Model Predictive Control of a Quadrotor on Manifold for Reference Trajectory Tracking

James Jackson

April 18, 2018

Abstract

While the capability of MAVs has grown significantly in recent years, we have yet to reach human performance in control of multirotor MAVs. Model Predictive Control has been proposed as one way to acheive human-level performance and has been demonstrated in a variety of cases with multirotors, however previous approaches have suffered from difficulties in the non-vector nature of the rotation group $SO(3)$. In this work, we show how to apply linear model predictive control of a quadrotor on the $SO(3)$ manifold by performing control on error state between some reference trajectory and the current state. This parameterization allows to perform direct control on the lie algebra associated with $SO(3)$, and provides a principled method to deal with the rotation group. First, the error-state dynamics are explained, then their application to L-MPC and a simple simulation study is used to show the effectiveness of the approach.

Figure 1: Quadcopter Used for Modeling

1 Introduction

In the last decade, we have seen the emergence of miniature aerial vehicles (MAVs) in everyday life for many industries. The price and weight of processing power, sensors, and motors have decreased dramatically in recent years and has resulted in modern MAVs being incredibly agile and acrobatic. To highlight this point, the sport of quadcopter racing among MAV enthusiats has seen regular top speeds exceeding 100 kph.

The most skilled human operators provide angular rate and throttle commands to an embedded flight controller at a rate of 50Hz using a radio transmitter. Many of these operators receive feedback from the multirotor only from a radio video link transmitted from the multirotor. From a state estimation and control persepective, the abilities of human operators of MAVs are truly incredible and we have yet to see equivalent performance by an autonomous MAV. This is due to a variety of factors including limitations in state estimation, path planning, and control. In this work, I hope to apply linear model predictive control (MPC) to a quadcopter to perform waypoint following, as one step towards reaching human performance in MAV control.

There are many existing approaches to performing MPC on quadrotors, but most of these rely on a flight controller to perform higher-level attitude control as opposed to the lower-level angular rates given by skilled human operators [1, 2, 3, 4, 5, 6]. Providing angular rate commands, as opposed to full attitude control allows for acrobatic maneuvers, outside the limitations of the flight controller attitude control scheme. Many of these approaches developed previously rely on an euler-angle decomposition [1, 2, 3, 4, 5, 6], of the attitude or ignored attitude dynamics [7, 8] to avoid dealing with the non-vector nature of rotations. Aswani et al. fixed the heading angle of their quadrotor [9] and could therefore perform acrobatics, but were limited in the types of trajecotories available.

While euler angles are not a vector space, they approximate a vector space given a linearization. However, they suffer from a singularity when pitched up to 90 degrees, suffer from linearization error far from level flight, and they require some book-keeping to deal with wrapped angles. Therefore, most exist-

ing MPC problems do not actually perform acrobatic flight where they would have to leave the linearization point used to cast euler angles into a vector space.

What I propose is to use a unit quaternion to represent attitude, and formulate the optimization problem of MPC in the Lie algebra associated with $SO(3)$ to smoothly optimize on the $SO(3)$ manifold. MPC schemes on manifolds are not new, and were shown in [10], but to my knowledge have not been applied to quadrotors.

1.1 Nomenclature

$p^I_{b/I}$: position of MAV body frame with respect to inertial frame, expressed in the inertial frame

$v^b_{b/I}$: velocity of MAV body frame with respect to inertial frame, expressed in the body frame

q^b_I : rotation from inertial to body frame, expressed in the inertial frame as a unit quaternion

$\omega^b_{b/I}$: angular velocity of MAV in body frame

T^b : Total thrust of quadrotor (in body frame)

s : "Throttle signal" between 0 and 1 which is used by a flight controller to control T^b

$\lfloor \cdot \rfloor$: The skew-symmetric operator

$\mathbf{i}, \mathbf{j}, \mathbf{k}$: The orthonormal unit vectors which describe the x, y, and z axes of a standard coordinate frame

For this I will be modeling the quadrotor MAV shown in Figure 1. This multirotor weighs approximately 14 oz with a battery and measures 220mm diagonally from rotor center to rotor center. I will be using 5" 3-blade propellers and appropriately sized motors, battery and speed controllers.

2 Derivation

2.1 Multirotor Dynamics

The free-body diagram in Figure 2 can be used to derive our model for the dynamics of a quadrotor with a linearized drag constant term μ, mass M and gravity constant g, given in Equation 1, and shown in [11, 12]

$$\begin{aligned} \dot{p}^I_{b/I} &= R\left(q^b_I\right)^\top v^b_{b/I} \\ \dot{v}^b_{b/I} &= R\left(q^b_I\right) g^I - \frac{T^b}{M} - \mu v^b_{b/I} \\ \dot{q}^b_I &= \omega^b_{b/I}. \end{aligned} \tag{1}$$

Both because the dynamics of a quadrotor of this size are extremely fast, and to mimic the operators

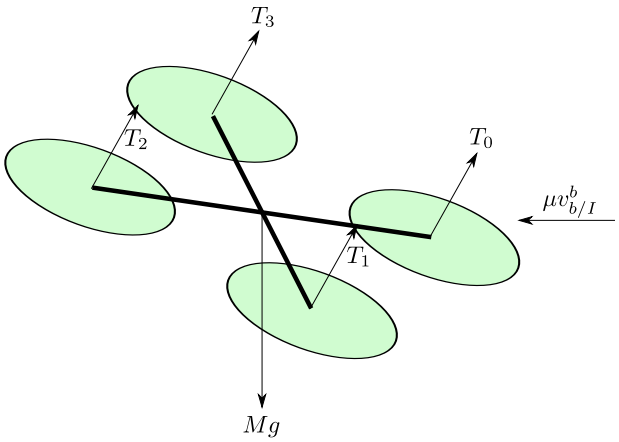

Figure 2: Forces acting on Quadcopter

available to a human operator, we will command throttle signal and angular rates, rather than actual forces and torques.

Unfortunately, we cannot command thrust directly, as the flight controller does not know the motor model. Instead, we are able to command a "throttle" value between 0 and 1, with 1 being maximum thrust and 0 being none, but a non-linear mapping between throttle and thrust in between. To obtain this mapping, I collected thrust and torque data on a test stand and fitted an approximate model. The relationship between the throttle signal s and thrust T_i shows a strong quadratic trend with almost no transient response and is shown in Figure 3.

The thrust from each motor (and therefore the total thrust) is somewhat easy to calculate. I just used a least-squares quadratic fit from the data shown in figure 3. The coefficients of this quadratic fit are shown in Table 1.

$$T_i = \left(a_T s_i^2 + b_T s + c_T\right)\mathbf{k} \tag{2}$$

The total thrust for all four motors is then

$$T^b = \sum_{i=0}^{4} T_i.$$

Torque is slightly more complicated, as we must make use of the geometry of each arm. Let $r^b_{i/b}$ be the vector which describes the location of motor i with respect to the body in the body frame. There are two components of torque from each motor, the first is due primarily to drag of the propeller, and is fitted with a quadratic fit from the data shown in Figure 3. The second is due to the thrust of the motor acting on the lever arm

$$\tau_i = \left\lfloor r^b_{i/b} \right\rfloor T_i + \left(a_\tau s_i^2 + b_\tau s + c_\tau\right)\mathbf{k}.$$

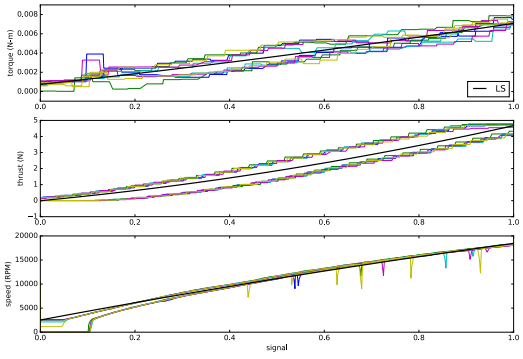

Figure 3: Experimental data from thrust and torque stand. It is unclear what causes what appears to be hysteresis in the torque and thrust plots. The two main paths are the sweep up (higher) and sweep down (lower). The reason this difference is unclear is because the speed plot does not have the same hysteresis, and therefore is unlikely to be due to transience in motor speed response or actual hysteresis in the motor. One hypothesis is that the mass of air in the test stand environment introduces a lag in torque readings, as thrust and torque are higher in static air environments. On a downward sweep, the torque and thrust are less, because the air is already moving.

The total torque is simply calculated as the sum of torques from the four motors

$$\tau^b = \sum_{i=0}^{4} \tau_i.$$

As mentioned before, we could, in theory, use this model to control torque directly, however, like a human operator, we are going to rely on the low-level flight controller to perform angular rate control at much higher rates. It is unreasonable to directly control the angular velocity of a quadcopter of this size with MPC due to computational and communication limitations. Instead, we will convert T^b to a desired throttle signal s, assuming that all motors are commanded the same signal (the case that the aircraft is in hover).

To get s, we simply take Equation 2 and solve for s.

$$\begin{aligned} T^b &= 4\left(a_T s^2 + b_T s + c\right) \\ 0 &= a_T s^2 + b_T s + c - \frac{T^b}{4} \\ s &= \frac{-b_T + \sqrt{b_T^2 - 4a_T\left(c_T - \frac{T^b}{4}\right)}}{2a_T} \end{aligned}$$

This calculation will return a valid signal, so long as

$$\begin{aligned} b_T^2 - 4a_T\left(c_T - \frac{T^b}{4}\right) &> 0 \\ \frac{-b_T^2}{a_T} + 4c_T &< T^b \end{aligned}$$

which in this case, using the values in Table 1 leads us to a lower-bound on T^b for a valid signal of

$$-4.1867 \quad < \quad T^b.$$

This is not a concern for us, as this equates a value of s outside of valid bounds, which are $[0, 1]$

	τ	T
a	0.0010008	1.87953563
b	0.00533861	2.79829004
c	0.00074126	-0.00514948

Table 1: Coefficients to quadratic approximation to motor response for torque and thrust found using a least-squares regression of test data.

If we substitute this expression for throttle control into the thrust term of our dynamics (Eq. 1), we get the following expression for our quadcopter dynamics:

$$\begin{aligned} \dot{p}^I_{b/I} &= R\left(q^b_I\right)^\top v^b_{b/I} \\ \dot{v}^b_{b/I} &= R\left(q^b_I\right) g^I - \frac{4}{M}\left(a_T s^2 + b_T s + c\right)\mathbf{k} - \mu v^b_{b/I} \\ \dot{q}^b_I &= \omega^b_{b/I}, \end{aligned} \tag{3}$$

and we can define our state and inputs as

$$\mathbf{x} = \left[p^I_{b/I}, v^b_{b/I}, q^b_I\right]^\top \qquad \mathbf{u} = \left[s, \omega^b_{b/I}\right]$$

2.2 Error State Tracking

At this point we must face the difficulty of dealing with the quaternion attitude object. Quaternions are not vectors, and instead form a group. (It should also be mentioned that there is no sensible vector representation of attitude, due to the nature of $SO(3)$).

The large majority of nonlinear optimization solvers make the assumption that the optimization variables are vectors, and therefore the optimization algorithm is allowed to

However, if we use the Lie algebra associated with $SO(3)$, we can perform our control in the vector space of the difference between quaternions. However, we have to re-cast our problem into an error-state formulation and perform control over the error between a reference trajectory and our current state.

In this work, our desired trajectory, $\mathbf{x}_r$, is a series of piecewise-constant waypoints in the form of a 2 meter by 2 meter square, located 2 meters above the ground. This trajectory is not dynamically feasible (as it would require instant teleportation between waypoints) but is easy to calculate. It would be much better to generate dynamically feasible methods such as trajectory optimization [13, 14, 15], or approaches which leverage the differential flatness of the multirotor [16, 8]. This is outside the scope of this work, however trying to track non-dynamically feasible trajectories actually exercises better the effectiveness of MPC, but better trajectories will result in better overall performance.

The reference trajectory dynamics are the same as our normal quadrotor dynamics,

$$\begin{aligned}
\dot{p}^I_{b/I,r} &= R\left(q^b_{I,r}\right)^\top v^b_{b/I,r} \\
\dot{v}^b_{b/I,r} &= R\left(q^b_{I,r}\right) g^I - \frac{4}{M}\left(a_T s_r^2 + b_T s_r + c\right)\mathbf{k} - \mu v^b_{b/I,r} \\
\dot{q}^b_{I,r} &= \omega^b_{b/I,r}
\end{aligned}$$

and in our peicewise constant waypoints, the reference inputs are the just the equilibrium throttle to remain at hover.

$$\mathbf{u}_r = \begin{bmatrix} \frac{-b_T + \sqrt{b_T^2 - 4a_T\left(c_T - \frac{Mg}{4}\right)}}{2a_T} \\ 0 \end{bmatrix}.$$

Given a reference trajectory $\mathbf{x}$, Let us define the error to our trajectory as $\tilde{\mathbf{x}}$.

$$\tilde{\mathbf{x}} = \mathbf{x}_r \boxminus \mathbf{x}$$

where $\boxminus$ and $\boxplus$ are the generalized subtraction and addition operator defined by [17] which allows us to notationally deal with the quaternion manifold object as if it were a vector. We then can define the dynamics of $\tilde{\mathbf{x}}$, using error state dynamics described in [18] (dropping second-order terms). We also will be notationally dropping frame dependence to simplify syntax (from now on, frames can be inferred unless explicitly described.)

$$\begin{aligned}
\dot{\tilde{p}} &= \dot{p}_r - \dot{p} \\
&= R(q_r)^\top v_r - R(q)^\top v \\
&= R(q \boxplus \tilde{q})^\top (v + \tilde{v}) - R(q)^\top v \\
&\approx R(q)^\top (I + \lfloor \tilde{q} \rfloor)(v + \tilde{v}) - R(q)^\top v \\
&= R(q)^\top \lfloor \tilde{q} \rfloor (v + \tilde{v}) + R(q)^\top (v + \tilde{v}) - R(q)^\top v \\
&= -R(q)^\top \lfloor v \rfloor \tilde{q} + R(q)^\top \lfloor \tilde{q} \rfloor \tilde{v} + R(q)^\top \tilde{v} \\
&\approx -R(q)^\top \lfloor v \rfloor \tilde{q} + R^\top(q)\tilde{v}
\end{aligned}$$

$$\begin{aligned}
\dot{\tilde{v}} &= \dot{v}_r - \dot{v} \\
&= R(q_r) g^I - \frac{4}{M}\left(a_T s_r^2 + b_T s_r + c\right)\mathbf{k} - \mu v_r \\
&\quad -R(q) g + \frac{4}{M}\left(a_T s^2 + b_T s + c\right)\mathbf{k} + \mu v \\
&= R(q \boxplus \tilde{q}) g - R(q) g \\
&\quad -\frac{4}{M}\left(a_T\left((s+\tilde{s})^2 - s^2\right) + b_T \tilde{s}\right)\mathbf{k} - \mu\tilde{v} \\
&\approx (I - \lfloor \tilde{q} \rfloor) R(q) g - R(q) g \\
&\quad -\frac{4}{M}\left(a_T\left(2s\tilde{s} + \tilde{s}^2\right) + b_T\tilde{s}\right)\mathbf{k} - \mu\tilde{v} \\
&= -\lfloor \tilde{q} \rfloor R(q) g - \frac{4}{M}\left(2a_T s\tilde{s} + a_T\tilde{s}^2 + b_T\tilde{s}\right)\mathbf{k} - \mu\tilde{v} \\
&= \lfloor R(q) g \rfloor \tilde{q} - \frac{4}{M}\left(a_T\tilde{s}^2 + (2a_T s + b_T)\tilde{s}\right)\mathbf{k} - \mu\tilde{v}
\end{aligned}$$

$$\begin{aligned}
\dot{\tilde{q}} &= \dot{q}_r - \dot{q} \\
&= \tilde{\omega}
\end{aligned}$$

In summary, our error-state dynamics are

$$\begin{aligned}
\dot{\tilde{p}} &\approx -R(q)^\top \lfloor v \rfloor \tilde{q} + R(q)^\top \tilde{v} \\
\dot{\tilde{v}} &= \lfloor R(q) g \rfloor \tilde{q} - \frac{4}{M}\left(a_T\tilde{s}^2 + (2a_T s + b_T)\tilde{s}\right)\mathbf{k} - \mu\tilde{v} \\
\dot{\tilde{q}} &= \tilde{\omega} \qquad (4)
\end{aligned}$$

and input for our error-state system will be

$$\tilde{\mathbf{u}} = [\tilde{s}, \tilde{\omega}]^\top$$

where, as before

$$\tilde{\mathbf{u}} = \mathbf{u}_r - \mathbf{u}.$$

In performing L-MPC, we will make frequent use of the Jacobians of Equation 4 with $A = \frac{\partial f}{\partial \tilde{\mathbf{x}}}$ and $B = \frac{\partial f}{\partial \tilde{\mathbf{u}}}$.

$$A = \begin{bmatrix} 0 & R(q)^\top & -R(q)^\top \lfloor v \rfloor \\ 0 & -\mu I & \lfloor R(q) g \rfloor \\ 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ -\frac{4}{M}\left(2a_T\tilde{s} + 2a_T s + b_T\right)\mathbf{k} & 0 \\ 0 & I \end{bmatrix}$$

2.3 Model Predictive Control

We will implementat Linear Model Predictive Control (L-MPC) with a finite horizon of N timesteps. In this problem we wish to solve

$$\begin{aligned} \min \quad & J = \sum_{i=t}^{N} \tilde{\mathbf{x}}[i]^\top Q \tilde{\mathbf{x}}[i] + \tilde{\mathbf{u}}[i]^\top R[i]\,\tilde{\mathbf{u}}[i] \\ \text{s.t.} \quad & \tilde{\mathbf{x}}[i+1] = \tilde{\mathbf{x}}[i] + dt\left(A|_{\mathbf{x}[t]}\tilde{\mathbf{x}}[i] + B|_{\mathbf{u}[t],\tilde{\mathbf{u}}[t]}\tilde{\mathbf{u}}[i]\right) \\ & \begin{bmatrix} 0 \\ -\omega_{\max} \end{bmatrix} < \tilde{\mathbf{u}} < \begin{bmatrix} 1 \\ \omega_{\max} \end{bmatrix}. \end{aligned}$$

3 Implementation and Results

We used the python convex optimization library [19] to solve the MPC problem. Human operators of MAVs typically provide control to MAVs at 50 Hz, therefore, simulation was performed with a timestep of 0.02s. To strike a balance between computational load and robustness, the optimization time horizon was set to 10 timesteps, or 0.5 seconds.

The optimization cost gains and input constraints on angular velocity were set as

$$Q = \text{diag}\left(\begin{bmatrix} 10 & 10 & 10 & 1 & 1 & 1 & 0.01 & 0.01 & 0.01 \end{bmatrix}\right)$$

$$R = \text{diag}\left(\begin{bmatrix} 0 & 0.001 & 0.001 & 0.001 \end{bmatrix}\right)$$

$$\omega_{\max} = \begin{bmatrix} 6\pi \\ 6\pi \\ 3\pi \end{bmatrix}$$

and the MAV was given 2 seconds to reach each waypoint before the reference trajectory was changed to a new waypoint. For this model, $\mu = 0.2$. No dynamics were simulated for the flight controller's response to angular velocity inputs. In reality, there is some transient response to angular velocity inputs, which would slow down the response of all other higher-level control. This was not simulated in this project, but certainly could be considered in future implementations.

Simulation, plotting and solving the MPC problem ran at about 0.2x real time on a laptop computer. Results are shown in Figures 4, 5, 6 and 7

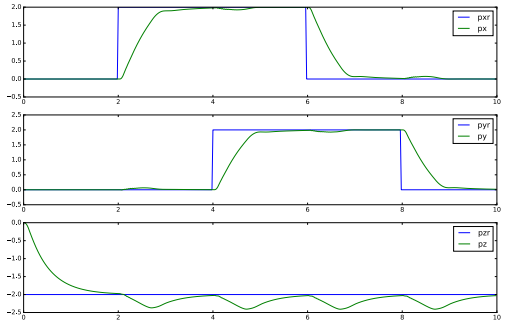

Figure 4: Position vs reference response

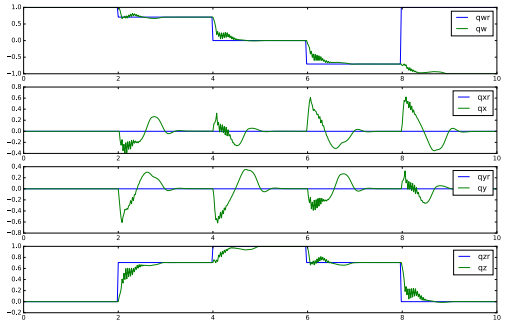

Figure 5: Attitude response vs reference - note that quaternions provide double coverage of $SO(3)$, therefore the resultant state is actually the same, despite the response converging to $-q_{w,r}$

4 Discussion

This implementation in its current form is limited, however, the approach accurately deals with the quaternion manifold without relying on linearizing assumptions, or euler-angle decompositions which suffer from gimbal lock. Due to the lack of dynamics on angular velocity inputs, the response saturates the ω input at almost every opportunity. This is likely due to an over-correction of linearization error, and the lack of transient dynamics make this possible. In reality, the response to this type of behavior be much smoother, because ω would not be allowed to change instantly.

Considering these drawbacks, the response is ideal. The cool thing about MPC is the fact that we can impose constraints on our inputs (and anything else for that matter) - however we pay for it in computation time. My python implementation was only able to run at about one-fifth real time, which is obviously

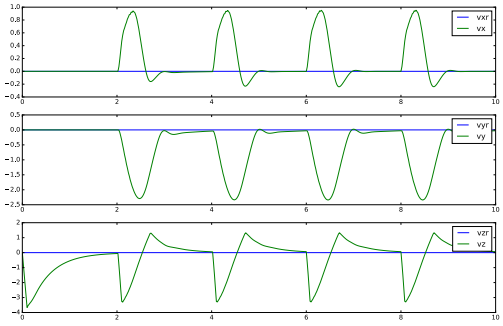

Figure 6: Velocity response of vs reference

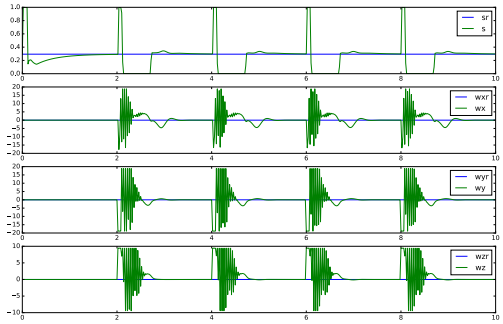

Figure 7: Inputs from MPC controller

insufficient for hardware experiments. However, the python implementation was not optimized and can be considered a prototype. An optimized implementation, without plotting routines would very likely exceed real-time requirements.

5 Conclusions

This report derived the error state dynamics for a quadrotor so that we could perform MPC on the $SO(3)$ manifold. This method is extensible to any reference trajectory and should operate at real time given a more efficient implementation. It performed well in simulation, and behaved in an explainable manner given the limitations of the simulation framework, such as exploiting the lack of transient response in ω inputs. A better simulation model would better show the response of this controller to such realities.

Future work would include generating smooth feasible trajectories (instead of static waypoints) which could include potentially inverted flight sections. This would exercise the strength of the manifold parameterization because there are no singularities, as opposed to the more common euler-angle based approaches. Further work could be to improve the modeling of low-level flight controller dynamics, and exercise the response to disturbances. These additions would provide much more realistic response in simulation. A final obvious extension would be to implement the system in a hardware experiment.

References

[1] P. Ru and K. Subbarao, "Nonlinear Model Predictive Control for Unmanned Aerial Vehicles," *Aerospace*, vol. 4, no. 2, p. 31, 2017.

[2] P. N. Chikasha and C. Dube, "Adaptive Model Predictive Control of a Quadrotor," *IFAC-PapersOnLine*, vol. 50, no. 2, pp. 157–162, 2017.

[3] G. V. Raffo, M. G. Ortega, and F. R. Rubio, *MPC with nonlinear H-infinity control for path tracking of a quad-rotor helicopter*, vol. 17. IFAC, 2008.

[4] M. Kamel, M. Burri, and R. Siegwart, "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[5] C. Liu, H. Lu, and W. H. Chen, "An explicit MPC for quadrotor trajectory tracking," *Chinese Control Conference, CCC*, vol. 2015-September, pp. 4055–4060, 2015.

[6] G. Ganga and M. M. Dharmana, "MPC Controller for Trajectory Tracking Control of Quadcopter," in *MPC Controller for Trajectory Tracking Control of Quadcopter*, 2017.

[7] M. Bangura and R. Mahony, *Real-time model predictive control for quadrotors*, vol. 19. IFAC, 2014.

[8] M. Greeff and A. P. Schoellig, "Model Predictive Path-Following for Constrained Differentially Flat Systems," 2017.

[9] A. Aswani, P. Bouffard, and C. Tomlin, "Extensions of Learning-Based Model Predictive Control for Real-Time Application to a Quadrotor Helicopter," *Proc American Control Conference ACC to appear*, pp. 4661–4666, 2012.

[10] U. Kalabić, R. Gupta, S. Di Cairano, A. Bloch, and I. Kolmanovsky, "MPC on manifolds with an application to SE(3)," *Proceedings of the American Control Conference*, vol. 2016-July, no. 3, pp. 7–12, 2016.

[11] R. C. Leishman, J. C. MacDonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems*, vol. 34, no. 1, pp. 28–41, 2014.

[12] D. O. Wheeler and D. P. Koch, "Relative Navigation : A Keyframe-Based Approach for Observable GPS-Degraded Navigation," 2017.

[13] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[14] M. Posa and R. Tedrake, "Direct Trajectory Optimization of Rigid Body Dynamical Systems Through Contact," *Algorithmic Foundations of Robotics X*, vol. 86, pp. 527–542, 2013.

[15] J. Pajarinen, V. Kyrki, M. Koval, S. Srinivasa, J. Peters, and G. Neumann, "Hybrid control trajectory optimization under uncertainty," 2017.

[16] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2688–2693, 2011.

[17] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[18] D. P. Koch and D. O. Wheeler, "Relative Multiplicative Extended Kalman Filter for Observable GPS-Denied Navigation," 2017.

[19] S. Diamond and S. Boyd, "CVXPY: A Python-Embedded Modeling Language for Convex Optimization," vol. 17, pp. 1–5, 2016.