2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Vector Field Following for Quadrotors using Differential Flatness

Dingjiang Zhou and Mac Schwager

***Abstract*— This paper proposes a differential flatness-based method for maneuvering a quadrotor so that its position follows a specified velocity vector field. Existing planning and control algorithms often give a 2D or 3D velocity vector field to be followed by a robot. However, quadrotors have complex nonlinear dynamics that make vector field following difficult, especially in aggressive maneuvering regimes. This paper exploits the differential flatness property of a quadrotor's dynamics to control its position along a given vector field. Differential flatness allows for the analytical derivation of control inputs in order to control the 12D dynamical state of the quadrotor such that the 2D or 3D position of the quadrotor follows the flow specified by a given vector field. The method is derived mathematically, and demonstrated in numerical simulations and in experiments with a quadrotor robot for three different vector fields.**

I. Introduction

In this paper we propose a control design method for driving a quadrotor helicopter along a desired velocity vector field using tools from differential flatness theory, as shown in Fig. 1. This method is intended to make existing planning and control algorithms that assume simple robot dynamics useful for quadrotor robots with complex nonlinear dynamics. Our method produces a low-level feedback controller so that the quadrotor will reliably follow a desired vector field, even in aggressive dynamic flight regimes. The method uses differential flatness theory to produce an input signal to effectively cancel the complex internal dynamics of the quadrotor, so it can be controlled as if it were an integrator.

Many control and planning techniques give an output in the form of a vector field along which the robot is intended to move. For example, navigation, obstacle avoidance, collision avoidance, swarming and flocking, and mobile network connectivity maintenance controllers often give a vector field for the robot or robots to follow to accomplish the desired task. This is effective for robots moving in a pseudo-static regime in which dynamics do not play a major role in the motion, for example ground robots with significant gear reduction, or slow moving aerial robots. However, in a dynamic regime in which inertial effects cannot be neglected, it is challenging to control a robot along an arbitrary vector field. Specifically, quadrotor helicopters have high dimensional, nonlinear dynamics to contend with, so algorithms that give a vector field are of questionable applicability without an "inner loop" controller to guide the quadrotor along the vector field. For example, a typical obstacle avoidance algorithm will give a vector field along which a robot can drive to avoid obstacles. Using this vector field as a control input for a quadrotor may be effective at low velocities, but as the commanded velocities increase, the quadrotor will deviate more and more from the desired path, eventually hitting the obstacle. Using our control method, the quadrotor will maneuver itself to precisely follow the vector field, even for high desired velocities.

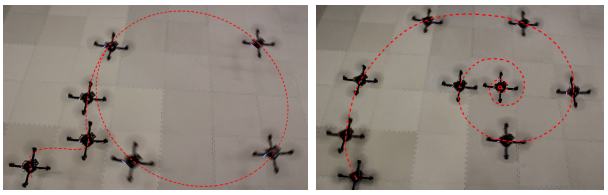


Fig. 1. A quadrotor flying to a limit cycle (left) and along a logarithmic spiral (right) using our proposed vector field following controller.

The chief difficulty lies in effectively compensating for the dynamics of the quadrotor so that it achieves the velocity commanded by the vector field. We do this by exploiting the differential flatness property of quadrotor dynamics. This property allows for the 12 dimensional state of the quadrotor to be specified analytically from a desired position and yaw angle, and their time derivatives. Furthermore, the required control inputs for the quadrotor can also be specified analytically in a similar way. Typically, differential flatness is used to plan *trajectories* for a quadrotor, and to give *open-loop* control inputs to achieve this trajectory (for example [10], [12]). Instead, we use differential flatness to produce a *closed-loop* feedback controller, that effectively causes the quadrotor to move along the flow of a given vector field as if it were a point particle, even in highly dynamic regimes.

We use the endogenous transformation for quadrotor dynamics described in [10], with some modifications, together with a low level SE(3) controller, also used in [10], and originally presented in [8]. The main contribution in our work is to derive the instantaneous velocity, acceleration, jerk, and snap required in the quadrotors endogenous transformation analytically from a given vector field. The vector field can be specified numerically with a given interpolation function between points, or it can be specified analytically. In either case, we require that up to third order spatial derivatives can be computed from the vector field description (for acceleration, jerk, and snap). From these derivatives we find the state of the quadrotor to achieve the velocity required by the vector field, and then control the quadrotor to that state

This work was supported in part by ONR grant N00014-12-1-1000. We are grateful for this support.

D. Zhou is with the Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA, zdj@bu.edu.

M. Schwager is with the Department of Mechanical Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215, USA, schwager@bu.edu.

with the low-level SE(3) controller. The overall architecture does not require any explicit trajectory planning or trajectory representation, and gives a reactive closed-loop controller. The control method is demonstrated in simulations and in experiments with a quadrotor robot flying along three different vector fields.

A. Related Work

Our control method renders many existing vector-field based control algorithms useful for quadrotors, which would otherwise be ineffective. For example, classical techniques that employ artificial potential fields and navigation functions for navigation to a goal [15], [4] can be used with our controller to navigate quadrotors. The foundational work on navigation functions [15] can be used with our method to navigate quadrotors. Also, algorithms for controlling linear dynamics in polytopic regions of a state space [4] have been used as building blocks for more complex control strategies [7], [1]. Our method can be used with [4] to control quadrotors through polytopic regions, extending these high level control techniques to quadrotors. Methods for obstacle avoidance also commonly generate vector fields to avoid obstacles, and these can now be used for quadrotors given our control method. For example, the seminal work [6] uses time varying artificial potentials that result in time varying velocity fields for a robot to avoid moving obstacles. These can be used directly with our control method for quadrotors. Similarly, the concept of reciprocal velocity obstacles [19] allows multiple robots to avoid collisions with one another, but relies on integrator dynamics of the robots (which is mathematically equivalent to giving a velocity vector field). Our method renders this approach useful for quadrotors as well.

In multi-robot control, it is particularly common to assume the robots have integrator dynamics. For example, well known control algorithms for flocking [5], herding [14], and consensus [13] assume either single or double integrator dynamics. Also, controllers for multi-agent sensor coverage and surveillance often assume integrator dynamics, as in [17], [2], [16], and controllers for connectivity maintenance in mobile wireless networks commonly use this assumption [21], [3]. Our method in this paper controls quadrotors so that they behave as single integrators, thus making all of these multi-agent control strategies directly applicable to quadrotors.

The rest of this paper is organized as follows. Differential flatness theory as it applies to quadrotor dynamics is formulated in Sec. II. The method for deriving a vector field following controller from a given vector field is described in Sec. III. Results of simulations and experiments are given in Sec. IV and Sec. V, respectively. We offer our conclusions in Sec. VI.

II. Differential Flatness for Quadrotors

In this section we define differential flatness, provide a model for the quadrotor dynamics, and we give a theorem that compactly states the endogenous transform for the quadrotor from the flat outputs (and their time derivatives) to the states and inputs. This material summarizes and formalizes the derivation from [10].

A. Differential Flatness

Differential flatness is a property of some nonlinear control systems that allows the state vector and the input vector to be written in terms of a smaller number of, so called, flat outputs, and some number of time derivatives of those outputs. The function that maps from the outputs and their time derivatives to the states and inputs is known as the endogenous transformation [12]. Quadrotor dynamics are known to be differentially flat. This is typically useful in trajectory planning, because a desired trajectory in the flat outputs can be used to analytically find the open-loop trajectory of the inputs required to reproduce the trajectory. In contrast, here we use differential flatness to find a closed-loop controller to drive a quadrotor as if it were a simple integrator traveling through a desired velocity vector field.

More formally, a nonlinear control system $\dot{\xi} = f(\xi, \mu)$ is called differentially flat if there exists an invertible function α such that

$$\sigma = \alpha(\xi, \mu, \dot{\mu} \cdots, \mu^{(p)}),$$

for some finite number of time derivatives p, where σ is called the flat output. Furthermore, the inverse of α gives the trajectories of ξ and μ as functions of the flat outputs and q time derivatives,

$$\begin{cases} \xi = \beta(\sigma, \dot{\sigma}, \cdots, \sigma^{(q)}), \\ \mu = \gamma(\sigma, \dot{\sigma}, \cdots, \sigma^{(q)}). \end{cases} \tag{1}$$

We call the two functions (β, γ) together the endogenous transformation. In this section we give the endogenous transformation for a quadrotor robot, with the position and yaw angle as flat outputs.

B. Quadrotor Dynamics

A quadrotor is well-modeled as a rigid body with forces and torques applied from the four rotors and gravity [10]. This is similar to the dynamics of a fixed wing aircraft, but without the aerodynamic forces and moments from the wings and stabilizers [18]. The relevant forces, moments, and coordinate frames are shown in Fig. 2.

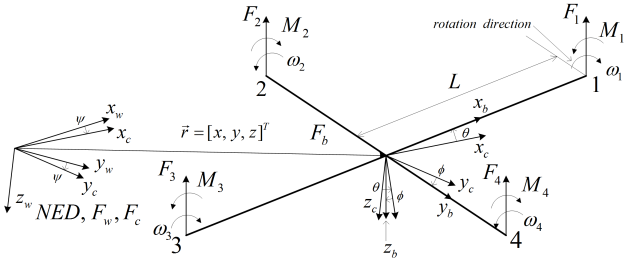


Fig. 2. Quadrotor coordinate frames with a North-East-Down world coordinate system. The world frame is denoted F_w, the aircraft body-fixed frame is F_b, and F_c is an intermediate frame after the yaw angle rotation.

We define the rotation matrix from the body frame to the world frame using ZYX Euler angles as

$$R = R_{z,\psi} R_{y',\theta} R_{x'',\phi} = \begin{bmatrix} C\theta C\psi & S\phi S\theta C\psi - C\phi S\psi & C\phi S\theta C\psi + S\phi S\psi \\ C\theta S\psi & S\phi S\theta S\psi + C\phi C\psi & C\phi S\theta S\psi - S\phi C\psi \\ -S\theta & S\phi C\theta & C\phi C\theta \end{bmatrix} \quad (2)$$

where ϕ is roll, θ is pitch, ψ is yaw, and $S\cdot$ is $\sin(\cdot)$, and $C\cdot$ is $\cos(\cdot)$ for simplicity. The quadrotor dynamics are given by the nonlinear system of equations

$$\dot{v} = ge_3 + \frac{1}{m} R f_z e_3 \quad (3)$$

$$\dot{R} = R\Omega \quad (4)$$

$$\dot{\omega}_b = J^{-1}\tau - J^{-1}\Omega J \omega_b \quad (5)$$

$$\dot{h} = v \quad (6)$$

where $v = [v_x, v_y, v_z]^T$ is the velocity in the world frame F_w, g is the acceleration due to gravity, m is the mass, f_z is the total thrust force from the rotors, $e_3 = [0, 0, 1]^T$, hence $f_z e_3$ is aligned with the negative vertical body-fixed direction $-z_b$ of the body frame F_b. The rotation matrix R is from (2), $\omega_b = [p, q, r]^T$ is the angular velocity of the quadrotor expressed in F_b, $\Omega = \omega_b^\wedge = [0, -r, q; r, 0, -p; -q, p, 0]$ is the tensor form of ω_b. The torque on the quadrotor is given by $\tau = [\tau_x, \tau_y, \tau_z]^T$ in the body frame F_b, J is the inertia matrix of the quadrotor, and $h = [x, y, z]^T$ is the position of the quadrotor in F_w. The quadrotor's inputs are the total thrust and the three torques $\mu = [f_z, \tau_x, \tau_y, \tau_z]^T$, and the system has a 12 dimensional state, $\xi = [x, y, z, v_x, v_y, v_z, \psi, \theta, \phi, p, q, r]^T$.

C. Endogenous Transformation

Consider the flat outputs consisting of position and yaw angle $\sigma = [\sigma_1, \sigma_2, \sigma_3, \sigma_4]^T = [x, y, z, \psi]^T$. In this section we give the functions $\beta(\cdot)$ and $\gamma(\cdot)$ such that, given $(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}, \ddddot{\sigma})$, we can find the associated state ξ and input μ. The following theorem summarizes the results of [10].

Theorem 1: The quadrotor dynamics (3)–(6), with state $\xi = [x, y, z, v_x, v_y, v_z, \psi, \theta, \phi, p, q, r]^T$ and input $\mu = [f_z, \tau_x, \tau_y, \tau_z]^T$, are differentially flat, with the flat outputs

$$\sigma = [\sigma_1, \sigma_2, \sigma_3, \sigma_4]^T := [x, y, z, \psi]^T, \quad (7)$$

such that $\xi = \beta(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma})$, with

$$\begin{cases} [x, y, z, v_x, v_y, v_z, \psi]^T = \beta_{1:7}(\sigma, \dot{\sigma}) \\ \qquad = [\sigma_1, \sigma_2, \sigma_3, \dot{\sigma}_1, \dot{\sigma}_2, \dot{\sigma}_3, \sigma_4]^T \\ \theta = \beta_8(\sigma, \dot{\sigma}, \ddot{\sigma}) = \text{atan2}(\beta_a, \beta_b) \\ \phi = \beta_9(\sigma, \dot{\sigma}, \ddot{\sigma}) = \text{atan2}(\beta_c, \sqrt{\beta_a^2 + \beta_b^2}) \\ [p, q, r]^T = \beta_{10:12}(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}) = (R^T \dot{R})^\vee, \end{cases} \quad (8)$$

where

$$\begin{cases} \beta_a = -\cos\sigma_4 \ddot{\sigma}_1 - \sin\sigma_4 \ddot{\sigma}_2 \\ \beta_b = -\ddot{\sigma}_3 + g \\ \beta_c = -\sin\sigma_4 \ddot{\sigma}_1 + \cos\sigma_4 \ddot{\sigma}_2, \end{cases} \quad (9)$$

and R is from (2) with the Euler angles (ϕ, θ) defined in (8). Furthermore, $\mu = \gamma(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}, \ddddot{\sigma})$, with

$$\begin{cases} f_z = \gamma_1(\sigma, \dot{\sigma}, \ddot{\sigma}) = -m \parallel \ddot{\sigma}_{1:3} - g e_3 \parallel \\ [\tau_x, \tau_y, \tau_z]^T = \gamma_{2:4}(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}, \ddddot{\sigma}) \\ \quad = J(\dot{R}^T \dot{R} + R^T \ddot{R})^\vee + R^T \dot{R} J (R^T \dot{R})^\vee, \end{cases} \quad (10)$$

where $\ddot{\sigma}_{1:3} = [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3]^T$ for short and the $^\vee$ map is the inverse operation of $^\wedge$.

Proof: (Abbreviated) The expression $\beta_{1:7}(\cdot)$ in (8) follows immediately from (7). Equation (3) shows that $z_b = -\frac{\ddot{\sigma}_{1:3} - g e_3}{\| \ddot{\sigma}_{1:3} - g e_3 \|}$ and $f_z = -m \parallel \ddot{\sigma}_{1:3} - g e_3 \parallel \equiv \gamma_1(\cdot)$, since $f_z e_3$ is aligned to z_b but in the opposite direction. Comparing z_b to the third column of rotation matrix R in (2) we can solve for $\cos\phi \sin\theta$, $\cos\phi\cos\theta$ and $\sin\phi$, which are β_a, β_b and β_c in (9), respectively, and find $\theta = \beta_8(\cdot)$ and $\phi = \beta_9(\cdot)$, as well as the rotation matrix R. Equation (4) shows that $\Omega = R^T \dot{R}$, hence $\omega_b = (R^T \dot{R})^\vee$. Finally, (5) gives $\tau = [p, q, r]^T = J\dot{\omega}_b + \Omega J \omega_b = J(\dot{R}^T \dot{R} + R^T \ddot{R})^\vee + R^T \dot{R} J (R^T \dot{R})^\vee \equiv \gamma_{2:4}(\cdot)$. ∎

The $^\vee$ map is feasible due to the following lemma.

Lemma 1: For all rotation matrices R, we have $R^T \dot{R}$ and $\dot{R}^T \dot{R} + R^T \ddot{R}$ are skew-symmetric matrices. In other words, $\forall R \in SO(3)$, $R^T \dot{R} \in so(3)$, and $\dot{R}^T \dot{R} + R^T \ddot{R} \in so(3)$.

Proof: Take the time derivative of $R^T R = I$ to get $\dot{R}^T R + R^T \dot{R} = 0$. Hence we have $(R^T \dot{R})^T + R^T \dot{R} = 0$, satisfying the $so(3)$ condition for $R^T \dot{R}$. Differentiate $(R^T \dot{R})^T + R^T \dot{R} = 0$ to get $((\dot{R}^T)\dot{R} + R^T \ddot{R})^T + ((\dot{R}^T)\dot{R} + R^T \ddot{R}) = (\dot{R}^T \dot{R} + R^T \ddot{R})^T + (\dot{R}^T \dot{R} + R^T \ddot{R}) = 0$, implies that $\dot{R}^T \dot{R} + R^T \ddot{R} \in so(3)$. ∎

III. Vector Field Following

In many scenarios, it is desirable for a robot to follow a given velocity vector field, for example for navigation, collision avoidance, connectivity maintenance, flocking, and herding applications. In this section, we describe our method for deriving the time derivatives of the flat outputs from a given velocity vector field. This is the main contribution of our work.

The key to our method is to find the required time derivatives $(\dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}, \ddddot{\sigma})$ by taking spatial derivatives of the vector field. With these time derivatives, the states and inputs of the quadrotor can be generated from the endogenous transformation as described in Theorem 1. The $SE(3)$ controller described in [10] and [8] is then used to stabilize the quadrotor along the required trajectory $\xi(t)$. Here we only consider vector fields specifying spatial velocity, hence the yaw angle σ_4 is irrelevant. We arbitrarily set $\sigma_4(t) \equiv 0$, and the expressions for the time derivatives of $\sigma_{1:3}(t)$ are computed with the following theorem.

Theorem 2: The flat output derivatives required to compute the endogenous transformation $\dot{\sigma}_{1:3}, \ddot{\sigma}_{1:3}, \dddot{\sigma}_{1:3}, \ddddot{\sigma}_{1:3}$ at any point x in a vector field $V(x)$ can be calculated by

$$\begin{cases} \dot{\sigma}_{1:3}(x) &= V(x) \\ \ddot{\sigma}_{1:3}(x) &= \mathcal{J}(\dot{\sigma}_{1:3}(x), x)\dot{\sigma}_{1:3}(x) \\ \dddot{\sigma}_{1:3}(x) &= \mathcal{J}(\ddot{\sigma}_{1:3}(x), x)\ddot{\sigma}_{1:3}(x) \\ \ddddot{\sigma}_{1:3}(x) &= \mathcal{J}(\dddot{\sigma}_{1:3}(x), x)\dddot{\sigma}_{1:3}(x), \end{cases} \quad (11)$$

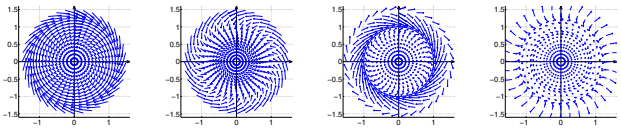
where $\mathcal{J}(f(x), x)$ denotes the Jacobian matrix of the function $f(x)$.

Proof: $\dot{\sigma}_{1:3}(x) = V(x)$ is true by definition. Then $\ddot{\sigma}_{1:3}(x) = \frac{d(\dot{\sigma}_{1:3}(x))}{dt} = \frac{\partial(\dot{\sigma}_{1:3}(x))}{\partial x}\frac{d(x(t))}{dt} = \mathcal{J}(\dot{\sigma}_{1:3}(x), x)\dot{\sigma}_{1:3}(x)$, and the procedure can be recursively applied to obtain the expressions for the third and forth time derivatives. ■

The method is applicable to 1D, 2D, or 3D vector fields, but we show examples only with 2D vector fields for clarity. For example, consider a velocity vector field given by the equation

$$V(x_1, x_2) = \begin{bmatrix} x_2 \\ -x_1 \end{bmatrix} - 0.5\sin(x_1^2 + x_2^2 - 1)\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad (12)$$

in which all trajectories converge to the limit cycle $x_1^2 + x_2^2 = 1$. Fig. 3 shows the velocity field (12) and its three time derivatives.



(a) Velocity (b) Acceleration (c) Jerk (d) Snap

Fig. 3. Vector field as in (12), with scaled vector length for plotting.

Using the endogenous transformation described in Sec. II, we then obtain the states and inputs for the quadrotor to be used as a control reference to control it along the desired vector field. Specifically, for the vector field in (12), the simulation in Sec. IV and the experiment in Sec. V will show that the trajectory of the quadrotor converges asymptotically to the limit cycle $x_1^2 + x_2^2 = 1$, with the origin be translated.

IV. Simulation

To illustrate the concept, in this section we consider simulation with a dynamical model of the quadrotor flying in three different velocity vector fields: 1), limit cycle; 2), logarithmic spiral and 3), obstacle avoidance. Sec. V gives the results of hardware experiments with a quadrotor executing our control method in the same three vector fields.

A. Limit Cycle

As in (12) and Fig. 3, the vector field is designed to cause the quadrotor to converge to a circle centered at $[0.9, 1.5, -0.5]^T$. The trajectory of the quadrotor is shown in Fig. 4, with the quadrotor image drawn at various time instants to give an indication of pitch and roll angles. The quadrotor does not actually enter the vector field until the third image. Before then, it is executing a planned maneuver to take off and enter the vector field with the appropriate velocity.

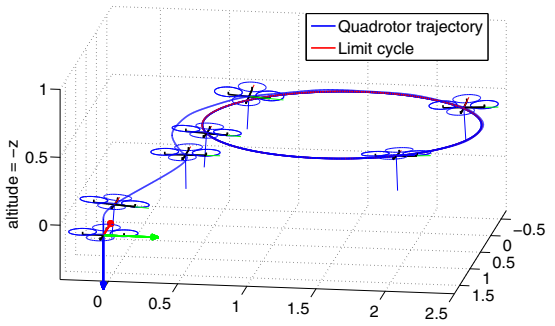


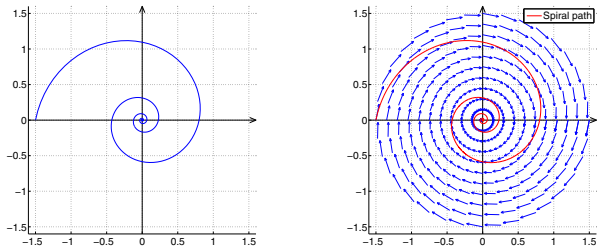
Fig. 4. The 2D vector field is the same with Fig. 3(a) but translated to $[0.9, 1.5, -0.5]^T$, with a constant altitude $0.5m$. The trajectory converges to the circle $(x-0.9)^2 + (y-1.5)^2 = 1$, with $z = -0.5$, as expected.

B. Logarithmic Spiral

The equation of a logarithmic spiral in polar coordinates is $r = ae^{-b\theta}, a, b > 0, \theta \in [0, \infty)$, with the corresponding vector field in Cartesian coordinates defined by

$$V(x) = \begin{bmatrix} x_2 \\ -x_1 \end{bmatrix} - b\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad (13)$$

as shown in Fig. 5. The trajectory of the quadrotor is shown in Fig. 6. Again, the first two images show a pre-planned take-off trajectory, and the quadrotor enters the vector field at the third image.



(a) Logarithmic spiral with $a = 1.5$, $b = 0.2$, and $\theta \to \infty$. (b) A path following the vector field as eq. (13) with $b = 0.2$.

Fig. 5. Starts from $[-1.5, 0]^T$ in the vector field, the path eventually spires to $[0, 0]^T$, showing the same trajectory as the logarithmic spiral.

C. Obstacle Avoidance

In this scenario, the quadrotor travels to a destination point, with an obstacle in its path. The destination generates an attracting field to pull the quadrotor to it, while the obstacle generates a repelling field to push the quadrotor away.

The attracting and repelling vector fields in Fig. 7 are formulated from the gradient of a 2D Gaussian $G(x) = Ae^{(-(x-x_0)^T(x-x_0))/2\sigma^2}$. Taking the gradient, the attracting vector field of the destination at x_a is defined as $v_a(x) = \frac{G(x)}{\sigma^2}(x_a - x), \forall x$. And the repelling vector field of the obstacle at x_r is defined as $v_r(x) = \frac{G(x)}{\|x - x_r\|}(x - x_r), \forall x$, such that $\| x - x_r \| \geq r_{ob}$, where r_{ob} is the minimum

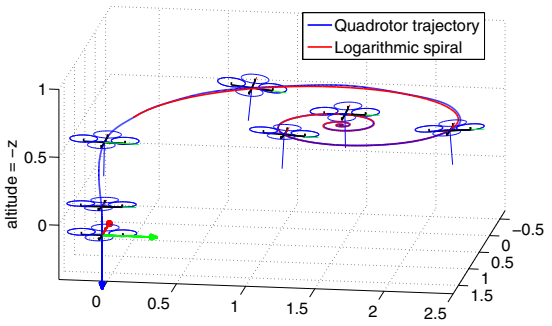


Fig. 6. The quadrotor starts from $[0.9, 0, -0.5]^T$ to follow the vector field, and spire to the center $[0.9, 1.5, -0.5]^T$, to which the vector field is translated. The quadrotor achieves a trajectory the same with the logarithmic spiral as in Fig. 5.

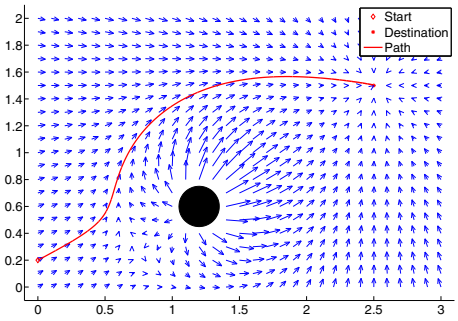


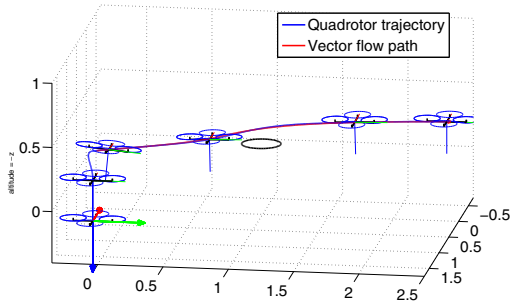
Fig. 7. A composite 2D vector field of an attracting field from the destination at $[2.5, 1.5]^T$ and a repelling field from the obstacle at $[1.2, 0.6]^T$. The path shows the expected trajectory of a robot starts from the point $[0, 0.2]^T$, and then follow the vector field to the destination, where the vector vanishes to zero.

safety distance to the center of the obstacle. Fig. 8 shows the simulation results.

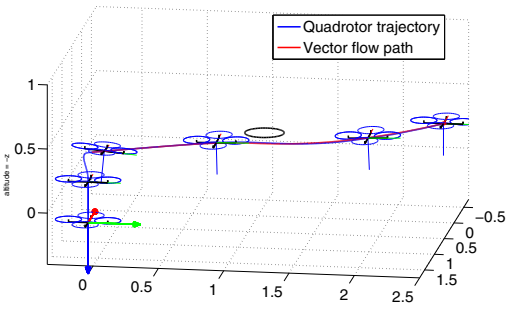
V. Experiments

Hardware experiments were conducted with a K500 quadrotor, from KMel Robotics (`http://kmelrobotics.com/`). The position and orientation of the quadrotor are observed by an OptiTrack motion capture system (`http://www.naturalpoint.com/optitrack/`) at the default update rate 120Hz and obtained by MATLAB via Java scripts reading UDP packets from the OptiTrack. The velocity and angular velocity are computed by numerical differentiation from the position and Euler angles, as well as with an Extended Kalman Filter (EKF) in MATLAB.

The UDP packet parsing, the EKF algorithm, and the $SE(3)$ controller are implemented in C language and are called by MATLAB via the MEX interface to speed up the calculation, making it possible to run the algorithm online without degrading the OptiTrack update rate, even when flying multiple quadrotors. Meanwhile, the K500 receives the desired thrust, Euler angles and Euler angles' rate as the inputs, and executes its onboard $SO(3)$ controller based on



(a) The virtual obstacle is at $[0.6, 1.2, -0.5]^T$, the quadrotor pass it from left.



(b) The virtual obstacle is at $[1.0, 1.2, -0.5]^T$, the quadrotor pass it from right.

Fig. 8. The quadrotor avoids the obstacle with different trajectories when the virtual obstacles (black circle) are at different positions.

the onboard IMU data at a 500Hz update rate. The control framework is shown in Fig. 9, which is a typical feedback control system.

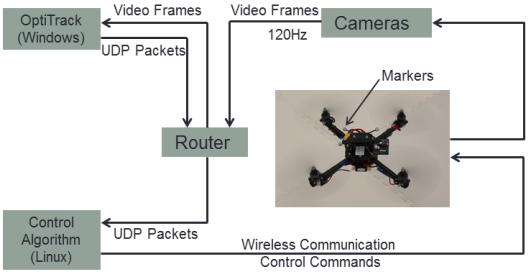


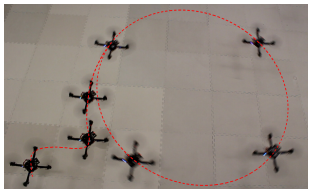
Fig. 9. System diagram, the markers are used for tracking.

All experiments were conducted corresponding to the same setup as in the simulations in Sec. IV. As in the simulations, the quadrotor takes off and flies into the vector field by following a predefined trajectory, which is generated from spline curves [20],[12]. The final state of this trajectory is designed to match the vector field so that the quadrotor can transition smoothly into from the trajectory following to the vector field following control.

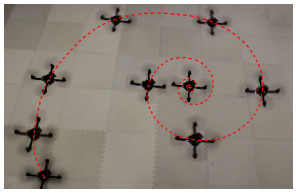
A. Limit Cycle

The quadrotor first flies to $[0.5, 0.5, -0.5]^T$, which is outside of the limit cycle $(x-0.9)^2 + (y-1.5)^2 = 1$, with

a predefined trajectory, then in the vector field following, it converges to the limit cycle as expected, with a small position error. Along the cycle, the quadrotor flies at a constant speed $1.4m/s$, as in Fig. 10(a).



(a) Composite image of a single quadrotor flying a limit cycle, http://tinyurl.com/qbus6c7.



(b) Composite image of a single quadrotor flying a logarithmic spiral, http://tinyurl.com/kx2meq4.

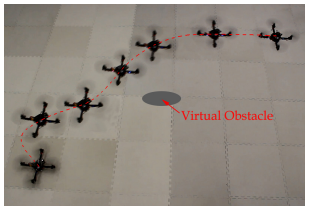
Fig. 10. A single quadrotor flying a limit cycle and a logarithmic spiral path at a constant altitude $0.5m$.

B. Logarithmic Spiral

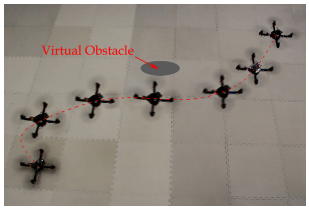
The quadrotor first flies to $[0.9, 0, -0.5]^T$ where it enters the vector field, as shown in Fig. 10(b). The maximum speed in this case is $1.4m/s$, and the quadrotor quickly slows down when approaching the center of the logarithmic spiral.

C. Obstacle Avoidance

We present two experiments to demonstrate the obstacle avoidance with the vector field following strategy. In each experiment, the virtual obstacle is presented at a different position. In the vector field, the quadrotor flies from the same start position at $[0.2, 0, -0.5]^T$ to the same destination at $[1.5, 2.5, -0.5]^T$.



(a) The virtual obstacle is at $[0.6, 1.2, -0.5]^T$, http://tinyurl.com/o475fna.



(b) The virtual obstacle is at $[1.0, 1.2, -0.5]^T$, http://tinyurl.com/qdmucrm.

Fig. 11. Composite image of a single quadrotor avoiding an obstacle at different positions.

VI. Conclusions and Future Work

In this paper we persent a method for controlling a quadrotor robot along a desired vector field. This is difficult because quadrotors have complex nonlinear dynamics. It is useful because many algorithms for robot navigation, collision avoidance, surveillance, flocking, and herding give a vector field for a robot to follow to accomplish its task. Our control method leverages the tools of differential flatness, and obtains the desired velocity, acceleration, jerk, and snap of the quadrotor directly from the vector field. The method is demonstrated in simulations and experiments with three different vector fields. In the future we will consider the yaw angle and its derivatives to the second order from the vector field, and explore extensions to multiple quadrotors interacting with vector fields. Also, The conditions of the vector field under which the quadrotor dynamics can satisfy should be studied.

References

[1] Nora Ayanian and Vijay Kumar. Decentralized feedback controllers for multi-agent teams in environments with obstacles. *IEEE Transactions on Robotics*, 26(5):878 – 887, October 2010.

[2] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, April 2004.

[3] D. V. Dimarogonas and K. H. Johansson. Decentralized connectivity maintenance in mobile networks with bounded inputs. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1507–1512, 2008.

[4] L. Habets and J. H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40(1):21–35, 2004.

[5] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[6] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[7] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1):287–297, 2008.

[8] Taeyoung Lee, M Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425. IEEE, 2010.

[9] P. Martin, R. M. Murray, and P. Rouchon. Flat systems, equivalence, and trajectory generation. Technical report, California Institute of Technology, Control and Dynamical Systems, April 2003.

[10] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[11] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The grasp multiple micro-uav testbed. *Robotics & Automation Magazine, IEEE*, 17(3):56–65, 2010.

[12] Richard M Murray. Optimization-based control. *California Institute of Technology, CA*, 2009.

[13] R. Olfati-Saber and R. R. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, September 2004.

[14] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21:25–34, 1987.

[15] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[16] M. Schwager, B. Julian, M. Angermann, and D. Rus. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. *Proceedings of the IEEE*, 99(9):1541–1561, September 2011.

[17] M. Schwager, D. Rus, and J. J. Slotine. Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment. *International Journal of Robotics Research*, 30(3):371–383, March 2011.

[18] Brian L Stevens and Frank L Lewis. Aircraft control and simulation. 2003.

[19] J. Van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *International Conference on Robotics and Automation (ICRA)*, pages 1928–1935, 2008.

[20] Fujio Yamaguchi and Fujio Yamaguchi. *Curves and surfaces in computer aided geometric design*. Springer-Verlag Berlin, 1988.

[21] M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4):812–816, August 2007.