Georgia Tech Team Entry for the 2016 AUVSI International Aerial Robotics Competition

Takuma Nakamura* Stephen Haviland† Dmitry Bershadsky‡
Georgia Institute of Technology, Atlanta, GA, 30332-0150

Eric Johnson§
Georgia Institute of Technology, Atlanta, GA, 30332-0150

THIS paper describes some of the key technologies used to accomplish the control, guidance, and navigation required in the mission 7 of the international aerial robotics competition (IARC). We also discuss the details of our quadrotor aerial vehicle, Georgia Tech quadrotor mini (GTQ-mini), which is used for the 2016 IARC. The GTQ-mini carries a high performance computer, and the system platform consists of an inertial measurement unit, a laser altimeter, and a monocular camera. The monocular camera is used for two purposes: localization of the vehicle and target tracking. Our localization is powered by a Bierman Thornton extended Kalman filter which utilizes feature points extracted with the corner Harris algorithm. We use two vision techniques to detect target ground vehicles: the Haar-like feature detection and the normalized-correlation-coefficient-based template matching. These detections are plugged into multiple extended Kalman filters, and that achieves the fusion of the multiple outputs of the two different techniques and the tracking of multiple agents. The 3D trajectory generation is required when the vehicle approaches a ground target, and that is achieved by an optimal control method called the receding horizon differential dynamic programming. The strategy for the mission is analyzed by using the image-in-the-loop simulator. We discuss the results of the simulations and the flight tests operated using the GTQ-mini.

Nomenclature

K	Kalman gain matrix
P	Error covariance matrix
Q	Process noise covariance matrix
R	Measurement noise covariance matrix
Z	The Z value, Mahalanobis distance
r	Residual of an estimator
f_x, f_y	Horizontal and vertical focal lengths of a camera
$\gamma_x \gamma_y$	Horizontal and vertical field of view of a camera
L_{ij}	Rotation matrix from a j frame to an i frame
Upperscript, Subscript represents	
t	Target
b	Body of a vehicle
c	Camera
i	Inertial frame
$-$	A priori variable in the extended Kalman filter
$+$	A posteriori variable in the extended Kalman filter
$\hat{A}$	Estimated value of A
${}^a x_c^b$	A variable x of a with respect to b expressed in the c frame, e.g, the x-coordinate position of the target with respect to the origin of the camera expressed in the inertial frame is denoted ${}^t X_i^c$

*Graduate Research Assistant, School of Aerospace Engineering, takuma.nakamura@gatech.edu
†Graduate Research Assistant, School of Aerospace Engineering, shaviland3@gatech.edu
‡Graduate Research Assistant, School of Aerospace Engineering, dbershadsky3@gatech.edu
§Associate Professor, School of Aerospace Engineering, eric.johnson@gatech.edu

I. Introduction

UNMANNED aerial vehicles (UAVs) would become a central technology of the new generation. We have already used UAVs as a reliable solution for photo-shooting and video-filming, and UAVs have become a massive industrial product. UAVs have played a huge role in the various areas from entertainment to military service, and their future application would apply to a lot more elements such as delivery and communication. However, there are a lot of problems to which the current UAV systems do not have a reliable and affordable solution. A indoor flight is one of the remaining challenges we need to solve because we are not able to use a global positioning system (GPS), which current UAV navigation highly relies on. We also need a robust and accurate object detection to approach a target and avoid an obstacle. The design of an aerial vehicle needs to be reconsidered to achieve a long-enduring flight. The mission 7 of the international aerial robotics competition (IARC) aims to solve these problems and tries to move the current state-of-the-art one step forward.

I.A. Problem Statement

For mission 7 the task is to develop a vehicle that can navigate over a 20 meter by 20 meter arena that can autonomsly herd ground targets while avoiding moving obstacles. Inside this arena, we have 10 iRobot Creates used as ground targets and 4 iRobot Creates acting as dynamic obstacles. We need to herd at least 7 of the ground targets towards the goal line while avoiding the obstacles. Herding requires an interaction between an aerial vehicle and a ground target which is achieved in two ways: landing in front of the target to hit the dumper sensor and flying over the target to activate a hull effect sensor. The mission duration is 10 minutes, and we are expected to navigate as many ground targets as possible in the duration. Only the onboard sensors can be used to percept a target and localize an aerial vehicle, but an external computational power is allowed.

I.B. Conceptual Solution

The Georgia Tech Aerial Robotics Team (GTAR) has developed an indoor micro aerial vehicle (MAV) that is able to explore a GPS-denied environment without any external aid. This vehicle is custom made by using the optimization tool called the electronic multirotor sizing tool (EMST) optimizer that selects the appropriate drive system for the specified mission, and achieves a flight duration enough to achieve the mission. We fuse the onboard sensor information — an inertial measurement unit (IMU), a laser altimeter, and a camera — using a Bierman Thornton extended Kalman filter (BTEKF) that estimates the position, velocity, and attitude of the vehicle. This filter is also capable of estimating the 3D biases of the IMU. This filter is aided by the vision system that extracts the feature points of an indoor structure; thus, we can know the relative position to the initial position. Our vision system enables target tracking by using multiple extended Kalman filters (EKFs). Perception of the target is achieved by the combination of the Haar-like feature detector and the template matching using normalized cross correlation (NCC). We run up to 10 EKFs to track all the ground targets. Since we know the physical sizes, speed, and turn rate of the targets, we use those a priori information to distinguish the targets from the obstacles. Once an appropriate target is found, we generate a landing trajectory by using the receding horizon differential dynamic programming (DDP). The receding horizon approach allows the vehicle to take appropriate time to land. In other words, the navigation system does not force the vehicle to land with an aggressive maneuver. These technologies are described in detail in the following sections.

I.C. Yearly Milestones

GTAR is entering the 7th mission for the first time. The core of our software used for control, guidance, and navigation that achieves the fundamentals (waypoint tracking, sensor fusion, state estimation) comes from the system used for the 6th mission; however, the following improvements and new capabilities are added to the system:

1. Improved inertial-navigation system aided by the vision system that achieves greater numerical stability,
2. A custom made MAV vehicle with a new stability augmentation system (SAS)
3. Vision-based target tracking that enables multiple agent and multiple model tracking
4. A capability to generate a 3D generation for autonomous landing

These and other improvements are described in detail in this paper.

II. Description of Vehicle

II.A. Hardware

GTAR uses the Georgia Tech quadrotor mini (GTQ-mini) for the 2016 IARC, and Figure 1 shows this vehicle. The GTQ-Mini weighs 500-700 grams depending on an installed battery and is less than 450mm in length in any dimension. It is able to fly without any external aid such as computing power and Vicon cameras. Test flights were operated at the Georgia Tech Indoor Flight Facility (Figure 2-(a)) using Vicon systems and in the AHS 2015 MAV student challenge, and the indoor flight capability is analyzed. This vehicle was developed with the UAV Research Facilitys, UAVRF, Electronic Multirotor Sizing Tool (EMST) optimizer, which allowed the vehicle to achieve the desired operation time and weight. The onboard computer, Gigabyte Brix 3, runs on Ubuntu 14.04 Operation System, and the computer communicates via USB interface with an Ardupilot. The Ardupilot provides IMU sensor readings to the Gigabyte Brix for processing. Custom USB cables were made to allow a MB1040 LV MAXSonar EZ45 sonar sensor and a downward facing Firefly-MV 6 USB monocular camera to connect to the onboard computer.

II.B. Software

The flight tests were conducted using Georgia Tech UAV Simulation Tool (GUST)[a]. GUST is a software framework that the UAVRF has installed to many vehicles for developing and testing purposes. GUST provides hardware-in-the-loop (HITL), software-in-the-loop (SITL) and ground station software. The full state vector of the system in GUST for the indoor flight is expressed as follows:

$$\hat{x} = \begin{bmatrix} \hat{q} & \hat{p}_i & \hat{v}_i & {}^{s}\hat{b} & {}^{\omega}\hat{b} & {}^{fp}\hat{p}_1 & \dots & {}^{fp}\hat{p}_N \end{bmatrix}^T, \tag{1}$$

where p, v, q is the position, velocity, attitude quaternion respectively, and ${}^{s}b, {}^{\omega}b$ is the bias of the accelerometer and gyroscope, respectively. N is the number of feature states, and ${}^{fp}\hat{p}_i$ denotes the position of the i-th feature point. The feature points are extracted using a corner Harris algorithm, and the feature states are assumed to be static. Instead of the conventional EKF, navigation uses a Bierman-Thornton EKF (BTEKF).[1] The BTEKF uses modified Cholesky factors[2] U and D on the covariance matrix P:

$$P = UDU^T, \tag{2}$$

where U is upper triangular matrix with a unit diagonal, and D is a diagonal matrix. The conventional EKF propagate is replaced by

$$U_k^- = L^T, D_k^- = G^T \begin{bmatrix} D_{k-1}^+ & 0 \\ 0 & Q_{k-1} \end{bmatrix} G, \tag{3}$$

where L is expressed in the equation below:

$$\begin{bmatrix} U_{k-1}^{+\,T} F_{k-1}^T \\ I \end{bmatrix} = GL. \tag{4}$$

The EKF measurement update equation is also expressed using U and D as follows:

$$U^+ = U^- U_a, \tag{5}$$

where

$$D^- - \frac{D^- U^{-T} H^T H U^- D^-}{H U^- D^- U^{-T} H^T + R} = U_a D^+ U_a^T. \tag{6}$$

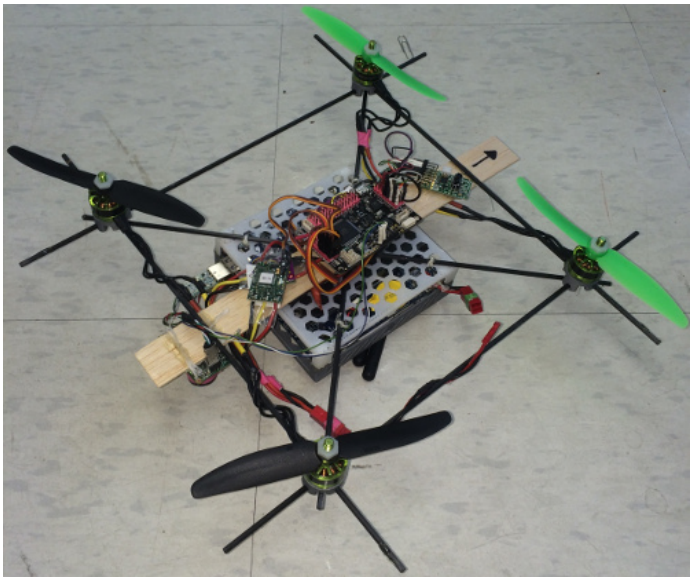
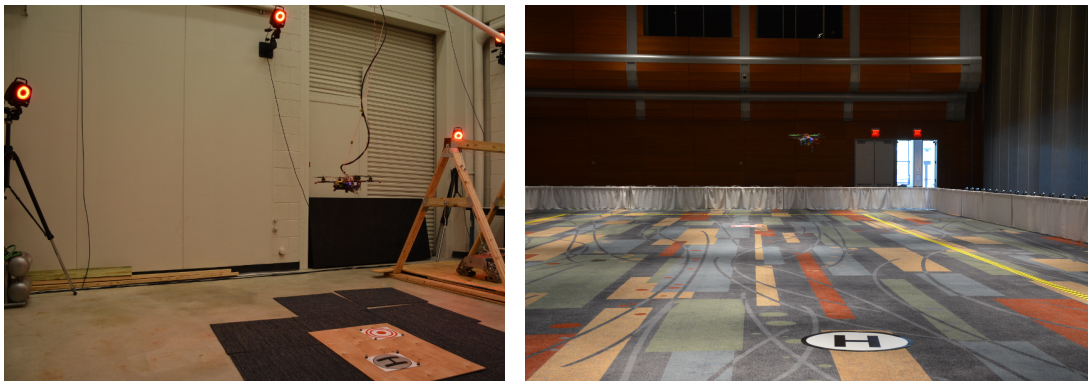


Figure 1. Figure shows our aerial vehicle GTQ-mini.



(a) (b)

Figure 2. Figure (a) shows the flight test in the Georgia Tech Indoor Flight Facility where the GTQ-mini is in flight with a safety cable. Figure (b) shows the 2015 AHS MAV Student Challenge where the GTQ-mini is in flight for practice.

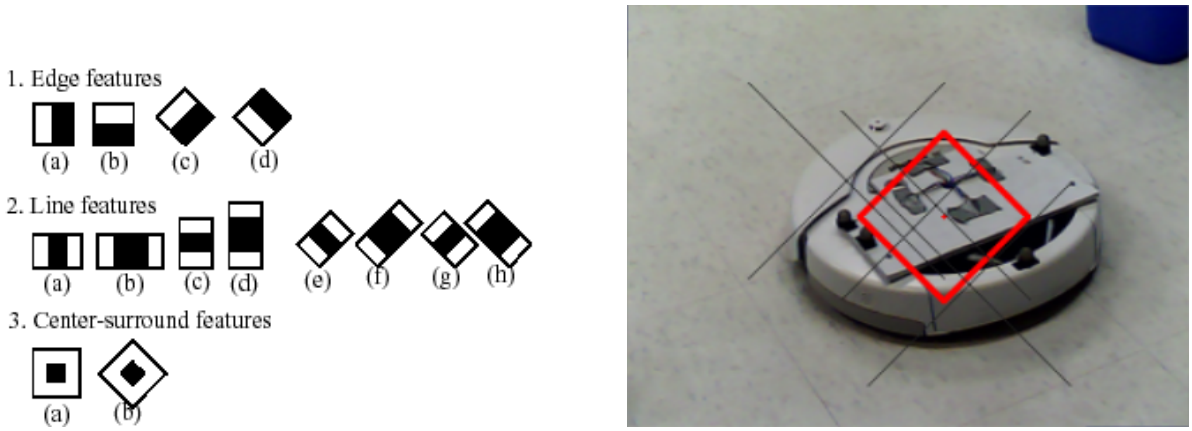
III. Target Tracking

III.A. Perception

We use two vision techniques to percept a ground target and an obstacle. The first technique is a machine-learning-based algorithm called a Haar-like feature detector.[3] The Haar-like feature detector is one type of feature-based approach to object classification. Typically, a feature-based approach functions operate faster than the pixel based; therefore, it is suitable for real-time applications such as target tracking using UAVs. We show in Figure 3-(a) some of the basic features. The Gentle Adaptive Boost Algorithm (Ada Boost) uses these simple features to train a classifier. Ada Boost needs images that contain the object of interest, which are so called positive images, and images that do not contain the object of interest, which are so called negative images. We used 500 positive images and 600 negative images to train the classifier for iRobot Creates. The raw outputs of the Haar-like feature detector are shown in the Figure 3-(b). Since we know the physical sizes of the target, the sizes of the target expressed in a camera frame $(a, b[\text{pixel}])$ can be estimated using the vehicle altitude as follows:

$$a = -\frac{\text{Width}A}{2^b \hat{Z}_i^i \tan(\frac{\gamma_x}{2})}, \quad b = -\frac{\text{Width}B}{2^b \hat{Z}_i^i \tan(\frac{\gamma_y}{2})}, \tag{7}$$

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



(a) (b)

Figure 3. Figure (a) shows examples of the Haar-like features, and Figure (b) shows a demonstration of the Haar-like feature detector finding an iRobot Create. The diagonal crossings are the raw measurements and the red rectangle shows the solution of the EKF that choose an appropriate measurement using the Mahalanobis distance.

where A, B(ft) are the dimensions of the target, γ_x, γ_y are the horizontal and vertical field of view, respectively. The classifier only searches the features with the estimated size. In order to allow the estimation error in target size, we give k-pixel margin, i.e., the detection starts with the smallest classifier size $a-k, b-k$ and ends with $a+k, b+k$. This technique not only reduces the wrong positive hit, but also speeds up the process. The measurement of the Haar-like feature detector is expressed as:

$$z_i = \begin{bmatrix} u \\ v \end{bmatrix}_i + v_k = \begin{bmatrix} f_x \frac{{}^t\hat{Y}_c^c}{{}^t\hat{X}_c^c} \\ f_y \frac{{}^t\hat{Z}_c^c}{{}^t\hat{X}_c^c} \end{bmatrix} + v_k, \tag{8}$$

where i is the index for the measurements at the same time, w_k is process noise, and v_k is measurement noise. Both of them are zero-mean Gaussian, i.e. $w_k \sim N(0, Q_k)$ and $v_k \sim N(0, R_k)$. Up to 20 raw measurements are stored for the Z-test, and the best measurements are plugged into the Kalman filter update. The state of the estimator, $\hat{x}$, is the position of the target in inertial frame, and the z position is overwritten by the output of the vehicle state estimation.

III.B. Filter Design

We use a standard state space form.

$$x_{k+1} = Fx_k + w_k, \tag{9}$$

$$y_k = H_k x_k + v_k, \tag{10}$$

This F matrix varies depending on the target dynamics models. The output matrix H cannot be directly computed, but using the position of the target expressed in the camera frame[4] it can be expressed as:

$$\begin{aligned} H &= \frac{\partial z}{\partial r_i}|_{x=\hat{x}} \\ &= \frac{\partial z}{\partial r_c}\frac{\partial r_c}{\partial r_i}|_{x=\hat{x}} \\ &= \begin{bmatrix} \frac{\partial u}{\partial {}^t\hat{X}_c^c} & \frac{\partial u}{\partial {}^t\hat{Y}_c^c} & \frac{\partial u}{\partial {}^t\hat{Z}_c^c} \\ \frac{\partial v}{\partial {}^t\hat{X}_c^c} & \frac{\partial v}{\partial {}^t\hat{Y}_c^c} & \frac{\partial v}{\partial {}^t\hat{Z}_c^c} \end{bmatrix} L_{ci} \\ &= \begin{bmatrix} -f_x \frac{{}^t\hat{Y}_c^c}{{}^t\hat{X}_c^c} & f_x \frac{1}{{}^t\hat{X}_c^c} & 0 \\ -f_y \frac{{}^t\hat{Z}_c^c}{{}^t\hat{X}_c^c} & 0 & f_y \frac{1}{{}^t\hat{X}_c^c} \end{bmatrix} L_{ci}, \end{aligned} \tag{11}$$

where f_x, f_y is the focal length of the camera, and the rotation matrix from camera to inertial is denoted $L_{ic} = L_{ci}^T$. Using the camera tilt (α), and pan (β) angle which are defined with a right-handed coordinate system with x-axis being the optical axis, the transformation matrix from body to camera frame can be expressed as follows:

$$L_{cb} = \begin{bmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ sin(\beta)sin(\alpha) & cos(\beta) & sin(\beta)cos(\alpha) \\ cos(\beta)sin(\alpha) & -sin(\beta) & cos(\beta)cos(\beta) \end{bmatrix}, \tag{12}$$

The transformation matrix from inertial to camera frame is obtained as

$$L_{ci} = L_{cb}L_{bi}. \tag{13}$$

The focal length f_x, f_y is expressed as follows using the width and height of the image:

$$f_x = \frac{\text{Width}}{2tan(\frac{\gamma_x}{2})}, \qquad f_y = \frac{\text{Height}}{2tan(\frac{\gamma_y}{2})}, \tag{14}$$

In the propagation of the EKF estimation, the covariance matrix, P, is updated using the discrete Lyapunov equation as follows:

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1}. \tag{15}$$

Since the target is known to be static, the state of the estimator, $\hat{x}$, is replaced by the previous posterior state, i.e. $\hat{x}_k^- = x_{k-1}^+$.The propagation of the error covariance matrix is expressed as $P_k^- = P_{k-1}^+ + Q_{k-1}$. In the update phase, the state and covariance matrix are updated using the Kalman gain matrix, which is obtained from the below equation:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}. \tag{16}$$

Although the linear model is available for the update of the state, the non-linear function, $h(x)$ that takes in the current state and computes an expected measurement vector is used for the update to reduce the issues related linearization.

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k)), \tag{17}$$

$$P_k^+ = (I - K_k H_k)P_k^-, \tag{18}$$

The statistical test, which is also called Z-test, is used to solve the corresponding problem. Since there are multiple outputs from the Haar-like feature detector, we need to choose an output used for the measurement update. The Z value, also called Mahalanobis distance, is defined as follows:

$$Z = r_k^T (H_k P_k H_k^T + R_k)^{-1} r_k, \tag{19}$$

where r_k is the residual. The measurement with the smallest Mahalanobis distance is used for the EKF update. The constant threshold for the Z value is set. This way, when the estimation is less confident, the measurement with a large residual is allowed for the update, and vice versa.

We use multiple model adaptive estimation technique to tell the conditions of the targets (straight, spin, and turn) and distinguish targets from obstacles. For N mutually exclusive values $x_1, x_2, \cdots, x_N$, the probability $P(y)$ can be expressed as follows:

$$Pr(y) = Pr(y|x_1)Pr(x_1) + Pr(y|x_2) + \cdots + Pr(y|x_N)Pr(x_N). \tag{20}$$

This result is known as the total probability theorem. We use this theorem to compute the probability that $x_{j,k} = x_{\text{true},k}$ given the observation z_k. From Bayes' rule this probability can be written as

$$Pr(\mathbf{x}_j|z_k) = \frac{\text{pdf}(z_k|\mathbf{x}_j)Pr(\mathbf{x}_j)}{\Sigma_{j=1}^N \text{pdf}(z_k|\mathbf{x}_j)Pr(\mathbf{x}_j)}. \tag{21}$$

The probability density function can be approximated by using the S matrix, which is computed as a by-product of Kalman gain in (16).

$$\text{pdf}(z_k|\mathbf{x}_j) \approx \frac{\exp(-r_k^T S_k^{-1} r_k/2)}{(2\pi)^{q/2}|S_k|^{1/2}}, \tag{22}$$

where q is the number of measurements, and r_k is a residual. Finally, we can use the parameter set with the highest conditional probability as follows:

$$\hat{\mathbf{x}}_{\text{true}} = \text{argmax}_{\mathbf{x}_j} Pr(\mathbf{x}_j | z_k), \tag{23}$$

When the probability becomes too large or too small (i.e. $Pr \approx 0$ or 1), it is nearly impossible for the estimator to switch the mode; therefore, we restrict each probability to [0.01, 0.99].

III.C. Target model

Our estimator is designed to track the position and heading of a ground target which has multiple known modes. The target is differential driven as shown in Figure 4, and difference of the velocity between the left wheel (v_l) and the right wheel (v_r) changes the heading ψ of the target. This dynamics can be expressed as follows:

$$^t\dot{X}_i = \frac{v_l + v_r}{2} cos(^t\psi), \tag{24}$$

$$^t\dot{Y}_i = \frac{v_l + v_r}{2} sin(^t\psi), \tag{25}$$

$$^t\dot{\psi} = \frac{v_l - v_r}{b}. \tag{26}$$

From the IARC rule, the target could have three modes: straight, spin, and turn. All the three modes are established by giving a different set of $[v_l, v_r]$ for a certain period of time. In each mode the wheel velocities $[v_l, v_r]$ are given as shown in TABLE 1. While in the straight mode, the vehicle runs straight with the constant velocity 0.33 m/s. We use this velocity as the default target velocity ^tV. While spinning, $\pm{}^tV/2$ is given to each wheel. Since v_l is positive, the target always spins clockwise as defined in Figure 4. The turn mode gives up to $\pm$0.032 m/s to each wheel in addition to ^tV. The additional velocity is decided randomly. We suggest to track the target by separating this dynamics into two modes $f_1(\mathbf{x}), f_2(\mathbf{x})$ where $\mathbf{x} = \begin{bmatrix} ^tX_i & ^tY_i & ^t\psi \end{bmatrix}^T$ as follows:

$$\frac{d\mathbf{x}_1}{dt} = f_1(\mathbf{x}) = \begin{bmatrix} ^tV cos(^t\psi) \\ ^tV sin(^t\psi) \\ 0 \end{bmatrix} + w_1, \tag{27}$$

$$\frac{d\mathbf{x}_2}{dt} = f_2(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ \Omega \end{bmatrix} + w_2, \tag{28}$$

where w_1, w_2 is zero-mean Gaussian noise $w_1 \sim N(0, Q_1), w_2 \sim N(0, Q_2)$ and Ω is a constant turn rate. The turn mode is dealt with the first model (27) by including a relatively large process noise. The Jacobians of each model are obtained by liniarizing the dynamics around the estimated state $\hat{\mathbf{x}}$ as $A_1 = \frac{\partial f_1(\mathbf{x})}{\partial \mathbf{x}}_{\mathbf{x}=\hat{\mathbf{x}}}$, $A_2 = \frac{\partial f_2(\mathbf{x})}{\partial \mathbf{x}}_{\mathbf{x}=\hat{\mathbf{x}}}$, and the discrete state matrices are $F_1 = I + \Delta A_1$, $F_2 = I + \Delta A_2$, where Δ is a time step.

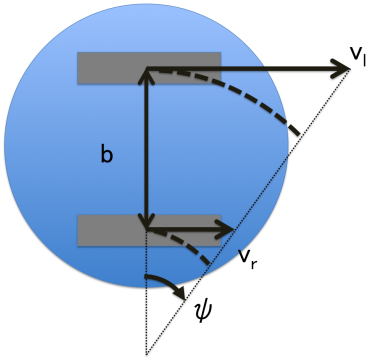


Figure 4. The model of the target is shown in the figure. This is a differential wheeled robot, and the speed differential of the two wheels $v_l - v_r$ generates a turn rate $\dot{\psi}$. The dotted lines are paths of wheels through a turn with respect to the wheelbase b.

Table 1. SET OF VELOCITIES GIVEN TO EACH MODE

Mode	v_l[m/s]	v_r [m/s]
Straight	0.33	0.33
Spin	0.165	-0.165
Turn	$0.33 \pm [0, 0.032]$	$0.33 \pm [0, 0.032]$

IV. 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,[5] and a DDP with inequality constraints is discussed in.[6]

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

$$\min_{\mathbf{u}}[\phi(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t)dt] \tag{29}$$

subject to the dynamics

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t), \mathbf{x}_0 = \mathbf{x}(t_0). \tag{30}$$

Our DDP takes the state vector of

$$\mathbf{x} = \begin{bmatrix} {}^bX_i & {}^bY_i & {}^bZ_i & {}^b\dot{X}_i & {}^b\dot{Y}_i & {}^b\dot{Z}_i \end{bmatrix}^T, \tag{31}$$

and the control vector is

$$\mathbf{u} = \begin{bmatrix} {}^b\ddot{X}_i & {}^b\ddot{Y}_i & {}^b\ddot{Z}_i \end{bmatrix}^T. \tag{32}$$

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

$$Q(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) = L(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) + V(\mathbf{x}(t_{k+1}), t_{k+1})). \tag{33}$$

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:

$$\begin{aligned} L(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) = &\frac{1}{2}(\mathbf{x}(t_k) - \mathbf{x}^*)^T Q_w(\mathbf{x}(t_k) - \mathbf{x}^*) \\ &+ \frac{1}{2}\mathbf{u}(t_k) R_w \mathbf{u}(t_k), \end{aligned} \tag{34}$$

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.

$$\mathbf{x}^* = \begin{bmatrix} {}^t\hat{X}_i + {}^tV\cos({}^t\hat{\psi})t_f & {}^t\hat{Y}_i + {}^tV\sin({}^t\hat{\psi})t_f & {}^tZ_i \end{bmatrix}^T, \tag{35}$$

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 $(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k)$.

$$Q_x(t_k) = \nabla_x L(t_k) + \nabla_x V(t_k) \nabla_x f(t_k), \tag{36}$$

$$Q_u(t_k) = \nabla_u L(t_k) + \nabla_x V(t_k) \nabla_u f(t_k), \tag{37}$$

$$Q_{xx}(t_k) = \nabla_{xx}L(t_k) + \nabla_x V(t_k)\nabla_{xx}f(t_k) + \nabla_x f(t_k)^T \nabla_{xx}V(t_k)\nabla_x f(t_k), \tag{38}$$

$$Q_{ux}(t_k) = \nabla_{ux}L(t_k) + \nabla_x V(t_k)\nabla_{ux}f(t_k) + \nabla_u f(t_k)^T \nabla_{xx}V(t_k)\nabla_x f(t_k), \tag{39}$$

$$Q_{uu}(t_k) = \nabla_{uu}L(t_k) + \nabla_x V(t_k)\nabla_{uu}f(t_k) + \nabla_u f(t_k)^T \nabla_{xx}V(t_k)\nabla_u f(t_k), \tag{40}$$

$$V_x(t_{k-1}) = Q_x(t_k) - Q_x(t_k)Q_{uu}^{-1}(t_k)Q_{ux}(t_k), \tag{41}$$

$$V_{xx}(t_{k-1}) = Q_{xx}(t_k) - Q_{xu}(t_k)Q_{uu}^{-1}(t_k)Q_{ux}(t_k). \tag{42}$$

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

$$l_{fb} = Q_{uu}^{-1}(t_k)Q_{ux}(t_k), \tag{43}$$

$$l_{ff} = Q_{uu}^{-1}(t_k)Q_u(t_k). \tag{44}$$

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

$$\delta\mathbf{u} = l_{ff} + l_{fb}\delta\mathbf{x}(t_k), \tag{45}$$

and the forward path is generated by a new control sequence

$$\mathbf{u}_{\text{new}} = \mathbf{u} + \gamma\delta\mathbf{u}, \tag{46}$$

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.

V. Simulation Results

We developed a six-degree-of-freedom quadcopter flight simulation to test the suggested vision system. This simulator can render a texture, and the vision system processes a rendered image. We estimate the vehicle states with the EKF, which is separated from the other filter used for target tracking. The vehicle is equipped with a monocular camera, an inertial measurement unit (IMU), and a magnetometer. We can choose how to update the position and the velocity: GPS, VICON cameras, and vision-aided inertial navigation. Here, we calculate the vehicle position error ${}^b e_{\text{width}}$ and ${}^b e_{\text{height}}$ as a deviation of the center of the camera from the target in the pixel. The optical axis lies on the positive x-axis, and we use a right-handed coordinate frame for $[y,z]_c$ to define the width and height of the image. Since our camera resolution is 320 by 240, the line of sight in the camera frame is $[y,z]_c = [\pm 160, \pm 120]$ in the pixel. We show in Fig. 5 the results of the image-in-the-loop simulation. The target vehicle turns from 15 to 18 seconds, and 45 to 49 seconds. The results show that the estimator was able to track the target with the f_1 mode estimation. The heading estimation also immediately follows the truth in real time. At 30 seconds, the target enters the spin mode, remains in the mode for 2.456 seconds, and makes a 180-degree turn. Soon after the initiation of the turn, the probability that estimator 1 drops and switches to the mode 2 control. The controller uses mode 2 controller temporarily even after the end of the spin mode. This is because the mode 1 estimator still has a huge residual in the heading estimation, and increases the uncertainty of the mode 1 estimator. This prevents the vehicle from aligning the velocity vector in a misguided direction. While the target is in the straight mode, we can successfully capture the target at the center of the image by using the mode 1 estimator. While the target reverses its heading, the uncertainty of mode 1 increases and switches to mode 2 before the target moves in the opposite direction. Since the mode 2 control law does not have heading information, the controller only reacts to the position signals; therefore, it is challenging to place the target at the center of the image. However, the mode 1 estimator catches up with the heading signal, and finds a correct heading. The controller swaps back to the mode 1, and relocates the target at the center.

We have developed a simulation that allows us to analyze the strategy for the IARC, and Figure 6 shows the simulation displays. We first herd a target that is close to the goal line and not heading toward it. When all the robots close to the goal are oriented to the goal line, we fly toward other targets. In other words, the herding is prioritized than 'rescuing' the target. This way, we can minimize the number of the target in the field as the time elapses. We

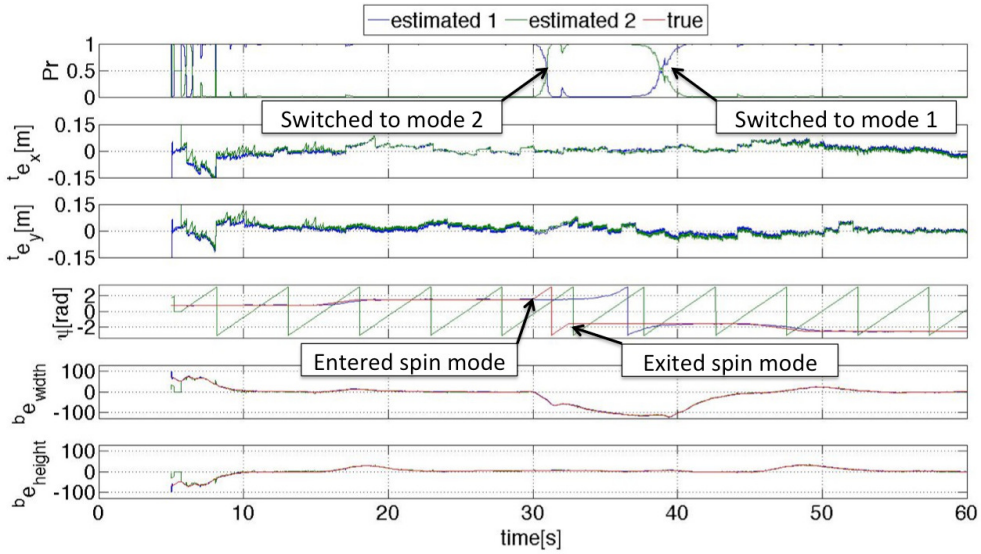
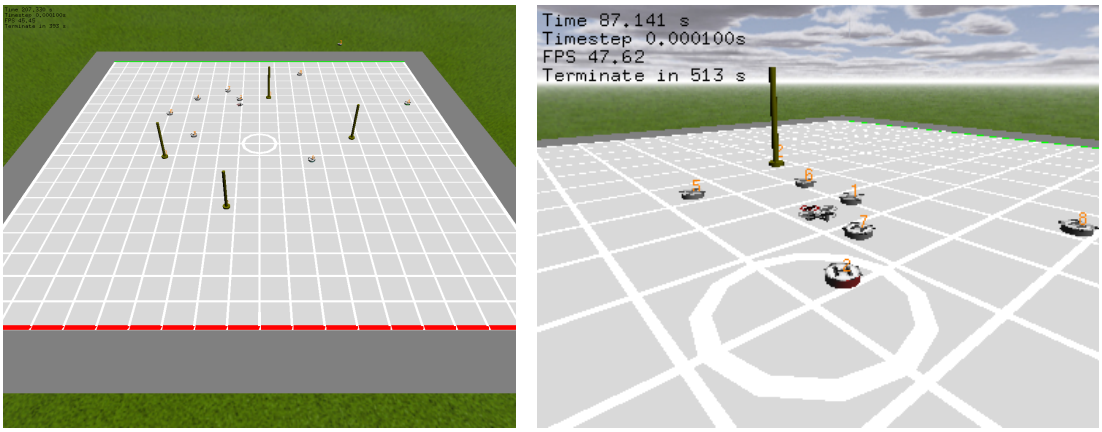


Figure 5. The figure shows the results of the image-in-the-loop simulation. The target tracker found the target at 5 seconds. The error of the target position estimation $({}^{t}e_x, {}^{t}e_y)$ was initially large, and the probability estimation was noisy; however, the probability soon converged to the correct model. The target tracker was able to estimate the heading with mode 1 while the target turned from 15 to 18 seconds, and 45 to 49 seconds. When the target entered the spin mode at 30 seconds, the probability that mode 1 is true quickly dropped and switched to mode 2. When the target reconfigured the constant speed, mode 1 started estimating the heading and re-established the mode 1 control. The ${}^{b}e_{\text{width}}, {}^{b}e_{\text{height}}$ stayed within the ± 120 pixels. Therefore, the target was in the line of sight throughout the process.



(a) (b)

Figure 6. Simulation displays

share in Fig. 7 the results of the simulation with the 1000 runs. The results display the probability density functions of the number of robots that goaled, removed from the field, and remained in the field after 10 minutes. The maximum velocity and acceleration of an aerial vehicle are restricted to 2.13 m/s and 3.05 m/s^2 in any direction. The position control of the aerial vehicle is achieved with a PID controller. We assume that the vehicle only has a capability to land in front of the target; therefore, the hall effect sensor is not activated. The results show that 24.5% of the simulation runs accomplished the mission, which is the summation of the probability that had more than or equal to the 7 goaled robots.

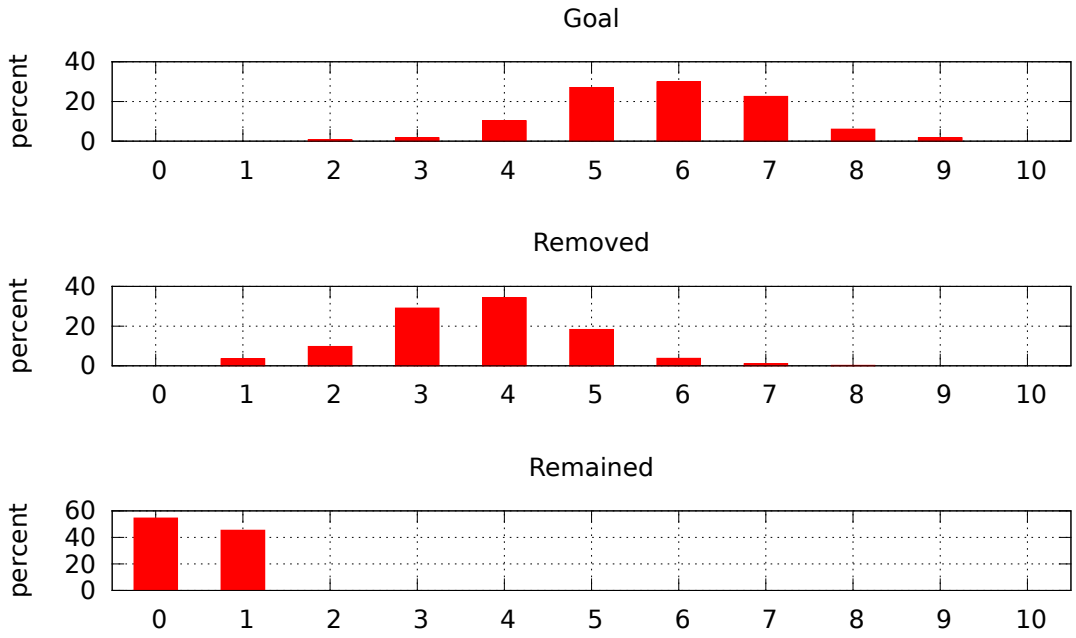


Figure 7. The figure shows the results of the IARC simulation. The top figure shows the probability density function of the number of the robots that reached the goal line. The 'removed' and 'remained' represent the probabilities of the number of the robot that removed from the field and remained in the field when the simulation was over in 10 minutes.

VI. Conclusion

This paper presents a new target-tracking system that utilizes MMAE from airborne images. Also, this paper introduce a receding horizon DDP that enables 3D trajectory generation for autonomous landing. We introduce our method to separate the target dynamics into mutually exclusive models and propose a method to use the statistics of the EKFs to give a capability to an aerial vehicle that switches the control laws for another mode of the target. By quickly judging the current mode of the target, the vehicle can capture the target in the line of sight. Simulation results presented in this paper illustrate the effectiveness of the algorithm for tracking mobile targets that replicate the motion specified in the IARC and show the benefit of using MMAE over a monotonic position estimator. We also propose our vision-based solution for Mission 7 of the IARC and share a strategy that achieves 24.5% success of the mission.

References

[1]Magree, D., & Johnson, E. N. (2015, January). A Monocular Vision-aided Inertial Navigation System with Improved Numerical Stability. In Proceedings of the AIAA Guidance Navigation and Control Conference, Kissimmee, Florida (pp. 6-2015).

[2]Grewal, M. S., & Andrews, A. P. (2014). Kalman filtering: Theory and Practice with MATLAB. John Wiley & Sons.

[3]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.

[4]Chowdhary, G., Johnson, E. N., Magree, D., Wu, A., Shein, A. (2013). GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. Journal of Field Robotics, 30(3), 415-438.

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

[6]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.