Vision-Based Optimal Landing On a Moving Platform

Takuma Nakamura
takuma.nakamura@gatech.edu
Graduate Research Assistant
Georgia Institute of Technology
Atlanta, GA, USA

Stephen Haviland
shaviland3@gatech.edu
Graduate Research Assistant
Georgia Institute of Technology
Atlanta, GA, USA

Dmitry Bershadsky
dbershadsky@gatech.edu
Graduate Research Assistant
Georgia Institute of Technology
Atlanta, GA, USA

Eric N. Johnson
eric.johnson@ae.gatech.edu
Associate Professor
Georgia Institute of Technology
Atlanta, GA, USA

ABSTRACT

This paper describes a vision-based control architecture designed to enable autonomous landing on a moving platform. The landing trajectory is generated by using the receding-horizon differential dynamic programming (DDP), an optimal control method. The trajectory generation is aided by the output of a vision-based target tracking system. The vision system uses multiple extended Kalman filters which allows us to estimate the position and heading of the moving target via the observed locations. The combination of vision-based target tracking system and the receding-horizon DDP gives an unmanned aerial vehicle the capability to adaptively generate a landing trajectory against tracking errors and disturbances. Additionally, by adding the exterior penalty function to the cost of the DDP we can easily constrain the trajectory from collisions and physically infeasible solutions. We provide key mathematics needed for the implementation and share the results of the image-in-the-loop simulation and flight tests to validate the suggested methodology.

INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have become omnipresent, and been recognized as a solution for a wide range of problems in many realms: surveillance, photo/videography, and delivery. Many tasks that are currently operated by manned vehicles could be replaced by UAVs, which would avoid endangering the life of human pilots and limit the man power required for the mission. However, the autonomy of UAVs is currently limited to simple tasks and often require human in the loop presence. Landing is one of the maneuvers that typically needs humans in the loop for multiple purposes: recognizing the landing spot, checking for obstacles, controlling the vehicle position highly precisely, and aborting the maneuver. UAVs equipped with the capability to automatically detect the landing target and generate a landing trajectory will enable various complicated tasks. For example, we can use a UAV to let it land on an uninstrumented truck or a simple object that does not communicate with external entities. Another scenario could be a fire engine launching UAVs to let them collect data from disaster zones and relaying the information back to rescuers to aid in recovery. In this work, we suggest a system architecture that enables vision-based landing on a static/moving platform using the target recognition system and receding-horizon differential dynamics programming (DDP).

The first part of this work is the development of the vision system that allows us to estimate the dynamics of a moving target through the observed position. We use the Haar-like feature detector to detect our landing place, which is a typical helipad or landing pad that consists of the letter "H" surrounded by a circle. There are many methodologies available to detect this sort of a distinct object such as to use a line detection with Hough transform (Ref. 1), feature point detection with corner Harris algorithm (Ref. 2), and Lucas-Kanade algorithm (Ref. 3) to name a few, but a complete evaluation of each method is far beyond the scope of this work. When a target is very specific and asymmetric, we have a method to measure the heading of the target just with a single static image, e.g. using AprilTags (Ref. 4), but we consider a method that is applicable to a more general target. Some researchers used the estimator for a vison-based maneuver (Ref. 5), (Ref. 6), (Ref. 7) whose approaches augment the state of the estimator to simultaneously estimate the vehicle states. Our estimator is configured differently to estimate the additional states of the target, and also we use the statistics obtained from the filter to solve the corresponding problem (i.e. which measurement is used to update which filter) and to initiate/abort an operation.

The second part of this work is motion planning for landing. Our approach is to use receding horizon DDP to generate an optimal landing trajectory. The DDP efficiently converges

Draft

to a local optimum, and gives open and closed loop control policies; therefore, it is suitable for real-time applications. In fact, it is already used for suspended load operations (Ref. 8), humanoid robots (Ref. 9) and many other robotics applications. Landing maneuvers are sensitive to uncertainties and disturbances which can cause the UAV to diverge from the desired trajectory. By using the receding horizon DDP we have the ability to recompute the trajectory throughout the landing maneuver. Saripalli et al achieved an autonomous landing on a moving target using a cubic polynomial (Ref. 10); however, this method lacks in the adaptivity against uncertainty that happens at some time over the future interval. Also, the efficiency of a polynomial-based method is highly dependent on the initial condition; the receding horizon DDP, on the other hand, is less dependent. Second, the receding horizon DDP gives the UAV the capability to optimize the landing time. When there is too little time before achieving landing, the vehicle should not attempt the landing. The receding horizon approach purely navigates the vehicle to the optimal points in the near future, and does not force the vehicle to land with aggressive control sequences. Third, the constraints on a trajectory such as collision avoidance and flight envelope are easily considered by using a constrained optimization technique. In this work, we use exterior quadratic penalty function (EQPF) (Ref. 11), but other methods (e.g. multiplier penalty function) can solve constrained optimization problems.

In this paper, we develop a fully autonomous self-contained vision-based quadrotor for landing. Simulation and experimental results are provided to validate the suggested methodology, and the full formulation of control architecture is described. The several key contributions presented in this paper are:

1. Development of the vision system that can track a moving target.
2. Derivation of the control architecture to enable an optimal landing.
3. Demonstration of the suggested landing methodology through successful simulations and flight tests.

This work is an extension of the results reported in our previous paper (Ref. 12) that describes static target tracking; therefore, we omit some of the details addressing the hardware and software implementation.

TARGET TRACKER

This section explores our methodology of target tracking. Our target tracker processes the image to extract the target candidate points, and the raw outputs are plugged into multiple extended Kalman filters (EKFs) to estimate the target position and heading. Utilizing the EKF has some advantages in tracking a moving target, one of the most significant of which is that we can propagate the desired waypoint even when we have no outputs from the image processing and/or the target is out of sight. This would allow us to re-establish the tracking. However, updating the way points without any measurement is dangerous at some point. Another benefit of using EKFs is that it can give us criteria about when we should stop updating the way point. We use the statistics of the filter to initiate/abort a maneuver. Also, the corresponding problem is solved using one of the statistics of the filter, the covariance of the residual.

Target recognition

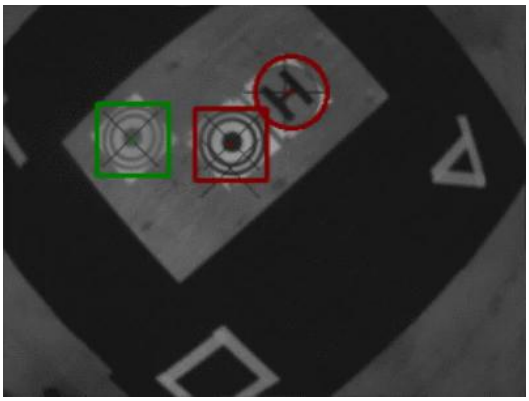


Fig. 1. The figure shows the results of the target tracking with the Haar-like feature detector. The target tracker can store up to 20 measurements and run up to 3 independent EKFs, each of which appears with a different color. The rectangles are the outputs from the double-red-circle finding, and the circle is from the helipad finding. The sizes of the rectangles and the circle correspond to the estimated dimensions of the target as shown in (1).

We use the Haar-like feature detector (Ref. 13) for target recognition. This method is widely used for face detection, and one of the most popular machine-learning-based template matching method. The quality of this method is highly dependent on the time used to train a classifier and the quantity of the resources. This method is one of the feature-based approaches to object classification; therefore, it typically works faster than the pixel based (Ref. 14). It is still computationally too expensive for a real-time application, but we boost the processing by limiting the size of the pixels searched in the algorithm. We assume that we know the target height, true target dimensions, and the camera position with respect to the center of gravity of the vehicle; therefore, we can calculate the distance to the target. Using the distance we can estimate the size of the target in the camera frame (a, b[pixel]) as follows:

$$a = -\frac{\text{Width}A}{2^b\hat{Z}_i^i \tan(\frac{\gamma_x}{2})}, b = -\frac{\text{Width}B}{2^b\hat{Z}_i^i \tan(\frac{\gamma_y}{2})}, \quad (1)$$

where A, B(ft) are the dimensions of the target, γ_x, γ_y are the horizontal and vertical field of view, respectively and ${}^b\hat{Z}_i^i$ is the z-direction position of the vehicle (b, left upper-script) with respect to the origin of the inertial frame (i, right upper-script) expressed in the inertial frame (i, subscript). The same notation is used throughout this paper. The classifier only searches

the features with the estimated size with k pixel margins, i.e., the detection starts with the smallest classifier size $a-k, b-k$ and ends with $a+k, b+k$.

Discrete EKF

We design an estimator for a constant-speed target. The nonlinear dynamics of the target is expressed as follows:

$$f(\mathbf{x}) = \begin{bmatrix} {}^tV cos({}^t\psi) \\ {}^tV sin({}^t\psi) \\ 0 \\ w \end{bmatrix}, \tag{2}$$

where $w_k \sim N(0, Q_k)$. The state vector of our estimator is as follows, and we estimate the position and the heading of the target:

$$\mathbf{x} = \begin{bmatrix} {}^tX_i^i & {}^tY_i^i & {}^tZ_i^i & {}^t\psi \end{bmatrix}^T. \tag{3}$$

We measure target positions through the Haar-like feature detector, and the measurements are expressed as follows:

$$z_i = \begin{bmatrix} u \\ v \end{bmatrix}_i + v_k = \begin{bmatrix} f_x \frac{{}^t\hat{Y}_c^c}{{}^t\hat{X}_c^c} \\ f_y \frac{{}^t\hat{Z}_c^c}{{}^t\hat{X}_c^c} \end{bmatrix} + v_k, \tag{4}$$

where v_k is measurement noise, which is a zero-mean Gaussian random variable: $v_k \sim N(0, R_k)$, and i is the index for the measurements at the same time. Since the Haar-like feature detector provides multiple locations including false hits, up to 20 measurements are stored, and the best measurements are plugged into the Kalman filter updates. We choose a measurement for the update using the residual covariance, which is described in the following subsection. The covariance matrix, P, is propagated using the discrete Lyapunov equation, and the states are propagated using the nonlinear model (2). In the update phase, the Kalman gain matrix is obtained from the equation below:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}. \tag{5}$$

The state and covariance matrix are updated with the equations below:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k)), \tag{6}$$

$$P_k^+ = (I - K_k H_k) P_k^-, \tag{7}$$

The h function is a nonlinear function that maps a position expressed in the inertial frame to a position expressed in the camera frame. We calculate the residual with this nonlinear function to avoid the issue caused by linearization, but the output matrix H requires linearization. We calculate the H matrix with the standard pinhole camera model. r_i and r_c denote the position of the target in the inertial frame and the one expressed in the camera frame, respectively. Then,

$$H = \begin{bmatrix} \frac{\partial z}{\partial r_c} \frac{\partial r_c}{\partial r_i}|_{x=\hat{x}} & 0 \end{bmatrix} \tag{8}$$

$$= \begin{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial {}^t\hat{X}_c^c} & \frac{\partial u}{\partial {}^t\hat{Y}_c^c} & \frac{\partial u}{\partial {}^t\hat{Z}_c^c} \\ \frac{\partial v}{\partial {}^t\hat{X}_c^c} & \frac{\partial v}{\partial {}^t\hat{Y}_c^c} & \frac{\partial v}{\partial {}^t\hat{Z}_c^c} \end{bmatrix} L_{ci} & 0 \end{bmatrix} \tag{9}$$

$$= \begin{bmatrix} \begin{bmatrix} -f_x \frac{{}^t\hat{Y}_c^c}{{}^t\hat{X}_c^c} & f_x \frac{1}{{}^t\hat{X}_c^c} & 0 \\ -f_y \frac{{}^t\hat{Z}_c^c}{{}^t\hat{X}_c^c} & 0 & f_y \frac{1}{{}^t\hat{X}_c^c} \end{bmatrix} L_{ci} & 0 \end{bmatrix}, \tag{10}$$

where f_x, f_y is the focal length of the camera expressed as follows:

$$f_x = \frac{\text{Width}}{2\tan(\frac{\gamma_x}{2})}, \quad f_y = \frac{\text{Height}}{2\tan(\frac{\gamma_y}{2})}, \tag{11}$$

and the rotation matrix from camera to inertial is denoted $L_{ic} = L_{ci}^T$.

Corresponding Problem

We use the residual and its covariance to solve the corresponding problem using the Z-test. Since we have three EKFs and an undefined number of outputs from the Haar-like feature detector, we need to choose a measurement for the update and which EKF the measurement goes to. The covariance of the residual can be computed as follows:

$$\begin{aligned} E[r_k r_k^T] &= E[(z_k - H_k\hat{x}_k^-)(z - H_k\hat{x}_k^-)^T] \\ &= E[\{H_k(x_k - \hat{x}_k^-) + v_k\}\{H_k(x_k - \hat{x}_k^-)^T + v_k\}^T] \\ &= H_k E[\varepsilon_k \varepsilon_k^T] H_k^T + E[v_k v_k^T] \\ &= H_k P_k^- H_k^T + R_k. \end{aligned} \tag{12}$$

Therefore, the Z value, also called the Mahalanobis distance, is defined as follows:

$$Z = r_k^T (H_k P_k H_k^T + R_k)^{-1} r_k. \tag{13}$$

In our case, this gives a measure regarding how much a measurement is separated from the current estimation. The measurement with the smallest Z-value is used for each of the EKF updates. We set a constant threshold for the Z value; thus, when the estimation is less confident, the measurement with a large residual is allowed for the update, and vice versa.

MOTION PLANNING

In this section, we provide our implementation of the differential dynamic programming (DDP) and introduce one way to deal with a constrained optimization, an exterior quadratic penalty function (EQPF). A full description of DDP algorithm can be found in (Ref. 15), and a DDP with inequality constraints is discussed in (Ref. 11).

Differential Dynamic Programming

We consider the cost minimization that involves finding a control function $\mathbf{u}(t)$ below.

$$\min_{\mathbf{u}}[\phi(\mathbf{x}(t_f),t_f)+\int_{t_0}^{t_f} L(\mathbf{x},\mathbf{u},t)dt] \tag{14}$$

subject to the dynamics

$$\frac{d\mathbf{x}}{dt}=f(\mathbf{x},\mathbf{u},t),\mathbf{x}_0=\mathbf{x}(t_0). \tag{15}$$

Our DDP takes the state vector of

$$\mathbf{x}=\begin{bmatrix}{}^bX_i & {}^bY_i & {}^bZ_i & {}^b\dot{X}_i & {}^b\dot{Y}_i & {}^b\dot{Z}_i\end{bmatrix}^T, \tag{16}$$

and the control vector is

$$\mathbf{u}=\begin{bmatrix}{}^b\ddot{X}_i & {}^b\ddot{Y}_i & {}^b\ddot{Z}_i\end{bmatrix}^T. \tag{17}$$

We utilize the standard "Q function" notation from reinforcement learning, which is expressed as below:

$$Q(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k)=L(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k)+V(\mathbf{x}(t_{k+1}),t_{k+1})). \tag{18}$$

where $L(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k)$ is the running cost, and $V(\mathbf{x}(t_{k+1}),t_{k+1})$ is the cost-to-go. Here, the running cost is expressed as:

$$\begin{aligned}L(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k)=&\frac{1}{2}(\mathbf{x}(t_k)-\mathbf{x}^*)^TQ_w(\mathbf{x}(t_k)-\mathbf{x}^*)\\&+\frac{1}{2}\mathbf{u}(t_k)R_w\mathbf{u}(t_k),\end{aligned} \tag{19}$$

where Q_w is positive semidefinite, and R_w is positive definite, both of which are diagonal matrices and decide the weight of states and control, respectively. The target vector $\mathbf{x}^*$ is computed by using the output of the target tracker described in the prevous section and known target information: target height and velocity.

$$\mathbf{x}^*=\begin{bmatrix}{}^t\hat{X}_i+{}^tV\cos({}^t\hat{\psi})t_f & {}^t\hat{Y}_i+{}^tV\sin({}^t\hat{\psi})t_f & {}^tZ_i\end{bmatrix}^T, \tag{20}$$

where ${}^t\hat{X}_i,{}^t\hat{Y}_i$, and ${}^t\hat{\psi}$ are the output of the target tracker. According to Bellman's principle of optimality, any sub-trajectory of an entire optimal trajectory is also optimal. This provides us with one of the essential qualities of the DDP: backward integration. By using a quadratic expansion of the cost-to-go at each stage which is achieved by a second-order Taylor series, we can express the backward path as follows. For conventional notations, (t_k) signifies that the function is evaluated at $(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k)$.

$$Q_x(t_k)=\nabla_xL(t_k)+\nabla_xV(t_k)\nabla_xf(t_k), \tag{21}$$

$$Q_u(t_k)=\nabla_uL(t_k)+\nabla_xV(t_k)\nabla_uf(t_k), \tag{22}$$

$$Q_{xx}(t_k)=\nabla_{xx}L(t_k)+\nabla_xV(t_k)\nabla_{xx}f(t_k)+\nabla_xf(t_k)^T\nabla_{xx}V(t_k)\nabla_xf(t_k), \tag{23}$$

$$Q_{ux}(t_k)=\nabla_{ux}L(t_k)+\nabla_xV(t_k)\nabla_{ux}f(t_k)+\nabla_uf(t_k)^T\nabla_{xx}V(t_k)\nabla_xf(t_k), \tag{24}$$

$$Q_{uu}(t_k)=\nabla_{uu}L(t_k)+\nabla_xV(t_k)\nabla_{uu}f(t_k)+\nabla_uf(t_k)^T\nabla_{xx}V(t_k)\nabla_uf(t_k), \tag{25}$$

$$V_x(t_{k-1})=Q_x(t_k)-Q_x(t_k)Q_{uu}^{-1}(t_k)Q_{ux}(t_k), \tag{26}$$

$$V_{xx}(t_{k-1})=Q_{xx}(t_k)-Q_{xu}(t_k)Q_{uu}^{-1}(t_k)Q_{ux}(t_k). \tag{27}$$

Plus, we obtain our feedback and feedforward policies (l_{fb} and l_{ff}) as follows:

$$l_{fb}=Q_{uu}^{-1}(t_k)Q_{ux}(t_k), \tag{28}$$

$$l_{ff}=Q_{uu}^{-1}(t_k)Q_u(t_k). \tag{29}$$

These two policies makes the update for control ($\delta\mathbf{u}$),

$$\delta\mathbf{u}=l_{ff}+l_{fb}\delta\mathbf{x}(t_k), \tag{30}$$

and the forward path is generated by a new control sequence

$$\mathbf{u}_{\text{new}}=\mathbf{u}+\gamma\delta\mathbf{u}, \tag{31}$$

where γ is a learning rate. We set the learning rate to 0.8, and the trajectories are evaluated $N=40$ times at each update. Among the N trajectories, the trajectory which has the minimum cost is used for position control, and cascaded to the attitude controller.

Receding Horizon Control

In our controller, the DDP updates the trajectory at 10 Hz and evaluates the trajectory up to 1 second ahead of time; that is, our terminal time t_f is 1 second. Since the DDP update rate is 10 Hz, we apply only the first 10% of the control sequence. Receding horizon control is known to be more robust against uncertainty and disturbances than fixed horizon control. The DDP stops updating the trajectory when the vehicle can land on the target in t_f, which is

$$|{}^bX_i(t_f)-x_1^*|\le\varepsilon_x,|{}^bX_i(t_f)-x_2^*|\le\varepsilon_y,|{}^bZ_i(t_f)-x_3^*|\le\varepsilon_z, \tag{32}$$

where $\varepsilon_x,\varepsilon_y$ and ε_z are the design parameters that decide the position error threshold for the landing. In our implementation, $\varepsilon_x=\varepsilon_y=\varepsilon_z=0.2$ ft.

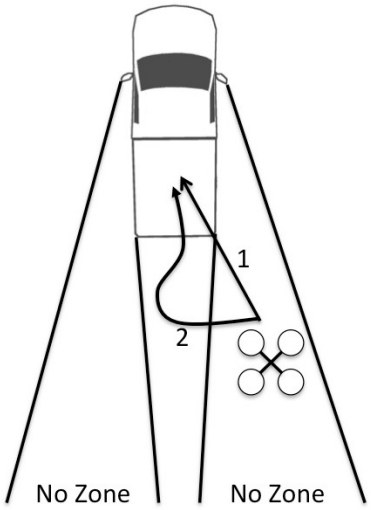


Fig. 2. The figure shows an example of the difference between constrained and unconstrained trajectory planning. Suppose a UAV attempts to land on an truck by avoiding the line of sight of the driver. The unconstrained optimal trajectory (shown as 1) is to fly directly to the target, which results in the minimum control and state costs. However, when we penalize the states in the line of sight of the driver, the UAV prioritizes going outside of the line of sight, and then approaches the target as shown in 2.

Exterior Quadratic Penalty Function

Trajectory planning for landing requires considering some constraints: avoiding a collision with the ground, aggressive landing, and the line of sight of a driver. Fig. 2 shows an example of constrained and unconstrained optimal trajectory. The constrained optimization in DDP can be dealt with by various methods. In this work, the constraints are treated with an EQPF. We define inequality constraints as follows:

$$g_i(\mathbf{x}(t)) \leq 0, (i = 1,2,\cdots,n), \tag{33}$$

where n is the number of constraints. Then, the below represents the augmented integrand of the running cost.

$$\tilde{L}(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k) = L(\mathbf{x}(t_k),\mathbf{u}(t_k),t_k) + w\sum_{i=1}^{n} A_i g_i^2(\mathbf{x}(t)), \tag{34}$$

where $A_i = 0$ if the g_i constraint is satisfied, and $A_i = 1$ otherwise. When all the constraints are satisfied, the above cost function is equivalent to the unconstrained cost defined in (19). The weight of the penalty function is determined by w. The DDP derivation shown above needs a slight modification to take the penalty function into consideration as follows:

$$\tilde{Q} = Q + w\sum_{i=1}^{n} A_i g_i^2, \tag{35}$$

$$\tilde{Q}_x = Q_x + 2w\sum_{i=1}^{n} A_i g_i \nabla_x g_i, \tag{36}$$

and

$$\tilde{Q}_{xx} = Q_{xx} + 2w\sum_{i=1}^{n} A_i (g_i \nabla_{xx} g_i + \nabla_x g_i \nabla_x g_i^T), \tag{37}$$

CONTROL

Our controller has the attitude controller and the position controller; each of which runs at 1000 Hz and 100 Hz, respectively. The position controller is switched between hover and 3D-trajectory control depending on the phases of the landing. Before we send the landing command to the vehicle, we use the hover control to chase the moving target. Once it has received the landing command, the vehicle follows the 3D trajectory generated by the receding horizon DDP as described in the previous section. In this section, we present our designs of the attitude and position controllers. Although the coordinates are defined differently, and the hover target is static, the basic controller design follows the standard quadrotor controller described in (Ref. 16)

Position Control

The hover controller uses the PID controller to chase a moving target. We know the velocity of the moving target, and that it is constant. The target tracker described in the section outputs the position and heading of the target; thus, we can line up the position and velocity of the vehicle (denoted ${}^b\mathbf{r}$ and ${}^b\dot{\mathbf{r}}$) with the ones of the target (denoted ${}^t\mathbf{r}$ and ${}^t\dot{\mathbf{r}}$). Then, the command acceleration can be expressed as follows:

$$\ddot{\mathbf{r}}^d = k_p({}^t\mathbf{r} - {}^b\mathbf{r}) + k_d({}^t\dot{\mathbf{r}} - {}^b\dot{\mathbf{r}}) + k_i\int({}^t\mathbf{r} - {}^b\mathbf{r}), \tag{38}$$

where k_p, k_d and k_i denotes the gain for proportional, derivative, and integral controller. Here, ${}^t\mathbf{r}$ and ${}^t\dot{\mathbf{r}}$ are calculated as follows:

$${}^t\mathbf{r} = \begin{bmatrix} {}^tX & {}^tY & {}^tZ \end{bmatrix}^T, \tag{39}$$

$${}^t\dot{\mathbf{r}} = \begin{bmatrix} {}^tV\cos({}^t\psi) & {}^tV\sin({}^t\psi) & 0 \end{bmatrix}^T. \tag{40}$$

We assume the height of the landing platform ${}^tZ$ is known. Those 3D accelerations (38) are used to calculate the desired attitude of the vehicle, which is described in the following subsection. After the landing command is sent, the optimal control sequence $\mathbf{u}$ from the receding horizon DDP is substituted for the desired acceleration instead.

Attitude Control

Let $\ddot{\mathbf{r}}^d = \begin{bmatrix} \ddot{r}_1^d & \ddot{r}_2^d & \ddot{r}_3^d \end{bmatrix}^T$; then, by using the linearized dynamics we can lead the desired accelerations to the following expressions:

$$\ddot{r}_1^d = g(\theta^d\cos{}^b\psi + \phi^d\sin{}^b\psi), \tag{41}$$

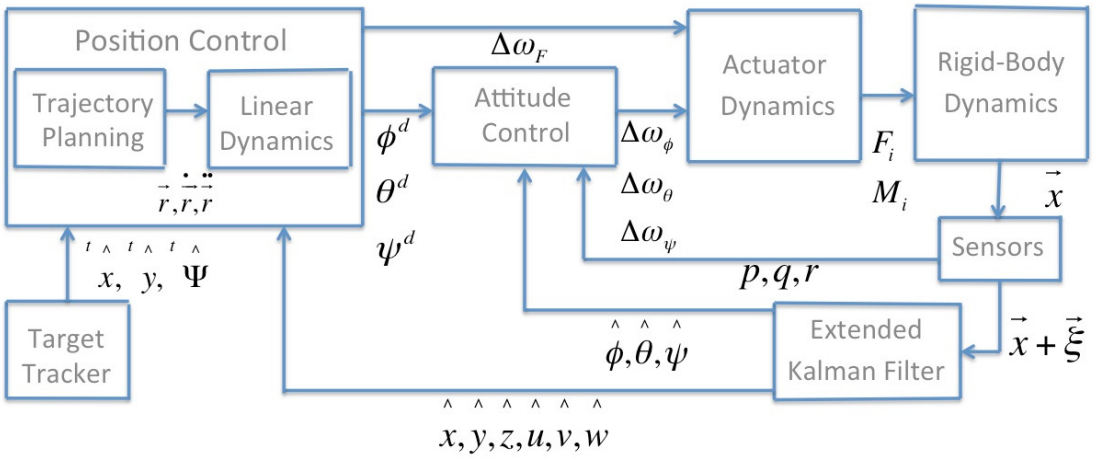


Fig. 3. The figure displays the control architecture we suggest. The inner attitude control loop uses onboard gyros and estimates of attitude to control the roll, pitch, and yaw angle, and runs at 1000 Hz. The outer position control loop uses the estimates of position and velocity, and runs at 100 Hz. The trajectory planning uses the output from the target tracker, and computes the trajectory using the receding horizon DDP that achieves landing.

$$\ddot{r}_2^d = g(\theta^d \sin^b \psi - \phi^d \cos^b \psi), \tag{42}$$

and

$$\ddot{r}_3^d = \frac{8k_F\omega_h}{m}\Delta\omega_F. \tag{43}$$

By solving the equations for ϕ^d, θ^d and $\Delta\omega_F$ we can obtain the desired attitude and throttle that achieves the commanded acceleration.

$$\phi^d = -\frac{1}{g}(\ddot{r}_1^d \sin^b \psi + \ddot{r}_2^d \cos^b \psi), \tag{44}$$

$$\theta^d = \frac{1}{g}(\ddot{r}_1^d \cos^b \psi + \ddot{r}_2^d \sin^b \psi), \tag{45}$$

$$\Delta\omega_F = -\frac{m}{8k_F\omega_h}\ddot{r}_3^d. \tag{46}$$

Since the heading of the vehicle is not determined in the desired acceleration (in other words, we can achieve the specified acceleration with any heading), the heading is our direct design parameter for attitude control. One design is that we choose the heading of the vehicle so that the target velocity vector is aligned with the lateral direction of an onboard image. This way, we would have more chance to keep the target in the line of sight. In this work, the vehicle stays in the initial heading. This method is efficient because it does not require extra control. Let $\Delta\omega_\phi, \Delta\omega_\theta$ and $\Delta\omega_\psi$ are the deviations from a nominal attitude; then, they are computed as follows using the PD controller:

$$\Delta\omega_\phi = k_{p,\phi}(\phi^d -^b \phi) + k_{d,\phi}(p^d -^b p), \tag{47}$$

$$\Delta\omega_\theta = k_{p,\theta}(\theta^d -^b \theta) + k_{d,\theta}(q^d -^b q), \tag{48}$$

$$\Delta\omega_\psi = k_{p,\psi}(\psi^d -^b \psi) + k_{d,\psi}(r^d -^b r). \tag{49}$$

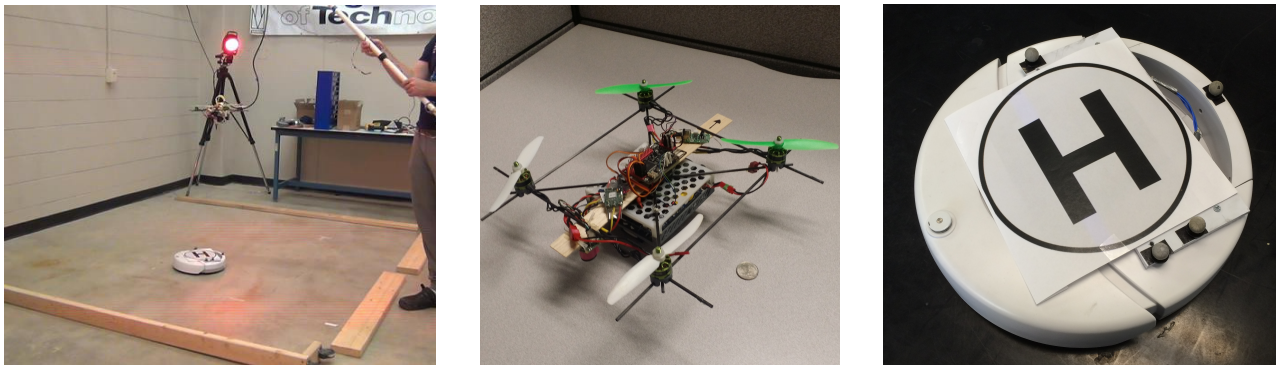
These commands produce moments by changing the desired rotor speeds. The conversion to the rotor speed varies depending on how we defined the body axis. When we use the standard "+" quadrotor configuration, that is, the body x and y axes lie on the legs of the quadrotor, two rotors control the vehicle roll, and the other two control the pitch. In the "+" configuration the rotor speeds can be expressed as the linear combinations of (46), (47), (48), and (49).

$$\begin{bmatrix} {}^d\omega_1 \\ {}^d\omega_2 \\ {}^d\omega_3 \\ {}^d\omega_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 1 \\ 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_h + \Delta\omega_F \\ \Delta\omega_\phi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix}, \tag{50}$$

where ω_h is a rotor speed needed to sustain hover.

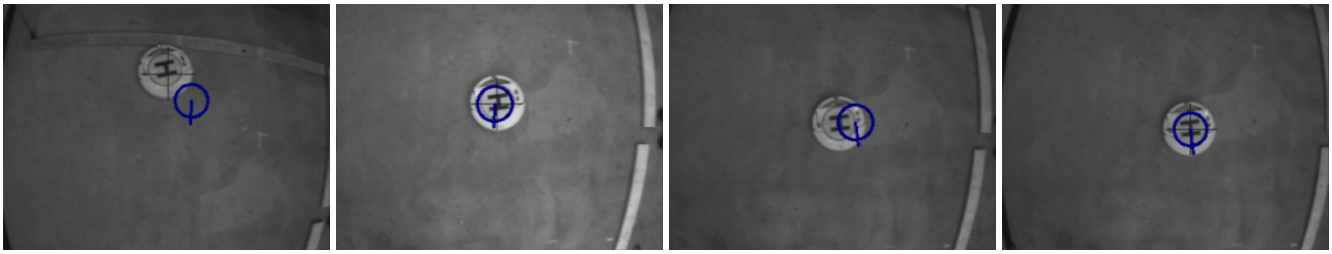
SIMULATION AND FLIGHT TEST

In this section, we provide our flight test resutls to validate the suggested vision system. Also, we provide the results of an image-in-the-loop simulation using the vision system to test the entire landing. We have operated the flight test in our indoor flight facility (shown in Fig. 4-(a)) where the Vicon system is installed so that we can measure the position and attitude of our aerial vehicle and our moving target. We have developed the GTQ-Mini (shown in Fig. 4-(b)) for the 2015 American Helicopter Society Micro Air Vehicle (MAV) Student Challenge (Ref. 17), and this vehicle is used for flight tests. This MAV vehicle is less than 450 mm in length in any dimension and weighs under 500 grams. It is able to fly without any external aide such as the Vicon cameras, but we



(a) (b) (c)

Fig. 4. The figure (a) shows our indoor flight facility, in which the GTQ-mini (shown in (b)) is in flight with a safety cable. Our target is an iRobot Create (shown in (c)), which is powered by Arduino and can be remote-controlled. Both of the GTQ-mini and iRobot Create have the reflection balls with them, and the position and attitude are recorded by the Vicon system.



(a) (b) (c) (d)

Fig. 5. The figures show the results of the flight test. The vertical and horizontal crossing is the measurement from the Haar-like feature detector. The position of the circle represents the estimated position, the radius of the circle corresponds to the estimated size of the target, and the arrow signifies the estimated heading of the target. After obtaining the consistent measurements, the system initialized the EKF at (a). The initial position estimate is the mean of the measurement we obtained before the initialization. The figure (b) shows that we have estimated the position and the heading of the target accurately. While there is no measurement (shown in (c)), the estimation propagated using the known dynamics of the target. This results in the deviation in estimation from the actual position and heading. After we reconfigure the measurement, the estimation established a more correct estimation.

have operated in the Vicon room to record the data. We a use Gigabyte Brix[1] as the onboard computer which runs on Ubuntu 14.04. The vehicle has a downward facing Firefly-MV[2] USB monocular camera. We have a software framework, the Georgia Tech UAV simulation tool (GUST)[3], which allows us to obtain an onboard image, connect to the Vicon system, and record relevant data. GUST separates the computer vision system from the inner loop navagation, running both in a seperate thread; thus, the vehicle control is achieved at a constant frequency. We use an iRobot Create (shown in Fig. 4-(c)) for a moving target, which carries the landing pad with the letter "H". This is powered by Arduino and can be remotely controlled. Fig. 5 shows the results of the target detection and estimation. We ran the target in a constrained area, and the target reverses the heading when it hits the boundary. Since we only measure the position of the target and estimate the heading of the target through the position measurement, our estimator deviated from the true position while the target is spinning to change the heading. In other words, we cannot detect when the target entered the spin at this time. We used the vehicle position and attitude obtained from the Vicon system, and the onboard images. The measurement was not as consistent as we obtain in the simulation, but it was successful in finding the target most of time, and we were able to estimate the heading as well as the position as shown in Fig. 5. We use this vision system to test the suggested landing methodology.

Now, we test the case where a UAV chases a moving target vehicle from behind and attempts landing, subject to constrained approaches. We start the simulation in which the target is in the line of sight of the vehicle; thus, we do not consider how to find a target outside the camera's field of view. First, the UAV establishes the position and heading estima-

[1] **http://www.gigabyte.us/**

[2] **http://www.ptgrey.com/**

[3] **http://www.uavrf.gatech.edu/platforms/gust/**

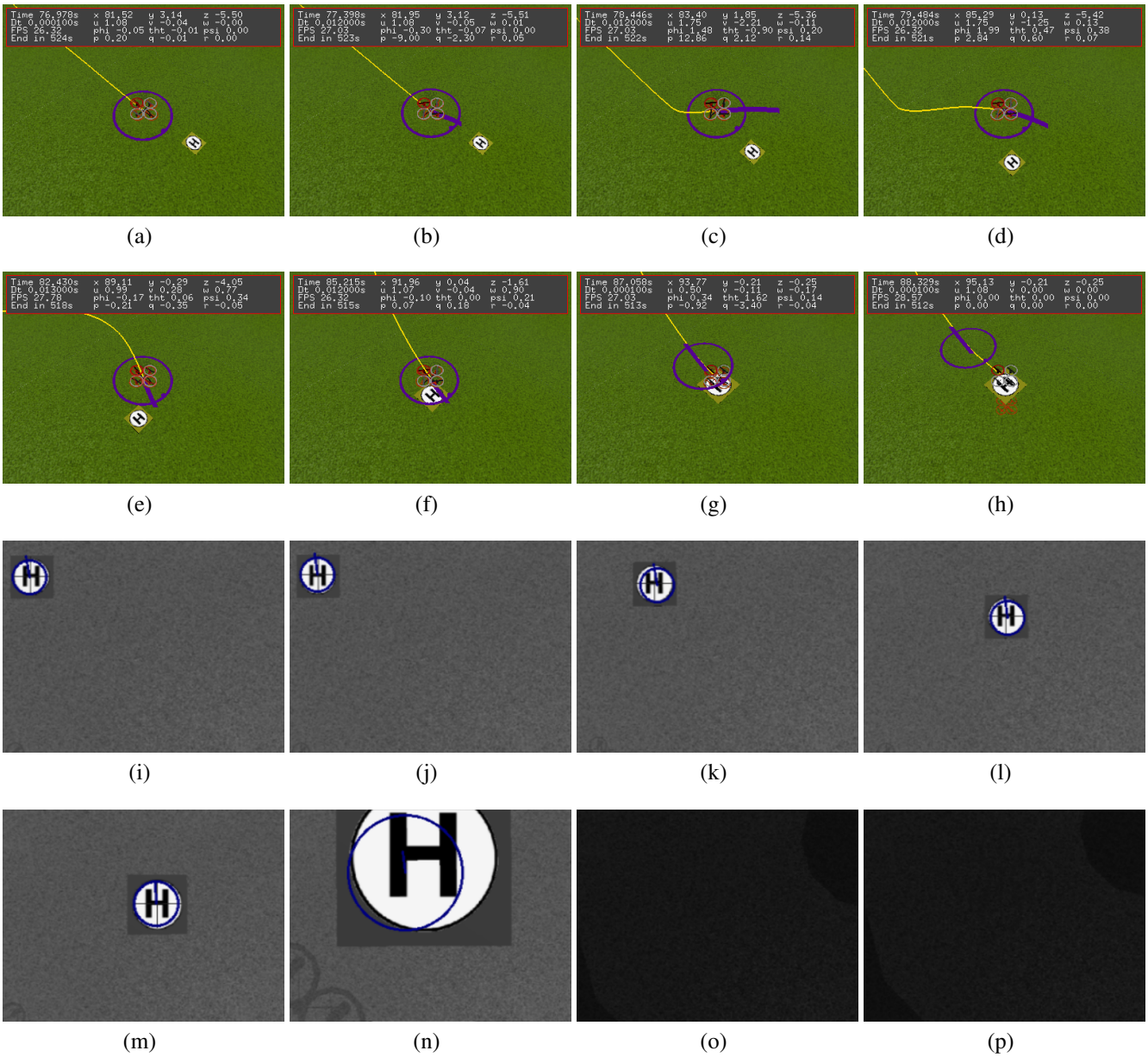
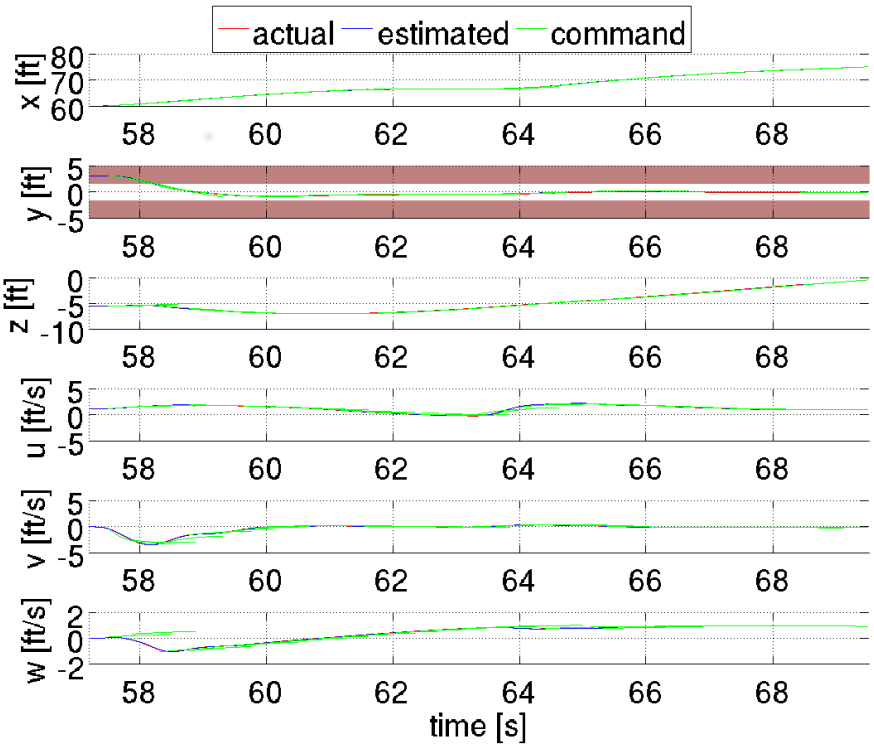
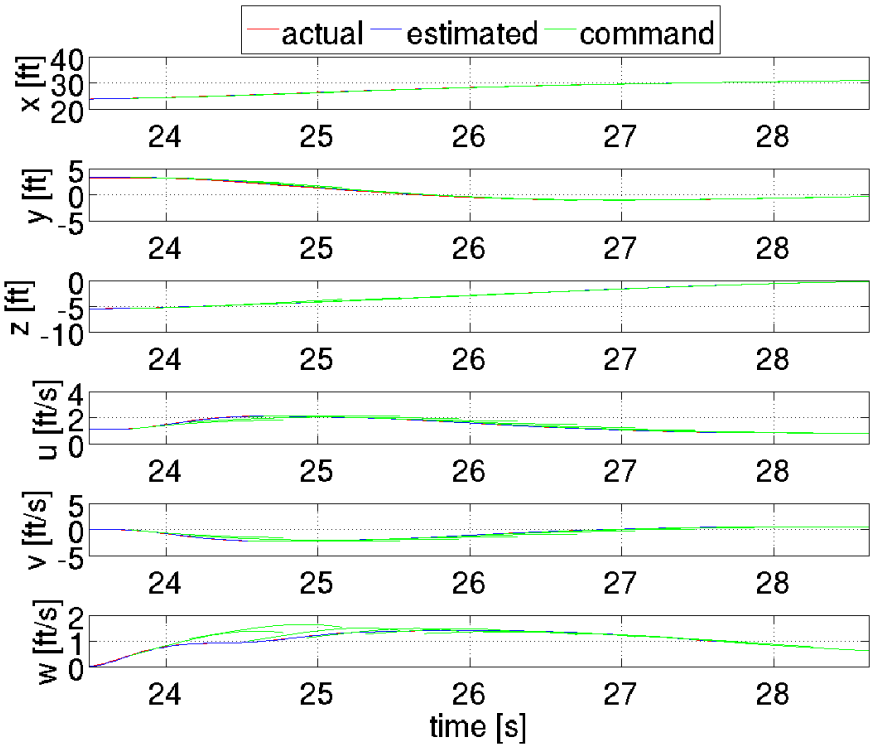


Fig. 6. The sequential figures show the simulation display and the simulated onboard images while the aerial vehicle tracks the target, approaches the target, and achieves landing. These figures are obtained while the vehicle lands on the target using the constrained trajectory. The purple circle represents the current way point, and the purple curve is the expected trajectory in one second. The yellow line is the vehicle path. The simulated onboard images are used for the vision system. The images are updated at approximately 50 frames per second; figures (i) to (p) show some of the simulated onboard images. The vehicle achieves hover behind the target at (a) and (i). The receding horizon DDP is activated at (b) and (j), and the vehicle approaches the target ((c), (d), (e), (k), (l), and (m)). When the receding horizon DDP computes the trajectory that allows the vehicle to land in one second ((f) and (n)), it uses the trajectory as a final approach trajectory. The vehicle follows the last trajectory ((g) and (o)), and lands ((h) and (p)). As shown in the figures, about one second before the landing, the vehicle loses visual of the target.

tion of the moving target using the hover control described in the control section. Then, we manually switch to the receding horizon DDP control. The receding horizon DDP computes the optimal landing trajectory using the output of the target tracker. The trajectory is calculated up to one second ahead of time. When the vehicle determines that it is infeasible to land in one second, it is simply navigated toward the local optimal, which typically results in approaching the target. When the terminal state of the trajectory is close enough to the target, the DDP stops updating the trajectory, and the vehicle attempts landing. This procedure is shown in the simulation images in Fig. 6. We also consider the cases of constrained



(a) With side constraints



(b) Without side constraints

Fig. 7. The figures show the results of the image-in-the-loop simulation including the true and estimated vehicle positions. The green lines are the DDP trajectory that has the minimum cost at the stage. The DDP updates at 10 Hz, but the figure shows only part of the trajectory for clarity. The shaded regions on the y-axis plot of (a) correspond to the constraied areas.

and unconstrained optimization. In the constrained optimization, we add two side constraints with respect to the target position. This simulates the case where a UAV pursuits landing by avoiding the line of sight of the driver as shown in Fig. 2. The constraints are given as follows:

$$g_1 = ({}^bY - {}^tY) - 1.5 \leq 0, \tag{51}$$

$$g_2 = -1.5 - ({}^bY - {}^tY) \leq 0. \tag{52}$$

These constraints penalize the portion of the trajectory which is more than 1.5 ft away from the target in the y-axis direction. The weight for each constraint is $w = 8$. We constrain the trajectory using the EQPF method described in the motion planning section. The target has the constant speed of 1.1 ft/s, which is the same speed as the iRobot Create had in the flight test. The vehicle is equipped with GPS, sonar, magnetometer, and inertial measurement unit. The vehicle state is estimated using the hybrid EKF that propagates the state of the filter using the IMU data and updates the states with the measurement of other sensors. The GPS update rate is 10 Hz, and the sonar and magnetormeter updates at 1 Hz. This EKF can estimate the position, velocity, attitude, and accelerometer and gyroscope biases. The details of the filter are described in (Ref. 12).

We show the results of the image-in-the-loop simulation in Fig. 7 of testing the landing case described above. The figure shows the actual and estimated states of the vehicle as well as the command trajectories from the DDP. Although the DDP updates at 10 Hz, we only show some of the trajectories for clarity. We used the weight matrix (described in (19)) below:

$$Q_w(t_f) = \text{diag}\begin{bmatrix}100 & 100 & 100 & 50 & 50 & 50\end{bmatrix} \tag{53}$$

$$R_w = \text{diag}\begin{bmatrix}40 & 40 & 400\end{bmatrix} \tag{54}$$

where $Q(t_f)$ is the weight for terminal states, and we do not weigh the states getting to the final states at each stage of the optimization. The vertical control is penalized more than the other two controls. This is because our vehicle model is a standard quadrotor which does not have much of the control authority to descend as compared to other directions. The results (Fig. 7) show the implementation of the constrained optimization's functionality. For the constrained trajectory, the vehicle is initially in the penalized area; therefore, it aggressively moves sideways to quickly go outside of the constrained area. This resuled in greater acceleration than the unconstrained trajectories. Also, it takes more time for the constrained DDP to achieve landing than the unconstrained version. One reason for this is that in the constrained trajectory, descent is not prioritized as much as going outside of the penalized area. Therefore, it uses the y-axis control more aggresively to move sideways, and descends slowly. The unconstrained method, on the other hand, results in smoother landing. The results validates our implementation and display the efficacy of the receding horizon DDP for a vision-based landing.

CONCLUSION

The authors suggest a new landing method using a vision system and the receding horizon DDP. We explore our method for vision-based tracking and our control architecture that enables optimal landings. We validate our implementation of the vision system with the flight test using onboard images. We use the vision system for an image-in-the-loop simulation to test vision-based autonomous landings. Both of the constrained and unconstrained methods are successful.

REFERENCES

[1]Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1), 11-15.

[2]Harris, C., & Stephens, M. (1988, August). A combined corner and edge detector. In Alvey vision conference (Vol. 15, p. 50)

[3]Lucas, B. D., & Kanade, T. (1981, August). An iterative image registration technique with an application to stereo vision. In IJCAI (Vol. 81, pp. 674-679).

[4]Olson, E. (2011, May). AprilTag: A robust and flexible visual fiducial system. In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 3400-3407). IEEE.

[5]Masselli, A., Yang, S., Wenzel, K. E., & Zell, A. (2014). A cross-platform comparison of visual marker based approaches for autonomous flight of quadrocopters. Journal of Intelligent & Robotic Systems, 73(1-4), 349-359.

[6]Yang, S., Scherer, S. A., & Zell, A. (2013). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. Journal of Intelligent & Robotic Systems, 69(1-4), 499-515.

[7]Yang, S., Scherer, S. A., Schauwecker, K., & Zell, A. (2014). Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision. Journal of Intelligent & Robotic Systems, 74(1-2), 27-43.

[8]De La Torre, G. (2015). Autonomous Suspended Load Operations via Trajectory Optimization and Variational Integrators.

[9]Tassa, Y., Erez, T. and Smart, W.D., 2008. Receding horizon differential dynamic programming. In Advances in neural information processing systems (pp. 1465-1472).

[10]Saripalli, S., & Sukhatme, G. S. (2006, January). Landing on a moving target using an autonomous helicopter. In Field and service robotics (pp. 277-286). Springer Berlin Heidelberg.

[11]Ruxton, D. J. (1993). Differential dynamic programming applied to continuous optimal control problems with state variable inequality constraints. Dynamics and Control, 3(2), 175-185.

[12]Nakamura, T., Haviland, S., Bershadsky, D., Magree, D., & E. N. Johnson, (2016, March) “Vision-Based Closed-Loop Tracking Using Micro Air Vehicles”, In IEEE aerospace conference.

[13]Lienhart, R., & Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002 International Conference on (Vol. 1, pp. I-900). IEEE.

[14]Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on (Vol. 1, pp. I-511). IEEE.

[15]Jacobson, D., & Mayne, D. (1970). Differential dynamic programming.

[16]Michael, N., Mellinger, D., Lindsey, Q., & Kumar, V. (2010). The grasp multiple micro-uav testbed. Robotics & Automation Magazine, IEEE, 17(3), 56-65.

[17]Haviland, S., Bershadsky, D., Magree, D., & Johnson, E., Development of a 500 gram Vision-based Autonomous Quadrotor Vehicle Capable of Indoor Navigation Proceedings of the 71st AHS Anunual Forum, Virginia Beach, VA, May 5-7, 2015.