu}, z_{off}^{pu})$ is dependent on two factors: 1) the transformation and offset between quadrotor frame and delta arm frame; 2) the workspace of delta arm. The calibration of the the two frames are discussed in experiments. The state transits to grabbing when receive a SR signal.

4) Grabbing: In grabbing state, the controller keeps the quadrotor hovering at the block. Given the quadrotor position and block position, the system estimates the block position in delta arm frame, and try grabbing the block until the feedback from motor indicating the block is grabbed. The feedback is based on motor torque and a grabbing flag is set to true if the block is grabbed, and false otherwise. When the grabbing flag is true, the waypoint is set to DO position and the quadrotor transits to next state.

5) Hovering at DO: The transition to DO position is similar with previous hovering states, except that the state machine keeps verifying if the grabbing flag is true. If the block is lost, the state machine enters a error handing logic and rolls back to block position to grabbing the block again.

For error handling, the state machine keeps reading grabbing flag. If the grabbing flag becomes false, which indicates the block is lost flying towards DO position, the state machine transits to ”hovering over the block state” and repeats grabbing. With the error handling logic, the ground station command is needless.

In this state, the state machine of task 5 updates the waypoint according to the varying DO pose from Optitrack.

6) Dropping: At reaching the DO position, the quadrotor drop the block and update the grabbing state to false. For task 3, the state machine transits to ”hovering at initial position”; for task 4, the state machine transits to ”hovering over the block” and repeats grabbing operation; task 5 is same as task 3.

Before running the state machine in flying, we first test it in simulation with flight log and log player, which is discussed in Experiments.

III. Experiments

A. Hovering and PID Tuning

For thrust, we did not use filter at first. The quadrotor turned out to be too sensitive to the noise from OptiTrack system and quickly falled over. Then we added exponential filter to process the raw data. After adding filter, the quadrotor became stable along x and y axes but it will response slowly along z axis. This phenomenon indicated that we should regulate P parameter for altitude channel. When hovering, quadrotor need to maintain its altitude and overcome gravity. So the quadrotor need to response quickly when perturbated along z direction. After we increased P, the quadrotor became oscillating. Oscillation shows that we used enough P term, next step we need to regulate D term to dampen the oscillation. But D term should be little because large D term can also cause oscillation and increase steady-state error. Finally, three-dimensional position hold performance is tuned well. Along x and y axes, there is no oscillation and steady-state error. Along z axis, there is no steady-state error and very little oscillation. And as shown from Fig.11, when given a disturbance along z axis, the quadrotor can response quickly to became stable within 5 seconds.

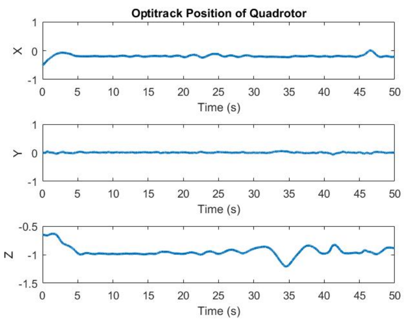

Fig. 11: Optitrack Position of Quadrotor(hover)

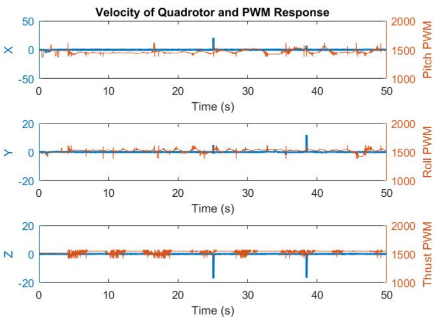

Fig. 12: Velocity of Quadrotor and PWM Response(hover)

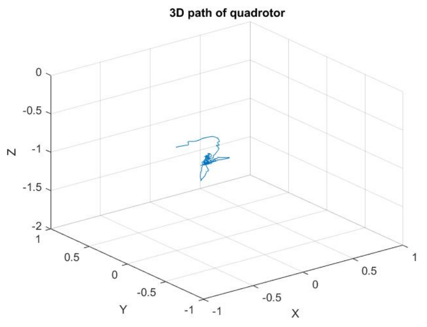

Fig. 13: 3D path of quadrotor(hover)

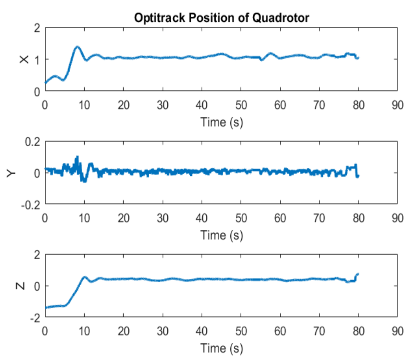

Fig. 14: Optitrack Position of Quadrotor(transition)

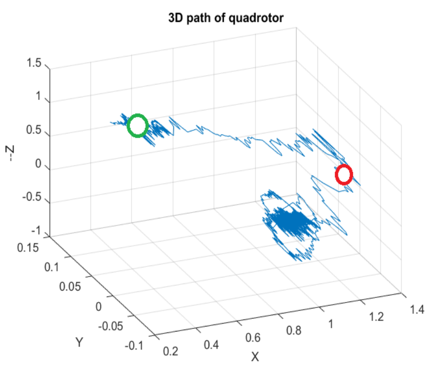

Fig. 16: 3D path of quadrotor(transition)

B. *Waypoint Transition*

In waypoint transition, we move the quadrotor from its start point (the point where it starts to autonomous fly) to a new point. The quadrotor moves up with one meter along the z axis and moves backward along x axis which is shown in Fig.16. The red circle (smaller one) is where the quadrotor starts to fly autonomous and we define it as the start point. The blue circle (the bigger one) is the goal position where the quadrotor would hover.

Fig.14 and Fig.15 shows the quadrotor can move smoothly from its start point to the goal point and the speed is moderate (see its average, there are some noise). The transition is very fast and only takes 4-5 second for one meter along one axis. The z axis is upside down in Fig.16 to make the z axis point upward. The position error in y and x axis is less than 0.05 meter and less than 0.1 meter in the z axis.

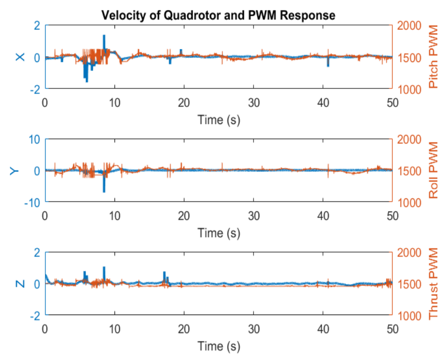

Fig. 15: Velocity of Quadrotor and PWM Response(transition)

C. *State Machine and Simulation*

One disadvantage of this project is that each group has limited test time and some operations caused by program bugs are dangerous for the quadotor and surround people. To compensate the deficiencies, we build a simulation framework with a processing pipeline:

1) Record LCM log during flying;
2) Replay the channels of interest and re-record the log:
   - `QUADROTOR_POSE_CHANNEL`
   - `CHANNELS_1_RX`
3) Play the processed log and test the system.

The reason why we select the channels and re-record the log is that the original log has messages from `ARM_COMMAND`, which conflict with the realtime messages published by `lcm_publish_loop`. The `QUADROTOR_POSE_CHANNEL` simulates the Optitrack system, and the `CHANNELS_1_RX` simulates the controller. Channel selection function is provided by the LCM software itself.

This simulation framework is used to test state machine logic. We fake several target waypoints in the trajectory in flight logs to test the states of waypoint transition logic. To test grabbing and dropping logic, we command the quadrotor to run a square and hover at four corners for a while. With this log, we set initial hovering position, PU, DO, and final destination in the four corners respectively. When the message from `lcm_publish_loop` indicates the quadrotor reaches PU position, the state transits to grabbing and delta arm starts working. The error handling logic is tested by manually remove the block from gripper. The grabbing flag changes to false and the quadrotor is observed to trying rolling back to PU position.

This simulation framework neither needs real flying nor Optitrack system feedback, which saves a lot of time waiting in queue.

D. *Grabbing and Dropping Blocks*

Quadrotor frame and dynamixel frame calibration and grabbing state verification are discussed in this section. The rotation between the two frames are easy to observe, thus we only calibrate the offsets in three dimensions.

Grabing the block is achieved by commanding an grab angle to the motor. We use $70\,\mathrm{deg}$, where the gripper proves enough friction force and the motor is not overloaded. For grabbing and dropping, two signals are available from motor to verify the state, namely torque and angle. The performance of the two measurements are presented below.

1) Quadrotor frame and dynamixel frame calibration: To estimate the $(x^{pu}_{off}, y^{pu}_{off}, z^{pu}_{off})$, we follow a two-step process:

1) Calibrate the frame offset in static status;
2) Optimize the offset in grabbing.

In the first step, we command the end effector to a know position (x_D, y_D, z_D) in dynamixel frame. The coordinate of the position in world frame from Optitrack system is (x_W, y_W, z_W). Let the quadrotor position in world frame is $(\bar{x}_W, \bar{y}_W, \bar{z}_W)$. Then the relationship between the coordinates are expressed as:

$$\begin{aligned} x_D &= (y_W - \bar{y}_W) + x_{off} \\ y_D &= (x_W - \bar{x}_W) + y_{off} \\ x_D &= -(z_W - \bar{z}_W) + z_{off}. \end{aligned} \tag{15}$$

The offsets are

$$\begin{aligned} x_{off} &= x_D - (y_W - \bar{y}_W) \\ y_{off} &= y_D - (x_W - \bar{x}_W) \\ z_{off} &= z_D + (z_W - \bar{z}_W). \end{aligned} \tag{16}$$

We then test grabbing the blocks at different positions and refine the offsets. The final calibrated offsets are shown in Table.

TABLE IV: Calibrated Offsets

x_{off}	y_{off}	z_{off}
-0.030	-0.030	0.275

2) Grabbing status verification: The torque and angle measured from motor can be used to verify if the block is grabbed or dropped. We notice that the angle is unstable when the end effector is moving. Thus we only use the torque. When the gripper is free, $torque = 0 \pm 0.02$, while when the block is grabbed, $torque = 0.43 \pm 0.03$. We set the threshold of the motor torque as 0.02: grab flag is true if $torque > 0.2$ and false otherwise.

The problem is that even the gripper does not grab the block, the torque also has spikes over 0.4, especially when the motor is rotating. We find it is hard to solve this problem by mechanical ways. Luckily, the state machine error handling logic handle this perfectly, because if the blocked is not grabbed, the torque recovers to 0 rapidly.

IV. Results

A. Position Hold

When doing position hold task, the quadrotor perform very stable as changed to autonomous mode. And when ground people pull the tether to give disturbance along x, y and z axes respectively, the quadrotor can also perform rapid response and little overshoot. What's more, the quadrotor basiclly has zero steat-state error.

B. Waypoint Transit

The competition assignment two requires us to hover in the start point and transit to the destination and hover in destination and then return to the start point. The process is shown in Fig.17. The error appears large because at this time, we didn't tune the thrust control well so it oscillate in z but it can become steady when it starts to hover.

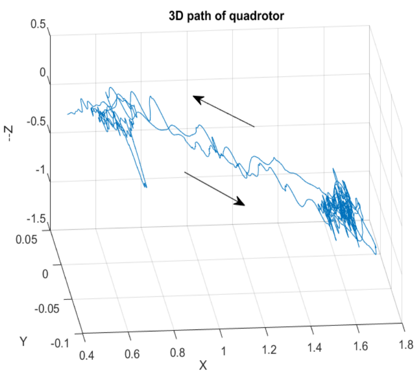

Fig. 17: Multiple Transition

C. Hand-Carried Block Pick and Place

We conducted hand-carried tests in the OptiTrack arena with block pick and place fixtures at known and then detected locations to demonstrate full hand-carried pick and place tests. When we tested this function, we calibrate the offset of each coordinate frame during test at first. After we finished calibration, hand-carried tests could be carried out successfully. This means that our state machine and delta arm inverse kinematics software work well.

D. Autonomous Flight with Block Pick and Place

Because of the time limit, we did not finish the last task. But we already test it can hover between the grasp position and the drop position. First, the quadrotor would detect the block holder robot position and hover in that position until the completion of grasping. Then, the quadrotor carries the block to the dropping robot position. At last, it would go down to the dropping altitude and drop the block. The result shown in Fig.18. The altitude values are not designed at that time.

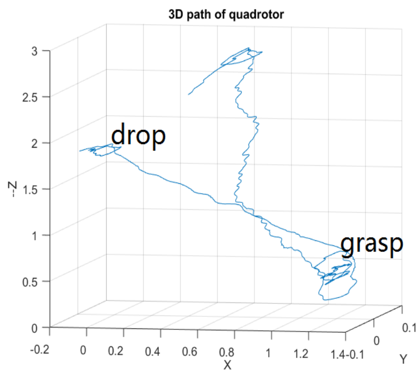

Fig. 18: Hover in grasp and drop positions

V. Conclusion

In this project, we have implemented an outer loop flight controller for an quadrotor. The quadrotor can hover at a specific position and conduct waypoint transition. And we mount an delta arm robot on the quadrotor to realize blocks pick-and-place. Finally, we discuss the performance of the controller and delta arm. From the results, we can see that our system performs reasonably well in the controller and state machine. However, we didn't tune the z axis controller in advance for the competition. We might better use a simpler controller such as position controller in the thrust controller since the velocity controller has many parameters to consider. For test3,4,5, it is a pity that we didn't finish them. If we have the further tests, we would try to make it have some reasonable oscillation during grasping because it can give it have more chance to reach the good grasp configuration.

Acknowledgment

We would like to thank Prof. Ella Atkins, Dr. Peter Gaskell, Mr. Theodore Nowak, Mr. Abhiram Krishna for supporting us in the flight experiments and give us guidance during the lab. And thank all of our fellow students in ROB 550 course for the helping and inspiring us through out the duration of the project. We are grateful to the Robotics Institute, University of Michigan for providing us with all the resources needed for the project.

References

[1] Hoffmann, Gabriel, et al. , *Quadrotor helicopter flight dynamics and control: Theory and experiment.* AIAA Guidance, Navigation and Control Conference and Exhibit. 2007.

[2] Di Donato, Pedro F., Peter E. Gaskell, and Ella M. Atkins., *Small Unmanned Aircraft Systems for Project-Based Engineering Education* AIAA Information Systems-AIAA Infotech@ Aerospace. 2017. 1377.