Autonomous Hybrid Ground/Aerial Mobility in Unknown Environments

*David D. Fan[1,2], *Rohan Thakker[1], Tara Bartlett[1], Meriem Ben Miled[1],
Leon Kim[1], Evangelos Theodorou[2], Ali-akbar Agha-mohammadi[1]

***Abstract*— Hybrid ground and aerial vehicles can possess distinct advantages over ground-only or flight-only designs in terms of energy savings and increased mobility. In this work we outline our unified framework for controls, planning, and autonomy of hybrid ground/air vehicles. Our contribution is three-fold: 1) We develop a control scheme for the control of passive two-wheeled hybrid ground/aerial vehicles. 2) We present a unified planner for both rolling and flying by leveraging differential flatness mappings. 3) We conduct experiments leveraging mapping and global planning for hybrid mobility in unknown environments, showing that hybrid mobility uses up to five times less energy than flying only.**

I. INTRODUCTION

In recent years, the development of Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) has been increasingly maturing in various applied fields, including agriculture [1], search and rescue [2] [3], delivery [4], and military applications [5]. However, the widespread adoption of autonomous vehicles still faces significant challenges, including those introduced by effective mobility in diverse environments.

While UAVs have great advantages over UGVs in terms of their ability to traverse difficult terrain, they are constrained by high energy requirements, short flight times, and low payload capabilities. They are also fragile, non-robust to collisions, and may be adversely affected by large wind or gust disturbances. On the other hand, UGVs can be designed to carry larger payloads, be more resistant to damage with heavier, more robust frames, and carry larger sources of power. The trade-off is that many points of interest are often inaccessible from the ground. Hybrid UAV/UGV vehicles are designed to combine the advantages of both classes of systems by moving on the ground to save power when possible, and flying when terrain constraints do not allow otherwise.

There has been some considerable interest in hybrid ground/aerial vehicles in recent years. The most common approach seems to be the use of passive or actuated wheels attached to the frame of a drone or quadrotor in various configurations [6], [7]. Other designs include a quad-rotor hinged inside a rotating cage [8], modular or adaptive structures which change configuration for flight or ground mobility [9], [10], [11], and even legged drones [12], [13]. These efforts are interesting explorations into the design space of hybrid ground/air drones. The focus of this work, however, is a unified guidance, navigation, controls, and autonomy stack specifically geared towards Unmanned Hybrid Vehicles (UHVs).

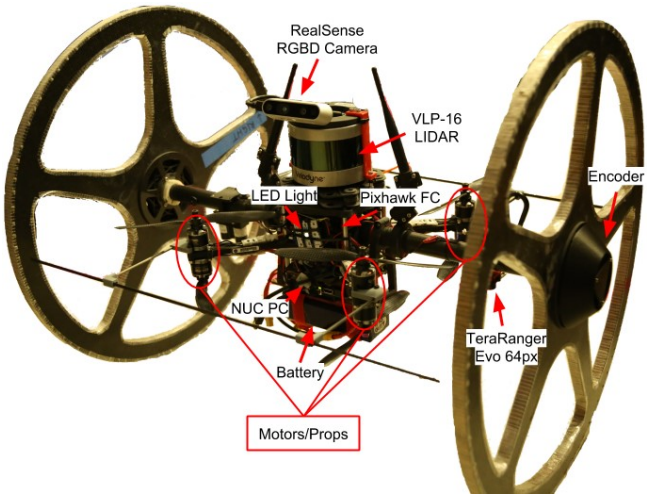

Fig. 1. Rollocopter hardware configuration

In this work, we make use of a hybrid vehicle which consists of a quad-rotor aerial vehicle with two passive wheels attached to each side of the frame of the quad-rotor. This design has been previously considered in [14], [15] and has the advantage of possessing the aerial power and agility of a quad-rotor while maintaining a low additional mass profile from the added wheels.

The objective of this paper is to describe a unified framework for controls, planning, and autonomy of hybrid ground/air vehicles. We take a unified planning approach which generalizes both rolling and flying by taking advantage of differential flatness mappings for quadrotors [16] as well as for two-wheeled non-holonomic wheeled mobile robots [17]. This results in a planner with the same structure for both mobility modes. We present a study of autonomous navigation of our hardware platform in environments which require hybrid mobility, showing 5x energy savings of hybrid mobility vs. flying only.

The main contributions of this work are as follows:

1) Control of non-actuated two-wheeled hybrid quadrotor vehicles.
2) A unified hybrid trajectory planner for both rolling and flying.
3) Mapping and global planning for hybrid mobility in unknown environments.

*These authors contributed equally to this work.

[1]NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

[2]Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta GA, USA

II. System Overview

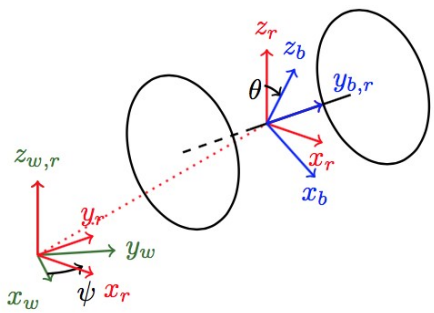

Fig. 2. Figure shows reference frames: world frame w (green), body frame b (blue) i.e. fixed w.r.t the quadrotor frame and rolling frame r (red), whose z-axis points in the direction of the surface normal of the ground and is separated from the body frame only by a rotation along y-axis.

We first outline the overall system architecture and describe the various components. We begin with a description of the frames used in this work. Three frames are used which we call world, body, and rolling, the last of which is unique to the hybrid design (See Figure 2). Note that in this work we are assuming that the ground is flat, hence, z_w is parallel to z_r.

The full state of the quadrotor while in flight is given as $x_a = ({}^w\vec{p}, {}^w\vec{v}, R_a, {}^w\vec{\omega}_a) \in \mathbb{R}^3 \times \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3$, and the input is given by $u = (|F|, {}^bM_x, {}^bM_y, {}^bM_z)$ which maps to $\vec{F}$ i.e. the thrust of four motors. Similarly, the full state while rolling is given by $x_g = ({}^wp_g, {}^wv_g, R_g, {}^w\omega_g) \in \mathbb{R}^2 \times \mathbb{R}^2 \times SO(2) \times \mathbb{R}$ while the control input is same that for flying.

As shown in Figure 3, our system architecture takes a unified approach to hybrid ground and aerial autonomy. Sensors in the hardware layer feed raw perceptual information to the perception layer. Visual Inertial Odometry (VIO) is used to estimate the current pose. The pose, along with pointclouds from range sensors, is used to create a map of the environment in the Local Mapper. This map is passed along to a geometric planner, the Hybrid Global Planner, which plans hybrid trajectories in pursuit of some objective, e.g. reach some global goal or explore unexplored space. Hybrid trajectories are generated which contain information about when the robot should roll, fly, land, takeoff, etc. A local waypoint from this global trajectory is passed along to the lower level Local Planner. This planner generates kinodynamically feasible trajectories which balance collision avoidance with reaching the goal. It is able to handle both rolling and flying trajectories in a unified manner. Once a feasible trajectory is selected, a setpoint along that trajectory is passed to the low-level Controller module. The controller is a hybrid controller which can perform both rolling and flying maneuvers and can track given setpoints. It sends desired attitudes/motor thrusts to the low-level flight controller or ESCs. In the next section we describe the hybrid controller and show experimental verification of its efficacy.

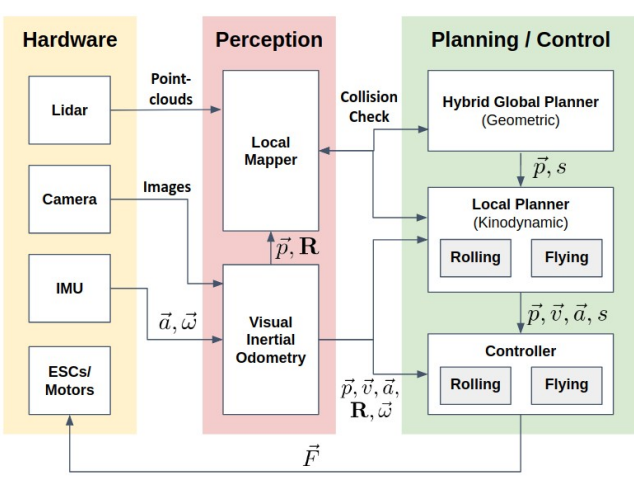

Fig. 3. System architecture

Notation

$f \in \{w, r, b\}$	Reference frames defined in Figure 2
$s \in \{g, a\}$	Mobility state: g ground/rolling and a aerial/flying
${}^f\vec{p}_g \in \mathbb{R}^2$	Position x, y of rolling frame w.r.t. world frame represented in f frame
${}^f\vec{p}_a \in \mathbb{R}^3$	Position x, y, z of body frame w.r.t. world frame represented in f frame
$R_g \in SO(2)$	Orientation of rolling frame w.r.t. world frame
$R_a \in SO(3)$	Orientation of body frame w.r.t. world frame
${}^f\vec{v}_g \in \mathbb{R}^2$	Linear x, y velocity of rolling frame w.r.t world frame represented in f frame
${}^f\vec{v}_a \in \mathbb{R}^3$	Linear x, y, z velocity of body frame w.r.t world frame represented in f frame
${}^f\vec{\omega}_g \in \mathbb{R}$	Angular z velocity of rolling frame w.r.t world frame represented in f frame
${}^f\vec{\omega}_a \in \mathbb{R}^3$	Angular velocity of body frame w.r.t world frame represented in f frame
${}^f\vec{a}_g \in \mathbb{R}^2$	Linear x, y acceleration of rolling frame w.r.t world frame represented in f frame
${}^f\vec{a}_a \in \mathbb{R}^3$	Linear x, y, z acceleration of body frame w.r.t world frame represented in f frame
${}^f\vec{\alpha}_g \in \mathbb{R}$	Angular z acceleration of rolling frame w.r.t world frame represented in f frame
${}^f\vec{\alpha}_a \in \mathbb{R}^3$	Angular acceleration of body frame w.r.t world frame represented in f frame
$\vec{F}$	Thrust vector whose elements represent thrust of each motor
θ	Angle between z-axis of rolling frame and z-axis of body frame
ψ	Yaw between rolling frame and world frame
x^d	Desired state
$\hat{x}$	Estimated state
x^{ff}	Feedforward state

III. Controls for Flying and Rolling

We first describe the standard control architecture that used for flying and then extend this architecture for ground mobility.

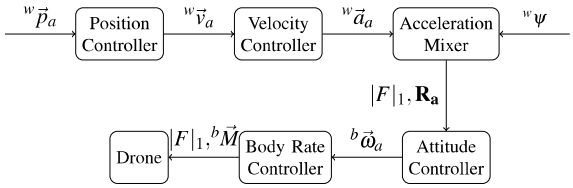

Fig. 4. Architecture of the flying controller

A. Flying Controller

Figure 4 shows the non-linear back-stepping controller used for flying [18], [19]. The controller receives the desired state ${}^{w}\vec{p}, {}^{w}\vec{v}, {}^{w}\vec{a}, {}^{w}\psi$ from the local planner. The feedback position controller uses a proportional control law to generate the desired velocity. The velocity controller generates a desired acceleration based on the desired velocity from the position controller and a feed-forward term from the position controller using a PID control law. The acceleration mixer calculates the desired thrust $|F|_1$ and attitude $\mathbf{R}$ based on desired acceleration, feed-forward acceleration and yaw. Similar to the position controller, the attitude controller generates a desired angular velocity in body frame based on error in attitude based on proportional control law and the body rate controller uses this to generate to generate a desired moment based on a PID control law. We refer readers to [19] for the exact equations of the dynamics and control.

B. Rolling Controller

The following are the key differences between rolling and flying modes:

- In rolling mode, we assume a no-slip condition: the robot cannot have a velocity along body y-axis due to the non-holonomic constraint.
- The bandwidth of yaw dynamics for rolling is lower than that for flying due to interaction with the ground.
- The pitch dynamics for rolling are as fast as flying assuming the friction in the bearing of the wheels is negligible.
- The roll of the robot is constrained such that the wheels always maintain contact with the terrain while rolling.

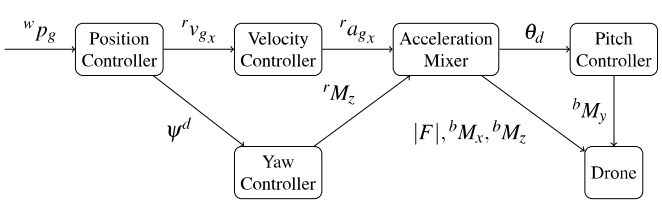

Fig. 5. Architecture of the Rolling controller

Based on this intuition we extend the flying control architecture for rolling as shown in Figure 5. The following shows the equations of the control law for rolling mode.

Position Controller: Desired velocity along x-axis in rolling frame is calculated using a proportional control law on error in position as follows

$$ {}^{w}e_g = {}^{w}p_g^d - {}^{w}\hat{p}_g \tag{1} $$

$$ {}^{r}e_g = R(\psi_{yaw})^{w}e_g \tag{2} $$

$$ {}^{r}v_{g_x} = {}^{pos}k_p({}^{r}e_g) \tag{3} $$

Yaw Controller: The yaw controller generates a desired moment in rolling frame using a PD control law on S^1 as follows:

$$ {}^{w}\psi^d = \tan^{-1}({}^{w}e_{g_y}/{}^{w}e_{g_x}) \tag{4} $$

$$ {}^{w}e_\psi = d_{S(1)}({}^{w}\psi^d, {}^{w}\psi) \tag{5} $$

$$ {}^{r}M_z = {}^{\psi}k_p({}^{w}e_\psi) + {}^{\psi}k_d({}^{w}\dot{e}_\psi) \tag{6} $$

Velocity Controller: The velocity controller generates a desired acceleration based the desired velocity and feed-forward velocity as follows:

$$ {}^{r}\dot{e}_{g_x} = ({}^{r}v_{g_x}^{d} + {}^{r}v_{g_x}^{ff}) - {}^{r}\hat{v}_{g_x} \tag{7} $$

$$ {}^{r}a_x = {}^{vel}k_d({}^{r}\dot{e}_x) + {}^{vel}k_I\chi_1 \tag{8} $$

Acceleration Mixer:

$$ {}^{b}\theta_{pitch}^d = \sin^{-1}({}^{r}a_x m/|F|) \tag{9} $$

$$ {}^{b}M_x, {}^{b}M_z = R({}^{b}\hat{\theta}_{pitch})^{r}M_z \tag{10} $$

Pitch Controller:

$$ {}^{b}e_{pitch} = d_{S(1)}({}^{b}\theta_{pitch}^d, {}^{b}\hat{\theta}_{pitch}) \tag{11} $$

$$ {}^{b}M_y = {}^{pitch}k_p({}^{b}e_{pitch}) + {}^{pitch}k_d(-{}^{b}\dot{\theta}_{pitch}) + {}^{pitch}k_I\chi_2 \tag{12} $$

where ${}^{pos}k_p$, ${}^{\psi}k_p$, ${}^{\psi}k_d$, ${}^{vel}k_d$, ${}^{\theta}k_p$, ${}^{\theta}k_d$ and ${}^{\theta}k_I$ are constant gains, (χ_1, χ_2) the integral position tracking error of velocity and pitch angle respectively.

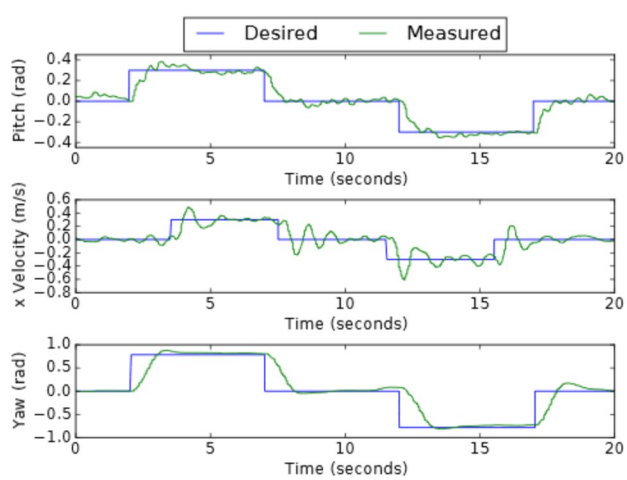

Fig. 6. Step responses for the pitch, velocity and yaw controllers for rolling.

1) Experimental Validation: We first show performance of the pitch controller while keeping the wheels fixed (Figure 6). After tuning the pitch controller, we give desired step inputs in x velocity and plot the response. Finally we give step inputs in yaw and tune the yaw controller.

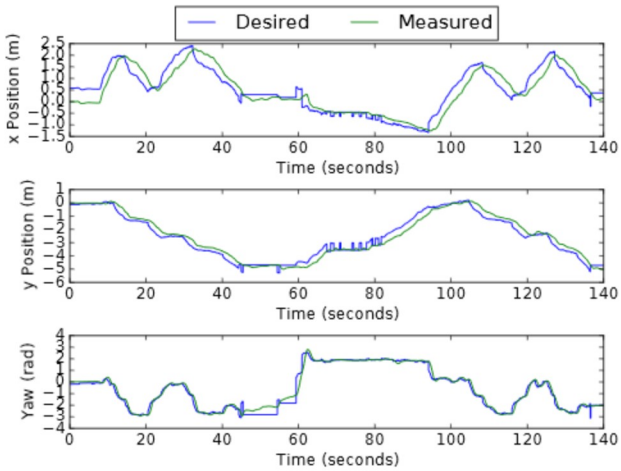

Fig. 7. Performance of rolling controller in response to changing set-points given by the local planner while traversing through the test course. Plots show position responses in x and y as well as yaw.

After tuning the controller with step responses we test the performance of rolling in a test course, as discussed in Section VI-B. This demonstrates that the robot is able to successfully navigate with course without colliding, proving the efficacy of the rolling controller.

C. Transition between Rolling and Flying

As shown in Fig 3 transition between flying and rolling is made by the Local planner. Besides position, velocity and acceleration, the hybrid controller receives the mobility mode s : ground or aerial.

Take-Off: Take-off is triggered by increasing the set-point along the z-axis in world frame and finished when the ground clearance is above a certain threshold.

Landing: Landing is triggered by asking for a negative velocity along the z-axis in world frame and finished when the ground clearance is lower than a certain threshold for 1 second.

IV. Local Planning

A. Hybrid Differential Flatness Planning

The design requirements of our local planner are as follows:

1) Accept position waypoint goals in the world frame.
2) Plan kino-dynamically feasible trajectories which are collision-free, for both rolling and flying.
3) Weigh competing costs of reaching the goal vs. maintaining a safe distance from obstacles.
4) Replan at the rate of received instantaneous point-clouds for agile responsiveness to dynamic obstacles.
5) Return a position and velocity setpoint for the controller to track.

Various trajectory planning methods exist in both the literature and in practice. Our approach seeks to fulfill these design requirements in a way that takes into account hybrid mobility. To this end we use a motion primitive-based approach. Generally motion primitives are computed via alternative paths [20] or by generating trajectories to a selection of waypoints within the field of view [21]. In this work we take the latter approach, generating kino-dynamically feasible trajectories using the same representation for both aerial and ground mobility.

Prior work has shown that differential flatness mapping-based planning for quadrotor flight has advantages in terms of agility, computational efficiency, and simplicity of design [22], [23]. Similar differential-flatness based mappings exist for a variety of non-holonomic vehicle configurations, including two-wheeled mobile robots [24], [17]. We first begin with a brief overview of differential flatness mappings for planning, then show how the dynamics of two-wheeled mobile robot maps to a subspace of the flat space of quadrotor dynamics (Figure 8). This enables us to plan using the same representation for rolling and flying, leading to a unified hybrid local planner.

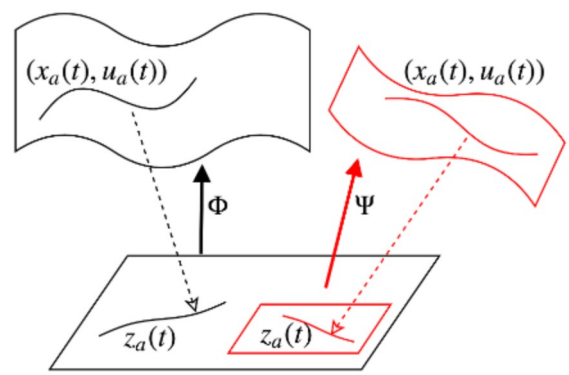

Fig. 8. Differential flatness

It has been shown [22] that one can find a mapping from the state x_a and control u to a flat space consisting of four variables: $z_a = ({}^w\vec{p}_a, \psi)$ and the derivatives of z. Define a trajectory $z_a(t) : [t_0, T] \to \mathbb{R}^3 \times SO(2)$ as a smooth curve parameterized by t. There exists a smooth mapping Φ, such that $(x_a, u) = \Phi(z_a, \dot{z}_a, \ddot{z}_a, \dddot{z}_a, \ddddot{z}_a)$. This is the differential flatness mapping from the dynamics of the quadrotor in flight to a flat space. Prior work has shown that one can construct minimum-snap polynomial trajectories in this flat space which map back to trajectories of the quadrotor dynamics [22]. Here we take a similar approach and define polynomial trajectories by their start and end points while assuming the accelerations, jerks, and snaps at these points are 0, i.e. $z_a(t) = (\vec{p}_a(t), \psi(t))$, $\dot{z}_a(t) = ({}^w\vec{v}, {}^w\dot{\psi}(t))$, $\ddot{z}_f(t) = \dddot{z}_f(t) = \ddddot{z}_f(t) = 0$ for $t = t_0$ and $t = T$.

We now review a similar deferentially flat mapping for two-wheeled mobile robots [17]. When the vehicle is rolling on flat ground, the state representation is reduced to $x_g = (\vec{x}_g, \psi)$ and the control inputs are $u_g = ({}^r v_{g_x}, \dot{\psi})$. There exists a differential flatness mapping from x_g and u_g to the flat space $z_g = ({}^w\vec{p})$, i.e. $(x_g, u_g) = \Psi(z_g, \dot{z}_g)$. Note that z_g is a subspace of z_a. We therefore can plan trajectories for rolling using the same polynomial representations we use for generating flight trajectories. Instead of generating 4-dimensional polynomials

in z_a, we generate two-dimensional trajectories in z_r. In this way we maintain a consistent architecture and pipeline for both mobility modes.

For both mobility modes, primitives are created based on a selection of endpoints. An arc of endpoints are created in the horizontal plane around the vehicle are determined based on a given distance horizon. In the aerial mode, additional endpoints at different heights are included at a variety of angles above and below the horizontal plane as seen in Figure 9.

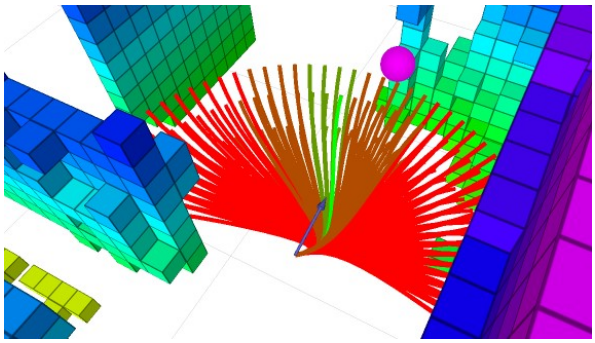

Fig. 9. Motion primitives radiating outward from the origin of the vehicle, with color-coded costs. Red represents primitives which are in collision. Dark red are primitives within the near-collision buffer. Dark green primitives are free-space primitives. Bright green indicates the lowest cost primitive. Note that primitives radiate outwards in x,y and z. The pink sphere is the local goal, and the blue arrow is the current pose of the vehicle. Colored boxes represent occupied space with color representing the height.

B. Selecting Optimal Trajectories

After generating a set of possible kinodynamically feasible trajectories, the planner needs to select the best one. The motion primitives are checked for collisions by using multiple query points along each primitive. For each query point, the distance to the nearest collision point in the representation of the environment is determined using a KD-tree search. If the distance to the nearest obstacle $d_{obstacle}$ is less than the radius of the vehicle, the trajectory is considered to be in collision and rejected by assigning it a large collision cost $C_{collision} = 10^6$. On the other hand, if $d_{obstacle}$ is larger than some buffer value, say 1 meter, then we consider it to be in *near collision* and it is given a collision cost of $C_{collision} = 100 \times W_g - d_{obstacle}$, where W_g is a constant factor which weighs moving towards the goal against staying away from obstacles. We then assign the total cost of the primitive as

$$C_{goal} = W_g \times d_{goal}$$
$$C_{total} = C_{collision} + C_{goal}$$

C. Choosing a controller setpoint

The lowest cost primitive is used to generate a setpoint for the controller. This setpoint can be the point along the trajectory corresponding to the current time. When replanning with a high frequency, this setpoint can be a fixed time interval ahead of the initial time at which the trajectory begins.

When the local planner receives a new goal, that goal has a mobility mode associated with it. If the mobility mode of the goal is different from the current mobility mode, the local planner will switch mobility modes, either from rolling to flying or vice versa. The local planner will then send a flying controller setpoint to command takeoff or landing. The takeoff command is achieved by sending an increasing z position setpoint to the controller. This transition is declared complete when the ground clearance is sufficiently high. Similarly, the landing command is sent to the controller as a negative velocity in the z direction. When the ground clearance is approximately equal to the wheel radius for a specified time interval, the vehicle is declared as having landed.

D. Localization-free robustness

In GPS-denied environments, or where position information is not available (i.e. $\vec{p}$ is unknown), position control is no longer feasible. A common approach is to perform collision avoidance on instantaneous collision information and follow walls (see [25], [26], [27], [28], [29], and many others). Our hybrid motion primitive approach is well-suited to this type of behavior. When localization is unavailable, we send desired velocities only, since velocity estimates are more reliable even when position-based localization fails. Because the differentially flat trajectories are very cheap and easy to generate, it is possible to re-plan at a high frequency, even at the frequency of instantaneous collision point clouds. Even without mapping and localization, basic collision-avoidance behaviors emerge from this framework. This provides a level of robustness to unreliable localization methods. More discussion about localization and mapping follows in the next section.

V. Localization, Mapping, and Global Planning

A. Localization

The ability to safely navigate and traverse unknown and GPS-denied environments necessitates a reliable autonomous localization system that can run in real-time. The system utilizes ORB-SLAM2 with RGB-D data from the Intel Realsense cameras for pose estimation. ORB-SLAM2 is an open-source SLAM system for monocular, stereo, and RGBD cameras which leverages the speed of ORB features and back-end bundle adjustment to enable lightweight and accurate real-time localization [30]. This pose estimate $(\vec{p}, \psi)$ is fused with the IMU onboard the Pixhawk flight controller via an onboard EKF to produce state estimates.

B. Mapping

Mapping for mobile robots is a well-studied problem [31]. In this work we take a traditional approach and leverage existing work. The point clouds from a 3D LIDAR along with the state estimates are combined into a local map using OctoMap that can be used for higher fidelity planning and navigation in geometrically complex environments. OctoMap is an open-source mapping framework which utilizes the octree data structure and probabilistic occupancy estimation

to efficiently generate three-dimensional models of the environment [32]. A local map of obstacles is generated in a 5m $\times$ 5m $\times$ 3m region around the current position, and is cleared upon localization failure. The hybrid mobility of the vehicle makes mapping especially important due to the limited field of view in elevation angle of the VLP-16 LIDAR as well as occlusion by the wheels.

C. Hybrid Global Planning

The vehicle autonomously navigates through the freshly mapped environment to a goal position through the use of an A* search planner [33]. Nodes of the search graph are placed in free space. An incrementally updated Euclidean Distance Transform (EDT) is calculated from the occupancy map through a brushfire algorithm to determine whether a node is in collision or close enough to an obstacle to be considered collided [34]. Each node has a cost which is determined by its distance from obstacles. An additional cost for flight is added to nodes which are above the ground. This cost is formulated to account for the increased energy required to fly compared to rolling and results in paths which prioritize rolling over flying when the terrain allows for it. See Figure (10).

D. Local Waypoint Generation

Given the hybrid path planned by A*, the global planner sends a goal to the local planner. This goal is chosen by taking the nth node further along the path from the current position. If the next node in the path is a different mobility mode than the current state of the vehicle, then a transition is requested of the local planner and controller, either from rolling to flying or vice versa. The replanning frequency is determined by octomap updates and the length of time taken to perform A* search, generally on the order of 1-2Hz.

VI. Experiments

In this section, we show experiments to compare energy consumption for different mobility modes of the rollocopter.

A. Hardware

Figure 1 shows the hardware configuration of the Rollocopter system. The electronic speed controllers (ESCs), motors, and propellers are used to control the drone. The wheels, connected to the drone by the main shaft, are made of a light carbon-fiber honeycomb structure. The vehicle is actuated by the propellers and uses the wheels for rolling, thus enabling hybrid mobility. The platform uses the following sensors onboard:

- RealSense RGB-D Camera: This camera provides the stereo and RGB-D images which can be used for simultaneous localization and mapping algorithms, such as ORBSLAM.
- Velodyne 360° VLP-16 LIDAR: The LIDAR provides a point cloud around the vehicle with a 360° azimuth angle and $\pm 15^o$ elevation angle field of view. This is used for collision checking and local mapping.
- IR LED-based height sensor (TeraRanger Evo 64px): Two TeraRanger Evo 64px height sensors [35] are used to provide bottom and top clearances. These sensors are critical for mitigating the limited field of view of the Velodyne sensor in the downwards and upwards directions.
- Pixhawk flight controller: We use a Pixhawk flight controller with onboard IMU, gyroscope and attitude estimation.
- Encoders: We utilize wheel encoders to aid in velocity estimation while operating on the ground.
- Intel NUC: The system is equipped with onboard computing, an Intel i7 Core processor and 32GB RAM. The system runs Ubuntu 16.04 and we use ROS [36] for message passing and handling.

The total mass of the system is 4.231 kg. The wheels, bearings, and mounting, weigh around 500g. We experimentally verify a maximum thrust rating of 62.5 N.

B. Test Environment

The test course in Figure 11is suitable for both flying and rolling and contains narrow corridors and hairpin turns. It also contains removable obstacles for testing hybrid mobility. The walls are 1.5m high and removable obstacles are 0.4m high. The top of the course is covered to create an enclosed maze. The total length of one circuit around the track is approximately 18m. Due to the narrow width of the corridors (1.2m) and the frequent 180° turns, a high degree of collision avoidance, and mobility is required.

C. Energy Comparison of Different Mobility Modes

Our first experiment was a comparison between rolling and flying. We collected rolling and flight data in the test course and log battery voltage and current levels. The desired velocity of both air and ground traversal was set at 0.3 m/s. A comparison between the average power used during both mobility modes is found in Table I. In these experiments we found that on average, rolling is 5x more efficient than flying. This is a conservative lower bound, and further gains in energy efficiency while rolling should be possible, since we did not dynamically adjust the thrust of the rolling controller. We leave this for future work.

As an initial estimate of the hybrid vehicle's energy efficiency over a purely aerial vehicle of a similar configuration/mass without the hardware needed to roll, the average power consumption of both cases can be compared. While we experimentally verified the power consumption of the vehicle with wheels in Table I, we approximate the average power consumption of an aerial vehicle without wheels to be roughly 650 W. From this approximation and our experimentally determined average power consumption values, the minimum percentage of rolling time to total travel time to justify a hybrid vehicle configuration over a purely aerial one is calculated as 41.4%. This value is obtained under the assumption that the average power consumption of the hybrid vehicle is a time-weighted average of the experimentally determined power consumption values of rolling and flying.

Next, we conducted experiments to quantify the power consumption of the vehicle in rolling only, flying only

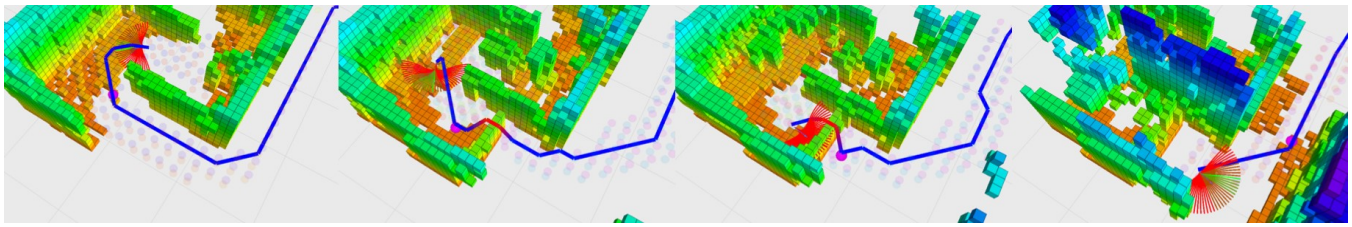

Fig. 10. Time sequence of planned hybrid transition from rolling to flying and back to rolling. Colored blocks are occupied voxels colored by z height. Blue/Red path indicates hybrid planned path where blue is rolling and red is flying. Transparent spheres denote A* nodes. Pink sphere is goal waypoint sent from the global planner to the local planner. Motion primitives from local planner are shown. Note that as the vehicle moves forward, an obstacle is revealed and a small hop over it is planned and executed.

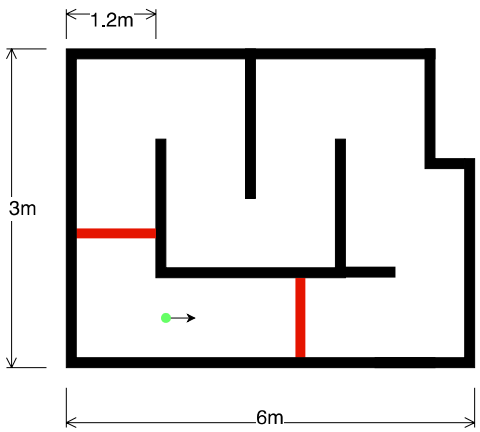

Fig. 11. Top-down view of test course with walls represented by black lines. The red lines represent removable obstacles. The green point and arrow represent the starting pose of the vehicle.

Mobility Mode	Avg. Power (Watts)	σ
Rolling	194.5	17.6
Flying	971.9	66.3

TABLE I

COMPARISON OF AVERAGE POWER USAGE DURING ROLLING VS. FLIGHT OVER 7 MINUTES. ROLLING IS 5X MORE EFFICIENT.

and hybrid modes while planning autonomously in the test course. We first removed any obstacles to allow the vehicle to traverse the entire course without flying. We verified that the planner chooses to roll at all times. We plotted the power used as well as energy consumption as a function of the ground distance the vehicle travels (Figure 12. We then add the obstacles to the course. The hybrid planner autonomously plans trajectories through the course, flying when it encounters an obstacle, then landing again, in an energy-optimal manner (Figure 10). Finally, we force the local planner to fly only, traversing through the course on flight alone to perform a comparison between hybrid mobility vs. flight.

In these experiment shown in Figure 12 we observed about a 3.5x reduction in power consumption while rolling vs. flying, due to the use of a higher desired thrust while rolling. However, >5x power reductions are entirely feasible by using lower rolling thrust. This reduction depends on a number of factors including wheel friction, terrain traversability,

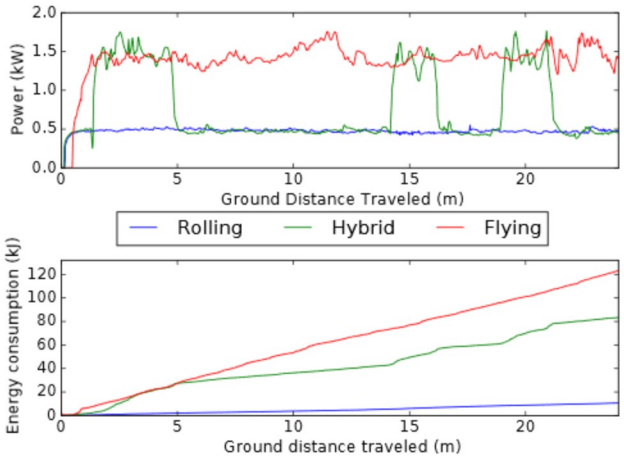

Fig. 12. Comparison of power and distance capabilities of the hybrid platform while rolling, hybrid planning, and flying in the test course. The hybrid planner flies over three obstacles, while rolling was tested with all obstacles removed.

vehicle configuration, etc. See [14], [15] for a more thorough treatment of the energy trade-offs associated with this design.

VII. CONCLUSIONS

With the hybrid vehicle system design, we are able to show advantages in reduced energy consumption, increased maximum run times, and increased distance traveled, which translates to further exploration of unknown environments. We have presented a novel controller for rolling, a unified planning framework for both ground and aerial mobility, and experiments quantifying the energy savings of hybrid mobility while exploring unknown, constrained environments. Future work includes a more comprehensive analysis and design of controllers for ground mobility on rough terrain, consideration of differential flatness trajectory optimization in the context of Model Predictive Control, and investigation of other hybrid vehicle designs.

ACKNOWLEDGMENT

The authors would like to thank the JPL SubT team for their support in the development of the hybrid aerial/ground vehicle. We would particularly like to thank the hardware team consisting of: Scott Harper, Matt Anderson, Arash Kalantari, Fernando Chavez, Kyle Strickland, and Thomas

Touma. We would also like to thank Brett Lopez for discussions regarding formulating the hybrid controls and planning framework.

This research was partially carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by Ali-akbar Agha-mohammadi. Copyright ©2019. All rights reserved.

REFERENCES

[1] S. Herwitz, L. Johnson, S. Dunagan, R. Higgins, D. Sullivan, J. Zheng, B. Lobitz, J. Leung, B. Gallmeyer, M. Aoyagi, *et al.*, "Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support," *Computers and Electronics in Agriculture*, vol. 44, no. 1, pp. 49–61, 2004.

[2] R. Bähnemann, D. Schindler, M. Kamel, R. Siegwart, and J. Nieto, "A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects," in *Proceedings of 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, p. 8088150, IEEE, 2017.

[3] M. Nieuwenhuisen, M. Beul, R. A. Rosu, J. Quenzel, D. Pavlichenko, S. Houben, and S. Behnke, "Collaborative object picking and delivery with a team of micro aerial vehicles at mbzirc," in *European Conference on Mobile Robots (ECMR)*, pp. 1–6, IEEE, 2017.

[4] A.-a. Agha-mohammadi, N. K. Ure, J. P. How, and J. Vian, "Health aware stochastic planning for persistent package delivery missions using quadrotors," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3389–3396, IEEE, 2014.

[5] D. D. Fan, E. A. Theodorou, and J. Reeder, "Model-based stochastic search for large scale optimization of multi-agent uav swarms," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2216–2222, IEEE, 2018.

[6] Y. H. Moyuru Yamada, Manabu Nakao and N. S. Development, "Development and field test of novel two-wheeled uav for bridge inspections," *International Conference on Unmanned Aircraft (ICUAS)*, 2017.

[7] J. R. Page and P. E. I. Pounds, "The quadroller: Modeling of a uav/ugv hybrid quadrotor," *International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[8] A. Kalantari and M. Spenko, "Modeling and performance assessment of the hytaq, a hybrid terrestrial/aerial quadrotor," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1278–1285, 2014.

[9] S. Mintchev and D. Floreano, "A multi-modal hovering and terrestrial robot with adaptive morphology," in *Proceedings of the 2nd International Symposium on Aerial Robotics*, 2018.

[10] S. Morton and N. Papanikolopoulos, "A small hybrid ground- air vehicle concept," *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[11] M. Zhang, B. Chai, L. Cheng, Z. Sun, G. Yao, and L. Zhou, "Multi-movement spherical robot design and implementation," *Proceedings of 2018 IEEE International Conference on Mechatronics and Automation, ICMA 2018*, pp. 1464–1468, 2018.

[12] K. Li, B. Han, Y. Zhao, and C. Zhu, "Motion planning and simulation of combined land-air amphibious robot," in *IOP Conference Series: Materials Science and Engineering*, vol. 428, p. 012057, IOP Publishing, 2018.

[13] Y. Mulgaonkar, B. Araki, J.-s. Koh, L. Guerrero-Bonilla, D. M. Aukes, A. Makineni, M. T. Tolley, D. Rus, R. J. Wood, and V. Kumar, "The flying monkey: a mesoscale robot that can run, fly, and grasp," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4672–4679, IEEE, 2016.

[14] A. Kalantari and M. Spenko, "Design and experimental validation of hytaq, a hybrid terrestrial and aerial quadrotor," in *2013 IEEE International Conference on Robotics and Automation*, pp. 4445–4450, IEEE, 2013.

[15] A. Kalantari and M. Spenko, "Hybrid aerial and terrestrial vehicle," June 23 2015. US Patent 9,061,558.

[16] S. Formentin and M. Lovera, "Flatness-based control of a quadrotor helicopter via feedforward linearization," *Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011.

[17] C. P. Tang, "Differential flatness-based kinematic and dynamic control of a differentially driven wheeled mobile robot," in *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2267–2272, IEEE, 2009.

[18] M. L. Taeyoung Lee and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," *49th IEEE Conference on Decision and Control*, 2010.

[19] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control: Technical report," tech. rep., ETH Zurich, 2013.

[20] J. Zhang, R. G. Chadha, V. Velivela, and S. Singh, "P-cap: Pre-computed alternative paths to enable aggressive aerial maneuvers in cluttered environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8456–8463, IEEE, 2018.

[21] A. A. Paranjape, K. C. Meier, X. Shi, S.-J. Chung, and S. Hutchinson, "Motion primitives and 3d path planning for fast flight through a forest," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 357–377, 2015.

[22] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[23] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5759–5765, IEEE, 2017.

[24] R. MURRAY, "Differential flatness of mechanical control system: A catalog of prototype systems," in *ASME Intl. Mech Eng Congress and Expo*, 1995.

[25] M. Katsev, A. Yershova, B. Tovar, R. Ghrist, and S. M. LaValle, "Mapping and pursuit-evasion strategies for a simple wall-following robot," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 113–128, 2011.

[26] P. Van Turennout, G. Honderd, and L. Van Schelven, "Wall-following control of a mobile robot," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pp. 280–285, IEEE, 1992.

[27] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi, "Divergent stereo in autonomous navigation: From bees to robots," *International Journal of Computer Vision*, vol. 14, no. 2, pp. 159–177, 1995.

[28] J. Serres, F. Ruffier, S. Viollet, and N. Franceschini, "Toward optic flow regulation for wall-following and centring behaviours," *International Journal of Advanced Robotic Systems*, vol. 3, no. 2, p. 23, 2006.

[29] F. L. Roubieu, J. Serres, N. Franceschini, F. Ruffier, and S. Viollet, "A fully-autonomous hovercraft inspired by bees: Wall following and speed control in straight and tapered corridors," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pp. 1311–1318, IEEE, 2012.

[30] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[31] G. O. Francisco Bonin-Font, Alberto Ortiz, "Visual navigation for mobile robots: A survey," *Intell Robot System*, 2008.

[32] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[33] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[34] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1116–1130, 2013.

[35] TeraRanger, "Teraranger evo 64px," *https://www.terabee.com/shop/3d-tof-cameras/teraranger-evo-64px/*, 2019.

[36] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.