

Article

UAV Payload Transportation via RTDP Based Optimized Velocity Profiles

Abdullah Mohiuddin ^{1,*}, Tarek Taha ², Yahya Zweiri ^{1,3}  and Dongming Gan ¹

¹ Khalifa University Center for Autonomous Robotic Systems, Khalifa University of Science and Technology, P.O. Box 127788, Abu Dhabi, UAE

² Algorithma's Autonomous Aerial Lab, P.O. Box 112230, Abu Dhabi, UAE

³ Faculty of Science, Engineering and Computing, Kingston University London, London SW15 3DW, UK

* Correspondence: abdullah.mohiuddin@ku.ac.ae; Tel.: +971-2-501-8558

Received: 9 June 2019; Accepted: 12 July 2019; Published: 8 August 2019



Abstract: This paper explores the application of a real-time dynamic programming (RTDP) algorithm to transport a payload using a multi-rotor unmanned aerial vehicle (UAV) in order to optimize journey time and energy consumption. The RTDP algorithm is developed by discretizing the journey into distance interval horizons and applying the RTDP sweep to the current horizon to get the optimal velocity decision. RTDP sweep requires the current state of the UAV to generate the next best velocity decision. To the best of the authors knowledge, this is the first time that such real-time optimization algorithm is applied to multi-rotor based transportation. The algorithm was first tested in simulations and then experiments were performed. The results show the effectiveness and applicability of the proposed algorithm.

Keywords: UAV; energy optimization; dynamic programming; aerial transportation

1. Introduction

Unmanned aerial vehicles (UAVs) have been widely used by researchers, security and law enforcement agencies, search and rescue operators, firefighters, farmers, filmmakers, photographers and delivery companies. UAVs can perform tasks in confined spaces or in hazardous environments. UAVs are now evolving from just being a sensor to air-borne manipulators. Researchers have started to focus on attaching manipulators to the UAVs, which enables them to perform tasks such as opening a valve, and picking and placing objects. Other than manipulation, researchers claim that UAVs, specifically small UAVs, have the potential to significantly improve the research in remote sensing [1]. Recently, a Canadian firm has started drone based payload delivery services [2]; another firm started coffee delivery via drones in Australia [3]. Recently, a drone was tested for its ability to transport organs for organ transplants [4]. UAVs are used to transport deform-able linear objects such as hose and goods [5,6]. The Swiss post used UAVs to transport medical samples, which helped them to reduce a 45 min car journey to a few minutes of flight [6]. A recent study showed that UAVs delivery might even help with reducing greenhouse emissions caused by the freight industry [7].

Motivated by the potential of UAV based payload transportation, we have investigated the possibility of applying a real-time multi-criteria optimization approach. Continuous improvement of the computational capabilities and reduction of the size of computational platforms have allowed for installing them on-board small UAVs. Therefore, dynamic programming based optimization techniques can now be applied for controlling UAVs. Specifically, the contribution of this paper is the development, numerical and hardware testing of a real-time dynamic programming algorithm for achieving velocity optimized energy efficient aerial transportation using a multi-rotor platform. The optimization algorithm can be applied in real-time scenarios, and it considers the aerodynamic influences.

Relevant Work

Energy consumption in multi-rotors is critical since they are powered by batteries that have limited endurance. The major power consumption results from the motors that are rotating the propellers to generate thrust that keeps the UAV airborne. The electrical energy consumed by the motors depends on the thrust requirements, and also includes the electrical losses due to heat and friction and overall propulsion system efficiency. These include losses of motors and electronic speed controllers. Although several studies have focused on improving battery charging methods for multi-rotors such as wireless charging [8,9], still, on average, the flight time of multi-rotors after every charge is around 20 to 30 min [10], which limits the applications of the multi-rotors. Other causes of energy consumption include the autopilot or any companion computer attached to the aerial platform, sensors such as a camera for visual servoing or communication links. The lateral motion of hex-rotors also deals with parasitic drags, which results in a higher requirement of rotor torque that results in more energy consumption. The energy constraints in aerial transportation can be addressed via two approaches. One approach is the design stage of the aerial transportation. The second stage is the energy savings by the efficient planning of the operation. Energy savings in the design stage can be achieved in various ways, including reducing the amount of weight carried by the UAV. Batteries account for up to 50% of total UAV mass for small UAVs [11], the addition of further payload mass drastically affects the already burdened energy budget. Flying with an optimum mass can maximize the endurance of the UAV [12]. In some cases, an aerial platform could utilize a hybrid design such as [13] that travels on the ground when the flight is not necessary to save energy.

The second energy saving approach is related to efficient motion planning. It is possible in these cases to choose a minimum energy consumption path from multiple available paths, generated by a path planner [14] for a single UAV. The method developed by [14] performs an offline energy efficient path planning based on Dijkstra's Algorithm.

A paper by [15] showed that energy consumption increases with forward velocity after a first decrease, when compared to the energy consumption in a hover condition. Another study by [16] evaluated the direct relation between energy efficiency and the speed of a multi-rotor system. It was argued and proved by [16] that a hex-rotors system have to consume energy to continuously support their weight and minimizing the time in the air could result in overall energy savings. Experiments showed that there was a 29% difference between the hex-rotor transportation with two different speeds. A novel path-following controller was presented by [16] in which the speed of the rotor-craft is a dynamic profile that varies with the geometric requirements of the desired path.

Minimizing time in the air to save energy is significant in case of moderate speeds up to 10 m/s where parasitic drag can be ignored [17]. However, for higher speeds and for bigger hex-rotors, this parasitic drag can be significant and would lead to more energy consumption. In some cases, a cage surrounds the multi-rotor, which also adds to the parasitic drag [18]. It can be inferred from previous studies that energy efficiency can also be increased by incorporating the aerodynamic factors and making velocity decisions that result in less overall energy consumption. It is also possible to generate a minimum energy consumption path for a single UAV as performed by [10]. The proposed method in [10] takes the brush-less DC motor model into account and finds the optimal path by using a predefined initial and final configuration of the multi-rotor. However, the method proposed by [10] was performed offline and it does not take into account the uncertainties present between actuation and expected motion. Furthermore, a lack of feedback in the proposed strategy makes it unfeasible for real-time multi-rotor transportation.

This research study presents an optimization algorithm for a platform's velocity based on a real-time dynamic programming algorithm (RTDP). The RTDP algorithm is developed utilizing the original dynamic programming principle presented by [19]. The developed optimization algorithm is inspired by a real-time dynamic programming introduced in [20] for the optimization of velocity to minimize energy consumption in automobiles. To the best of the authors' knowledge, RTDP based energy optimization for multi-rotors energy savings has not been attempted before.

In Section 2, we will describe the fundamentals of the RTDP algorithm used for energy optimization. After that, Section 3 will discuss the experimental methodology used in this study, and results of the numerical experiments will be presented and discussed in Section 3.7, followed by software in the loop simulation results in Section 4. Finally, hardware experiments are also discussed in Section 5 followed by conclusions and references.

2. Algorithm Description

The optimization algorithm is based on minimization of a cost function as described in Equations (1) and (2). The optimization algorithm is applied in two steps as shown in Figure 1. The first step is applied on a pre-defined horizon length, while not considering the end of the journey. The cost function for first step with a non-finite stage is described in Equation (1):

$$\min_V J_N = g(x_N) + \sum_{k=0}^{N-1} J_k(J_e, J_t, \lambda), \tag{1}$$

where J_N is the cumulative cost at the last stage, V is the reference velocity of the UAV and decision variable, J_k is the cost at a particular time step k which is a function of the decision variable V , $J_e(x_k, u_k, z_k)$ is the energy cost of a particular step k , and is further explained in Section 2.1, $J_t(x_k, u_k, z_k)$ is the journey time cost of the step k and λ is the weighing factor as described further in Section 2.1; $g(x_N)$ is the terminal cost. The terminal cost is used to approximate the cost incurred after the horizon up to the end of the journey at an unknown distance, required to minimize the impact on future horizons of decisions made for the current horizon.

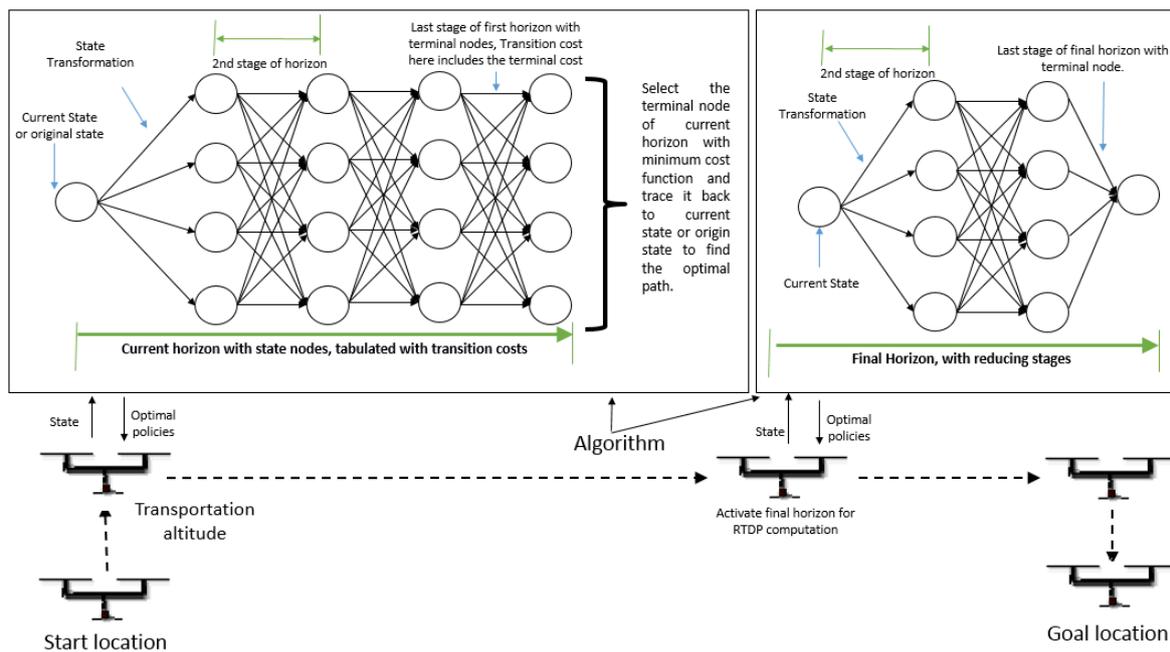


Figure 1. Process flow diagram; first, the multi-rotor takes off and reaches the transportation altitude, the optimization algorithm then sweeps through search-space in the rolling horizon to find an optimum policy. The multi-rotor moves with this optimum policy for this distance step. This repeats for every distance step, until the final horizon is activated. During the final horizon, at each computation, the number of stages is reduced and the final state is given.

The second step is when the aerial vehicle enters the last horizon; during this step, the end of the journey is considered. The main difference between the first step and the second step is that, in the former, the cost function includes a terminal cost, while the second step defines a final state of the system. The second step of the optimization is activated when the vehicle enters the last horizon.

This step is called the Dynamic programming (DP) algorithm with the finite stage. In this step, the DP algorithm is applied at the start of each stage, but the number of stages is reduced for each DP computation, based on the proximity of the vehicle to the goal position. The final state is predefined and, in this case, it is considered to be a velocity state equal to zero. Equation (2) describes the second step of the optimization algorithm:

$$\min_V J_N = \sum_{N-1}^{k=0} J_k(J_e, J_t, \lambda). \quad (2)$$

As described in [21], apart from the minimization of cumulative cost function, we also need to describe the system in a discrete time model. The overall model is discretized such that it can be written as Equation (3):

$$x_{k+1} = f_k(x_k, u_k, z_k), \quad (3)$$

where $k = 0, 1, 2, 3, \dots, N - 1$ is the time index showing the stages. Here, x_{k+1} represents the current state of the system and f_k is the transition relationship, as a function of the current state x_k and u_k , the control decisions which are defined further below. x_k is the state vector defined as

$$x_k = [V], \quad (4)$$

where V is the velocity of the payload. u_k is the decision variable defined as

$$u_k = [V_1], \quad (5)$$

where V_1 is the reference velocity of the UAVs in the transport direction. As a simple case, this optimization is performed for the transportation from point A to B having the same altitude. The disturbance can be defined as z_k

$$z_k = [wind \quad \rho], \quad (6)$$

which is based on wind speed and ρ is the air density at the transportation altitude.

In order to apply dynamic programming in real time, the problem is divided into horizons. If the DP sweep is performed on the complete journey from point A to point B, then the computational costs would increase, rendering the method to be incapable of working in real time. The application of DP sweep on the horizon length reduced the computational time, therefore the DP sweep is performed frequently. The DP sweep is performed every time the vehicle moves a predefined distance step. Each horizon includes a pre-defined set of distance intervals. The current horizon is the one that includes the current state of the agent. The application of the DP algorithm can be seen in Figure 1. It can be seen in Figure 1 that there are two types of DP sweeps performed: one type of DP sweep with terminal costs and the other type without terminal costs. The second step of DP sweep is required because the final state of the UAV is in hovering mode in order to drop the payload at goal position.

During the start of the journey, the DP sweep with terminal costs is performed as follows. The transition costs are calculated for all stages of current horizon only. Starting with the origin or taking the current state of the agent, the transition costs of all possible states will be calculated and stored for the first stage. Now, for the second stage, at each node, the cumulative transition costs of reaching this node from all states of the first stage will be calculated. This process will be repeated from $k = 0$ to $k = N - 1$ until the terminal node. After reaching the terminal node of the current horizon, the transition cost to the terminal node will include the terminal cost as shown in Figure 1 and explained in Equation (1). Starting from the terminal node $k = N$ with minimum cost, the optimal solution will be traced back to node $k = 0$. After obtaining the optimal policies, control actions will be immediately utilized until the agent reaches the next stage. When the agent reaches the next stage, we will take the current state of the agent and consider it as the origin and run the algorithm again until the end of the rolling horizon. After reaching the terminal node of the current horizon, we will again

trace back the solution and immediately implement the optimal policies. The process will be repeated until the vehicle enters the last horizon.

When the vehicle enters the last horizon, the DP sweep is performed following the cost function Equation (2) and as shown in the end part of the journey, as shown in Figure 1. In the last horizon, cumulative transition costs will be calculated while considering the final state, which is pre-defined. Hence, the terminal cost will not be calculated. After each stage, the number of stages will be reduced, until the vehicle reaches its destination.

2.1. Definition of Cost Function

The cost function is defined based on two parameters: energy consumption and time spent. A parameter λ is defined as the weighting factor to select the preference between the time spent to reach the goal or the energy consumption to reach the goal. If λ is close to one, the energy consumption is given a priority and, if λ is close to zero, then minimization of time spent is of higher priority. In order to make these factors comparable to each other, a normalization is required. The combined cost function can be written as follows:

$$J_k = J_e\lambda + J_t(1 - \lambda), \quad (7)$$

where J_e and J_t are the normalized energy and journey time cost functions, respectively.

2.1.1. Normalization of Time Spent

The time spent to reach the goal position can be optimized by minimizing the following cost function at each backup operation, when the algorithm is going through the options within the search space. In order to make it compatible with the other part of the cost function, the time cost function is normalized and made dimensionless using $J_t = \frac{T_a}{T_n}$, where T_a is the time spent by following an option from search space and T_n is the normalization factor to make it compatible with the power consumption factor in the combined cost function. This normalization factor is carefully selected to make sure that the normalized time spent is less than one. This allows for comparison between the energy and time. This dimensionless component of the cost function reduces the time spent to reach a goal position because of the cost function's minimization.

2.1.2. Normalization of Energy Consumption

Using the weighting factor, the cost function for the energy can be written as $J_e = \frac{E_c}{E_n}$, where E_c is the energy consumed during the transition between two states, whereas E_n is the normalization factor used to make it compatible with the time factor. The factor E_n is selected such that the normalized energy consumption factor turns out to be less than one.

2.1.3. Terminal Cost

The terminal cost is calculated during first step of the optimization. The terminal cost is calculated during DP computation; when the cost function of the last node is calculated, the terminal cost is considered a multiple of the last transition cost to reach the last node. Since a DP sweep is repeatedly performed at each distance step for the length of the horizon, the terminal cost penalizes the decisions that can benefit the current horizon at the cost of the future horizons. An example of such is the reduction of velocity significantly at the end of current horizon, which would result in high acceleration in the beginning of the next horizon.

2.2. Model of the Evaluated System

The system consists of a hex-rotor as shown in Figure 2. The UAV model was simplified by assuming that the UAV structure is rigid and CoG (center of gravity) and geometric centre is at origin B of the body frame and coincide. Let $P = [x, y, z]^T$ be the position vector of the center of mass of

the hex-rotor relative to the fixed inertial frame $\varepsilon = [X', Y', Z']^T$. The orientation of the hex-rotor is expressed in Euler angles as $\Phi = [\phi, \theta, \psi]^T$, where ϕ is the roll angle about the x -axis, θ is the pitch angle about the y -axis, and ψ is the yaw angle about the z -axis of the hex-rotor UAV. Six rotors attached to identical brush-less DC motors are rotating with a speed ω_N . The following equations best describe the translational and rotational dynamics model used in this paper for the hex-rotor UAVs [22]:

$$m\ddot{x} = (\sin \phi \sin \psi + \cos \phi \cos \psi \sin \theta) T + F_D(V_x + V_{wx}(z), \theta, \rho(z)), \quad (8)$$

$$m\ddot{y} = (\cos \phi \sin \theta \sin \psi - \cos \psi \sin \phi) T + F_D(V_y + V_{wy}(z), \phi, \rho(z)), \quad (9)$$

$$m\ddot{z} = (\cos \theta \cos \phi) T - mg + F_D(V_z + V_{wz}(z), \phi, \rho(z)), \quad (10)$$

$$I_x \ddot{\phi} = (I_y - I_z) \dot{\theta} \dot{\psi} + l u_1 - j \dot{\theta} u_4, \quad (11)$$

$$I_y \ddot{\theta} = (I_z - I_x) \dot{\phi} \dot{\psi} + l u_2 - j \dot{\phi} u_4, \quad (12)$$

$$I_z \ddot{\psi} = (I_x - I_y) \dot{\phi} \dot{\theta} + u_3, \quad (13)$$

where $m = M + m_p$, which is sum of mass of the platform M and mass of the payload m_p , g is the gravitational acceleration and $\ddot{x}, \ddot{y}, \ddot{z}$ are the translational accelerations of the hex-rotor UAV in x -, y - and z -axes. T is the total thrust produced by all motors calculated as $T = k_b \sum_{N=1}^{N=6} \omega_N^2$, $u_1 = k_b \cos(30)(\omega_1^2 + \omega_2^2 - \omega_4^2 - \omega_5^2)$, $u_2 = k_b (\sin(30)\omega_1^2 + \omega_6^2 + \sin(30)\omega_5^2 - \sin(30)\omega_2^2 - \omega_3^2 - \sin(30)\omega_4^2)$ and $u_3 = k_t (\omega_3^2 + \omega_1^2 + \omega_5^2 - \omega_2^2 - \omega_4^2 - \omega_6^2)$, k_t and k_b are the torque and thrust coefficients, $u_4 = (\omega_1 - \omega_2 + \omega_3 - \omega_4 + \omega_5 - \omega_6)$. The rotational inertia of the UAV is expressed as (I_x, I_y, I_z) .

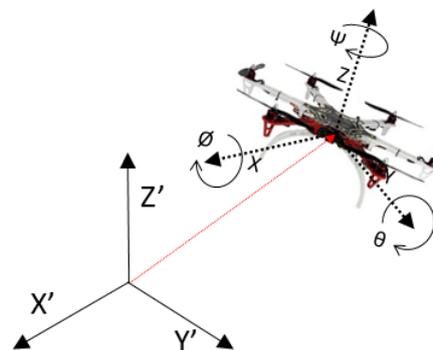


Figure 2. Multi-rotor schematic, with body frame and inertial frame.

The model detailed above is suitable for evaluating the optimization algorithm as shown in Section 3, but it cannot be used for cost function calculation because of the high computational costs. If we ignore complex maneuvers and only point A to point B transportation, we can then ignore the energy consumption variation during attitude changes. Therefore, only the Equations (8)–(10) describing the translational dynamics of the multi-rotor are used to develop a reduced model. The optimization algorithm developed considers the transportation from point A to point B without changing altitude or yaw. Keeping yaw angle fixed, and assuming a constant altitude allows the cost function calculations to be simple; further studies, however, will be done without these assumptions.

Therefore, in this case of no vertical motion, $\ddot{z} = 0$. The total thrust required to stay airborne would be $T = \frac{mg}{(\cos \theta \cos \phi)}$. If the yaw angle is fixed to zero and T from the above equation is substituted in equations of motion, the multi-rotor equations of motion can be reduced to Equations (14) and (15).

$$m\ddot{x} = m \cdot \tan \theta \cdot g + F_D(V_x + V_{wx}(z), \theta, \rho(z)), \quad (14)$$

$$m\ddot{y} = -\frac{m \cdot \tan \phi}{\cos \theta} \cdot g + F_D(V_y + V_{wy}(z), \phi, \rho(z)). \quad (15)$$

Equations (14) and (15) can be used to find the roll and pitch angles that can then be used to find the required thrust. The equations must be discretized in order to calculate the thrust required, which in turn is needed to calculate the power consumption as described in Section 2.2.2. A step by step description of power consumption calculation is shown in Algorithm 1

Algorithm 1: Calculation of average power consumption for a state transition

Result: Power consumption

Input : $Vx_{i+1}, Vx_i, V_{wx}, \Delta x, m,$

Output: E_s

$\theta_i = 0$

while While $(\theta_i - \theta) < 0.01$ **do**

$\theta = \theta_i$

 Calculate $F_D(V_x + V_{wx}, \theta)$

 Calculate θ_i from Equation (18)

end

Calculate $T = \frac{mg}{(\cos \theta_i \cos \phi)}$

Calculate P_{elec}

2.2.1. Discretized Model

Since the stage corresponds to the distance travelled by the agent, it is necessary to convert the translational dynamics equations from time domain to the distance domain. Chain rule can be used to perform such conversion. The chain rule $\frac{dx}{dt} = \frac{dx}{dt} \frac{dt}{dx}$ was applied to obtain the translational dynamics equation as follows. In order to reflect the change from time domain to distance domain, the index k in the cost function definition is replaced by index s in Equations (1)–(6):

$$mV_x(s) \frac{dV_x}{dx} = m \cdot \tan \theta_s \cdot g + F_D(V_x + V_{wx}, \theta), \quad (16)$$

$$mV_y(s) \frac{dV_y}{dy} = \frac{-m \cdot \tan \phi_s}{\cos \theta_s} \cdot g + F_D(V_y + V_{wy}, \phi). \quad (17)$$

The distance based translational dynamics model can be discretized using the forward or backward Euler method. The equations for the forward Euler approach are described as below:

$$\frac{Vx_{i+1} - Vx_i}{\Delta x} = \frac{1}{mVx_i} \cdot (m \cdot \tan \theta_s \cdot g + F_D(V_x + V_{wx}, \theta)), \quad (18)$$

$$\frac{Vy_{i+1} - Vy_i}{\Delta y} = \frac{1}{mVy_i} \cdot \left(\frac{-m \cdot \tan \phi_s}{\cos \theta_s} \cdot g + F_D(V_y + V_{wy}, \phi) \right). \quad (19)$$

2.2.2. Motor Energy Consumption

A thrust measurement stand is used to obtain the rotor speed, thrust and energy consumption plots for the DJI E310 propulsion system, which includes 2312 motors with the DJI 9450 rotors. All values obtained were compared with the values given by [23] for the same propulsion system. A power law based empirical equation between the rotor speed and the power consumption is obtained using curve fitting as shown in Equation (20):

$$P_N = 2 \times 10^{-8} \times \omega_N^{3.3659}, \quad (20)$$

where P_N is the power consumed by rotor N of the UAV, and ω_{iN} is the speed of the rotor in rad/s. The speed of the rotors is found using the relation $T_N = k_b \omega_{iN}^2$, where $k_b = 9.85 \times 10^{-6}$. The energy consumption during a particular step can be calculated by

$$E_s = \int P_{elec} dt, \quad (21)$$

which can be written in the discretized form as follows:

$$E_s = P_{elec} \Delta t, \quad (22)$$

where t is the time spent in that distance interval, $P_{elec} = \sum_1^N P_N$ and E_s is the energy consumed during that distance interval.

2.2.3. Calculation of Step Travel Time

The time spent for that particular distance step can be calculated using the following equation:

$$\Delta t = \int_0^{sf} \frac{1}{v} ds. \quad (23)$$

Equation (23) is continuous and we need to discretize this equation in order to evaluate the time consumed in that particular step:

$$\Delta t = \sum_{k=1}^N \frac{1}{v_k} \Delta s. \quad (24)$$

2.3. Parameter Selection

There are several parameters that have to be determined before executing the optimization algorithm. These parameters include: the horizon length; the distance interval; and the velocity interval. These parameters can influence the complexity of the algorithm and the computational time for one DP search space sweep. The complexity of the algorithm O as shown in Equation (25), and calculated and plotted in Figure 3a, depends on the number of distance intervals N_s between the start and goal position, and the number of velocity intervals N_v :

$$O(N_s N_v^2). \quad (25)$$

The complexity of the algorithm increases by increasing the number of velocity and altitude intervals. The selection of distance size and the velocity interval also depends on the hex-rotor's ability to reach that velocity in a particular distance step.

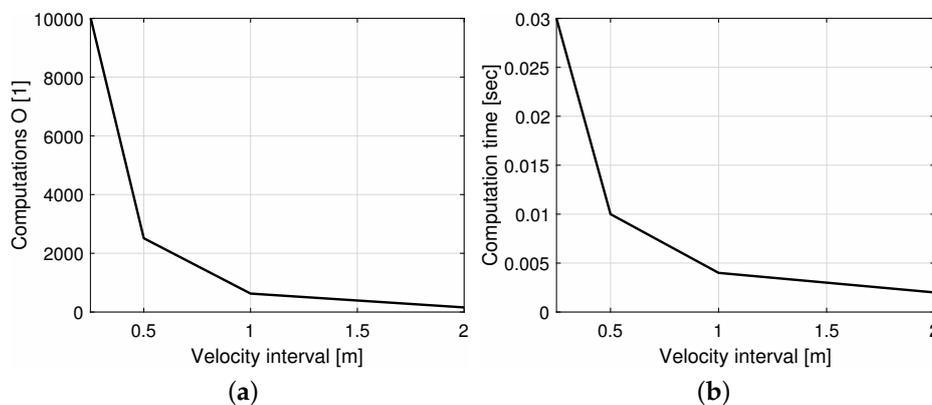


Figure 3. The effects of a decrease in velocity interval that results in an exponential increase in (a) number of computations O and (b) computation time.

Sample computation times and number of computations were plotted in Figure 3. The analysis in Figure 3 is based on distance interval of 10 m and velocity interval ranging from 0.25 m/s to 2 m/s. The computation is done using a 2.4 GHz computer with 8 GB RAM. An open-source autopilot hardware “pixhawk” is used with “PX4” firmware. The position controller in this system works at a rate of 100 Hz, which means it takes 0.01 s for one position control cycle. Currently, the computation time for one DP sweep for only the x -axis takes between 0.002 s to 0.03 s depending on the number of velocity intervals. If we consider matching the frequency of position controller of PX4 with this DP sweep, we can consider a 0.5 m/s velocity interval. Ideally, if the DP sweep is to replace the position controller, we should try to optimize the computation time so that it is equal to or below 100 Hz. This could be achieved with very small horizon lengths.

2.3.1. Distance Interval Selection

An ideal distance interval would have constant acceleration for the application of discretized translational dynamics equation. The change in velocity states is performed via PID velocity controller. Hence, the ideal case of constant acceleration can only be approximated to the response of velocity controller, if the distance interval is such that the time spent in this distance interval is equal to or greater than the rise time (90% of steady state) of the velocity response. In cases where the velocity change in the two states is such that, for that distance interval, it will take a time equal to the rise time to reach the desired velocity, then the assumption of constant acceleration can be applied. If the required velocity change in the two states will take less time than the time vehicle travels a particular distance step, it would result in the vehicle accelerating and then coasting in one distance step. This simply means that, in such a case, a constant acceleration assumption cannot be applied. In such a case, the transition cost must be calculated in two steps. The first step is to use the rise time to calculate the transition cost and the second step is to calculate the transition cost from rise time to the end of the distance interval. In other words, in this case, the distance interval is divided into two parts: the first part with an acceleration and the second part without any acceleration. In cases where the time required to travel the distance interval and reach the desired velocity is greater than the rise time, this desired velocity is then not feasible and should be discarded from the search-space. The distance interval should have a value high enough that for the minimum velocity intervals it has possible states to explore.

2.3.2. DP Sweep Trigger

Each time the UAV wants to travel a distance equal to the distance interval, the DP algorithm is activated. In order to achieve this, the following equation is used to convert the position vector into a periodic function $\zeta = \sin(P + \varrho)$, where P is the position (currently considering only x), and ϱ is used to select the distance interval value to trigger the DP algorithm. The values of ζ range from -1 to 1 , and whenever the value ζ crosses *zero*, the DP algorithm is triggered.

When the DP sweep is not triggered, the last set-point velocity is continuously being sent to the drone. This is important since, apart from sending the velocity commands for the transport direction, the algorithm is also continuously sending the velocity commands for vertical and lateral position control. The lowest allowable publish rate for position controller is 2 Hz since the firmware used in autopilot of software in loop simulations and hardware experiments triggers the fail-safe mode if the velocity commands are publishing below the threshold frequency.

2.3.3. DP Sweep Sample Plots

Two sample DP sweeps are plotted in Figure 4. These plots are developed using the DP code with the following specifications: the DP sweep consists of only one horizon length with 10 distance intervals. The velocity states ranged from 0 m/s to 14 m/s. In Figure 4, the weighted cost function is represented by colored lines with varying thickness. The cost function values were normalized and, based on the cost function value, the thickness of the line was given. Figure 4 shows the cost

function values distribution for all possible states. We can see that the cost function has a cumulative increasing effect since the first stages show thin blue lines and the last stages show thick and red lines. After doing this search space sweep, the DP algorithm traces back the the optimal path by starting from the last transition costs and selects the minimum cost all the way back to the first node.

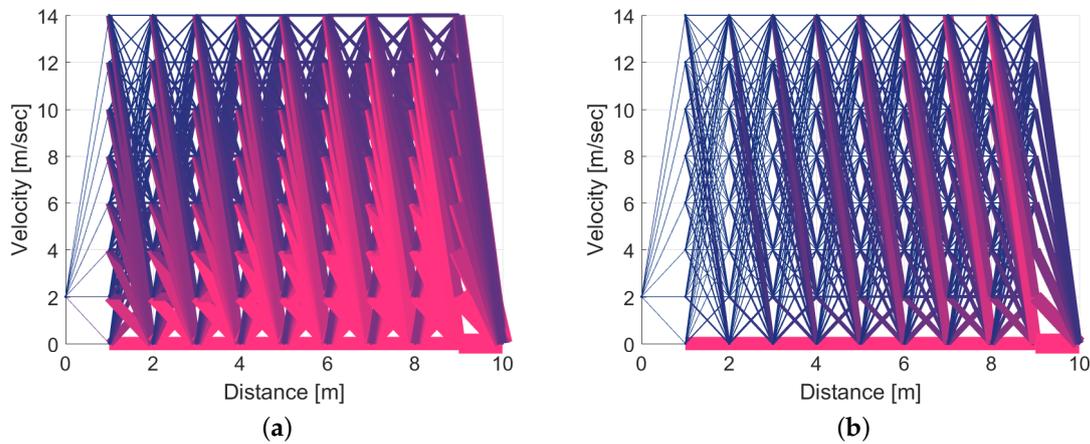


Figure 4. A Dynamic programming (DP) sweep with velocity interval 2 m/s and horizon length 10 m, initial velocity 2 m/s and final velocity 0 m/s. Width of the lines and intensity in redness shows a high value of combined cost function (a) $\lambda = 0.5$; (b) $\lambda = 0.9$.

3. Numerical Simulation Experiments

The algorithm was tested on a complete hex-rotor model developed in MATLAB Simulink. The purpose of developing a simulation tool is two-fold: first to asses the proposed algorithm, secondly, the simulation tool can also be used to asses the mission completion feasibility prior to the actual flight or software in loop simulations. It also assists in deciding the parameters of the algorithm, which otherwise would be very difficult to do with the real flight. The Simulink model consisted of a hex-rotor model, control block and optimization algorithm block as shown in Figure 5. The hex-rotor model uses Equations (8)–(13) as described in Section 2.2 previously in detail. The optimization block consists of the real-time dynamic programming optimization algorithm.

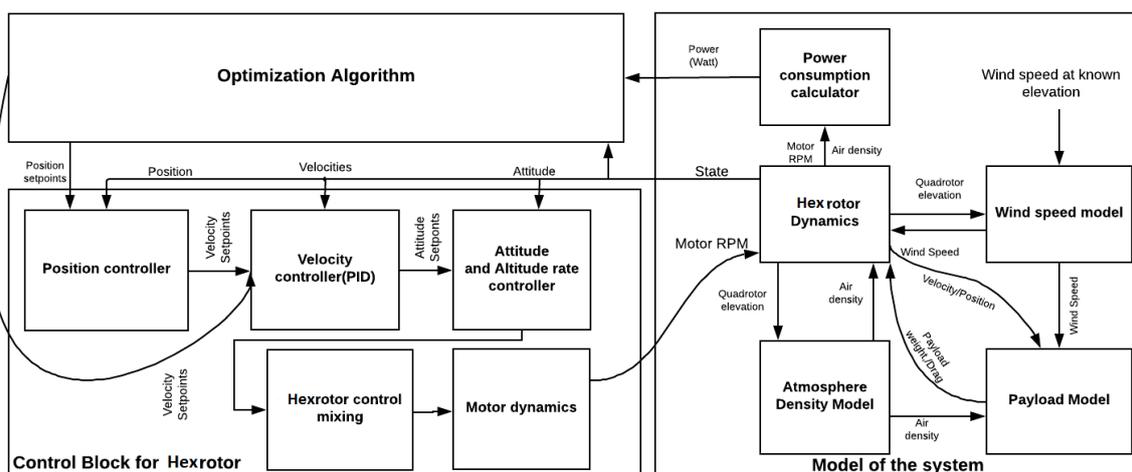


Figure 5. Description of the complete system for simulation tests of the algorithm, including the controller block, the hex-rotor block, and the optimization algorithm block.

Several factors affect the energy consumption during an aerial transport using a UAV. Some of these factors include the drag caused by wind speed and the density of air. Wind speed and air density are known to vary with elevation. Air density decreases at higher altitudes and wind speed increases

with elevation. Several blocks were added to the hex-rotor model, which include a power consumption estimation block, atmospheric density calculation block, wind speed estimation block, and the payload model block. The interactions of all these blocks and information inflow and outflows from these blocks are explained in Figure 5. The controller block, atmospheric density model, wind speed model, and payload model are described further below in Sections 3.1–3.4, respectively. The inertial parameters of the DJI-F550 UAVs were found by using the developed CAD model and a CAD software was used to find the inertial parameters. The mass of the UAV was calculated after taking measurements from a weight scale. The altitude, attitude and position controllers were added to the Simulink model and tuned. All parameters used in the simulation are presented in Table 1. The controller gains mentioned in Table 1 were obtained by tuning the system manually.

Table 1. List of all the parameters for DJI F-550 drone model.

Parameter	Component	Value	Parameter	Component	Value
K_{pvx}	Controller	0.15	l_{min}	UAV	0.88m
K_{ivx}	Controller	0.001	K_{ppx}	Controller	0.4
K_{dvx}	Controller	0.04	K_{ppy}	Controller	0.4
K_{pvy}	Controller	0.15	Max Velocity	UAV	12 m/s
K_{ivy}	Controller	0.001	Max Altitude	UAV	2.2 m
K_{dvy}	Controller	0.04	Min Altitude	UAV	1 m
$K_{p\dot{\theta}}$	Controller	20	UAV mass	UAV	3.4 Kg
$K_{i\dot{\theta}}$	Controller	0	Payload mass	Payload	0.586 Kg
$K_{d\dot{\theta}}$	Controller	8	Arm length	UAV	0.27 m
$K_{p\phi}$	Controller	20	Iz	UAV	0.05
$K_{i\phi}$	Controller	0	Ix	UAV	0.037
$K_{d\phi}$	Controller	8	Iy	UAV	0.037

3.1. Controller Block

The controller block consists of a position controller, which generates velocity set-points and a velocity controller, which generates the attitude set-points, the attitude controller, generates the required motor speeds.

3.1.1. Position Controller

The position controller is simply a proportional controller, which generates the velocity set-points to maintain the desired position:

$$V_{sp} = e_p K_{pp},$$

where K_{pp} is the proportional gain for position controller and

$$e_p = P_{des} - P_{current}.$$

3.1.2. Velocity Controller

The velocity controller is a PID controller, which generates the attitude set-points:

$$\theta_{sp} = e_v K_{pv} + K_{iv} \int e_v + K_D \frac{\partial V}{\partial t},$$

where K_{pv} is the proportional gain for velocity controller, and K_{iv} is the integral gain for velocity controller, K_{dv} is the derivative gain for velocity controller, and

$$e_v = V_{des} - V_{current}.$$

3.2. Atmospheric Density Model

The atmospheric density model is based on a US standard atmosphere [24]. It is implemented in this study using an international standard atmosphere block, which takes the altitude as an input and provides the density as an output.

3.3. Wind Speed Model

Wind speed is an important factor which can affect the power consumption of a multi-rotor during the flight. Off-line energy assessment would require the measurements of wind speed. Wind speed measurements are usually available online from the meteorological department or can be obtained via an on-board wind speed sensor, and it is measured at a certain altitude. However, by using a power law, wind speed can be estimated at the required altitude, provided that we have measurements of wind speed at the near ground altitude. The power law is explained as Equation (26) and is obtained from [25]:

$$V_w = v_g \cdot \left(\frac{z}{z_g} \right)^\alpha, \quad (26)$$

where V_w is the estimated wind velocity at the elevation z , and v_g is the wind velocity at near ground elevation z_g . α is called the Hellman exponent, whose values can be found using Table 2.

Table 2. Hellman exponents for different locations

Location	Hellman Exponent
Unstable air above open water surface:	0.08
Neutral air above open water surface:	0.10
Unstable air above flat open coast:	0.11
Neutral air above flat open coast:	0.18
Stable air above open water surface:	0.27
Unstable air above human inhabited areas:	0.27
Neutral air above human inhabited areas:	0.30
Stable air above flat open coast:	0.40
Stable air above human inhabited areas:	0.60

3.4. Payload Model

Payload is currently considered as a block with a surface area A_p , and hence a drag co-efficient C_D , and a mass m_p . The drag force acting on the payload is calculated by using Equation (27):

$$F_{Dp} = \frac{1}{2} \cdot C_{Dp} \cdot \rho_a \cdot A_p (V_v + V_w)^2, \quad (27)$$

where F_{Dp} is the vector representing the drag forces acting on the payload, C_{Dp} is the co-efficient of drag for payload, A_p is the payload surface area exposed, V_v is the velocity of the hex-rotor, V_w is the wind speed and ρ_a is the density of air at an altitude a .

3.5. Thrust Irregularity in Forward Flight

The thrust is generally modeled as $T = k_b \omega^2$, which is called the thrust model at hover. This model, however, does not accurately predict the thrust and motor speed relation during high speed forward flights as it is observed in [26,27]. It was also concluded in [28] that, for the same rotor speed, the thrust decreases greatly at higher angles of attack and higher speeds. Another experimental study by [29] showed that the total lift produced by various multi-rotors decreases by decreasing pitch angles from 0° to -40° when subject to 6 m/s wind speed. In another study [30], a controller is proposed to be able to compensate for thrust irregularity during forward flights. This thrust deficit in high speed forward flight means that the propellers have to rotate faster than usual to produce the thrust required.

This simply means that more energy is consumed during forward flight to maintain the same thrust. In [30], this thrust irregularity is modeled using Equations (28)–(30).

The induced air velocity due to rotation of rotors is shown in Equation (28)

$$v_h = \sqrt{\frac{T_h}{2\rho A}}, \quad (28)$$

where v_h is the air velocity caused by the rotation of propellers in hovering conditions, T_h is the thrust in hovering condition, ρ is air density and A is the propeller sweep area:

$$\frac{T}{T_h} = \frac{v_h}{v_i - v_\alpha \sin \beta}, \quad (29)$$

where T is the actual thrust produced, v_i is the induced air velocity from the propellers caused by forward flight. v_α is the upstream air velocity and β is the attitude angle. The induced velocity v_i of the air from propellers can be calculated using the following equation as [31]

$$v_i = \frac{v_h^2}{\sqrt{(v_\alpha \cos \beta)^2 + (v_i - v_\beta \sin \beta)^2}}. \quad (30)$$

The above formulation was applied on results from [29] and was found to be matching with experimental results.

The effects of forward velocity and the pitch angle can be further seen via plotting the ratio $\frac{T}{T_h}$ from Equation (29) in Figure 6. Forward velocities v_α ranging from 0–12 m/s and pitch angles β ranging from 0–40° were plugged in Equation (30) along with the hover air velocity v_h as calculated from Equation (28) to calculate induced velocity v_i . The induced velocity v_i , pitch angle β , forward velocity v_α and air velocity in hover v_h were plugged in Equation (29) to obtain the ratio $\frac{T}{T_h}$ for each pitch angle value and forward velocity and plotted in Figure 6. At very low pitch angles and higher velocities, the ratio $\frac{T}{T_h}$ remains above *one*. However, at higher pitch angles and higher velocities, the ratio stays below *one*. This simply shows that, for the same power, the thrust produced in the hovering condition will be higher than the thrust produced in forward velocity. The energy consumption calculation in the numerical model accounts for this thrust deficit in high speed forward flight using the method described in Figure 7.

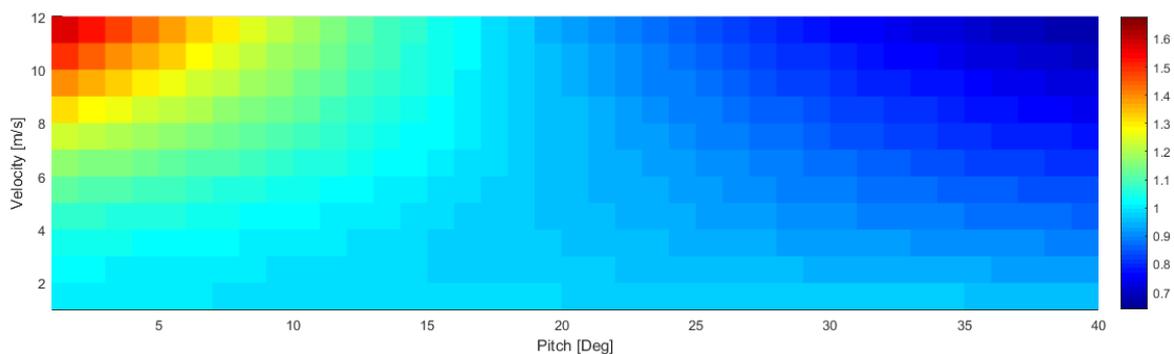


Figure 6. The variation of $\frac{T}{T_h}$ with respect to the pitch angle and the forward velocity is shown using the color-bar.

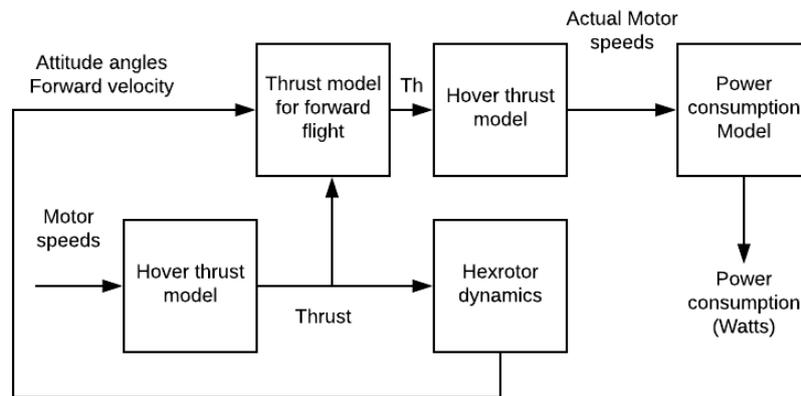


Figure 7. Corrected power consumption calculation based on thrust irregularity.

3.6. Assumptions

The attitude dynamics of the UAV is decoupled with the payload, therefore any drag force acting on the payload will not effect the attitude angle of the UAV. This assumption is supported by using a multi degree of freedom based free joints arm in software in the loop simulations. In hardware experiments, the payload attachment point can be a ball joint or a small cable; this will decouple the attitude dynamics of the multi-rotor with payload. We do not model the thrust irregularity happening at higher pitch angles and higher forward velocities in the dynamic model. Therefore, a thrust irregularity compensator is not required as it is suggested by [30,32]. The energy consumption calculation is, however, performed while considering the thrust irregularity in forward flight as discussed in Section 3.5. We also assume that the attitude, altitude and velocity controller of the UAV is robust enough to deal with extreme conditions (high velocities, high attitude angles). The velocity controller is not a model based controller as it is shown in [32], therefore drag force compensation during high speed forward flight is not required. The RTDP algorithm receives the current state of the UAV and provides a velocity decision. This velocity decision can be implemented using the controller presented in this paper or more calibrated ones such as [33]. We assume that the UAV is capable of lifting a cube with a frontal area of 0.12 m^2 . Energy losses during fast maneuvers are not considered. These losses include the braking of the motors to change the motor speeds to achieve differential thrust required for attitude change. This assumption can be supported by the fact that frequent attitude angle changes are not happening between point A to point B during the transport. Side wind is neglected in this analysis. Based on the lateral frame drag, it is necessary for the multi-rotor to balance the drag force to be able to maintain the lateral position. This drag can also influence the energy consumption in addition to the drag caused due to forward flight. A sudden change in air density is out of the scope of the current paper. The sudden change in air density can occur if there is a dust devil vortex or a thermal plume in the path. The horizontal length of such thermal plumes is considerably less than the journey length of our simulations. In real applications, the air density should be updated based on the elevation of the UAV.

3.7. Numerical Simulation Results

A numerical simulation test was conducted where the UAV was carrying a cubic payload of 0.12 m^2 surface area and negligible weight from $[0, 0, 10 \text{ m}]$ to $[250 \text{ m}, 0, 10]$. The initial velocity at point of origin $[0, 0, 10 \text{ m}]$ and at goal position $[250 \text{ m}, 0, 10]$ were 0 m/s . The velocity interval of 0.1 m/s was considered. Distance intervals of 1, 2, 3, 4 m were considered, whereas the horizon considered consisted of 10 distance interval steps. The DP sweep was triggered at every distance interval step. Three test cases with values of $\lambda = (0.2, 0.5, 0.7)$ were performed. The results of the test cases are shown in Tables 3–6. These results show the percentage of energy savings when the λ is increased from 0.2 to 0.7. Table 5 shows that the journey time decreases when we decrease the λ

from 0.7 to 0.2. Another test was performed with an increment of velocity interval of 0.5 m/s and the difference in energy saving was found to be less than 1%. Extended simulations were performed with distance interval ranging from 1 m–4 m for a velocity interval of 0.1 m/s. The results of the extended simulations are tabulated in Tables 3, 4 and 6.

Table 3. Numerical simulation results with distance interval 1 m.

Weight-age (λ)	Time (S)	Difference (%)	Energy (kJ)	Difference (%)	Velocity Interval (m/s)
0.2	29.73	−15.64%	28.2	+18.01%	0.1
0.5	34.39	0	23.12	0	0.1
0.7	40.446	+17.6%	22.39	−3.15%	0.1

Table 4. Numerical simulation results with distance interval 2 m.

Weight-Age (λ)	Time (S)	Difference (%)	Energy (kJ)	Difference (%)	Velocity Interval (m/s)
0.2	31.19	−6.7%	29.3	+11.12%	0.1
0.5	33.46	0	26.04	0	0.1
0.7	38.58	+15.3%	23.16	−11.05%	0.1

Table 5. Numerical simulation results with distance interval 3.12 m.

Weight-age (λ)	Time (S)	Difference (%)	Energy (kJ)	Difference (%)	Velocity Interval (m/s)
0.2	29	−14.48%	27.6	+11.44%	0.1
0.5	33.2	0	24.44	0	0.1
0.7	38.5	+15.96%	22.89	−6.34%	0.1
0.2	29	−15.37%	27.6	+10.9%	0.2
0.5	33.46	0	24.59	0	0.2
0.7	38.5	+15.06%	22.83	−7.15%	0.2
0.2	29.93	−9.75%	28.01	+12.85%	0.3
0.5	32.85	0	24.41	0	0.3
0.7	38.5	+17.19%	22.83	−6.47%	0.3
0.2	29.93	−9.62%	28.01	+12.7%	0.4
0.5	32.81	0	24.45	0	0.4
0.7	38.5	+17.34%	22.83	−6.62%	0.4
0.2	29.96	−11.64%	28.03	+11.48%	0.5
0.5	33.45	0	24.81	0	0.5
0.7	38.08	+13.84%	22.76	−8.26%	0.5

Table 6. Numerical simulation results with distance interval 4 m.

Weight-age (λ)	Time (S)	Difference (%)	Energy (kJ)	Difference (%)	Velocity Interval (m/s)
0.2	30.89	−2.39%	28.14	+5.044%	0.1
0.5	31.63	0	26.72	0	0.1
0.7	36.5	+15.39%	23.63	−11.56%	0.1

The results of the numerical simulation are also presented in Figure 8. These results include the transport trajectory, the velocity profiles, the pitch angles, the opposing drag forces, and the energy consumption profiles. The plots include the numerical simulation results for a value of $\lambda = (0.2, 0.5, 0.7)$. The velocity profile for $\lambda = 0.2$ shows the highest value, which results in more drag forces and thus requires higher pitch angles to maintain the velocity. The higher pitch angles and higher forward velocities supplement the increase of thrust deficit, which means that more power is consumed. Therefore, although $\lambda = 0.2$ results in the fastest transportation, it results in more power

consumption. However, for $\lambda = 0.7$, the velocity is lower, which results in lower drag forces, which in turns results in a lower pitch angle. The lower pitch angle and smaller forward velocities result in a reduction of thrust deficit, hence the total energy consumption during this transportation is low although the mission took more time to complete.

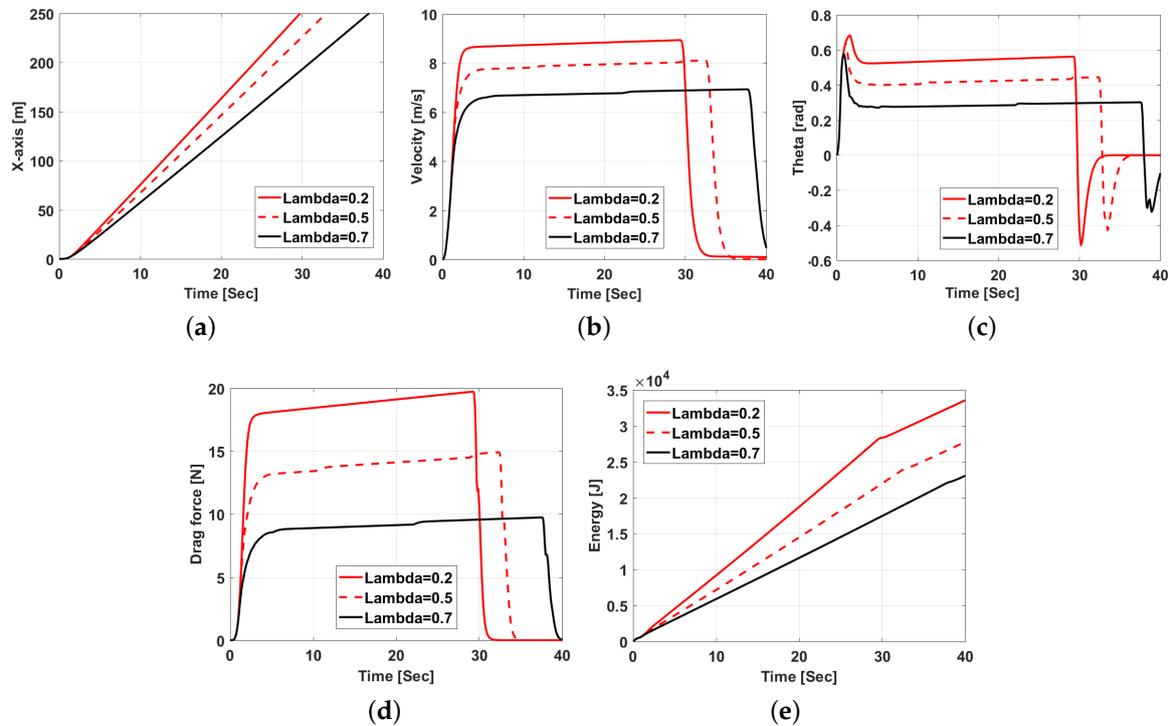


Figure 8. Flight plots for numerical simulations performed for $\lambda = (0.2, 0.5, 0.7)$. (a) trajectory of the UAV; (b) velocity of the UAV; (c) pitch angle of the UAV; (d) opposing drag force of the UAV; (e) energy consumption of the UAV.

4. Software in the Loop Simulations

Examples of Software in the loop (SITL) simulations in literature are [34–37]. More accurate simulation frameworks are also being developed [38] for multi-UAV simulations. Software in the loop simulations (SITL) are performed to verify the functionality of the code before performing the hardware experiments. In general, our SITL setup can be explained by the following Figure 9.

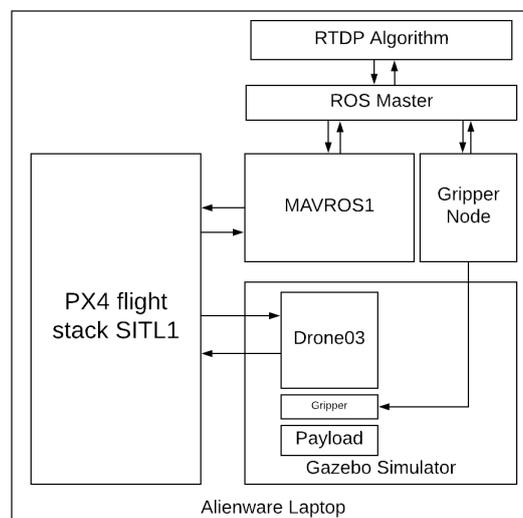


Figure 9. Software in the loop simulation architecture.

Using the Table 1, the DJI F-550 drone was modeled in Gazebo as shown in Figure 10, and the same PX4 flight stack was simulated as it is used in the real autopilot. The ROS, MAVROS package, PX4 flight stack simulator, Gazebo simulator and the RTDP algorithm were running on the same Linux computer. The computer has a 2.4 GHz processor and 32 GB RAM. All communication between MAVROS, Gazebo and gripper node were performed via the local ROS MASTER.

The RTDP algorithm was communicating with the flight stack through ROS Master using the MAVROS package. The flight stack was communicating with the DJI F550 drone in the Gazebo simulator. It is a widely accepted practice to test new algorithms in SITL simulations before testing them on real hardware. The autopilot firmware is developed by hundreds of researchers and it is available online with the source code. The RTDP algorithm developed can use MAVROS via the robotics toolbox of MATLAB to be able to communicate with the autopilot. A multi-link arm was fixed at the base of the DJI-F550 model as shown in Figure 10. All the joints in the multi-link arm were free and were mimicking the behavior of the cable. A gripper ROS node was running which can create and delete links between the payload and the gripper attached at the end of the multi-link arm.

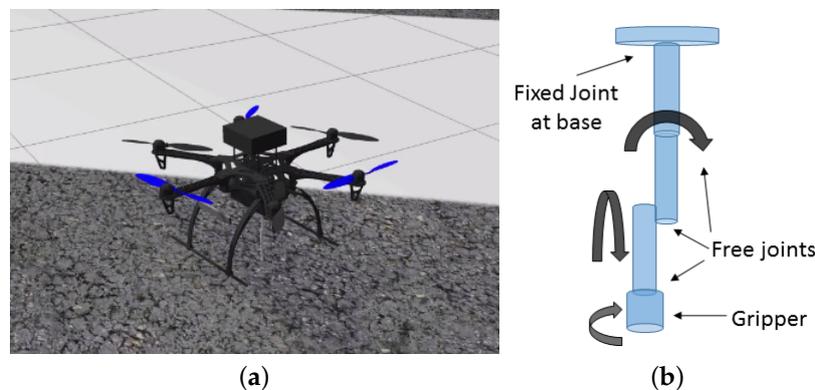


Figure 10. Software in the loop Gazebo DJI F550 model along with the description of the manipulator; (a) DJI-F550 model in Gazebo simulator with PX4 flight stack simulation; (b) the multi-link free joints based arm.

SITL Experiments

Similar parameters were used in the SITL simulation experiments as discussed in Section 3.7. RTDP algorithm was developed in Simulink, which was running on the same laptop where the Gazebo simulator was running. A velocity interval of 0.1 m/s was used. Similar to the autopilot, the simulated flight stack also records the flight data including the actuator signals, attitude, velocity and position estimates. The autopilot logs were used to plot the performance plots. The publish rates of the topics during the aerial transport in SITL simulations were observed using an ROSTOPIC tool and were plotted in Figure 11. A delay in publishing of the topic can trigger the fail-safe activation for off-board failure. The frequency of publishing is very high when the DP sweep is not triggered. When the DP sweep is triggered, we expect the publishing frequency to go down. The lowest publishing frequency is 4.8 Hz. The results of the simulations are plotted in Figure 12. The numerical simulation results and software in the loop simulation results are similar. In both cases, at $\lambda = 0.2$, the DP sweeps resulted in high velocities, which created higher parasitic drag. This resulted in higher pitch angles, which lead to high power consumption. The drone behavior was similar for numerical simulations and SITL simulations for values of $\lambda = 0.5, 0.7$. This shows that the algorithm developed is compatible with the autopilots' firmware and it satisfies the constraints imposed by the parameters of the firmware. These parameters include the multi-rotor constraints such as max velocity, max altitude, fail-safe mode activation conditions, etc. It also proves the applicability and usefulness of the algorithm.

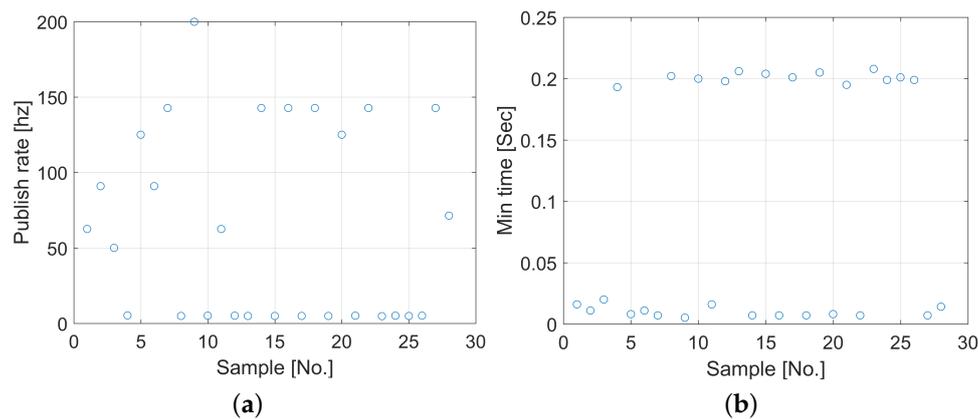


Figure 11. Publish rates and time as obtained by the ROSTOPIC tool with a sample of two consecutive messages. (a) publish rate of the velocity command topic; (b) minimum and maximum time between two commands.

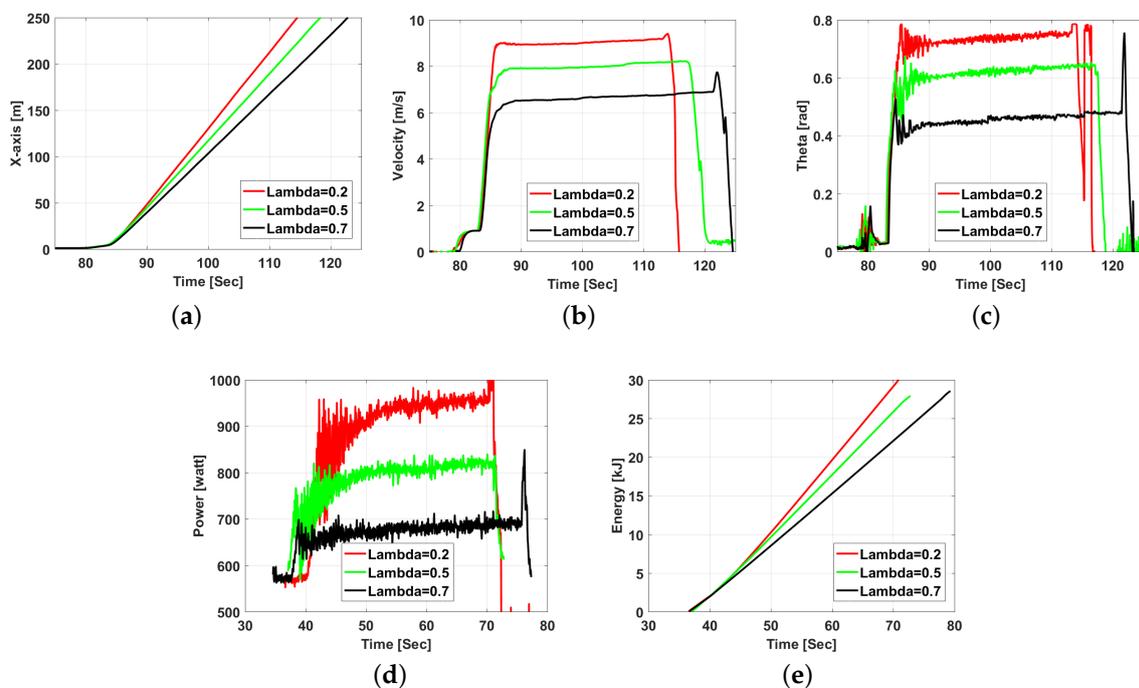


Figure 12. Flight plots for Gazebo SITL simulations performed for $\lambda = (0.2, 0.5, 0.7)$. (a) trajectory of the UAV; (b) velocity of the UAV; (c) pitch angle of the UAV; (d) power consumption profile; (e) energy consumption of the UAV.

5. Real-Time Flight Experiments

The system hardware as shown in Figure 13 consists of an external computer, and a DJI-F550 with Pixhawk autopilot with PX4 firmware for low level control. DJI-F550 also had an on-board companion computer, which was connected to the autopilot via MAVLINK. A cable-based gripper inspired by Gough–Stewart platform was developed and attached to the Hex-rotors. The gripper is a payload attachment mechanism that allows for automation of attachment and release of the payload.

The electro-permanent magnet was activated and deactivated by using an Arduino Nano connected to the companion computer via USB link. The gripper can be activated and deactivated during autonomous operation, using an ROS node. The electro-permanent magnet required power for activation and deactivation only; therefore, it was powered by the same battery that was powering the

drone. A current and voltage sensor called Elogger V4 is connected in series between the batteries and the ESCs of the UAVs while bypassing the power supply of the onboard computer as shown in Figure 13. Elogger was also used by [39,40] for logging the current and voltages of a UAV during the flight. The cable material was chosen to withstand the weight of the payload requirements. The software for the experiment is based on ROS interface MAVROS. An Optitrack system is used for position estimation. The algorithm is written in MATLAB Simulink and it communicates to the drone via MAVROS through ROS MASTER. Further details of experimental procedure are provided in “Supplementary Materials”. The videos of all experiments are available at [41].

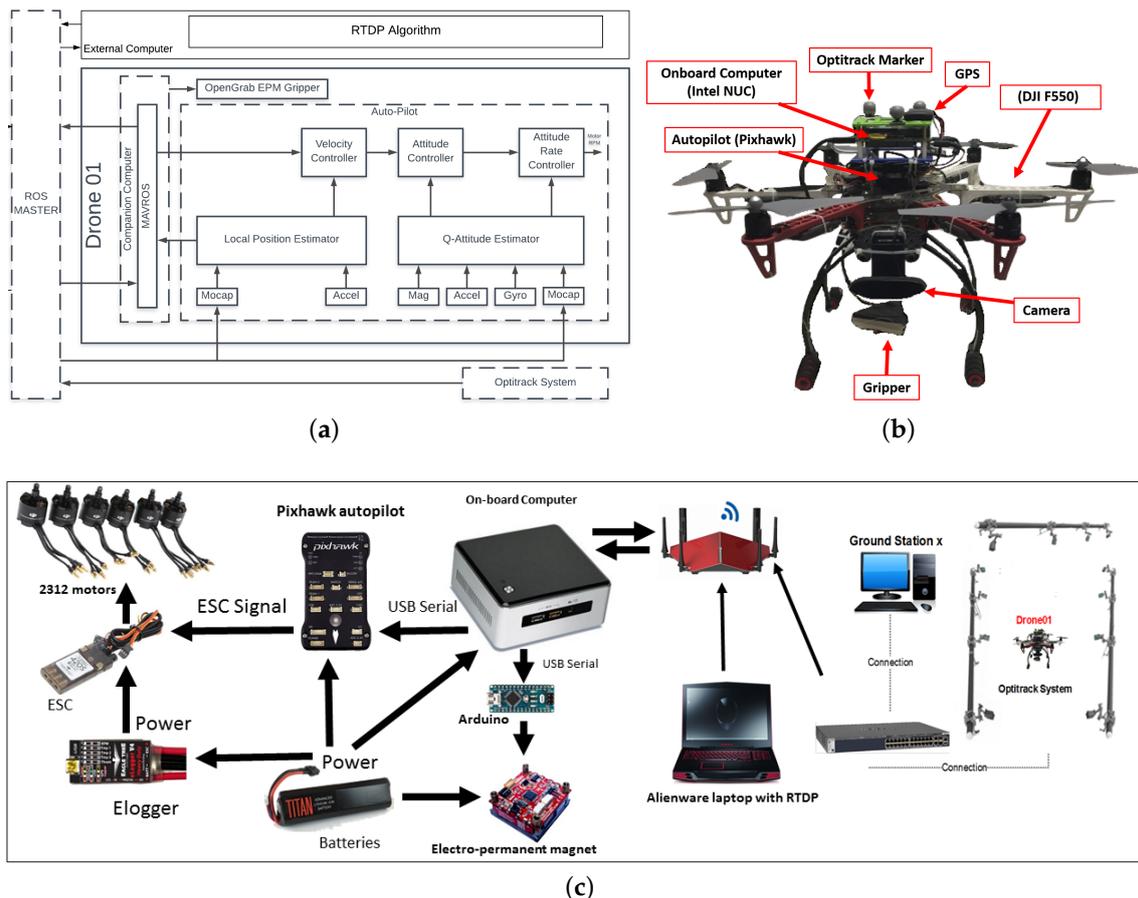


Figure 13. A description of the hardware used in the testing of the RTDP algorithm. (a) software architecture of experimental setup; (b) DJI-F550 drone used in the experiments; (c) power source and route for various components of the drone.

5.1. Baseline Experiment

An initial baseline experiment was performed, where the Drone01 was kept in hover at fixed altitude set-point. There was no payload attached to the Drone01. The drone01 is the same drone that was modeled in SITL and Simulink simulations. The power consumption was recorded and compared with the SITL and Simulink model hover power consumption. The results are plotted in Figure 14. It can be seen in Figure 14 that power consumption in SITL and Simulink models at hovering condition matches with the hardware experiments. However, there are minor discrepancies during the takeoff, which can be ignored for this study since the optimization is performed for transportation from point A to point B.

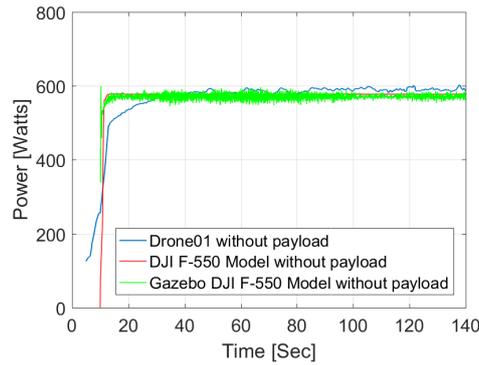


Figure 14. Hover power consumption in real hardware and the SITL and Simulink experiments.

5.2. Lab Scale RTDP Transportation

An experiment was performed to test the feasibility of transportation from point *A* to point *B* using RTDP experiments. A distance interval of 0.1 m was considered, with 10 distance intervals in one horizon. The starting point was [0, 0, 0.8], whereas the goal position was [0, 6, 0.8]. A velocity interval of 0.1 m/s was considered with a velocity range of 0–12 m/s. The time-energy weight-age factor of $\lambda = 0.7$ was chosen. The drone01 took off manually while it was transported to point *A*. At point *A*, the RTDP algorithm was triggered and, when the drone reached point *B*, the RTDP algorithm was switched off and drone was manually landed. The trajectory of the transportation and the velocity profiles of the numerical simulation and the experiments are plotted in Figure 15.

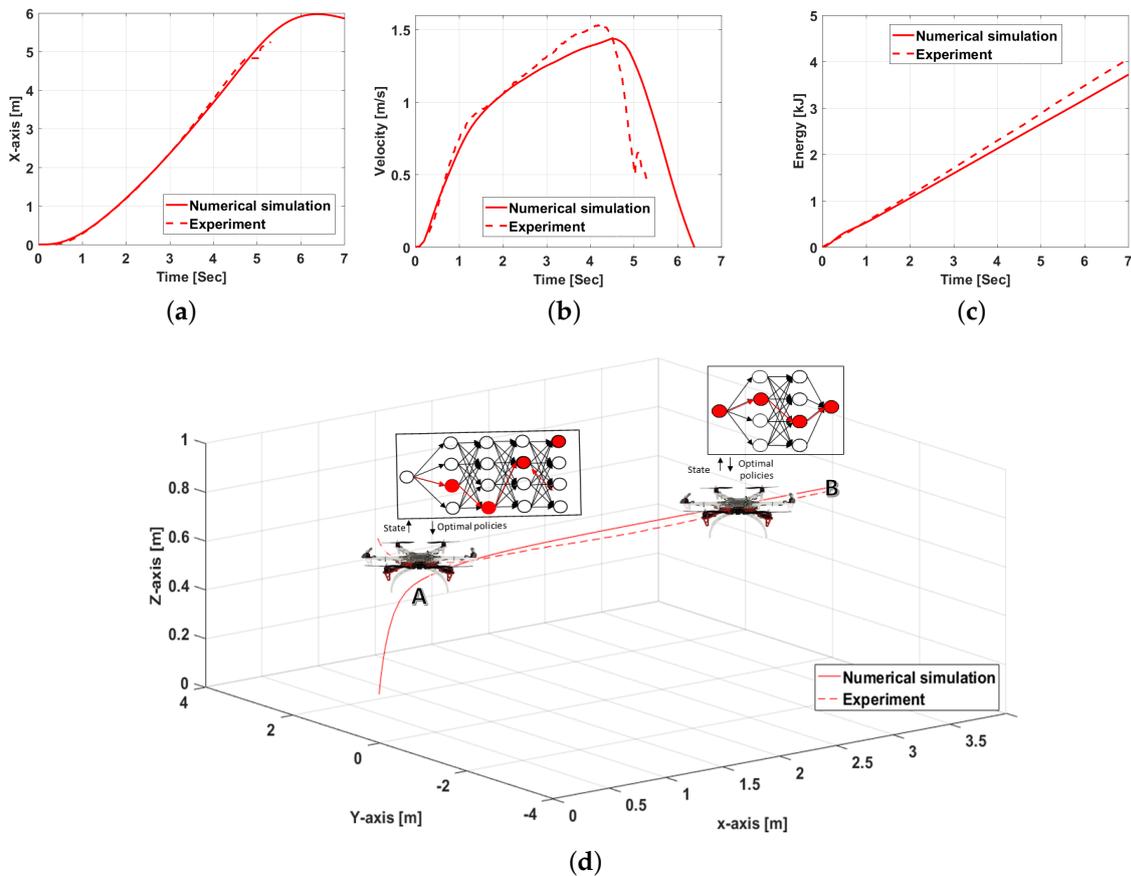


Figure 15. Lab scale RTDP based aerial transportation. (a) drone01 trajectory in simulation and experiment; (b) drone01 velocity profiles in experiments and simulations; (c) drone01 energy profiles in experiments and simulations; (d) drone01 trajectory in simulation and experiments also showing DP sweep with terminal costs in first horizons and DP sweep without terminal cost in the last horizon.

The experiment showed that the drone01 moved from point *A* to point *B* successfully. The numerical simulation results and experimental results are in close agreement. This proves the compatibility and experimental applicability of the algorithm.

After comparing the simulation and experiments for $\lambda = 0.7$, two more experiments were performed to compare $\lambda = 0.7$ with $\lambda = 0.5, 0.2$. The horizon length, distance interval, start point and goal point were kept the same and fixed. The results of the experiments are plotted in Figure 16 showing the comparison of velocity profiles and trajectories between $\lambda = 0.7, 0.5, 0.2$. As expected, the drone01 reaches the destination in the order $\lambda = 0.2, 0.5, 0.7$, which is also reflected in the velocity profiles. This test was performed in the lab environment where, due to space constraints and safety limitations, high speed flights were not achieved. However, these tests confirmed that the algorithm is capable of transporting the drone01 from point *A* to point *B* successfully, while manipulating the journey time. Minimization of journey time is useful in energy savings when the opposing drag forces are not significant, which is the case in these lab experiments.

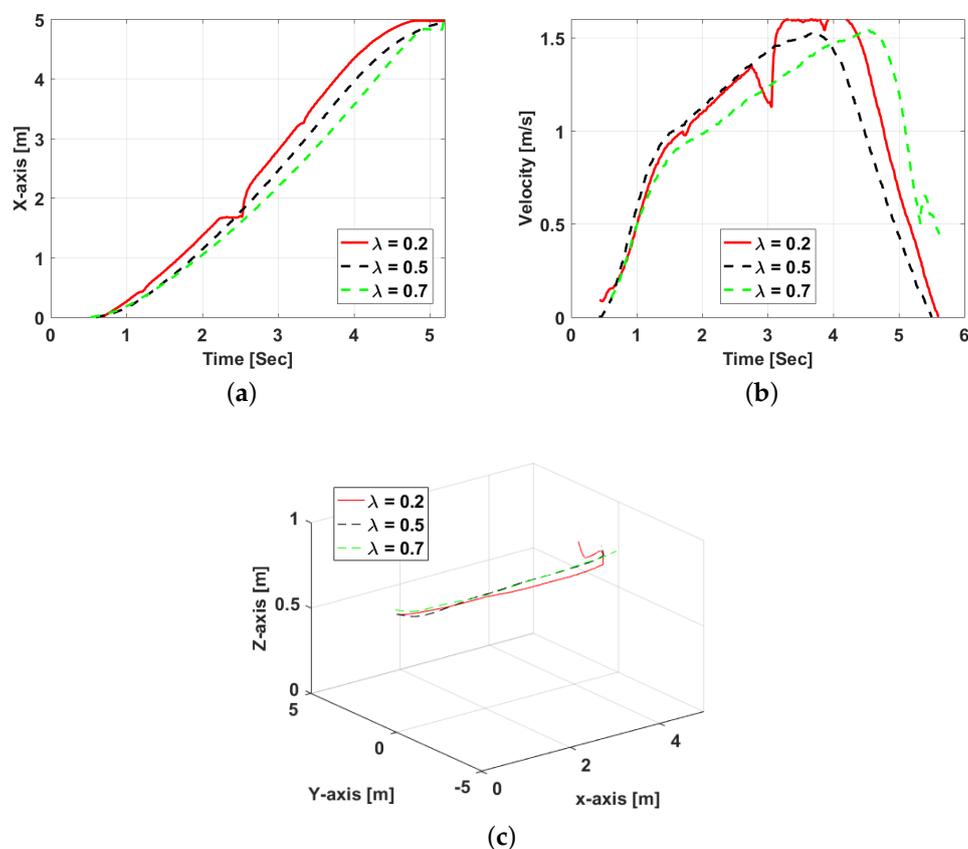


Figure 16. Lab scale RTDP based aerial transportation. (a) drone01 trajectory in experiments for $\lambda = 0.2, 0.5, 0.7$; (b) drone01 velocity profiles in experiments for $\lambda = 0.2, 0.5, 0.7$; (c) drone01 trajectory in experiments for $\lambda = 0.2, 0.5, 0.7$.

5.3. Variable Goal Position Experiment

The real-time DP algorithm's effectiveness in changing the velocity profile based on goal position change is tested in the lab environment. The results of the simulation and experiments for the variable goal position are plotted in Figure 17. In this simulation and experimental test, the goal position is initially provided as 2.5 m, whereas, after 3.5 s, the goal location is changed to 5 m. This is reflected in the velocity profile and the position-time profile. The *x*-trajectory of the drone shows a change at 3.5 s. The velocity profile also shows a sudden change between 3 and 4 s. It shows that the velocity profile

was first selected for a goal position at 3 m, but then, because of the goal position change, the velocity profile was also changed. The experiments and the numerical simulations are in good agreement.

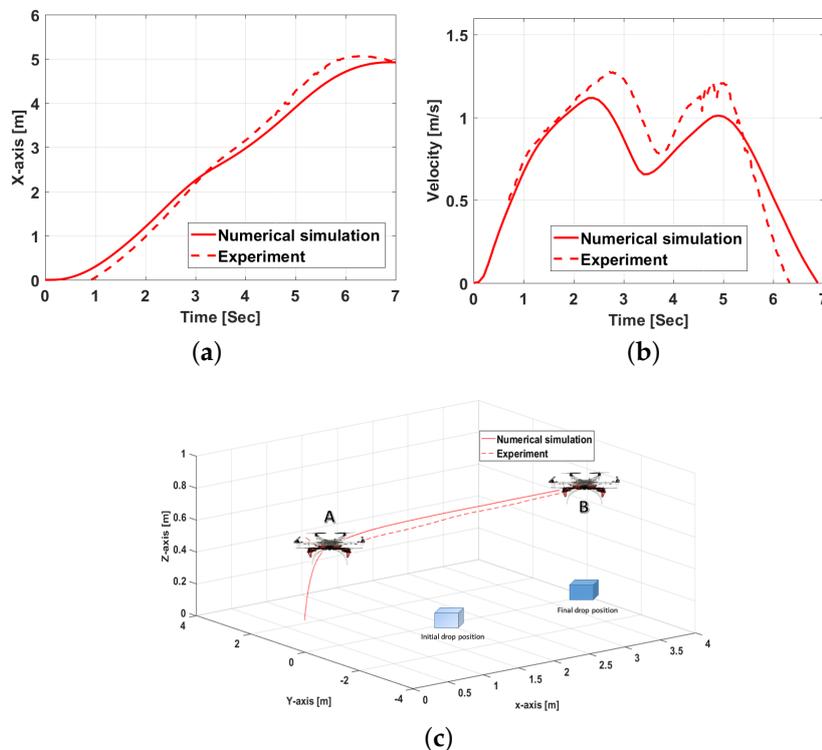


Figure 17. Lab scale RTDP based aerial transportation while the goal position changed at 3.5 s. (a) drone01 trajectory in simulation and experiment; (b) drone01 velocity profiles in experiment and simulations; (c) drone01 trajectory during experiment and numerical simulation, initial and final drop location are shown.

6. Conclusions

In this research study, a real-time dynamic programming algorithm is proposed for achieving energy efficient motion for aerial UAV transportation. The proposed algorithm is based on minimization of cost function. The cost function is the weighted sum of the energy cost and journey time cost. A factor λ is used to assign weight-age to favor either the time of the journey or energy consumption. After the description of the algorithm, a numerical model was developed and explained, which is used for the testing of the algorithm. After testing in the numerical model, the algorithm was also tested in Software-in-the-loop *Gazebo* simulation environment. The lab scale experiments were also conducted to validate the simulation results. After that, low speed flight tests were conducted for $\lambda = 0.2, 0.5, 0.7$, which showed that, by changing λ to a lower value, we can reduce the journey time for low speed flights. The real-time decision-making of the algorithm was tested by changing the goal position mid-flight in another experiment. The algorithm is capable of altering the journey time or energy spent based on the provided value of λ . Due to the definition of λ , it was found that increasing the value of λ decreases the energy consumption, while the journey time increases. The future works include expansion of cost function to include the 3D motion. The cost function will be modified to remove current assumptions such as assumption of constant altitude. After that, we plan to expand the current work to dual UAV payload transportation to test the scalability of the developed algorithms.

Supplementary Materials: Further description of experimental procedure is available online at <http://www.mdpi.com/1996-1073/12/16/3049/s1>.

Author Contributions: T.T., D.G. and Y.Z. supervised the design of experiments and algorithm development; A.M. developed the algorithm, developed the numerical and SITL model and performed the experiments; T.T. provided assistance with lab experiments and SITL simulations; Y.Z. supervised the Simulink based numerical model; A.M. wrote the paper, while T.T., D.G. and Y.Z. revised the manuscript.

Funding: This publication is based upon work supported by the Khalifa University of Science and Technology under Award No. RC1-2018-KUCARS.

Acknowledgments: The authors would like to give special thanks to the lab staff for the support.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CoG	Center of gravity
UAV	Unmanned aerial vehicles
SITL	Software in the loop
RTDP	Real-time dynamic programming
EPM	Electro permanent magnet
ESC	Electronic speed controller

References

- Lippitt, C.D.; Zhang, S. The impact of small unmanned airborne platforms on passive optical remote sensing: A conceptual perspective. *Int. J. Remote Sens.* **2018**, *39*, 4852–4868. [CrossRef]
- DDC. Drone Delivery Canada. Available online: <https://dronedeliverycanada.com/> (accessed on 6 April 2019).
- Wall Street Journal. Google Drones Can Already Deliver You Coffee In Australia. Available online: <https://www.youtube.com/watch?v=prhDrfUgpB0> (accessed on 6 April 2019).
- Zraick, K.. Like 'Uber for Organs': Drone Delivers Kidney to Maryland Woman. *The New York Times*. Available online: <https://www.nytimes.com/2019/04/30/health/drone-delivers-kidney.html> (accessed on 30 April 2019).
- Estévez, J.; Lopez-Guede, J.M.; Graña, M. Quasi-stationary state transportation of a hose with quadrotors. *Robot. Auton. Syst.* **2015**, *63*, 187–194. [CrossRef]
- Swiss Post. Drones: A Vision Has Become Reality. Available online: <http://www.swisspost.ch/drones> (accessed on 1 July 2019).
- Stolaroff, J.K.; Samaras, C.; O'Neill, E.R.; Lubers, A.; Mitchell, A.S.; Ceperley, D. Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery. *Nat. Commun.* **2018**, *9*, 409. [CrossRef] [PubMed]
- Junaid, A.; Konoiko, A.; Zweiri, Y.; Sahinkaya, M.; Seneviratne, L. Autonomous wireless self-charging for multi-rotor unmanned aerial vehicles. *Energies* **2017**, *10*, 803. [CrossRef]
- Campi, T.; Cruciani, S.; Feliziani, M. Wireless power transfer technology applied to an autonomous electric uav with a small secondary coil. *Energies* **2018**, *11*, 352. [CrossRef]
- Morbidi, F.; Cano, R.; Lara, D.; Morbidi, F.; Cano, R.; Lara, D.; Generation, M.E.P.; Morbidi, F.; Cano, R.; Lara, D. Minimum-Energy Path Generation for a Quadrotor UAV. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 2–8.
- Kumar, V.; Michael, N. Opportunities and challenges with autonomous micro aerial vehicles. *Int. J. Robot. Res.* **2012**, *31*, 1279–1291. [CrossRef]
- Abdilla, A.; Richards, A.; Burrow, S. Power and endurance modelling of battery-powered rotorcraft. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Hamburg, Germany, 28 September–2 October 2015; pp. 675–680.
- Morton, S.; Papanikolopoulos, N. A small hybrid ground-air vehicle concept. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 5149–5154.
- Economou, J.; Kladis, G.; Tsourdos, A.; White, B. UAV optimum energy assignment using dijkstra's algorithm. In Proceedings of the European Control Conference (ECC), Kos, Greece, 2–5 July 2007; pp. 287–292.

15. Karydis, K.; Kumar, V. Energetics in robotic flight at small scales. *Interface Focus* **2017**, *7*, 20160088. [[CrossRef](#)] [[PubMed](#)]
16. Gandolfo, D.C.; Salinas, L.R.; Brandão, A.; Toibero, J.M. Stable path-following control for a quadrotor helicopter considering energy consumption. *IEEE Trans. Control. Syst. Technol.* **2017**, *25*, 1423–1430. [[CrossRef](#)]
17. Bangura, M.; Mahony, R. Nonlinear dynamic modeling for high performance control of a quadrotor. In Proceedings of the Australasian Conference on Robotics and Automation, Wellington, New Zealand, 3–5 December 2012; pp. 1–10.
18. Salaan, C.J.; Okada, Y.; Hozumi, K.; Ohno, K.; Tadokoro, S. Improvement of UAV's flight performance by reducing the drag force of spherical shell. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 9–14 October 2016; pp. 1708–1714.
19. Bellman, R.E.; Dreyfus, S.E. *Applied Dynamic Programming*; Princeton University Press: Princeton, NJ, USA, 2015; Volume 2050.
20. Levermore, T.; Sahinkaya, M.N.; Zweiri, Y.; Naves, B. Real-time velocity optimization to minimize energy use in passenger vehicles. *Energies* **2016**, *10*, 30. [[CrossRef](#)]
21. Bertsekas, D.P.; Bertsekas, D.P.; Bertsekas, D.P.; Bertsekas, D.P. *Dynamic Programming and Optimal Control*; Athena Scientific Belmont: Belmont, MA, USA, 2005; Volume 1.
22. Yacef, F.; Rizoug, N.; Bouhali, O.; Hamerlain, M. Optimization of Energy Consumption for Quadrotor UAV. In Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV), Toulouse, France, 18–21 September 2017; pp. 215–222.
23. Filatov, D.M.; Friedrich, A.I.; Devyatkin, A.V. Parameters identification of thrust generation subsystem for small unmanned aerial vehicles. In Proceedings of the 2nd International Conference on Control in Technical Systems (CTS), St. Petersburg, Russia, 52–27 October 2017; pp. 381–383.
24. *U.S. Standard Atmosphere*; NASA: Greenbelt, MD, USA, 1976.
25. Bañuelos-Ruedas, F.; Angeles-Camacho, C.; Rios-Marcuella, S. Analysis and validation of the methodology used in the extrapolation of wind speed data at different heights. *Renew. Sustain. Energy Rev.* **2010**, *14*, 2383–2391. [[CrossRef](#)]
26. Gill, R.; D'Andrea, R. Propeller thrust and drag in forward flight. In Proceedings of the IEEE Conference on Control Technology and Applications (CCTA), Mauna Lani, HI, USA, 27–30 August 2017; pp. 73–79.
27. Svacha, J.; Mohta, K.; Watterson, M.; Loianno, G.; Kumar, V. Inertial Velocity and Attitude Estimation for Quadrotors. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–9.
28. Molter, C.; Cheng, P. Propeller performance calculation for multicopter aircraft at forward flight conditions and validation with wind tunnel measurements. In Proceedings of the International Micro Air Vehicle Conference and Flight Competition (IMAV), Stuttgart, Germany, 12 December 2017; pp. 307–315.
29. Russell, C.R.; Jung, J.; Willink, G.; Glasner, B. Wind Tunnel and Hover Performance Test Results for Multicopter UAS Vehicles. In Proceedings of the 72nd American Helicopter Society (AHS) International Annual Forum and Technology Display, West Palm Beach, FL, USA, 17–20 May 2016; NASA: Greenbelt, MD, USA, 2016; pp. 1–20.
30. Huang, H.; Hoffmann, G.M.; Waslander, S.L.; Tomlin, C.J. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3277–3282.
31. Leishman, G.J. *Principles of Helicopter Aerodynamics with CD Extra*; Cambridge University Press: Cambridge, UK, 2006.
32. Svacha, J.; Mohta, K.; Kumar, V. Improving quadrotor trajectory tracking by compensating for aerodynamic effects. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; pp. 860–866.
33. Spurný, V.; Báča, T.; Saska, M.; Pěnička, R.; Krajník, T.; Thomas, J.; Thakur, D.; Loianno, G.; Kumar, V. Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles. *J. Field Robot.* **2019**, *36*, 125–148. [[CrossRef](#)]
34. Aminzadeh, A.; Atashgah, M.A.; Roudbari, A. Software in the loop framework for the performance assessment of a navigation and control system of an unmanned aerial vehicle. *IEEE Aerosp. Electron. Syst. Mag.* **2018**, *33*, 50–57. [[CrossRef](#)]

35. Jaleel, H.; Abdelkader, M.; Shamma, J.S. Real-time distributed motion planning with submodular minimization. In Proceedings of the IEEE Conference on Control Technology and Applications (CCTA), Copenhagen, Denmark, 21–24 August 2018; pp. 885–890.
36. Schofield, S.D.; Edwards, M.J.; Green, R.D. Sensitivity analysis of multirotor position control. In Proceedings of the International Conference on Image and Vision Computing New Zealand (IVCNZ), Christchurch, New Zealand, 4–6 December 2017; pp. 1–6.
37. Hinas, A.; Ragel, R.; Roberts, J.; Gonzalez, F. A Framework for Vision-Based Multiple Target Finding and Action Using Multirotor UAVs. In Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 1320–1327.
38. Heintz, C.; Bailey, S.C.; Hoagg, J. Formation Control of Fixed-Wing Unmanned Aircraft: Theory and Experiments. In *AIAA Scitech 2019 Forum*; AIAA: Reston, VA, USA, 2019; p. 1168.
39. Gadalla, M.; Zafar, S. Analysis of a hydrogen fuel cell-PV power system for small UAV. *Int. J. Hydrog. Energy* **2016**, *41*, 6422–6432. [[CrossRef](#)]
40. Chekiri, R.; Lavoie, P.; Richarz, W.G. Experimental aeroacoustics of a two-strut, two-wheel landing gear in a propeller wake. In Proceedings of the 52nd Aerospace Sciences Meeting, National Harbor, MD, USA, 13–17 January 2014; p. 0020.
41. Abdullah Mohiuddin. UAV Velocity Optimized Aerial Transportation via RTDP—Youtube. Available online: https://youtu.be/VOTTZ_fBlyU (accessed on 11 April 2019).



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).