Real-Time Model Predictive Control for Keeping a Quadrotor Visible on the Camera Field-of-View of a Ground Robot

Wei Ding[1], Madan Ravi Ganesh[2], Robert N. Severinghaus[3], Jason J. Corso[2] and Dimitra Panagou[1]

***Abstract*— This paper considers a cooperative control design for an aerial/ground robot system, and addresses the problem of maintaining visibility of a quadrotor within the camera field-of-view of a ground robot in the presence of external disturbances. The quadrotor needs to be tracked by the ground robot with a monocular camera, and hence its motion should facilitate the ground vision-based tracking process by remaining in the effective camera sensing area. We design a model predictive controller (MPC) strategy where the visibility constraints of the camera and the control input constraints of the quadrotor are encoded into the cost function via barrier functions, and we adopt a fast MPC solver that is able to solve the optimization problem in real time. We also propose a method to enhance the robustness of the algorithm by suitably defining a restart method for the MPC solver. The applicability of the proposed algorithm is demonstrated through simulations and experimental results on real setups.**

I. Introduction

Aerial and ground robots are nowadays used in many applications, from first response to monitoring and surveillance [1], [2]. Their limited actuation, sensing and communication capabilities make the cooperative control between aerial and ground robots a topic worthy of investigation [3]–[5]. One relatively recent idea for ultimately achieving increased situational awareness for ground vehicles is to implement a cooperative aerial/ground robot system in which the aerial robot is tethered to the ground robot with cables for power supply and data transmission, and streams information about the surrounding environment to the ground vehicle. Alternatively, a vision-based localization method has been recently developed in [6], where the ground robot is equipped with a monocular camera looking upwards and processes the acquired video to obtain a 3D pose estimation of the aerial robot flying above the ground.

One of the challenges of this system is that the aerial robot should hover at a high enough altitude over the ground in order to provide a larger visual field and stream data to the ground robot, and at the same time maintain a reasonable position relative to the ground robot to facilitate the visual tracking process, despite the effect of measurement noise and disturbances, e.g., wind gusts. The camera mounted on the ground robot has an inverse-pyramid shape visual zone, and the aerial robot should always remain there (Fig. 1).

*This work was supported in part by U.S. Army TARDEC under agreement W56HZV-14-2-0001

[1]Wei Ding and Dimitra Panagou are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, USA. {weiding, dpanagou}@umich.edu. [2]Madan Ravi Ganesh and Jason J. Corso are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, USA. {madantrg,jjcorso}@umich.edu. Robert N. Severinghaus is with the US Army TARDEC, Warren, MI. robert.n.severinghaus.civ@mail.mil

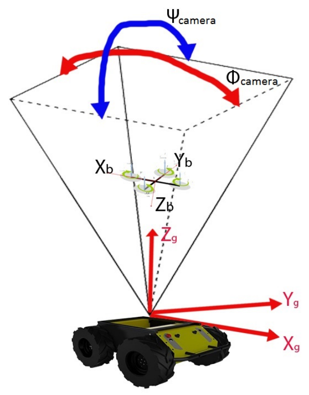


Fig. 1. Modeling of the camera field-of-view. The quadrotor should be restricted to fly in the effective sensing area.

Control of quadrotors has become a popular research topic recently [7]–[11]. Model Predictive Control (MPC) is a popular means to address constrained control problems for aerial vehicles, see for instance [12]–[16]. Most of the work on MPC for aerial vehicles focuses on the generation of trajectories that satisfy safety constraints such as avoiding physical obstacles or other vehicles. The visibility maintenance problem considered here differs in the sense that the quadrotor is not forced to track a desired trajectory, but rather to hover within a region defined by state constraints, despite the effect of disturbances. To the best of our knowledge this problem has not been addressed before, while the experimental implementation of MPC schemes on micro aerial vehicles has not been addressed extensively either [17]–[19].

This paper builds upon ideas in our earlier work [20] and implements a real-time MPC algorithm that forces the quadrotor hover in the effective sensing region of the camera of the ground robot. Compared to [20], here we consider the 3D motion of an aerial robot under the effect of external disturbances, we implement a Kalman filter to fuse information from the available position and acceleration sensors, and implement the proposed strategy experimentally on a real testbed. Furthermore, the adopted fast MPC solver is suitably modified to enhance robustness against extreme cases where the original algorithm failed during the experimental implementation.

The paper is organized as follows: Section II provides the mathematical modeling of the considered constrained control system. The proposed MPC strategy along with the adopted fast solver and the restart method that recovers from failures is presented in Section III. Sections IV and V present the simulation and experimental results that demonstrate the applicability and effectiveness of our MPC on real setups under various scenarios of external disturbances and measurement noise. Section VI summarizes our results and ideas on ongoing and future research.

II. Mathematical Modeling

1) Quadrotor Control: Quadrotors are typically controlled via a two-loop strategy, namely position (or outer) control and attitude (or inner) control [7], [9], [10], [13]. The linear acceleration of the vehicle is used as the input of the outer control system to regulate the position and the linear velocity of the quadrotor. Linear acceleration in the body frame is related to linear acceleration in an inertial frame through the kinematic transformation:

$$\begin{bmatrix} a_x \\ a_y \\ a_x \end{bmatrix} = \mathbf{C}_{gb}(\phi,\theta,\psi) \begin{bmatrix} 0 \\ 0 \\ a_b \end{bmatrix}, \tag{1}$$

where a_x, a_y and a_z denote the linear acceleration components along the axes of the inertial frame, a_b is the linear acceleration along the z-axis of the body frame, $\mathbf{C}_{gb}(\phi,\theta,\psi) \in SO(3)$ is the rotational transformation matrix from body frame to inertial frame, and ϕ, θ, ψ are the roll, pitch and yaw Euler angles, respectively. The inner control loop of the Hummingbird quadrotor that was used in the experimental trials is already set by the manufacturer (Ascending Technologies) and regulates the speed of the four motors so that the Euler angles ϕ, θ, the yaw rate $\dot{\psi}$, and the acceleration a_b converge to their desired values ϕ_{des}, θ_{des}, $\dot{\psi}_{des}$, and $a_{b,des}$, respectively. The $\dot{\psi}_{des}$ command is used to regulate the yaw angle ψ to a desired value ψ_{des}. The desired values are mapped to linear acceleration a_x, a_y and a_z along the axes of the inertial frame through Eq. (1). Thus, in practice we only need to address the outer (position) loop with the linear acceleration being the control input, and the derived control input is used to render the desired values of the Euler angels and acceleration in the body frame ϕ_{des}, θ_{des}, ψ_{des} and $a_{b,des}$, respectively, that are further fetched to the inner (attitude) controller.

Consequently, the translational motion of a quadrotor along the x, y and z axes of an inertial frame can be ultimately described via a double integrator along each axis, augmented with position error integration terms to reduce the steady state error. The system is described in matrix form:

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k + \boldsymbol{\omega}_k, \tag{2}$$

where $\mathbf{A}_d = diag\{\mathbf{A},\mathbf{A},\mathbf{A}\}$ and $\mathbf{B}_d = diag\{\mathbf{B},\mathbf{B},\mathbf{B}\}$, with $\mathbf{A} = \begin{bmatrix} 1 & dT & 0 \\ 0 & 1 & 0 \\ dT & \frac{dT^2}{2} & 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} \frac{1}{2}dT^2 & dT & \frac{1}{6}dT^3 \end{bmatrix}^\top$ being the state and control matrices of a double integrator augmented with error accumulation term, respectively. dT is the time interval of the discretized system. The state vector $\mathbf{x}_k \in R^9$, written analytically as: $\mathbf{x}_k = \begin{bmatrix} e_k^x & v_k^x & I_k^x & e_k^y & v_k^y & I_k^y & e_k^z & v_k^z & I_k^z \end{bmatrix}^\top$, incorporates the position error e_k, its time rate of change v_k, and the position error accumulation I_k along each axis of the inertial frame, where the superscripts indicate the direction of the movement. $\mathbf{u}_k = \begin{bmatrix} u_k^x & u_k^y & u_k^z \end{bmatrix}^\top$ is the control input denoting acceleration in the inertial frame, and $\boldsymbol{\omega} \in R^9$ is the vector incorporating external disturbances.

2) Constraints: The effective sensing area of the camera mounted on the ground robot is modeled as an inversed pyramid shaped zone (Fig. 1). Plausibly, it would make sense to have the quadrotor fly as high as possible. However, the visual tracking algorithm becomes less reliable as altitude increases [6]. Thus, the position of the quadrotor is constrained as follows: $-z\ \tan\frac{\phi_{\text{camera}}}{2} \le x \le z\ \tan\frac{\phi_{\text{camera}}}{2}$, $-z\ \tan\frac{\psi_{\text{camera}}}{2} \le y \le z\ \tan\frac{\psi_{\text{camera}}}{2}$, $z_{\min} \le z \le z_{\max}$, where x, y and z are the position coordinates of the quadrotor in an inertial frame with origin on the camera of the ground robot, ϕ_{camera} and ψ_{camera} are the camera angles-of-view, chosen smaller than their actual values to ensure that the entire quadrotor remains visible, and $z_{\min}$, $z_{\max}$ denote the height boundaries for the vision tracking algorithm to work properly. The constraints are written in matrix form:

$$\mathbf{F}_x \mathbf{x}_k \le \mathbf{b}_x, \tag{3}$$

where $\mathbf{F}_x \in R^{6\times 9}$ and $\mathbf{b}_x \in R^{6\times 1}$ are suitably defined matrices so that (3) reflects the constraints described above. Finally, to account for motor saturation, the control input is constrained as: $\mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}$, where $\mathbf{u}_{\min}, \mathbf{u}_{\max} \in R^3$ are known bounds. Similarly to the case above, the matrix form of input constraints is:

$$\mathbf{F}_u \mathbf{u}_k \le \mathbf{b}_u, \tag{4}$$

with $\mathbf{F}_u \in R^{6\times 3}$ and $\mathbf{b}_u \in R^{6\times 1}$ suitably defined matrices to reflect the input constraints.

III. Model Predictive Controller

3) Objective and Constraints: State and input constraints are encoded into the Model Predictive Control formulation by incorporating barrier functions to penalize violations of the constraints into the objective function. The optimization problem is defined as:

$$\begin{aligned} \underset{\substack{\mathbf{u}_0\ldots\mathbf{u}_{N-1} \\ \mathbf{x}_1\ldots\mathbf{x}_N}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} J(\mathbf{x}_{i+1},\mathbf{u}_i) + \phi_u(\mathbf{u}_i) + \phi_x(\mathbf{x}_{i+1}) \\ \text{subject to} \quad & \mathbf{x}_{i+1} = \mathbf{A}_d\mathbf{x}_i + \mathbf{B}_d\mathbf{u}_i + \boldsymbol{\omega}_i,\ \forall i \in \{0,\ldots,N-1\}, \end{aligned} \tag{5}$$

where N is the prediction horizon,

$$J(\mathbf{x}_{i+1},\mathbf{u}_i) = \mathbf{x}_{i+1}^\top \mathbf{Q}\mathbf{x}_{i+1} + \mathbf{u}_i^\top \mathbf{R}\mathbf{u}_i \tag{6}$$

is a quadratic cost function of the state and the control input, $\mathbf{Q} \in R^{9\times 9}$ and $\mathbf{R} \in R^{3\times 3}$ are positive definite and symmetric matrices, and $\phi_x(\mathbf{x})$, $\phi_u(\mathbf{u})$ are barrier functions of the state and control input, respectively, defined as:

$$\phi_x(\mathbf{x}_{i+1}) = -\mu_x \sum_{j=1}^{6} \ln(b_x^j - \mathbf{F}_x^j \mathbf{x}_{i+1}), \tag{7a}$$

$$\phi_u(\mathbf{u}_i) = -\mu_u \sum_{j=1}^{6} \ln(b_u^j - \mathbf{F}_u^j \mathbf{u}_i), \tag{7b}$$

where $\mathbf{F}_u^j$, $\mathbf{F}_x^j$ stands for the jth row of $\mathbf{F}_u$ and $\mathbf{F}_x$, respectively, and b_u^j, b_x^j stand for the jth element of $\mathbf{b}_u$ and $\mathbf{b}_x$, respectively. For all the inequality constraints to be satisfied, the values of the barrier functions (7a), (7b) must remain real, finite numbers.[1]

4) Disturbance Estimation: External disturbances due to wind gusts, or the fact that the response of the inner loop controller might not be fast enough to render the desired acceleration in time, may force the quadrotor out of the effective sensing area. Earlier work on disturbance estimation includes, for instance, [21], which estimates wind velocity and direction through bayesian estimation, [22], which considers the aerodynamic effect of a wind gust, and [23], which estimates wind disturbance using a Dryden wind-gust model.

Here we use a simple way of estimating external disturbances and other unmodeled phenomena by comparing the desired acceleration and actual acceleration. That is, in each time step k, we consider that the disturbance $\boldsymbol{\omega}_k$ in Eq. (2) is related to the difference between the onboard IMU acceleration measurement $\mathbf{a}_k^{imu}$ and the desired acceleration $\mathbf{u}_{k-1}$ of the previous control step through:

$$\boldsymbol{\omega}_k = \mathbf{B}_d(\mathbf{C}_{gb}\,\mathbf{a}_k^{imu} - \mathbf{u}_{k-1}). \tag{8}$$

We assume that the disturbance $\boldsymbol{\omega}_k$ does not change rapidly during the entire prediction horizon N, and we update the corresponding matrices of the optimization problem accordingly in every call of MPC solver, see later on.

5) Fast Solving Algorithm: Solving linear optimization problems has been extensively studied. [24] and [25] solve the linear optimization problem for various initial conditions off-line and implement the derived solutions online as a look-up table, yet require large hardware memory when used for a large horizon MPC. [26], [27] and [28] utilize an interior-point linear programming algorithm, which however is not fast enough for implementation on a real-time control system. [29] combines and improves the interior point algorithm through fixed barrier coefficients and iteration numbers, and solves the optimization problem with the Infeasible Start Newton Method, which uses an initial guess that satisfies inequality constraints, but does not need to meet the equality constraints.[2]

6) Infeasible Start Newton Method: The infeasible start newton method is introduced in [30]. At each time step k, one combines all predicted states $\mathbf{x}_k^1, \dots, \mathbf{x}_k^N$ and control inputs $\mathbf{u}_k^0, \dots, \mathbf{u}_k^{N-1}$ of the N prediction steps into one vector: $\mathbf{z}_k = \begin{bmatrix} \mathbf{u}_k^{0\top} & \mathbf{x}_k^{1\top} & \dots & \mathbf{u}_k^{N-1\top} & \mathbf{x}_k^{N\top} \end{bmatrix}^\top$, and concatenates equation (5) in compact form as in [29]:

$$\begin{array}{ll} \underset{\mathbf{z}_k}{\text{minimize}} & J = \mathbf{z}_k^\top \mathbf{H}_k \mathbf{z}_k + g_k(\mathbf{z}_k) \\ \text{subject to} & \mathbf{M}_k \mathbf{z}_k = \mathbf{h}_k \end{array}, \tag{9}$$

where: $\mathbf{H}_k = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q} \end{bmatrix}$, $\mathbf{h}_k = \begin{bmatrix} \mathbf{A}\mathbf{x}_k^0 + \boldsymbol{\omega}_k \\ \boldsymbol{\omega}_k \\ \boldsymbol{\omega}_k \\ \vdots \\ \boldsymbol{\omega}_k \end{bmatrix}$, $\mathbf{M}_k =$

$$\begin{bmatrix} -\mathbf{B}_d & \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{A}_d & -\mathbf{B}_d & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{A}_d & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{A}_d & -\mathbf{B}_d & \mathbf{I} \end{bmatrix},$$

and

$$g(\mathbf{z}_k) = \sum_{i=0}^{N-1} \phi_u(\mathbf{u}_k^i) + \phi_x(\mathbf{x}_k^{i+1}),$$

where $\mathbf{x}_k^0 = \mathbf{x}_k$ is the initial condition for problem (9). Note that in the computation of the $\mathbf{h}_k$ term, we assume that the disturbance $\boldsymbol{\omega}_k$ is given by (8), and maintains the same value during the entire prediction horizon. The condition for optimality of (9) is:

$$\begin{aligned} \boldsymbol{\rho}_J = & \quad \nabla J + \mathbf{M}_k^\top \boldsymbol{\eta}_k = \mathbf{0}, \\ \boldsymbol{\rho}_i = & \qquad\qquad \mathbf{M}_k \mathbf{z}_k = \mathbf{0}, \end{aligned} \tag{10}$$

where ∇J is the gradient of the objective function of the problem (9), $\boldsymbol{\rho}_J$ and $\boldsymbol{\rho}_i$ are residuals of the optimality condition. Equation (10) can be solved with Newton's Method, where the descending direction $\Delta\mathbf{z}_k$ and $\Delta\boldsymbol{\eta}_k$ are computed by solving:

$$\begin{bmatrix} \nabla^2 J & \mathbf{M}_k^\top \\ \mathbf{M}_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{z}_k \\ \Delta\boldsymbol{\eta}_k \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\rho}_J \\ \boldsymbol{\rho}_i \end{bmatrix}, \tag{11}$$

where $\nabla^2 J$ is the Hessian matrix of the objective function of the problem (9). We choose an initial guess $\mathbf{z}_{k,0} \in Z_f$ where $Z_f = \{\mathbf{z} | \mathbf{F}_x \mathbf{x} < \mathbf{b}_x,\ \mathbf{F}_u \mathbf{u} < \mathbf{b}_u\}$. Let $\boldsymbol{\rho}(\mathbf{z}_k, \boldsymbol{\eta}_k)$ denote $\begin{bmatrix} \boldsymbol{\rho}_J^\top & \boldsymbol{\rho}_i^\top \end{bmatrix}^\top$, then the procedure is highlighted in Algorithm 1 below. After solving the MPC, the first predicted input $\mathbf{u}_k^0$ is used as the control input $\mathbf{u}_k$ of the system (2).

7) Restoration upon Algorithm Failure: We considered cases that might lead to violation of constraints, such as strong wind gust, under which the algorithm fails to return meaningful control inputs. Such situations might deteriorate drastically the performance of the system in the following sense: The fast solving algorithm usually runs with a warm start method [31]. The warm start uses an initial guess $\mathbf{z}_{k,0}$ for the MPC solver at time k, which is a shift of elements of the previous MPC solution $\mathbf{z}_{k-1}^*$: $\mathbf{z}_{k,0} = \begin{bmatrix} \mathbf{u}_{k-1}^{1*\ \top} & \dots & \mathbf{x}_{k-1}^{N*\ \top} & \mathbf{u}_{k-1}^{0*\ \top} & \mathbf{x}_{k-1}^{1*\ \top} \end{bmatrix}^\top$, where $\mathbf{u}_{k-1}^{i*}$ and $\mathbf{x}_{k-1}^{i*}$ are the sequences of control inputs and predicted states in $\mathbf{z}_{k-1}^*$. The advantage of the warm start method is fast convergence, since $\mathbf{z}_{k,0}$ satisfies the inequality constraints and is close enough to the equality constraints, provided that the disturbance $\boldsymbol{\omega}_k$ does not change rapidly in one time step, that is, no strong additive disturbance occurs. However, if the MPC solver fails for some reason, then the warm start method will keep using initial guesses that lie out of the

[1]Note that the primary goal is rather to limit the quadrotor in the visibility zone, and not necessarily to keep it close to the center of the camera field of view, hence the matrices $\mathbf{Q}$ and $\mathbf{R}$ can be chosen with relatively small entries compared to the barrier coefficients μ_x, μ_u.

[2]Here we adopt the ideas of solving the MPC as done in [29] and [30], we keep a large barrier coefficient and improve the robustness of the algorithm with a restart mechanism for the cases when failures might happen.

Algorithm 1 Solve MPC

```
1: procedure SOLVE($\mathbf{x}_k^0$, $\boldsymbol{\omega}_k$)
2:     update matrix $\mathbf{h}_k(\mathbf{x}_k^0, \boldsymbol{\omega}_k)$
3:     select initial guess $\mathbf{z}_{k,0} \in Z_f$
4:     set $ErrorTolerance$, $0 < \beta < 1$, $0 < \alpha < 0.5$
5:     set $\mathbf{z}_k = \mathbf{z}_{k,0}$, $\boldsymbol{\eta}_k = \mathbf{0}$
6:     while $\|\boldsymbol{\rho}(\mathbf{z}_k, \boldsymbol{\eta}_k)\| > ErrorTolerance$ do
7:         calculate $\boldsymbol{\rho}(\mathbf{z}_k)$ through equation (10)
8:         calculate $\Delta\mathbf{z}_k$ and $\Delta\boldsymbol{\eta}_k$ through equation (11)
9:         initialize searching step $d = 1$
10:        while $\|\boldsymbol{\rho}(\mathbf{z}_k + d\Delta\mathbf{z}_k, \boldsymbol{\eta}_k + d\Delta\boldsymbol{\eta}_k)\| > (1 - \alpha d)\|\boldsymbol{\rho}(\mathbf{z}_k, \boldsymbol{\eta}_k)\|$ do
11:            $d = \beta d$
12:        end while
13:        $\mathbf{z}_k = \mathbf{z}_k + d\Delta\mathbf{z}_k$
14:        $\boldsymbol{\eta}_k = \boldsymbol{\eta}_k + d\Delta\boldsymbol{\eta}_k$
15:    end while
16:    Return $\mathbf{z}_k$
17: end procedure
```

constrained set for the next calls of the MPC solver. One typical case of failure is when the predicted state $\mathbf{z}_{k-1}^*$ does not satisfy the inequality constraints. For instance, when the quadrotor is very close to the boundary of the visibility region and has a large outward velocity (e.g., due to a sudden gust), then the MPC might not prevent the quadrotor from flying outside the constrained region. From the algorithmic perspective, this is due to complex numbers that result from the barrier term and yield useless acceleration control inputs.

Hence, for the real-time implementation of the MPC solver on the quadrotor, we added a mechanism to avoid such a chain of failures after a violation of constraints might occur. This mechanism judges the correctness of the MPC solution, and resets the initial guess of the warm start method in the case when Algorithm 1 fails to satisfy the constraints. More specifically, the mechanism will check the result returned by the MPC solver. If all predicted states and control inputs satisfy the inequality constraints, then these predicted states will be used to generate the initial guess for the next call, and this generated guess is stored. If any failure is detected, then the algorithm will restart with the initial guess that was used in the most recent successful call.

The improved robustness of the algorithm under the restart mechanism can be seen in Fig. 2, where we ran a Monte Carlo test with simulation parameters shown in Table I, in which position and velocity were randomly generated so that they satisfy the inequality constraints. The MPC solver returns an meaningless result at around 12s, when a near-boundary position and a large outward velocity lead to predicted states and inputs that violate the inequality constraints. The solver keeps failing ever after. In contrast, the use of the proposed mechanism adopts the most recent feasible solution. In physical terms, when the quadrotor approaches the boundary of the constrained area, it is expected to apply relatively big control effort that will bring it back to the interior of the constrained area. Even if the derived control input does not prevent the quadrotor from flying outside the constrained zone, keeping the MPC solver functional by applying the previous control inputs increases the chances of getting the quadrotor back into the visibility zone.

TABLE I
SIMULATION PARAMETERS

$\tan\frac{\phi_{\text{camera}}}{2}$	$\tan\frac{\psi_{\text{camera}}}{2}$	μ_u	μ_x
0.2	0.2	0.6	10
x_d	y_d	z_d	$x_{\text{min}}^{\text{simu}}, y_{\text{min}}^{\text{simu}}$
0	0	17	$-3\ m$
$x_{\text{max}}^{\text{simu}}, y_{\text{max}}^{\text{simu}}$	$z_{\text{min}}^{\text{simu}}$	$z_{\text{max}}^{\text{simu}}$	$u_{1\text{max}}, u_{2\text{max}}$
$3\ m$	$17\ m$	$19\ m$	$5\ m/s^2$
$u_{1\text{min}}, u_{2\text{min}}$	$u_{3\text{min}}$	$u_{3\text{max}}$	$v_{\text{max}}^{\text{simu}}$
$-5\ m/s^2$	$-5\ m/s^2$	$7\ m/s^2$	$1.0\ m/s$
$v_{\text{min}}^{\text{simu}}$	N	dT	
$-1.0\ m/s$	10	$0.1s$	

$^*\{\bullet\}^{\text{simu}}$ means the value is generated by program

$^*v_{\text{max}}^{\text{simu}}, v_{\text{min}}^{\text{simu}}$ are bounds of randomly generated velocity in 3 directions

$^*dT$ is the time step of the discrete system

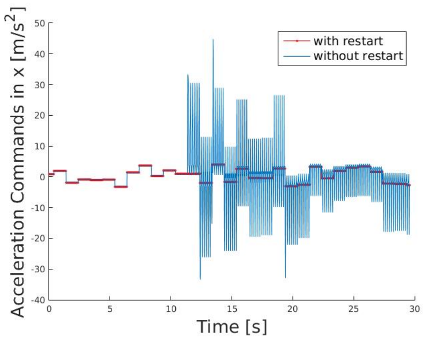


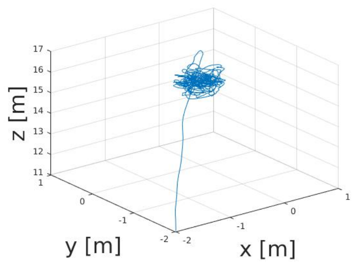
Fig. 2. Algorithm restart contributes to the robustness of the MPC solver.

IV. Simulation Results

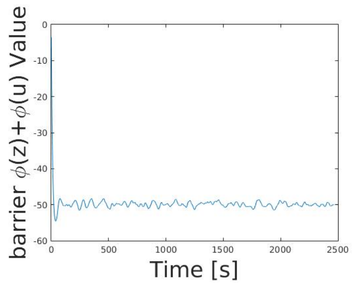
We simulated the response of the system (2) under the proposed MPC strategy. Uniformly distributed noise taking values in the interval $[-0.5, 0.5]$ is added to the state vector $\mathbf{x}_k$ of the system (2), and this noisy state vector is used as state feedback into the model predictive controller. The disturbance vector $\boldsymbol{\omega}_k$ is also uniformly distributed, with values taken in the interval $[-1, 1]$. The control inputs computed by the MPC are applied to (2). The simulated path of the quadrotor is shown in Fig. 3(a), and indicates that the vehicle hovers about the ground within a bounded zone. To demonstrate that the considered constraints were not violated, the value of the sum of the barrier functions (7a), (7b) is plotted in Fig. 3(b). This sum remains a finite real number, hence the states and control inputs remain within their constrained sets.

V. Experiments

The efficacy of the proposed MPC strategy is demonstrated through experiments performed with a quadrotor in an indoor



(a) The simulated path of the quadrotor under the MPC strategy.



(b) The value of the barrier function during the simulated scenario.

lab environment. We used a vicon motion capture system to measure the position of the quadrotor and close the MPC loop, while the integration of ground vision tracking with autonomous aerial control is ongoing work.

The motion capture system provides position measurements with precision 1mm at 100 Hz. The inner-loop controller of the quadrotor runs at 1KHz, processing commands from either a transmitter or XBee communication links. The IMU data is fetched from High Level Process through XBee links at 100 Hz. The navigation loop, described in the following section, processes the acquired data and generates the current state for the MPC solver. The on-line MPC solver runs in MATLAB at about 100 Hz, and is bridged with C/C++ code through Lightweight Communications and Marshalling (LCM) package [32]. The control command is sent through a XBee link connected to Low Level Processor at 100 Hz. `Armadillo`, a C++ linear algebra library, is used in the Kalman filter implementation for matrix operations.

8) Testing Under Perfect State Measurement and no Disturbance Compensation: We initially tested the system under the consideration that the vicon measurements of position and velocity (obtained via differentiating the position data) are nearly perfect. The quadrotor is forced to fly under the MPC strategy and is pushed by hand towards the boundary of the constrained region at given time instants during the flight trial. The applied disturbances are neither compensated for, nor measured, nor taken into consideration into the MPC. The position trajectories are depicted in Fig. 3.

More specifically, the quadrotor is pushed by hand at time $45s$ to a distance 0.07 m from the boundary of the constrained region with outward velocity about 0.09 m/s. The MPC forces the vehicle away from the boundary towards the interior of the constrained area, and results in an increase of its altitude so that a larger bounding area is obtained.

9) Testing Under Noisy Measurements and Disturbances: Vision-based tracking usually results in noisy position estimates, and merely differentiating this data does not provide reliable enough velocity estimates for implementation on the real system. In order to emulate the real (visual tracking) sensing system, we added zero-mean white Gaussian noise of standard deviation 0.25 m, which corresponds to about 8% of half of the width of the effective sensing area that varies between 3.2m and 3.8m, to the vicon position measurements so that their statistical properties are comparable to those of the visual tracking measurements. We implemented the Kalman filter to obtain an estimate of the full state vector. The resulting position trajectories are shown in Fig. 4. The quadrotor is disturbed by sudden pushes occasionally during the flight, however it reacts fast enough and compensates for the disturbance under the proposed MPC strategy, without much drifting in position. Fig.4(a) and Fig.4(b) demonstrate that the quadrotor flies within the constraint bounds indicated with red dash lines.

VI. Conclusions

Using model predictive control with barrier functions to encode state and input constraints, we were able to generate control inputs for a quadrotor so that it maintains its position within a constrained area, despite the effect of a class of external disturbances and noise. The fast MPC solving algorithm with warm start adopted from earlier work was shown to provide fast enough computational speed for real-time implementation, while its robustness was furthermore improved upon strong disturbances by reseting the initial guess of the warm start method in the case when the MPC solver fails to return a feasible solution.

Ongoing and future work includes the consideration of time-varying constrained sensing areas due to the motion of the ground robot, integrating the visual tracking algorithm into the system and performing outdoors flight trials.

References

[1] L. Merino, F. Caballero, J. Dios, and A. Ollero, "Cooperative fire detection using unmanned aerial vehicles," in *Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 1884–1889.

[2] K. Alexis, G. Nikolakopoulos, A. Tzes, and L. Dritsas, "Coordination of helicopter uavs for aerial forest-fire surveillance," in *Applications of intelligent control to engineering systems*, 2009, pp. 169–193.

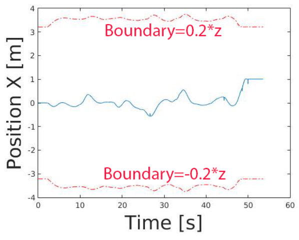
[3] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, 2006.

[4] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, and B. Jung, "Adaptive teams of autonomous aerial and ground robots for situational awareness," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 991–1014, 2007.

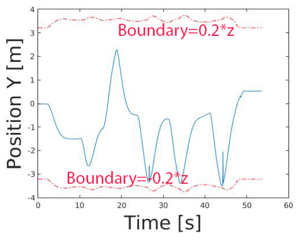
[5] J. Y. Chen and B. R. Clark, "Uav-guided navigation for ground robot operations," in *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, no. 19, 2008, pp. 1412–1416.

[6] V. Dhiman, M. R. Ganesh, W. Ding, D. Panagou, R. N. Severinghaus, and J. J. Corso, "Modeling 3d error propagation from 2d visual tracking with applications in quadrotor tracking and autonomous cars," 2015, $https : //dl.dropboxusercontent.com/u/43986138/ICRA16.pdf$.

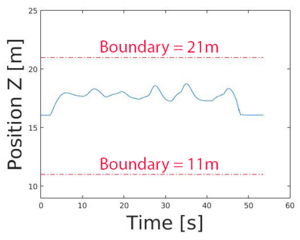
[7] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, 2010.



(c) X History

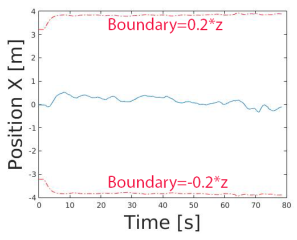


(d) Y History

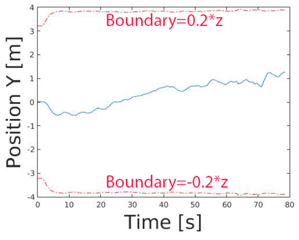


(e) Z History

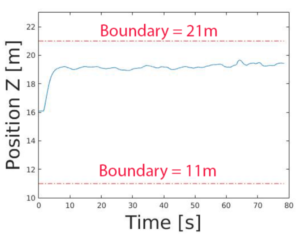
Fig. 3. Position history of real flight under perfect measurements and no disturbance compensation.



(a) X History



(b) Y History



(c) Z History

Fig. 4. Position history of real flight under disturbances and noise.

[8] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," *AIAA*, 2008.

[9] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *IEEE/RSJ Int. Conf. on Int. Robots and Systems*, 2007, pp. 153–158.

[10] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE Int. Conf. on Robotics and Automation*, vol. 5, 2004, pp. 4393–4398.

[11] M. Cutler and J. P. How, "Actuator constrained trajectory generation and control for variable-pitch quadrotors," in *AIAA Guidance, Navigation, and Control Conference*, 2012.

[12] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Constrained-control of a quadrotor helicopter for trajectory tracking under wind-gust disturbances," in *15th IEEE Mediterranean Electrotechnical Conference*, 2010, pp. 1411–1416.

[13] ——, "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances," *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[14] C. Liu, W.-H. Chen, and J. Andrews, "Piecewise constant model predictive control for autonomous helicopters," *Robotics and Autonomous Systems*, vol. 59, no. 7-8, pp. 571–579, July 2011.

[15] Z. Chao, S.-L. Zhou, L. Ming, and W.-G. Zhang, "UAV formation flight based on nonlinear model predictive control," *Mathematical Problems in Engineering*, pp. 1–16, 2012.

[16] M. Saska, Z. Kasl, and L. Preucil, "Motion planning and control of formations of micro aerial vehicles," in *19th IFAC World Congress*, Aug. 2014, pp. 1228–1233.

[17] T. S. Y. Kuwata, A. Richards and J. How, "Distributed robust receding horizon control for multivehicle guidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 627–641, 2007.

[18] Y. Kuwata and J. How, "Cooperative distributed robust trajectory optimization using receding horizon MILP," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 2, pp. 423–431, 2011.

[19] T. H. S. Stephan M. Huck, Marvin Rueppel and J. Lygeros, "Rcopterx - experimental validation of a distributed leader-follower MPC approach on a miniature helicopter test bed," in *2014 European Control Conference*, Strasburg, France, June 2014, pp. 802–806.

[20] S. Maniatopoulos, D. Panagou, and K. J. Kyriakopoulos, "A MPC scheme for the navigation of a nonholonomic vehicle with field-of-view constraints," in *Proc. of the 2013 American Control Conf.*, Washington DC, USA, June 2013, pp. 3967–3972.

[21] N. Sydney, B. Smyth, and D. Paley, "Dynamic control of autonomous quadrotor flight in an estimated wind field," in *52nd Conference on Decision and Control*, 2013, pp. 3609–3616.

[22] F. Schiano, J. Alonso-Mora, K. Rudin, P. Beardsley, R. Siegwart, and B. Siciliano, "Towards estimation and correction of wind effects on a quadrotor uav," in *International Micro Air Vehicle Conference and Competition 2014*, Aug. 2014.

[23] C. Hancer, K. T. Oner, E. Sirimoglu, E. Cetinsoy, and M. Unel, "Robust position control of a tilt-wing quadrotor," in *49th IEEE Conference on Decision and Control*, 2010, pp. 4908–4913.

[24] P. TøNdel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit mpc solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.

[25] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[26] K. Edlund, L. E. Sokoler, and J. Jrgensen, "A primal-dual interior-point linear programming algorithm for mpc," in *48th IEEE Conference on Decision and Control*, 2009, pp. 351–356.

[27] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 281–302, 2000.

[28] E. A. Yildirim and S. J. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM Journal on Optimization*, vol. 12, no. 3, pp. 782–810, 2002.

[29] 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.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[31] A. Shahzad, E. C. Kerrigan, and G. A. Constantinides, "A warm-start interior-point method for predictive control," 2010.

[32] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 4057–4062.