Learning-Based Model Predictive Control on a Quadrotor: Onboard Implementation and Experimental Results

Patrick Bouffard, Anil Aswani, and Claire Tomlin

***Abstract*— This paper presents details of the real time implementation onboard a quadrotor helicopter of learning-based model predictive control (LBMPC). LBMPC rigorously combines statistical learning with control engineering, while providing levels of guarantees about safety, robustness, and convergence. Experimental results show that LBMPC can learn physically based updates such as the ground effect to an assumed model, and how as a result LBMPC improves transient response performance. We demonstrate robustness to mis-learning. Finally, we demonstrate the use of LBMPC in an integrated robotic task demonstration. The quadrotor is used to catch a ball thrown with an *a priori* unknown trajectory.**

I. Introduction

There has been interest in the use of small unmanned aerial vehicles (UAVs) for security, surveillance/sensor networks [1], and search-and-rescue [2] applications, and such vehicles have even seen use in the recent rebel uprising in Libya [3]. Due to these applications, simplicity of mechanical design and maintenance, and desireable safety characteristics, quadrotor helicopter UAVs are a popular choice among researchers in control and robotics ([4], [5], [6], [7], [8]).

Recent results in the applications of learning techniques to robotic systems (e.g., [9], [10]) suggests exploring how they might integrate with control techniques; indeed this is an active area of research [11]. Learning-Based Model Predictive Control (LBMPC) [12] is a new model-based control strategy that also allows for online updates to the model to improve performance, while maintaining certain guarantees about safety, robustness, and convergence. LBMPC combines aspects of learning-based control and model predictive control (MPC, [13]). In contrast to adaptive control techniques [14], [15], the LBMPC based controller allows one to specify, *a priori,* a model based on the known physical system with uncertainty bounds. Like robust control, LBMPC can deal with uncertainty directly, but also allows the designer to specify performance objectives to optimize and explicitly incorporates online model updates to further improve performance. LBMPC is compatible with many learning techniques; previous work has employed a modified Nadaraya-Watson estimator with Tikhonov regularization [12] and a semi-parametric regression estimator [16].

In this paper, we present details and experiments of an implementation of LBMPC that runs in real time onboard a quadrotor UAV with limited computing performance and memory. A companion paper [17] explains the details of the modifications from LBMPC as it is described in [12]. Here, we outline a control architecture that uses a modified extended Kalman filter (EKF) to perform state estimates and learn updated model parameters. The LBMPC formulates the control problem as the solution of a convex optimization problem.

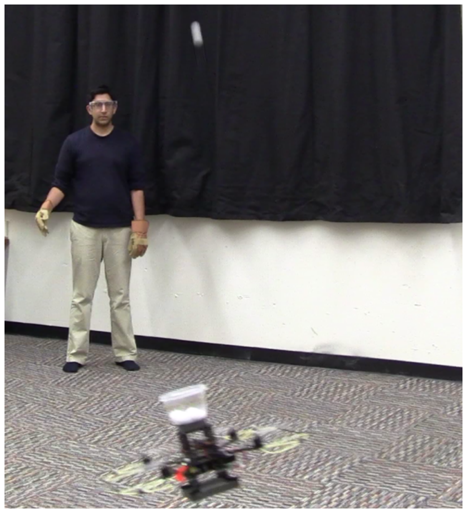
Fig. 1. “Ball catching” experiment. The quadrotor, controlled using LBMPC, is about to catch a ball. Video from the experiments can be viewed online at `http://eecs.berkeley.edu/~aaswani/LBMPC`.

Experiments show LBMPC has similar computational requirements to linear MPC, but can improve performance by allowing the models used to be updated online. The experiments demonstrate learning updates including the so-called ground effect (increased aerodynamic lift when operating close to the ground) [18]. LBMPC provides robustness against mis-learning; that is, even if the learning algorithm is poorly designed or tuned, the formulation provides safety. To demonstrate the precision control possible using LBMPC, we program the quadrotor to catch balls (Fig. 1).

The paper is organized as follows. We begin with preliminaries of notation, followed by mathematical models of the system. Next, the LBMPC controller is introduced, and the experimental apparatus is described. The paper concludes by providing experimental results.

II. Preliminaries

Here, we define the notation used in this paper. Vectors are not typeset specially, but will be identified as such when introduced (e.g., $v \in \mathbb{R}^{10}$). All vectors are column vectors,

This research is supported by the NSF CPS ActionWebs project, the ONR MURI SMARTS project, and by an NSERC fellowship (P. Bouffard).

A. Aswani, P. Bouffard, and C. Tomlin are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA

and the transpose of a vector or matrix is denoted with a superscript T (e.g., v^T).

Variables that change at each discrete timestep have the time index denoted by the subscript. However, in equations describing the update of such a variable, a superscript $+$ on the variable indexed by time indicates the subsequent time index of the variable. For example, $v^+ = 0.5v + 0.1$ is equivalent to $v_{i+1} = 0.5v_i + 0.1$.

Where a time-indexed variable is included without a subscript, this refers the value of the variable at the "most recent" timestep in a sense that should be clear from the context. The notation $\|v\|_M^2$ denotes the quadratic form $v^T M v$. The subscript N, E, or D is added to vectors to denote the vector component corresponding to the North, East, or Down (though the compass directions should not be interpreted literally) axis of an inertial frame, respectively. Symbols with a dot above are the time derivative of that symbol (e.g., $\dot{x} = \frac{d}{dt}x$).

III. Models

A. Quadrotor Vehicle Model

The basic principle of operation of a quadrotor helicopter consists in the generation of net force and torque through variation of the rotational speeds of the four rotors. Detailed treatment of the dynamics of quadrotor motion can be found in [4]. Here we assume a simplified model, that is more suitable for an operating regime around steady hover.

The quadrotor's position and orientation are expressed in terms of a body-fixed frame with axes $\mathbf{F}_B := \{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$, with respect to an inertial frame with axes $\mathbf{F}_I := \{\mathbf{x}_N, \mathbf{x}_E, \mathbf{x}_D\}$. Define the state of the system $x = \left[\, x_N\ \dot{x}_N\ \theta\ \dot{\theta}\ x_E\ \dot{x}_E\ \phi\ \dot{\phi}\ x_D\ \dot{x}_D \,\right]^T \in \mathbb{R}^{10}$, where (x_N, x_E, x_D) are the components of the vector from $\mathbf{F}_I$ to $\mathbf{F}_B$, expressed in $\mathbf{F}_I$, and ψ, θ, ϕ are the rotations (in radians) in a 3-2-1 (yaw-pitch-roll) Euler sequence taking $\mathbf{F}_I$ to $\mathbf{F}_B$. We assume that ψ is held fixed.

We assume that the closed-loop attitude dynamics (for pitch and roll) can be approximated by a second-order torsional inertia-spring-damper SISO system, with the commanded pitch/roll angle as input and the actual angle as output. Based on empirical data from tests using step inputs, we determined a transfer function model for the closed-loop attitude dynamics of each axis of the form $G(s) = n_0 \left(s^2 + d_1 s + d_0\right)^{-1}$. The pitch and roll dynamics are decoupled and identical; this is supported by the empirical data and vehicle symmetry.

For the translational degrees of freedom, we assume decoupled axes for the lateral (horizontal; $\mathbf{x}_N - \mathbf{x}_E$ plane) motion. We assume a frictionless point mass model driven by the quadrotor's total thrust T along with acceleration due to gravity $(-g/m \cdot \mathbf{x}_D)$, where $g = 9.81\,\mathrm{m/s^2}$, $m = 1.3\,\mathrm{kg}$. The no-input dynamics in each translational degree of freedom are simply a double integrator. The corresponding discretized (time step Δt) dynamics matrix is $A_t = \left[\begin{smallmatrix} 1 & \Delta t \\ 0 & 1 \end{smallmatrix}\right]$.

We now combine the translational and attitude dynamics for a given lateral axis (i.e. roll and $\mathbf{y}$, pitch and $\mathbf{x}$) to form a 4-state linear (affine) discrete-time model for that axis:

$$\begin{aligned} x_l^+ &= A_l x_l + B_l u_l + k_l \\ &= \begin{bmatrix} A_t & B_t C_r \\ 0 & A_r \end{bmatrix} \begin{bmatrix} x_t \\ x_r \end{bmatrix} + \begin{bmatrix} 0 \\ B_r \end{bmatrix} u_r + \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \end{aligned}$$

where $A_t, A_r \in \mathbb{R}^{2\times 2}$ are the discretized, linearized dynamics matrices of the translational and rotational subsystems respectively, $B_t, B_r \in \mathbb{R}^{2\times 1}$ are the input maps of translational and rotational subsystems respectively, and $k_l \in \mathbb{R}^{4\times 1}$ is a zero vector representing the nominal affine part of the dynamics. Based on the step input testing and with $\Delta t = 0.025\,\mathrm{s}$, the lateral dynamics are $k_l = 0$ and

$$[A_l | B_l] = \left[\begin{array}{cccc|c} 1 & 0.025 & 0.0031 & 0 & 0 \\ 0 & 1 & 0.2453 & 0 & 0 \\ 0 & 0 & 0.7969 & 0.0225 & 0.01 \\ 0 & 0 & -1.7976 & 0.9767 & 0.9921 \end{array}\right].$$

The vertical dynamics have no rotational component and can be written as $z^+ = A_t z + B_z + k_z$, where $B_z = -K_T \left[\, \frac{\Delta t^2}{2}\ \Delta t \,\right]^T$ ($K_T > 0$ is an empirically-determined thrust-to-command ratio) and $k_z = g \left[\, \frac{\Delta t^2}{2}\ \Delta t \,\right]^T$ represents the acceleration due to gravity. Finally, we combine the discrete time models blockwise to obtain an overall discrete-time linear-affine dynamics model,

$$x^+ = Ax + Bu + k + h(x,u) \tag{1}$$

$$y = Cx + \epsilon \tag{2}$$

where

$$\begin{aligned} A &= \mathrm{blkdiag}\left(A_l, A_l, A_z\right) \in \mathbb{R}^{10\times 10}, \\ B &= \mathrm{blkdiag}\left(B_l, B_l, B_z\right) \in \mathbb{R}^{10\times 3}, \\ k &= \begin{bmatrix} 0 & 0 & k_z \end{bmatrix}^T \in \mathbb{R}^{10\times 1},\ C \in \mathbb{R}^{5\times 10}, \end{aligned}$$

and C is a zero matrix except for unity entries such that $y = \left[\, x_N\ \theta\ x_E\ \phi\ x_D \,\right]^T + \epsilon$. The input is the commanded attitude and thrust $u = \left[\, \theta_s\ \phi_s\ T_s \,\right]^T$. The term $h(x,u)$ represents the unmodeled dynamics of the system. Thus the "nominal" dynamics state update (the case in which $h \equiv 0$) is,

$$x^+ = F_{\mathcal{N}}(x,u) := Ax + Bu + k. \tag{3}$$

The ϵ term represents measurement noise, assumed to be a bounded stochastic quantity, i.i.d. at each timestep.

B. Model of a Ball in Free Flight

A key part of the ball catching experiment, is estimating the ball's future trajectory based on an estimate of its current state. The trajectory of the ball in free flight is governed by the action of gravity, drag due to air resistance, the Magnus effect, buoyancy, and added (or "virtual") mass [19], [20]. The gravity force is $F_G = mg \cdot \mathbf{x}_D$, and drag acts in a direction opposite the ball's instantaneous velocity V. The magnitude of the drag force is proportional to the square of the ball's velocity $F_D = \frac{1}{2}\rho C_D \frac{V}{\|V\|} \|V\|^2$, where C_D is a drag coefficient that is typically determined empirically.

The Magnus effect induces a force perpendicular to both the velocity and the spin axis of the ball, thus causing the its trajectory to curve. This force appears to have an effect on our trajectory predictions, based on study of the ball's trajectory. However, a nonlinear EKF estimate that incorporated the Magnus effect did not converge fast enough to provide accurate estimates, and subsequently we used a more straightforward Luenberger observer—that has been proven to converge [20]—with a model that neglects Magnus. The buoyancy force and the "added mass" on the other hand, are both small enough to neglect.

Let $x_b = \left[\, x_{b,N}\ \dot{x}_{b,N}\ x_{b,E}\ \dot{x}_{b,E}\ x_{b,D}\ \dot{x}_{b,D} \,\right]^T$ represent the 3D position and velocity of the ball expressed in $\mathbf{F}_I$. The discrete-time (time step Δt_b) dynamics update is

$$\begin{aligned} x_b^+ = F_b(x_b) &= \operatorname{blkdiag}(A_b, A, A_b) x_b \\ &\quad + \left[\, 0\ \Delta t_b F_{D,N}\ 0\ \Delta t_b F_{D,E}\ \tfrac{\Delta t_b^2}{2} g\ \Delta t_b (g + F_{D,D}) \,\right], \end{aligned}$$

where $A_b = \left[\begin{smallmatrix} 1 & \Delta t_b \\ 0 & 1 \end{smallmatrix}\right]$, i.e., a discretized double integrator in each axis. Note that we neglect the small contribution of the drag force to the position update. We measure only the position of the ball; the output equation is $y_b = C_b x_b$ where C_b is zero except for unity entries such that $y_b = \left[\, x_{b,N}\ x_{b,E}\ x_{b,D} \,\right]^T$.

IV. Control System Design

In this section we describe the design of the quadrotor controller incorporating the LBMPC scheme. The overall control architecture is composed of (i) estimation of the vehicle state and learning of the unmodeled dynamics, and (ii) an optimization-based procedure for performing closed-loop control. Both are model-based: The state estimate uses a model of the system to make predictions of the current state based on the past state and input, and the optimization problem uses a system model to determine the cost of prospective control policies and to evaluate the result of those policies over a finite planning horizon.

1) Vehicle State Estimation and Learning: We assume a linear, time-varying oracle [12] $\mathcal{O}_m : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, parametrized by a vector of parameters $\beta \in \mathbb{R}^p$, $p = 12$, of the form $\mathcal{O}_m(x, u) = F(\beta)x + H(\beta)u + z(\beta)$, in which F, H, and z are linear in the entries of β. The parameters are constrained such that $\beta_{\min,i} \le \beta_i \le \beta_{\max,i}$, $i = 1, \dots, p$. The state update equation under the learned dynamics is then,

$$\begin{aligned} x^+ = F_{\mathcal{O}}(x, u) &:= F_{\mathcal{N}}(x, u) + \mathcal{O}_m(x, u) \\ &= (A + F)\,x + (B + H)\,u + k + z. \end{aligned} \quad (4)$$

The parameters $\beta = \{\beta_1, \dots, \beta_p\}$, can be thought of as "adjustments" to certain entries of the nominal dynamics matrices. In what follows, we simply write F, H, and z (dropping the explicit dependency on the parameters β). Estimates of the parameters $\hat{\beta}$ are determined jointly with estimates $\hat{x}$ of the state, using a variant of the extended Kalman filter (EKF) in which convergence is guaranteed for a model which is jointly nonlinear in the state and parameters but individually linear in each of these [21]. We assume that the parameters evolve according to $\beta^+ = \beta + \mu$ where μ is noise. The modified EKF is governed by the set of update equations,

$$\begin{aligned} \hat{x}^+ &= F_{\mathcal{O}}(x, u) + \hat{K}\zeta \\ P_2^+ &= (A + F)P_2 + MP_3 - \hat{K}\Xi L^T \\ P_3^+ &= P_3 - L\Xi L^T - \delta P_3 P_3^T + \Upsilon \\ \hat{\beta}^+ &= \operatorname{bound}(\hat{\beta} + L\zeta) \end{aligned}$$

Where $L := P_2^T C^T \Xi^{-1}$ and $M := \frac{\partial}{\partial \beta}(F\hat{x} + Hu + z)$. Here, $\zeta = y - C\hat{x}$ is the measurement innovation. The matrix $\hat{K}$ is a feedback matrix chosen such that $A + F - \hat{K}C$ is exponentially stable for all possible β. The matrices $P_2 \in \mathbb{R}^{10\times 12}$, $\Xi \in \mathbb{R}^{5\times 5}$, $P_3 \in \mathbb{R}^{12\times 12}$, and $\Upsilon \in \mathbb{R}^{12\times 12}$ are the cross-covariance between state and parameter estimates, the covariance of the parameter estimates, the measurement noise, and the parameter noise respectively. The tuning parameter $\delta > 0$ improves the numerics. The bound function clips each parameter to be within the specified limits.

A. LBMPC Design

At the heart of the LBMPC control scheme is the on-line solution of a convex optimization problem—specifically, a quadratic program (QP). At timestep m, we solve the QP,

$$\min_{c_\cdot, \theta}\ p(\tilde{x}_{m+N}) + \textstyle\sum_{j=0}^{N-1} q(\tilde{x}_{m+j}) + r(\check{u}_{m+j}) \quad (5)$$

$$\begin{aligned} \text{s.t.}\quad \tilde{x}_m &= \bar{x}_m = \hat{x}_m, & (6)\\ \tilde{x}_{m+i} &= F_{\mathcal{O}}(\tilde{x}_{m+i-1}, \check{u}_{m+i-1}), & (7)\\ \bar{x}_{m+i} &= F_{\mathcal{N}}(\bar{x}_{m+i-1}, \check{u}_{m+i-1}), & (8)\\ \check{u}_{m+i-1} &= K\bar{x}_{m+i-1} + c_{m+i-1}, & \\ \bar{x}_{m+i} &\in \mathcal{X},\ \check{u}_{m+i-1} \in \mathcal{U}, & \\ \bar{x}_{m+1} &\in \mathcal{X} \ominus \mathcal{D}, (\bar{x}_{m+1}, \theta) \in \omega & \end{aligned}$$

for $i \in \{1, \dots, N\}$ where N is the number of steps forward in time over which the optimization is performed (i.e., the "horizon"). The cost function (5) is the sum of final state error cost $p(x) = \|x - x_s\|_P^2$, and intermediate step costs on state $q(x) = \|x - x_s\|_Q^2$ and input $r(u) = \|u - u_s\|_R^2$. The different notions of the state are indicated by marks on the symbol; hence x (no marks) indicates the true state, $\hat{x}$ the estimated state, $\tilde{x}$ the predicted state incorporating the oracle, and $\bar{x}$ the predicted state using the nominal model. The desired state is x_s, and u_s is the steady-state control that would maintain the state at x_s, i.e. u_s solves $x_s = F_{\mathcal{O}}(x_s, u_s)$.

The matrices P, Q, and R are weights on the final state error cost, the stage error cost and the stage control input cost, respectively. The polyhedral sets $\mathcal{X}$ and $\mathcal{U}$ are bounded and convex; they encode the allowable states and inputs, respectively. These are typically expressed as sets defined by half-space inequalities. For example, $\mathcal{X} = \{x \mid F_x x \le h_x\}$. Note that, owing to the boundedness of β, $\mathcal{X}$, and $\mathcal{U}$, the oracle is also bounded: $\mathcal{O}_m(x, u) \in \mathcal{D}$ for some bounded, convex polytope $\mathcal{D}$. The set ω is an approximation of the maximal output admissible distubance invariant set [22] and $\theta \in \mathbb{R}^3$ is a parametrization of points that can be feasibly tracked with a linear controller.

The solution $\{c_i^*\}_{i=m}^{m+N-1}$ to this QP encodes the optimal—with respect to the cost function in (5)—sequence of controls to apply to the system over the next N steps based on the current parameter and state estimates. The actual controls are the $\check{u}$'s, (**??**) used to determine predicted oracle states $\tilde{x}$ (7) used in the cost function and predicted states $\bar{x}$ (8) used for constraint satisfaction.

From an implementation perspective, the key output of the QP is only the first control of the sequence of N controls, $\check{u}_m = K\hat{x}_m + c_m^*$. Note that the feedback K serves to limit the effects of model uncertainty [23]. This is the control that is actually applied to the system; at the next iteration through the control loop, the QP is solved once again with new state estimates and new oracle dynamics based on updated F, H, z matrices. While the above treats the salient points, [17] goes into greater detail regarding the development of the optimization problem.

V. Experimental Setup

In this section we describe our experimental apparatus, particularly the quadrotor vehicle used, our laboratory setup including the method of sensing the quadrotor's pose, as well as the software architecture.

The main element of the system is a quadrotor UAV, based on the "Pelican", a vehicle system geared towards research applications produced by Ascending Technologies. As configured for these experiments, the overall vehicle mass is 1.3 kg. Our quadrotor is equipped with an onboard computer with a 1.6 GHz Intel Atom N270 CPU, 1 GB of RAM, an 8 GB solid state (micro-SD card) disk, and wi-fi communications.

The quadrotor is supplied with onboard electronics and firmware that implement systems functionality as well as closed loop control for attitude angles, and open loop control of thrust running at 1 kHz on one of two ARM7 chips on a proprietary board. This controller accepts θ, ϕ, $\dot{\psi}$ and commands over a serial port interface; we issue these commands at a rate of 40 Hz. The serial interface also provides telemetry data, although this telemetry is used for debugging/diagnostic purposes only in this work.

Experiments are conducted in a laboratory environment equipped with a Vicon MX motion capture system. This system tracks the 3D position of small retroreflective markers using an array of cameras with nearinfrared illumination strobes. Provided with a model of a rigid body equipped with markers, it provides the full rigid-body position and orientation of the quadrotor at a rate of 120 Hz. We use this same system to obtain measurements of the 3D position of the ball in the "ball catching" experiment.

A ground station laptop computer provides the ability to control the quadrotor manually and initiate the automatic modes of operation. The various computers are interconnected on a local area network (LAN), with the onboard computer communicating via WiFi. A one-way radio link provides a safety backup and is required to arm the quadrotor for flight.

The onboard computer runs Ubuntu Linux and a software stack developed for quadrotor experimentation [24]**,** which uses the ROS (Robot Operating System [25]) framework. The LBMPC control architecture runs entirely onboard the quadrotor's computer, including the QP solver. Most of the software is implemented in C++, but the QP is solved using LSSOL [26] (FORTRAN). We are able to achieve the system's nominal control period of 40 Hz with a horizon of $N = 15$ steps with this solver; future investigations will investigate performance using different solving formulations [27], [28], [29].

VI. Experimental Results

In this section, we describe the results of several experiments that illustrate different aspects of the performance using LBMPC, with particular emphasis on the benefits of LBMPC over standard linear MPC.

A. Learning the ground effect

The ground effect is a well known aerodynamic effect in which the vehicle is subject to additional lift when in the vicinity of the ground. In helicopters, ground effect typically has a non-negligible impact on lift force when the main rotor is within 2 rotor diameters of the ground [18]. This effect has also been noted in in other quadrotors [30], [31], [32].

In this experiment, the quadrotor was commanded to hover at a specified height, out of the ground effect, and after some time (at approx. 249 s on the plot), the altitude command was changed to correspond to a ground clearance of 3 cm. At this height, the plane of propellors is approximately 0.19 m from ground, or about $3/4$ of one rotor diameter. In the parametrization used, β_7 is the learned change in the input mapping for the thrust input, with the nominal value being the $(10, 3)$ entry of B. As shown in Fig. 2, the parameter estimate quickly (within approximately 1 s) adjusts to reflect an increase in the total thrust per unit thrust command (ratio of β_7 to $B_{10,3}$). A clear increase in effective thrust per input thrust is seen when the quadrotor is in the vicinity of the ground; approximately 6% more thrust per unit command is observed. When the command is returned to the original value, β_7 reverts correspondingly, within about 2 s.

For the same experiment but with standard linear MPC (nominal model only, no learning), the quadrotor is not able to hover at the commanded distance above the ground, because the effective thrust is significantly greater than what the nominal model predicts. Thus when flying with standard linear MPC, it is not possible to perform a "soft landing"—one has to manually cut power to and let the quadrotor fall the remaining distance.

B. Decreased overshoot in step response

In this experiment, we investigated the effects of LBMPC on the transient response of the quadrotor to changes in hover setpoint. The quadrotor was commanded to initially hover at $x = -1\,\text{m}$. The setpoint was repeatedly changed to $x = 1\,\text{m}$ and then back to $x = -1\,\text{m}$ after a delay of $3.5\,\text{s}$. We performed this test with both linear MPC (using

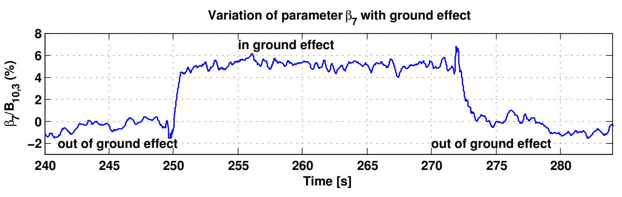

Fig. 2. Variation of thrust input mapping $(B + H)_{10,3}/B_{10,3}$ vs. time.

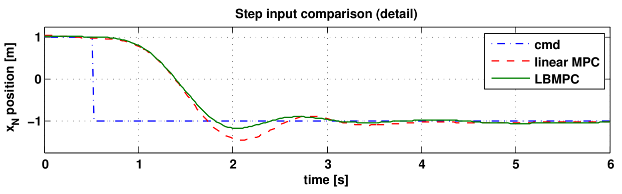

Fig. 3. Step response for linear MPC with nominal model and LBMPC with learned model. The reference command is the dotted blue line. The LBMPC response here is from the 4th step command after enabling learning.

only the nominal model) and with LBMPC. Fig. 3 shows a comparison of the x-axis position of the quadrotor during this maneuver between linear MPC and LBMPC. The LBMPC response exibits considerably less overshoot (62% less in the $x = 1\,\mathrm{m}$ maneuver shown in Fig. 3) than the linear MPC response. In addition, we observed that the LBMPC response characteristics would improve with repeated maneuvers; this is expected given that the model parameters continue to be refined with each maneuver. We also observed a greater decrease in overshoot when successive step maneuvers were more closely spaced in time. This reflects the fact that the parameter adjustments learned during the transient flight are important in improving the stopping characteristics, and it suggests that a possible avenue for improvement is to introduce a velocity-dependent drag term in the dynamics model. This example demonstrates the type of performance improvement that is possible with LBMPC and a well-behaved oracle.

C. Robustness to "incorrect learning"

In this experiment, we deliberately caused the dual EKF to be prone to mis-estimate the model parameters by grossly increasing the noise process covariance Υ. We allowed the quadrotor to hover at a height above the ground of 0.85 m using linear MPC (without learning updates; F, H, z all zero), and enabled learning. After some maneuvering, the parameter estimates diverged, hitting their bounding limits. At this time, the quadrotor's altitude dropped sharply, but the quadrotor did not contact the ground, and ended up in a stable hover approximately 0.1 above the ground (see Fig. 4). The optimization found a feasible solution throughout, and this demonstrates that even when the oracle degrades the learned model with respect to the nominal one, the system does not become unsafe or unstable.

D. Precise maneuvering: ball catching

In this experiment, we tested the dynamic performance of the quadrotor using LBMPC using a challenging robotic

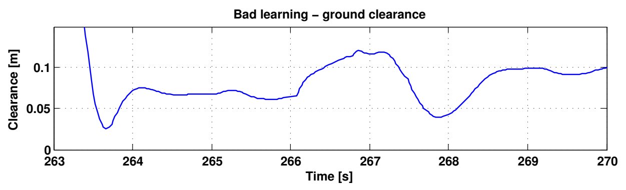

Fig. 4. Safety is maintained even if parameter learning goes awry.

demonstration task, of catching a ball thrown by a human, when the ball has an *a priori* unknown trajectory, before it hits the ground.

We equipped the quadrotor with a simple plastic cup, with a circular opening of radius 0.065 m directly above the main body. The quadrotor is programmed to hover in place at a fixed altitude, 0.5 m above the ground. A command is issued to ready the quadrotor to catch the ball. Next, the ball, which has a mass of 6 g and a diameter of 33 mm (similar to a ping-pong ball) is tossed towards the robot by hand.

The ball is covered with reflective tape so that the Vicon system can track it. The measurements of the ball's position are fed into a Luenberger observer that uses a nonlinear model incorporating a quadratic drag term for the state prediction step, and a linear correction step. The observer's velocity estimate is initialized using a finite difference estimate from two successive measurements to speed up the observer's convergence. Once 20 initial measurements have been processed, the state estimate is used to propagate the dynamics model forward to estimate the point $\hat{x}_c$ where the ball's trajectory will intersect the plane in which the quadrotor is hovering. The quadrotor's reference command is then set to $\hat{x}_c$, and it continues to track updates to $\hat{x}_c$.

The ball catching task is challenging because the quadrotor must arrive quickly and accurately at the location where the ball is predicted to be. Given the contraints of the experiment room, even for a ball thrown high the quadrotor still has roughly 1 second from the time that the initial $\hat{x}_c$ are available to when the ball actually crosses the plane. The estimates of x_c must be accurate enough from the beginning that that the quadrotor is not commanded initially in the wrong direction, thus losing ground when the estimate later improves. Furthermore, when the quadrotor is accelerating, the vehicle is tilted and so the effective "catch zone" for the ball is reduced compared to when the quadrotor is stationary; this favors an approach in which that can reach the destination and stabilize quickly.

We were able to achieve a very high rate of successful catches–over 90%. The vast majority of misses were also very close, within one or two ball diameters of the edge of the cup. We elected not to perform a more well-controlled study of success rates because this would require developing a repeatable ball-throwing device. At this stage, we believe that it would be more interesting to investigate a more detailed nonlinear model for the ball's dynamics. Indeed we observed the effects of the Magnus force, which caused a noticeable curvature in the ball's path. We attempted to throw the ball in a similar fashion each time, but some variable amount of spin

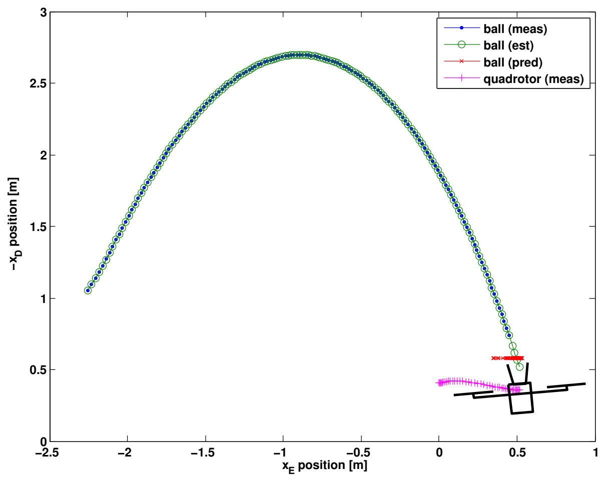

Fig. 5. Catching a ball—side view of data from ball catching experiment. Measurements and EKF estimates of the ball's position throughout its trajectory, the estimated final position of the ball, the trajectory of the quadrotor body frame $\mathbf{F}_B$ are shown. A cartoon approximation of the quadrotor in its final pose is also shown.

(usually topspin given the underhanded throw) is induced on each throw.

VII. Conclusions and Future Work

We have described the implementation of modified LBMPC onboard a quadrotor helicopter, and experiments that demonstrate some of the performance improvements that LBMPC can enable. Future work will examine whether the special structure of the MPC problem could enable improvements in computation time. A possible future direction for improvement in the ball catching task is to try and identify the spin based on the available measurements, using a model that incorporates the Magnus effect.

References

[1] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541 – 1561, 2011.

[2] G. Hoffmann, S. Waslander, and C. Tomlin, "Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications," in *Proc. of the AIAA Guidance, Navigation, and Control Conf. and Exhibit*, (Keystone, Colorado), Citeseer, Aug. 2006.

[3] I. Austen, "Libyan Rebels Reportedly Used Tiny Canadian Surveillance Drone," *The New York Times*, p. A11, Aug. 2011.

[4] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control Engineering Practice*, vol. 19, pp. 1023–1036, June 2011.

[5] A. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," in *15th International Symposium on Robotics Research*, (Flagstaff, AZ, USA), 2011.

[6] M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In-and Outdoor Environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[7] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, 2010.

[8] O. Purwin and R. D'Andrea, "Performing and extending aggressive maneuvers using iterative learning control," *Robotics and Autonomous Sys.*, vol. 59, pp. 1–11, Jan. 2011.

[9] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous Helicopter Aerobatics through Apprenticeship Learning," *The International Journal of Robotics Research*, vol. 29, pp. 1608–1639, June 2010.

[10] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification," *The International Journal of Robotics Research*, vol. 29, pp. 1038–1052, Apr. 2010.

[11] J. H. Gillula and C. J. Tomlin, "Guaranteed Safe Online Learning of a Bounded System," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Francisco, CA), 2011.

[12] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably Safe and Robust Learning-Based Model Predictive Control," July 2011. arXiv:1107.2487v1 [math.OC]. [Online].

[13] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, pp. 125–133, Jan. 2004.

[14] K. J. Å ström and B. Wittenmark, *Adaptive Control*. Prentice-Hall, 2nd ed., 1994.

[15] S. S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, 1994.

[16] A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin, "Reducing Transient and Steady State Electricity Consumption in HVAC Using Learning-Based Model-Predictive Control," *Proceedings of the IEEE*, vol. PP, no. 99, pp. 1–14, 2011.

[17] A. Aswani, P. Bouffard, and C. Tomlin, "Extensions of Learning-Based Model Predictive Control for Real-Time Application to a Quadrotor Helicopter," *submitted*, 2011.

[18] J. G. Leishman, *Principles of helicopter aerodynamics*. Cambridge University Press, 2006.

[19] R. L. Andersson, *A Robot Ping-Pong Player: Experiments in Real-Time Intelligent Control*. 1988.

[20] W. Yingshi, S. Lei, L. Jingtai, Y. Qi, Z. Lu, and H. Shan, "A novel trajectory prediction approach for table-tennis robot based on nonlinear output feedback observer," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pp. 1136–1141, IEEE, 2010.

[21] L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, pp. 36–50, Feb. 1979.

[22] I. Kolmanovsky and E. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical Problems in Engineering*, vol. 4, no. 4, pp. 317–363, 1998.

[23] L. Chisci, J. Rossiter, and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

[24] P. Bouffard, "starmac-ros-pkg ROS repository." `http://www.ros.org/wiki/starmac-ros-pkg`, 2011.

[25] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, (Kobe, Japan), 2009.

[26] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright, "User's Guide for LSSOL (Version 1.0)," 1986.

[27] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 267–278, 2010.

[28] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[29] M. Zeilinger, C. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *Automatic Control, IEEE Transactions on*, vol. 56, no. 99, pp. 1–1, 2008.

[30] S. Waslander, G. Hoffmann, J. Jang, and C. Tomlin, "Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3712–3717, IEEE, 2005.

[31] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, no. 1, pp. 153–158, IEEE, 2007.

[32] N. Guenard, T. Hamel, and L. Eck, "Control laws for the tele operation of an unmanned aerial vehicle known as an x4-flyer," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3249–3254, IEEE, 2006.