Trajectory Generation of a Quadrotor Transporting a Bulky Payload in the Cluttered Environments

SEONIL LEE¹, HYEONBEOM LEE¹*, (Member, IEEE)
¹School of Electronics Engineering, Kyungpook National University, Daegu 41566, South Korea
Corresponding author: Hyeonbeom Lee (e-mail: hbeomlee@knu.ac.kr).

This research was supported in part by the Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea(NRF) and Unmanned Vehicle Advanced Research Center(UV ARC) funded by the Ministry of Science and ICT, the Republic of Korea(NRF-2020M3C1C1A01086411) and in part by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2021R1A6A1A03043144).

ABSTRACT When an aerial robot carries a bulky and heavy payload, it suffers from a wavering effect because of the large inertia of the heavy mass of a payload, which can cause a serious crash or fall to the ground. We propose the path-planning algorithm for an aerial robot to pass a narrow space while transporting a bulky payload to resolve the issue. Unlike the existing conventional path-planning method that treats the payload as a point-mass, this paper generates the trajectory considering the shape of a payload and passage size of an Unmanned Aerial Vehicle (UAV). First, we design a map-generation algorithm with the Voronoi diagram by exploiting the fact that the UAV has an ellipsoidal shape in transport operation with a bulky object. The modified map data can be exploited for the global path generation such as A* path-planner. For the rotational motion of the UAV, the heading angle was determined by considering the lengths of the major and minor axes of the ellipse. The global path was optimized based on the elastic band optimization algorithm to solve the jerky motion on the trajectory. Through the simulations and experiments, we validate the efficiency of our algorithm in the environment with the narrow space. Experimental results showed that the proposed path-planning algorithm could be used for safe aerial transportation in realistic environments.

INDEX TERMS Trajectory optimization, collision-free path planning, constraints, multirotors.

I. INTRODUCTION
An increase in the popularity of online shopping has been driving the development of aerial robots for efficient logistics management. Aerial robots are used for aerial delivery [1]–[4] or warehouse management [5]–[7]. However, several issues, such as safety and computational efficiency for path planning must be addressed to transport a bulky payload. Conventional aerial transportation methods [2]–[6] are difficult to use in warehouses or factories with limited spaces because they assume an aerial robot or payload is shaped as a point-mass or circle. Considering a robot or cargo as a circle may not result in a path through a narrow space because of an empty space in the circle.

For the motion planning of a rigid body, rapidly exploring random tree (RRT) or rapidly exploring random tree star (RRT*) is a well-known planning algorithm to generate a path in a narrow space. In [2], the global path was generated using RRT* for multiple aerial robots to carry a bulky payload. In [8], an RRT-based planning algorithm and a model predictive control-based coordination of unmanned aerial vehicles (UAVs) were developed for cooperative aerial robots carrying large, cable-suspended payloads. However, the sampling-based method, such as RRT, has the disadvantage of time consumption in narrow corridors, because of random sampling characteristics. In [9], they resolved this bottleneck issue by calculating each exit as the goal position of RRT in a narrow passage. However, this method needs exact information of narrow space and is slower than conventional RRT. In addition, since the trajectory generated by RRT is not smooth, an aerial robot suffers from disturbances due to the large momentum of bulky payload when robots rotate.

For cooperative aerial robots [10]–[12], an optimization method is used. In [10,11], they presented a trajectory optimization algorithm for cooperative quadrotors with cable-suspended payloads. Despite showing promising experimen-
tal results, the algorithms in [10,11] can only be applied to aerial robots with cable-suspended payloads in which drones can change shapes. As a result, it cannot account for the effect of high momentum due to the payload. [12] presented a planning method for a fleet of rigidly attached quadrotors transporting heavy objects and trajectory optimization under these output wrench constraints. However, these methods also assumed that the obstacles were modeled as cylinders of infinite height [11] or a fleet of quadrotors forming a circle [12]. In addition, the method in [12] is only verified in simulation and the method in [11] takes a relatively long time to create a path in the narrow-space scenario.

Another method for handling a bulky payload is proposed based on Voronoi graph [13,14]. In [13], they used the overlapped region of generalized Voronoi graph (GVG) circles for sampling free space. Because this method reduces the sampling space, the time consumption of RRT, which has the disadvantage of calculating sampling in all free space, is reduced. They used RRT for sampling of space including a bubble shape with distance information from obstacles. However, it still has a bottleneck problem in confined spaces. In [14], they used k-dimensional tree (KD-tree) to find obstacle distance and made circles based on distance as radius such as GVG. They used the overlapped region for free space and optimized paths in the bounded overlapped regions to avoid obstacles. Although GVG is a useful method according to [13,14], there is still a bottleneck problem in the narrow passage [13]. The method in [14] cannot consider the heading angle of a robot, which can cause a collision of drones with a bulky payload when passing through narrow paths.

To resolve the mentioned issues, we proposed a viable planning framework for a quadrotor transporting a bulky payload (See Fig. 1). Our contributions are summarized as follows: First, a quadrotor with the desired trajectory for a bulky payload was generated in real-time in a narrow passage or complex environment. We created a map-generation algorithm based on a modified Voronoi diagram and an ellipsoidal payload model. The distance between the ellipse’s major and minor axes was used to calculate the rotation of an aerial robot. Unlike [11,13,14], our algorithm can generate the desired path of complex-shaped drone in real time. Second, we designed a heading-angle optimization algorithm to minimize the angular velocity to pass narrow passage, which minimized the wavering effect of the aerial robot. While our previous work [15] performed trajectory smoothing and speed optimization of a drone without a payload, this paper can handle a drone with a bulky payload in narrow passage. Third, we validated the performance of our proposed framework by conducting simulations and practical outdoor experiments. Moreover, we analyzed the computation time compared to the state-of-art planning algorithm for a rigid body payload. The experimental result showed the feasibility of the proposed algorithm for enabling aerial transportation.

II. METHOD

This section explains generating a global path using a Voronoi diagram with a proposed corner-extension rule. The overall structure is described in Fig. 2. First, to generate original map data in this paper, we used the FIESTA algorithm by HKUST [16]. By using Euclidean signed distance field (ESDF), which means distance from obstacles, FIESTA makes an ESDF map based on point cloud or depth image. This algorithm creates a 3D map as a point cloud (See Fig. 1). FIESTA also provides a 2D ESDF map exploiting
Algorithm 1 Corner Extension

1: Requires $r_l$, $r_s$, $V$, Obs
2: $n = \text{Obs.size}()$, $m = V.size()$
3: for $i = 0, 1, \ldots, n$ do
4:     $\text{point} = \text{Obs}[i]$
5:     for $j = 0, 1, \ldots, m$ do
6:         if $V_{rad}[j] = \text{distance}(\text{point}, V_{pos}[j])$ then
7:             $\nu = V_{rad}[j]$
8:         end if
9:     end for
10: $R = \nu^{\text{min}}$
11: if $R \geq r_l$ then
12:     extend(Obs) $\leftarrow r_l$
13: else
14:     extend(Obs) $\leftarrow r_s$
15: end if
16: end for

A. VORONOI DIAGRAM AND MAP MODIFICATION

Vertices of the Voronoi diagram can provide a safe region for path planning because the vertices are medial axis. This is very useful in determining whether a robot can pass through that space. However, in the conventional Voronoi diagram, we cannot consider the size of a payload. To resolve this issue, first, we assumed that a robot with a bulky payload is an ellipsoidal shape shown in Fig. 3 (a). The length of the major and minor radius of the ellipse was exploited to decide robot rotation and validity in space. To consider the shape of a payload, we proposed the map modification algorithm.

Our proposed method needs vertices of the Voronoi diagram based on a grid map. We generated the Voronoi diagram (i.e., Fig. 2(a)) using [17] because this algorithm provides vertices and distances from obstacles. We drew the Voronoi circles (i.e., Fig. 2(b)) which had the distance to the obstacle as the radius in the map generated by the Voronoi diagram. In this case, since the presence of numerous circles can cause degradation of computational efficiency for the map modification (i.e., Fig. 2(c)), we decreased the number of circles. If the distance between two random circles is below 20 percent of the sum of two radii of circles as follows

$$d \leq 0.2(r_1 + r_2),$$  \hspace{1cm} (1)

then one of the two circles is deleted. Here $d$ is the distance between two circles and $r_1$ and $r_2$ are the radii of $i$-th circles.

Before generating the path, the corner-extension method was used, as shown in Fig. 3 (b). In conventional A*, the shape of a robot or a payload could not be considered. The method using configuration space obstacles [18] is exploited conventionally. However, this method used only fixed shape of the robot without considering rotation. We modified the map using corner extension with the length of the major radius ($r_l$) and the minor radius ($r_s$) to address this issue.

Radius of Voronoi circles was used for operating the corner extension. In Alg.1, $V$ consists of the position ($V_{pos}$) and radius($V_{rad}$) of the Voronoi circles. $Obs$ consists of the position of obstacles in the grid map ($Obs_1$ and $Obs_2$ in Fig. 3 (b)). We found circles whose radius is the distance from the obstacle to the center point of the circle using $\text{distance}(\text{point}, V_{pos}[j])$ function (Line 6-8). For example, as shown in Fig. 3 (b), the nearest circle for $Obs_1$ is $C_1$ and $C_2$. The circle for $Obs_2$ is $C_3$. Next, we found the nearest circle and saved the radius of that circle among other circles (Line 10). The nearest circle for $Obs_1$ and $Obs_2$ is $C_2$ and $C_3$ respectively. If the shortest radius is longer than $r_l$, the obstacle size is extended with the radius $r_l$ (Line 12). For example, since $d_3$ in Fig. 3 (b) is larger than $r_l$, the corner for $Obs_2$ is extended with radius $r_l$. In this area, UAV (Unmanned aerial vehicle) can pass a narrow passage with any heading angle. On the other hand, the obstacle size is extended as $r_s$ when the radius is shorter than $r_l$ (Line 14). Since the radius of the nearest circle for $Obs_1$ (i.e., $d_2$) is shorter than $r_l$ and larger than $r_s$, the corner for $Obs_1$ is extended with radius $r_s$. In this area, UAV should rotate to travel a narrow passage, which means that only the thin shape of the UAV can pass. If the radius is shorter than $r_s$, this area is not used for the UAV with any heading angle in this narrow passage. An integrated map to use the former mentioned methods can be used A* algorithm. If the robot turns appropriately, the map makes the UAV pass the narrow corridor.
In this section, we explain our optimization method for the rotation flag. The flags are used in section III-B.

To simplify the problem, we assumed that the waypoints were densely distributed over a trajectory and the angle between \( Q_k \) and \( Q_{k+1} \) (i.e., \( \theta_k \)) was smaller than 1 radian. This assumption was easily solved by using trajectory interpolation. In this case, we could minimize the sum of the lateral force \( N_k \) for the whole trajectory, but the lateral force \( N_k \) constrained as

\[
\|N_k\| \approx 2\|F_k\|\sin(\theta_k/2) \approx \|Q_{k+1} - Q_k\|\theta_k
\]

\[
\approx \|Q_{k+1} - Q_k\|^2/R_k \leq \frac{d^2}{R_{\text{min}}},
\]

Here, \( d \) is the maximum distance between \( Q_k \) and \( Q_{k+1} \). \( R_{\text{min}} \) means the minimum radius of the safe region. If \( R_{\text{min}} \) is smaller, the angle between waypoints is close to 90 degrees. Finally, waypoint \( Q_k \) is minimized based on \( \|N_k\| = \|2Q_k - Q_{k-1} - Q_{k+1}\| \) as

\[
\min_{Q_1, \ldots, Q_m} \sum_{k=2}^{m-1} \|2Q_k - Q_{k-1} - Q_{k+1}\|^2
\]

s.t. \( Q_1 = P_1, Q_m = P_m \),

for \( k = 2, \ldots, m - 1 \),

\[
\|P_k - Q_k\| \leq R_{\text{radius}},
\]

\[
\|2Q_k - Q_{k-1} - Q_{k+1}\| \leq \left\{ \frac{d^2}{R_{\text{min}}} \right\}
\]

For the safety of a drone, \( Q_k \) should be regenerated within the region \( R_{\text{radius}} \) (See Fig. 5 (a)). In this paper, the safety regions vary depending on the radius of the nearest Voronoi circle. For example, if the radius of the nearest Voronoi circle is larger than \( r_s \) and smaller than \( r_l \), then the \( R_{\text{radius}} \) is set to \( r_s \), otherwise, \( R_{\text{radius}} \) is set to \( r_l \).

### B. HEADING-ANGLE SELECTION AND OPTIMIZATION

The algorithm mentioned in Sec. III-A was used only for the waypoints and not the heading angle. For the decision of the heading angle, we first decided the direction of the optimized path from section III-A and by exploiting the rotation flag described in Sec. II-C, a robot was rotated to pass a narrow passage. In this case, the heading angle was selected. The
Algorithm 2 Heading-angle decision

1: Requires $\Psi, Q, Y, m$
2: for $k = 0, 1, \ldots, m$ do
3: \quad $f \leftarrow flag(Q_k)$
4: \quad if $f == 1$ then
5: \quad\quad $p^* \leftarrow$ initialRotation($\Psi_k^f$)
6: \quad\quad $p^n \leftarrow$ nextRotation($\Psi_k^{f+1}$)
7: \quad\quad $p^c \leftarrow (p^* + p^n)/2$
8: \quad end if
9: \quad if $k == p^c$ then
10: \quad\quad addConstraint($Y_k = \Psi_k^{f=p^c}$)
11: \quad end if
12: end for
13: Optimization($\Psi, Y$) $\leftarrow$ eq.(8)

 drone rotated in a direction that minimized rotation in a positive or negative direction based on the target path. Until passing the narrow space, rotation direction was maintained.

Heading-angle optimization is needed for minimizing the wavering effect. Because this effect is caused by the momentum of the payload, reducing the heading-angle rate of the desired trajectory is helpful. To minimize the heading-angle rate, the optimization formula is set for minimizing the heading-angle acceleration. The heading-angle $\Psi$ and the optimized heading-angle $Y$ are defined as follows:

$$
\Psi = \{\Psi_1^f, \Psi_2^f, \ldots, \Psi_m^f\},
$$

$$
Y = \{Y_1, Y_2, \ldots, Y_m\}. \tag{7}
$$

The superscript $f$ means the flag for robot rotation. For the optimization process, we define the angular velocity (i.e., $\dot{Y}_k - Y_{k-1}$) and angular acceleration (i.e., $2\ddot{Y}_k - \dot{Y}_{k-1} - \dot{Y}_{k+1}$). For the heading angle optimization, we have to consider the following situation as constraints. First, to synchronize the moving direction of the drone with the direction of the camera, we added constraint that the difference between $\Psi_k^f$ and $Y_k$ is lower than $\pi/2$ (11). This constraint kept the drone looking forward. Second, because of the discontinuity of the heading angle near $-\pi$ and $\pi$ caused by arc-tangent function, the desired heading angle could be volatile during optimization process (i.e., the left blue box in Fig. 5(b)). To solve this problem, we represented the heading angle as a continuous variable by adding or subtracting $2\pi$ to the original value. Third, since the optimization process minimized the variation of the heading angle, the target angle might not change even at the point where the target heading angle should change at the rotation point (See the right red box in Fig. 5(b)). So, an additional constraint was required for both the target heading angle and the optimized heading angles to have the same value when passing the narrow passage (10).

The optimization process with the initial and final heading angle (9) can be written as

$$
min \sum_{k=2}^{m-1} \|2Y_k - Y_{k-1} - Y_{k+1}\|^2, \tag{8}
$$

$$
s.t. \, Y_1 = \Psi_1^f, Y_m = \Psi_m^f
$$

$$
Y_k = \Psi_k^{f=p^c}, \tag{10}
$$

for $k = 2, \ldots, m-1, \|\Psi_k^f - Y_k\| \leq \frac{\pi}{2}. \tag{11}$

The detailed process for the heading angle optimization is described in Alg. 2. We first found the rotation point (i.e., $p^s \in 1, \ldots, m$) given the optimized waypoints (Line

---

**FIGURE 7:** Trajectory tracking of cooperative aerial manipulators: (a) Gazebo simulation, (b) Single drone with a payload, (c) Two drones with a payload, (d) 3D map, (e) Trajectory tracking for a single drone, and (f) Trajectory tracking for two drones.
3-5). Then, by using \( p^e \) and the next rotation flag (i.e., \( p^e \in 1, \ldots, m \)) obtained by \textit{nextRotation} function (Line 6), we computed the middle waypoint \( p^e = (p^n + p^s)/2 \) between two rotations (See Fig. 6, Line 7). To maintain the rotation while traveling the narrow passage, we added further constraints for the optimization (Line 10). By solving optimization problem, we finally obtain the trajectory (Line 13).

### C. VELOCITY OPTIMIZATION

To apply our algorithm for the experiment, we interpolated the trajectory with respect to the user-defined arrival time and optimized the velocity in each waypoint. In a simulation or experiment, the distance is too far to command the UAV to position control, resulting in a very fast velocity. To resolve this issue, we interpolated the trajectory using a polynomial equation and the user-defined maximum velocity information. The detailed process is described in our previous work [15].

### IV. SIMULATION

To validate the proposed algorithm, we tested our method through simulations and experiments. Our trajectory generation is implemented in C++ and the optimization tool is developed based on the Gurobi optimizer [20]. Desktop specifications are 2.9 GHz Intel Core™ i7-10700 CPU and 32 GB RAM. System environments are robot operating system (ROS) in Ubuntu 18.04. We used a gazebo-based RotorS simulator manufactured by ETHZ [21]. The payload object has a mass of 300 g and 1.5 m. In the simulation, we used the position controller already implemented on the RotorS simulator without any modification.

#### A. SIMULATION RESULT

We designed the maze such as Fig. 7 (a). The UAV has the Visual-Inertial (VI) sensor that needs to generate a map by FIESTA [14]. The generated map is shown in Fig. 7 (d). For path planning, this map is converted to a grid map. We set resolution of the grid map as 0.1 m per pixel. The map size is 18.1 m × 21.3 m maze.

The modified map is used for the proposed path planners. When using a drone as illustrated in Fig. 3 (a), the generated trajectory is shown in Fig. 7 (e). The drone with a bulky payload can pass the narrow passage satisfactorily. For two drones with a bulky payload as shown in Fig. 7 (f), each trajectory can be easily obtained from the results as shown in Fig. 7 (e). To do so, the mean value of each trajectory is set to the same as a single UAV trajectory. This result shows that the proposed method is useful to not only for a single drone but also for multiple drones.

#### B. COMPARISON WITH OTHER METHODS

To compare our algorithm with the state-of-art planning algorithm for a rigid body payload such as RRT-Connect [22], RRT* [23], Informed-RRT* [24] and Fast Marching Tree star (FMT*) [25], we ran simulations using the Open Motion Planning Library (OMPL) [26]. This library can consider SE2 state-space condition that has the position and rotation information. To consider the robot shape, Flexible-collision Library (FCL) [27] is exploited. FCL is helpful to decide collision-free rotation at a position. We mixed both the libraries for collision-free path planning. Table 1 shows the calculation time and total path length. In Table 1, RTC means RRT-Connect algorithm and I-RRT* means Informed-RRT* algorithm. The total heading angle variations (Total yaw in Table 1) are measured based on the following equation

\[
E_{\psi,k} = \sum_{k=1}^{n} \mid \psi(k) - \psi(k-1) \mid.
\]  

Our method shows satisfactory performance at time consumption and rotation. Although our algorithm is a little bit longer than RRT*, I-RRT* and FMT* in point of total length, our algorithm is safer because the center of the payload and the wall are further apart in a narrow path.

#### C. ANALYSIS

We also compared the stability of a drone while transporting a bulky and heavy payload with or without an the proposed optimization approach. Fig. 9 shows the comparison result. The red line means pitch angle and the blue line means roll angle. If our method is not used when rotating in the corner, the heading-angle angular velocity fluctuates, as shown in Fig. 9 (a). It affects the pitch and roll of the UAV and become unstable (See Fig. 9 (a)). Our proposed method is described in the Fig. 9 (b). Because our method optimized the whole path and minimized the heading angle rotation, the heading angle was changed smoother than the conventional method. It makes the pitch and roll of the UAV more stable than the Fig. 9 (a). This result shows that optimizing the heading-angle in the whole path provides a stable trajectory.

### V. EXPERIMENT

#### A. UAV STRUCTURE

We used the IFO-S drone as a UAV and installed T265 and global positioning system (GPS) module for odometry. For generating the map by FIESTA [14], Intel Real-sense D455 was used. The altitude sensor was TF-mini LiDAR manufactured by Benewake. We used NVIDIA Jetson Xavier NX as an onboard computer and pixhawk-mini as a flight controller.
B. OUTDOOR EXPERIMENT

To validate our methods, we designed an obstacle that had a narrow passage (See Fig. 1). The passage width of the obstacle was 0.85 m. The UAV needed to rotate the heading-angle to pass the passage. We set the velocity constraint to 0.1 m/s and altitude to 0.6 m. Fig. 11 shows the trajectory and 3D map, that the UAV passes the obstacle. Fig. 12 shows the attitude and position of the UAV with the bulky object following trajectory. In the 28-30s, the UAV fluctuated because of the obstacle structure. At other times, the UAV followed the trajectory well.

VI. CONCLUSION

In this paper, we proposed the method for trajectory generation for the drone transporting a bulky payload through the narrow corridor. The map was modified to account for the lengths of ellipse’s major and minor radii by using the size of the Voronoi circle. In the modified map, our method found a safe free space, generated the desired trajectory, and selected suitable rotation for narrow space. An optimization approach minimized unstable movement at cornering. Through simulations and experiments, we verified that our method guaranteed safety and computational efficiency for path planning.

REFERENCES

[1] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, “Cooperative grasping and transport using multiple quadrotors,” in Distributed autonomous robotic systems. Springer, 2013, pp. 545–558.
FIGURE 11: Map generated by FIESTA and snapshot during the experiment

FIGURE 12: Position and roll, pitch, yaw data, linear velocity of each axis in experiments

[1] C. Y. Son, H. Seo, D. Jang, and H. J. Kim, “Real-time optimal trajectory generation and control of a multi-rotor with a suspended load for obstacle avoidance,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1915–1922, 2020.

[2] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, “Fast autonomous flight in warehouses for inventory application,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3121–3128, 2018.

[3] W. Kwon, J. H. Park, M. Lee, J. Her, S.-H. Kim, and J.-W. Seo, “Robust automated navigation of unmanned aerial vehicles (uavs) for warehouses’ inventory application,” IEEE Robotics and Automation Letters, vol. 5, no. 1, pp. 243–249, 2020.

[4] A. Ham, “Drone-based material transfer system in a robotic mobile fulfillment center,” IEEE Transactions on Automation Science and Engineering, vol. 17, no. 2, pp. 957–965, 2020.

[5] V. Spurný, M. Petřík, V. Vonasek, and M. Saska, “Cooperative transport of large objects by a pair of unmanned aerial systems using sampling-based motion planning,” in 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2019, pp. 955–962.

[6] J. Szkandera, J. Kolingerová, and M. Mařák, “Narrow passage problem solution for motion planning,” in International Conference on Computational Science. Springer, 2020, pp. 459–470.

[7] J. Fink, N. Michael, S. Kim, and V. Kumar, “Planning and control for cooperative manipulation and transportation with aerial robots,” The International Journal of Robotics Research, vol. 30, no. 3, pp. 324–334, 2011.

[8] B. E. Jackson, T. A. Howell, K. Shah, M. Schwager, and Z. Manchester, “Scalable cooperative transport of cable-suspended loads with uavs using distributed trajectory optimization,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 3368–3374, 2020.

[9] Z. Wang, S. Singh, M. Pavone, and M. Schwager, “Cooperative object transport in 3d with multiple quadrotors using no peer communication,” in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1064–1071.

[10] D. Kim, J. Lee, and S.-e. Yoon, “Cloud rrt*: Sampling cloud based rrt*,” in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 2519–2526.

[11] S. Kim, S. Choi, H. Kim, J. Shin, H. Shim, and H. J. Kim, “Robust control of an equipment-added multirotor using disturbance observer,” IEEE Transactions on Control Systems Technology, vol. 26, no. 4, pp. 1524–1531, 2017.
