On the Design and Use of a Micro Air Vehicle to Track and Avoid Adversaries

Ruijie He, Abraham Bachrach, Michael Achtelik, Alborz Geramifard,
Daniel Gurdan, Samuel Prentice, Jan Stumpf and Nicholas Roy

***Abstract*—The MAV '08 competition focused on the problem of using air and ground vehicles to locate and rescue hostages being held in a remote building. To execute this mission, a number of technical challenges were addressed, including designing the micro air vehicle (MAV), using the MAV to geo-locate ground targets, and planning the motion of ground vehicles to reach the hostage location without detection.**

In this paper, we describe the complete system designed for the MAV '08 competition, and present our solutions to three technical challenges that were addressed within this system. First, we summarize the design of our micro air vehicle, focusing on the navigation and sensing payload. Second, we describe the vision and state estimation algorithms used to track ground features, including stationary obstacles and moving adversaries, from a sequence of images collected by the MAV. Third, we describe the planning algorithm used to generate motion plans for the ground vehicles to approach the hostage building undetected by adversaries; these adversaries are tracked by the MAV from the air. We examine different variants of a search algorithm and describe their performance under different conditions. Finally, we provide results of our system's performance during the mission execution.

1. Introduction

The MAV '08 competition in Agra, India focused on the problem of using air and ground vehicles to locate and rescue hostages being held in a remote building. Executing this mission required addressing a number of technical challenges. The first technical challenge was the design and operation of micro air vehicles (MAVs) that were capable of flying the necessary distances and carrying the sensor payload to locate the hostages. The second technical challenge was the design and implementation of vision and state estimation algorithms to detect and track a ground adversary guarding the hostages. The third technical challenge was the design and implementation of robust planning algorithms to generate tactical motion plans for our ground vehicles to reach the hostage location without detection by the ground adversary.

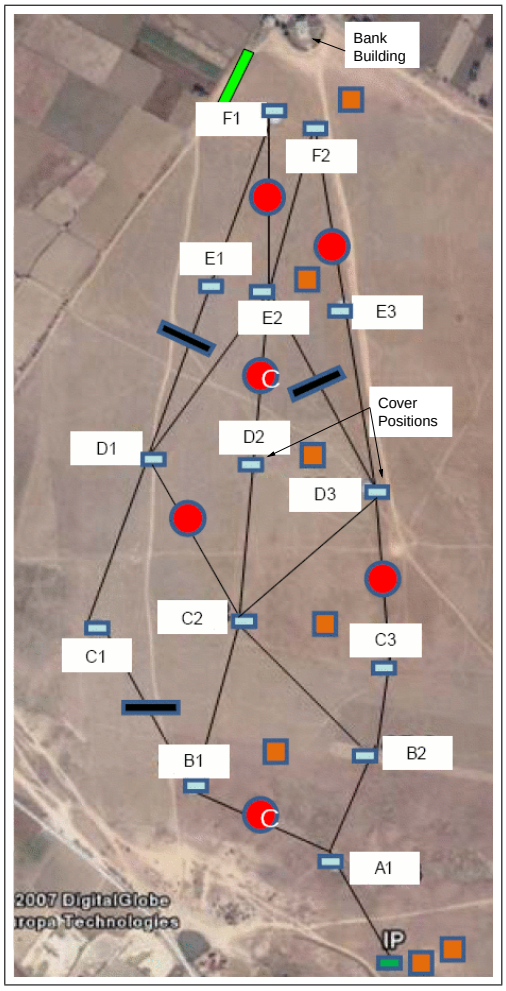
In this paper, we describe the complete system designed for the MAV '08 competition, and present our solutions to the three technical challenges described above. First, we summarize the design of our micro air vehicle, focusing on the navigation and sensing payload. We present a schematic of our complete system and discuss some of our design choices. Second, we describe the vision and state estimation algorithms used to track ground features through image sequences obtained by the MAV, including stationary obstacles and a moving adversary. Specifically, we use an adaptive algorithm (Avidan, 2007) that learns to discriminate the target from the background, coupled with standard Bayesian filtering to track the object from image to image in a global co-ordinate system. Third, we describe the planning algorithm used to generate motion plans for the ground vehicles to approach the hostage building undetected by the moving adversary. In order to plan with respect to the changing position of the ground adversary, we examine different variants of standard search algorithms that allow us to plan efficiently and react to unexpected or unmodeled changes in the ground adversary's position. Finally, we provide results of our system's performance during the mission execution.

Ruijie He, Abraham Bachrach, Alborz Geramifard and Samuel Prentice and Nicholas Roy are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St., Cambridge, MA 02139. Email: ruijie@mit.edu, abachrac@mit.edu, prentice@mit.edu, agf@mit.edu, nickroy@mit.edu
Michael Achtelik, Daniel Gurdan and Jan Stumpf are with Ascending Technologies GmbH. Graspergerstr. 8, 82131 Stockdorf Germany. Email: michael.achtelik@asctec.de, daniel.gurdan@asctec.de, jan@asctec.de

2. Related Work

We built on work from the different fields of robotics, computer vision, and planning to compete in the MAV '08 competition. There have been many rotorcraft UAV platforms developed, including quad-rotors (Hoffmann et al., 2007; Gurdan et al., 2007) that operate on the same principles as the hex-rotor developed here. Other autonomous rotorcraft morphologies include coaxial vehicles (Bouabdallah et al., 2006; Ng et al., 2004) and conventional helicopter platforms (Bagnell and Schneider, 2001).

The computer vision community has seen considerable work on object tracking, and an exhaustive survey of this literature is beyond the scope of this paper. However, surveys of the current state of the art include Yilmaz et al. (2006) and Porikli (2006). While there have been many successful algorithms such as background subtraction (Javed et al., 2002), mean shift tracking (Comaniciu et al., 2000), and ensemble tracking (Avidan, 2007) (which our algorithm is based on), all of these algorithms make the assumption of a relatively stationary camera, which does not hold for the camera on our vehicle. There has also been considerable work on tracking in the UAV literature (Casbeer et al., 2005; Furukawa et al., 2006; Quigley et al., 2005; Tisdale et al., 2008); however, much of this UAV work focuses on the abstract problem of tracking the target over time. Previous work does not leverage computer vision algorithms to effectively use a camera to track the objects without making major assumptions regarding their



(a) Prior Map

(b) Hostage building

(c) View from the ingress point

Fig. 1. (a) The map of the environment from the ingress point (IP, lower right) to the hostage building (top middle). The lightly shaded rectangles are cover points for the commandos, the circles are mine locations and the dark boxes are potential terrain obstacles. The cover points were provided *a priori* but the MAV was required to detect the mines and obstacles. (b) The view of the hostage building from the on-board MAV camera. (c) The view of the hostage building from the ingress point, 1km away. The rectangular cover positions can be seen faintly near the horizon.

appearance. In this work, we have been able to integrate modern computer vision inference algorithms with Bayesian filtering to effectively track the targets over time.

Similarly, we have built on existing work in spatial-temporal planning, showing the trade-offs present between full space-time planning and approximate techniques. Path planning with moving obstacles has been a challenging problem for researchers in many fields, including robotics and navigation. One option (van den Berg and Overmars, 2004) is to use a Probabilistic Road Map (PRM) to first generate discrete points in a continuous map that takes into account obstacle locations. Given a model of the dynamics of the moving obstacles, the resulting points are then extended into the time dimension to calculate the optimal path. A similar approach (Jaillet and Simeon, 2004) generates the initial map based on static obstacles and then uses a lazy-evaluation technique to generate a complete map of state-time space. Similarly, path planning in state-time space has been considered (Fraichard, 1999), where all dynamic obstacles in the 2-dimensional space are represented as static obstacles in a 3-dimensional space. A general path planning algorithm can then be used to find the optimal path in that space. One interesting fact is that path planning with dynamic obstacles can be viewed as a special case of cooperative path planning with multiple agents, where all dynamic obstacles are simply moving agents with predefined paths. Silver (2005) explored a similar approach to Fraichard's, resolving collisions in multi-agent path planning by using a reservation table.

3. The MAV '08 Mission

The MAV '08[1] mission was a hostage-rescue scenario, in which commandos had to be guided across a field by aerial vehicles to a remote building. The hostage building was guarded by a moving adversary; to allow the commandos to reach the building undetected, aerial surveillance was required to estimate the guard vehicle's position and its field-of-view. As the guard vehicle moved, the commandos were able to

[1] 1st US-Asian Demonstration & Assessment of Micro-Aerial & Unmanned Ground Vehicle Technology `http://www.nal.res.in/mav08/`

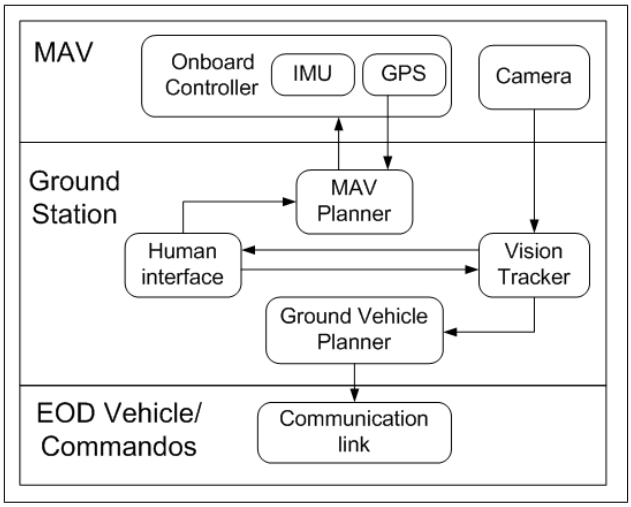


Fig. 2. Schematic of our sensing, tracking, planning and control system.

Fig. 3. Our six-rotor helicopter. The helicopter is 29cm in diameter and weighs 142g without the navigation electronics, camera or communication hardware.

take advantage of covered positions at known GPS locations throughout the field. When the guard vehicle's view of the field was occluded by obstacles such as the hostage building, the commandos were able to advance from covered position to covered position, remaining hidden at all other times. Complicating the problem further, some of the routes between covered positions were blocked by unknown obstacles and terrain, and some of the routes were seeded with mines at unknown positions. These obstructions had to be detected and geo-located aerially to plan safe paths for the commandos. Once detected and geo-located, the mines could also be disposed of using an explosive ordinance disposal (EOD) vehicle. Finally, the commandos were required to reach the hostage building within 40 minutes of the start of the mission, including completion of the surveillance, mine disposal and guard tracking tasks.

In order to obtain the position of the guard vehicle, detect route blockages and geo-locate mines, aerial surveillance was essential. However, the MAV '08 rules dictated a maximum size of air vehicle of 30 cm. Our approach to the mission was to use a series of rotorcrafts to survey the field, search for mines and obstacles, and also maintain a position estimate of the guard vehicle. Figure 1(a) shows a map of the field, containing the covered positions (A1 to F2) at known GPS locations. The circles along the edges are mine positions and the black bars are route blockages (these positions are shown here for the purposes of explanation, but were not provided before or during the competition). The ingress point for MAV launch, commando and EOD vehicle entry is shown at the bottom right (IP) and the hostage building is at the top middle (shown in Figure 1(b) from the on-board MAV camera). The view from the ingress point to the hostage building across the 1 km field is shown in Figure 1(c).

Figure 2 presents a schematic of our software architecture for the MAV '08 mission, describing the communication links between the different modules at the ground station, on the MAV, and with the ground agents. Via the human interface, ground station operators choose high-level goals for the MAV, such as searching areas for mines and tracking the guard vehicle. The MAV planner processes the human input and sends goal GPS waypoints to the MAV onboard controller (Section 4), which deals with the low-level controls for GPS-waypoint navigation. As the MAV surveys the environment, camera data is streamed back to the ground station, and the vision tracking and detection module (Section 5) enables the human operator to track particular objects of interest. GPS co-ordinates of the geo-located mines and guard vehicle are then conveyed to the ground vehicle planner (Section 6), which uses this real-time information to plan paths for the ground agents to reach the bank building safely.

4. The Micro Air Vehicle

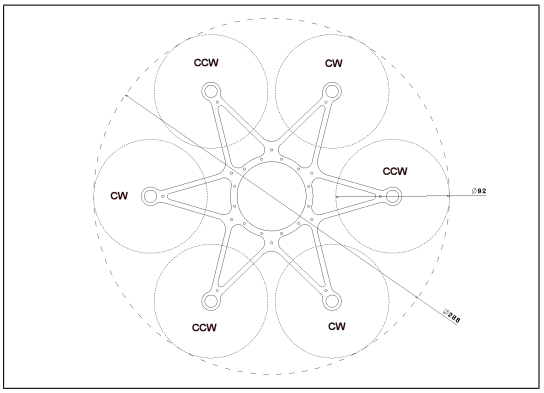
Our vehicle design consists of a custom-designed carbon-fiber airframe, with 6 brushless motors as the propulsion system. The vehicle is 29 cm rotor-tip to rotor-tip and weighs 142 grams without the navigation electronics, camera or communication hardware. The vehicle is shown in Figure 3. The total flight time of the vehicle is 10-12 minutes, with maximum speed of 10 m/sec, depending on wind conditions, temperature, etc.

4.1. Hex-rotor Vehicle Design

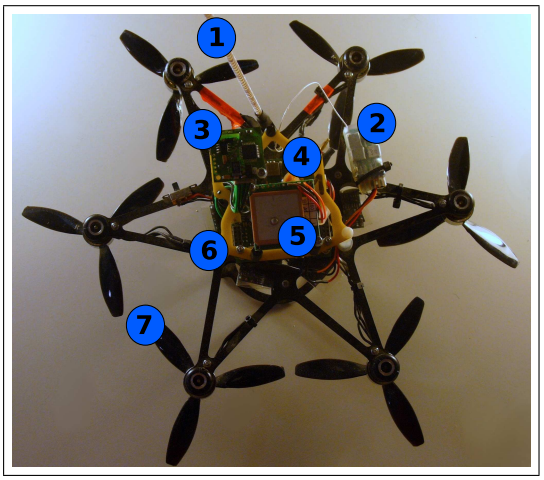
For an efficient design of a Vertical Take-Off and Landing (VTOL) vehicle, it is important to maximize the total rotor area. Lift is generated by the impulse of the air accelerated by the rotors, and the impulse p is linear in the velocity v_a and the mass m_a of the accelerated air: $p = m_a v_a$. The energy E required to accelerate the air is linear in the mass of the air, but is proportional to the square of the air's velocity: $E = \frac{1}{2} m_a {v_a}^2$. Thus, to produce a certain impulse, it is more energy efficient to accelerate more air to a lower velocity than to accelerate less air to a higher velocity. Consequently, a major goal in the design of our air vehicle was to maximize the overall rotor area.

Other driving factors of building a multi-rotor system include the increased agility, maximum wind-load and mechanical robustness of quad- or hex-rotor designs, relative to their coaxial counterparts. Additionally, the rules of the MAV '08

competition required that the vehicle be smaller than 30 cm in diameter. As custom propeller design and molding is very expensive, we sought the use of COTS propellers. The only COTS propellers that complied with the size limitations and were available in both the clockwise and counter-clockwise rotation variants necessary for a quad- or hex-rotor solution were 3-blade propellers with a 92 mm rotor-diameter.



(a) Hex-rotor design. CW: Clock Wise rotation, CCW: Counter Clock Wise rotation.



(b) Actual vehicle. 1) Antenna, 2) Analog RC link 3) IMU sensor 4) ARM microprocessor 5) GPS sensor 6) Shock absorber 7) Rotors

Fig. 4. Top-view of hex-rotor. (a) Vehicle schematic. (b) Photo of actual vehicle.

An initial quad-rotor prototype using these propellers was unfortunately unable to provide the required payload and flight time. However, it was possible to design a hex-rotor platform within the size limitations specified by the rules, as shown in Figure 4. By using six rotors instead of four, our helicopter was able to generate up to 50% more thrust with less than a 10% increase in vehicle weight.

The six motors are mounted underneath the frame such that air is displaced downwards without disturbance from the chassis, increasing energy efficiency and flight-time. The vehicle has three rotors spinning clockwise, and three rotors spinning counter-clockwise, as shown in Figure 4. With this rotor configuration, all rotational degrees of freedom can be controlled independently, as is the case for a quad-rotor helicopter. Thus, the vehicle controller ensures that a yaw command has no influence on pitch, roll or thrust, and the same is true for the other three command channels.

4.2. Onboard Sensing and Navigation

The navigation system consists of a 60 MHz Philips ARM microprocessor, u-blox GPS receiver, compass, IMU and pressure sensor. The software on the ARM microprocessor integrates the IMU and GPS measurements to provide a consistent state estimate at 1000 Hz. The on-board software accepts goal waypoints (x, y, z, θ) in the GPS (world) co-ordinate frame and uses PID control to achieve the desired position. Changing wind conditions affect how accurately the PID controller can achieve a specific waypoint; the integral (I) term can compensate for this error, but the time required for the I term to integrate the error and respond accordingly is frequently not worth the improved accuracy. We therefore use a dead-band in the position controller that describes the expected error in the controller, and we estimate the width of this dead-band online using a series of progressively smaller distance thresholds for each GPS waypoint. The position controller attempts to achieve the desired GPS position initially with a minimum 15 m accuracy, and then takes an additional 30 seconds to achieve the position with 2.5 m accuracy. If the waypoint is not achieved to within 2.5 m in those 30 seconds, the control software assumes that external factors (i.e., wind) are interfering and ends the attempt. In this way, we are guaranteed to get within 15 m of the desired waypoint, and the vehicle will attempt to achieve a higher level of accuracy without incurring excessive time delays.[2] However, the control loop will not keep trying indefinitely if wind or other external factors prevent it from achieving a 2.5 m accuracy.

The vehicle carries a Digi 900 MHz Xtend RF module operating at 100 mW, connected to the ARM microprocessor over an RS-232 serial line. The ground station communicates with the MAV via a USB-serial converter to the Xtend base station; the bandwidth is such that the ground station typically receives telemetry packets of 35 bytes at 30 Hz. The vehicle is configured to use the digital data link as the primary communication mechanism, over which GPS commands are sent to the vehicle and the vehicle's current GPS position is received in the telemetry data. Our assumption is that as long as the ground station can communicate with the vehicle, the vehicle can be controlled safely. If the digital data link is lost and the ground station and vehicle can no longer communicate, then the on-board controller reduces power to 30% of full throttle and attempts to land. The GPS waypoint controller can also be over-ridden with an auxiliary analog RC link operating at 72 MHz. If a safety pilot observes the vehicle behaving incorrectly, then an RC transmitter can be used to assume control over the vehicle and return it to base or land it safely. The onset of manual (analog) control signals is processed by the software on the ARM microprocessor. Since the vehicle is unstable without attitude control, switching

[2] In practice, we were always able to achieve 15 m accuracy in winds up to 10 knots, but were not able to make progress against headwinds of between 15 and 20 knots.

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

Fig. 5. Operation of tilt-camera. The camera has a 90° range of motion in the tilt direction. When the vehicle is below 5 m, the camera is automatically retured to the forward view.

to manual control disables GPS-waypoint following while keeping attitude control active.

The camera sensor is a Panasonic KX141 480 line CCD camera with a 90° field-of-view lens. Since the 900 MHz digital transmitter does not have sufficient bandwidth for real-time video, we use a LawMate TM-240500 2.4 GHz 500 mW transmitter to transmit the analog video data, and an Iftron Technologies YellowJacket 2.4 GHz diversity receiver at the ground station. This camera and transmitter provide excellent video capability at long ranges, and the 2.4 GHz frequency does not interfere with our 900 MHz data link. The camera is mounted on a small servo that provides 90° motion along one degree of freedom, allowing the camera to tilt towards any angle between the forward and straight down directions. The servo is controlled from the ARM navigation computer, which in turn receives servo instructions from the base station. The camera lens extends below the frame of the vehicle when pointing straight down, so that the camera is automatically returned to the forward view when the vehicle is below 5 m (Figure 5).

5. Object Tracking

The first phase of the MAV '08 mission involved surveying the field, identifying obstacles and mines, and then tracking the guard vehicle. These tasks presented the second challenge of identifying the positions of targets on the ground. Our approach was to locate objects in the image received from the camera, and use the known position of the MAV from GPS and a calibrated camera model to geo-locate the objects. However, due to noisy estimates of the vehicle pose, it was necessary to combine projections from many successive images to achieve an accurate geo-location estimate. For example, when we analyzed the geo-location estimates for an object with known world position, we see in Figure 6 that the individual estimates deviated from the ground-truth by up to 6 m. However, the distribution of measurements shows a systematic error or bias of approximately 3.3 m. This bias was likely due to errors in the calibration of the transformation between the body and camera co-ordinate frames.

We were given minimal prior information of the appearance of the guards, obstacles and mines; hence, we did not have enough information regarding a specific color, shape, or motion to allow general object detection of any of the target objects. We instead focused on the problem of object tracking, first relying on a human operator to detect the initial appearance of each object in the scene before tracking the object in successive frames. For object tracking, we developed

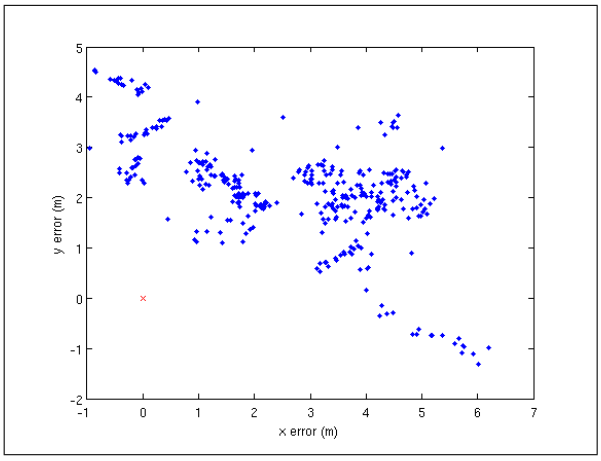


Fig. 6. Target geo-location estimates for an obstacle with a known location. The true location is mapped to $(0, 0)$.

a modified version of the classifier-based adaptive ensemble tracker, developed by Avidan (2007). Our algorithm, which we call Agile Ensemble Tracking (AET), uses the same object appearance classifier as Avidan. However, instead of using mean-shift to track the object across frames, we use a more robust, particle-filter based, Bayesian filter approach that is able to handle the fast motion of the MAV-mounted camera. While this approach does not allow completely autonomous operation, it significantly reduces the amount of attention required from the operator.

5.1. Learning Object Appearance Models

Once an initial estimate of the target object in an image is identified by a human operator, we use a machine learning classifier to learn a model of the object's appearance. The classifier is trained to distinguish pixels that belong to the object from background pixels. To train the classifier, we assume that the object is localized within a known $n \times m$ sub-block of the image; pixels within that sub-block are given positive labels, and pixels outside that sub-block are given negative labels. Each pixel is described by d local features, e.g., local color features and a histogram of local oriented gradient features (Dalal and Triggs, 2005). Each pixel i at image location $\mathbf{p}_i$ is therefore a separate training instance consisting of a d-dimensional feature vector $\mathbf{x}_i \in X$ and a label $y_i \in Y$. To distinguish the object from the background, we learn a classifier that predicts the label for each pixel based on the local image features. Following Avidan's work, we use a boosting method inspired by AdaBoost (Schapire, 2003) to

learn this classifier. AdaBoost requires a weak classifier, which in this algorithm is implemented as a linear separating hyper-plane $\mathbf{h}$, such that

$$\hat{y}(\mathbf{x}_i) = h(\mathbf{x}_i) = \text{sign}(\mathbf{h}^T \mathbf{x}_i) \quad (1)$$

where $\hat{y}(\mathbf{x})$ is the classifier output label for instance $\mathbf{x}$. The separating hyper-plane for a set of examples is computed using weighted least squares. This weak classifier is then boosted to learn an ensemble of classifiers $H = \{\mathbf{h}_1, \ldots, \mathbf{h}_K\}$ with associated weights $\alpha_1, \ldots, \alpha_K$. K is the total number of classifiers that are maintained by the algorithm. These weights are chosen iteratively, as shown in Algorithm 1.

Algorithm 1 : ADABOOST

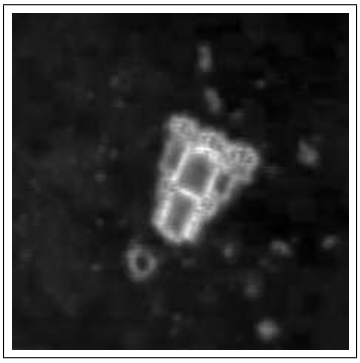
Require: N training instances $\{\mathbf{x}_i, y_i\}$
1: Initialize weights $\{w_i\}_{i=1}^N$ to be $\frac{1}{N}$
2: **for** $k = 1 \ldots K$ **do**
3: Normalize $\{w_i\}_{i=1}^N$ to sum to 1
4: Train weak classifier $\mathbf{h}_k$
5: $err = \sum_{i=1}^N w_i |\mathbf{h}_k(\mathbf{x}_i) - y_i|$
6: $\alpha_k = \frac{1}{2} log \frac{1-err}{err}$
7: Update $w_i = w_i e^{\alpha_k |\mathbf{h}_k(\mathbf{x}_i) - y_i|}$ for $i = 1 \ldots n$
8: **end for**
9: **return** $H(\mathbf{x}_i) = \sum_{k=1}^{K} \alpha_k \mathbf{h}_k(\mathbf{x}_i)$

In order to capture the appearance characteristics of an object at different scales, we train a separate ensemble of classifiers for a range of image scales. We then classify the pixels of a new image using the multi-scale, boosted, ensemble classifier, such that each pixel receives a (normalized) weighted vote for each label from each classifier based on the local image features at each pixel. The output of the classifier is a new image where each pixel has an associated likelihood value that it belongs to the tracked object.

Figure 7(a) illustrates an example training image, where the pixels in the inner block are positive training instances and the pixels in the outer block are negative training instances. Figure 7(b) shows the weighted classifier response to the same image after training. Notice that pixels along the sharply distinct color boundaries have the greatest classifier response.

During tracking, the appearance of both the object and the background will vary over time; for instance, the orientation of edge features will change as objects rotate in the image frame. We therefore continually learn new classifiers from the incoming images. After tracking has been performed on each image, the image is used as a new training instance for learning new classifiers. Using boosting, the B best classifiers are retained from the current K classifiers, while $K - B$ additional classifiers are trained and added to the set of weak classifiers. This process of updating the classifiers is shown in Algorithm 2. In order to ensure that this retraining does not result in a drift over time away from the original image, we also investigated a variation where a subset of the original K classifiers are kept, regardless of the performance of this classifier subset at the current time step. This modification would ensure that there always exist at least some classifiers that are known to be correct.

(a) Original Image



(b) Ensemble Filter Response

Fig. 7. (a) An example training sub-block. The pixels in the smaller, inner block are assumed to be positive training instances, and the pixels in the outer block are negative training instances. (b) The response of the weighted classifiers across the sub-image of the detected vehicle. The intensity of each pixel is the likelihood of belonging to the object as provided by the classifier.

Algorithm 2 : ADABOOST ONLINE UPDATE

Require: N training instances $\{\mathbf{x}_i, y_i\}$, existing strong classifier $H_{in} = \{\mathbf{h}_1, \ldots, \mathbf{h}_K\}$
1: Initialize weights $\{w_i\}_{i=1}^N$ to be $\frac{1}{N}$
2: $H_{out} = \{\emptyset\}$
3: **for** $k = 1 \ldots B$ **do**
4: Normalize $\{w_i\}_{i=1}^N$ to sum to 1
5: **for** $\mathbf{h}_j \in H_{in}$ **do**
6: Compute $err_j = \sum_{i=1}^N w_i |\mathbf{h}_j(\mathbf{x}_i) - y_i|$
7: **end for**
8: Choose $\hat{\mathbf{h}} \in H_{in}$ with minimum $\widehat{err}$
9: $\hat{\alpha} = \frac{1}{2} log \frac{1-\widehat{err}}{\widehat{err}}$
10: Remove $\hat{\mathbf{h}}$ from H_{in} and add to H_{out}
11: Update $w_i = w_i e^{\hat{\alpha} |\hat{\mathbf{h}}(\mathbf{x}_i) - y_i|}$ for $i = 1 \ldots n$
12: **end for**
13: **for** $k = B + 1 \ldots K$ **do**
14: Normalize $\{w_i\}_{i=1}^N$ to sum to 1
15: Train weak classifier h_k as in ADABOOST
16: Add h_k to H_{out}
17: **end for**
18: **return** H_{out}

5.2. Image Space Object Tracking

In the original ensemble tracker (Avidan, 2007), the estimate of the object's location is found using mean-shift on

the likelihood image computed from the classifier response. Starting from the previous target rectangle, mean-shift uses a hill-climbing technique to find the $m \times n$ rectangular region which contains the greatest aggregate response. While this approach works quite well for relatively stationary cameras, we found that the mean-shift approach was unable to handle the fast motion of our MAV platform.

As a result, we modified the tracking algorithm to use a particle-filter based Bayes filter to update the position estimate of the object. We incorporate an estimate of the camera ego-motion as a prior for predicting the location of the object in a subsequent image. This ego-motion estimate is essential for compensating for unpredictable motions of the camera, which would otherwise cause the tracker to lose track of the object. The attitude of the vehicle, as estimated by its onboard IMU, was too noisy to provide an adequate estimate of this ego-motion. Instead we estimate it directly from the imagery by computing optical flow between the entire previous and current images. We make use of the Pyramidal-Lucas-Kanade optical flow implementation available in the OpenCV package.[3] The optical flow algorithm computes a sparse set of f feature matches $\{\mathbf{p}_i^{t-1}, \mathbf{p}_i^t\}_{i=1}^f$, where $\mathbf{p}_i^{t-1}$ is the $2D$ pixel location of feature i in the image I_{t-1}, and $\mathbf{p}_i^t$ is its corresponding location in image I_t at the next time-step. Using these feature matches, we can estimate the camera ego-motion as a 3×2 affine transformation matrix Δ such that:

$$\mathbf{p}_i^{t+1} \approx \begin{bmatrix} 1 & \mathbf{p}_i^t \end{bmatrix} \Delta \tag{2}$$

This affine transformation captures translation, rotation, scaling, and shearing effects in image space. Due to the height of the vehicle, and the nearly planar ground surface, an affine transformation is generally a reasonable approximation.

Since some of the feature matches may be wrong or correspond to moving objects, we refine the ego-motion estimate by performing expectation-maximization (EM) to identify the affine transformation that best explains the apparent camera motion. Other methods such as RANSAC could also be used. The affine transformation Δ is then used in the motion model of a Bayes filter, while the learned object appearance model H is used in the associated sensor model. We use a particle filter to approximate the posterior distribution $p(\mathbf{p}_t|z_{0:t})$ according to

$$p(\mathbf{p}_t|z_{0:t}) = \alpha p(z_t|\mathbf{p}_t) \int_{X_{t-1}} p(\mathbf{p}_t|\mathbf{p}_{t-1}) p(\mathbf{p}_{t-1}|z_{0:t-1}) dt. \tag{3}$$

where $\mathbf{p}_t$ is the location of the object in the image at time t, z_t is the object measurement calculated from the image at time t, $p(\mathbf{p}_t|\mathbf{p}_{t-1})$ is our motion model, $p(z_t|\mathbf{p}_t)$ is our sensor model, and $p(\mathbf{p}_{t-1}|z_{0:t-1})$ is the prior distribution of the object's location.

The object measurement z_t is obtained by using the learned object appearance model to classify the image at time t. The classifier outputs a real value in the interval $[0, 1]$ for each pixel $\mathbf{q}_t$ in the image, and Figure 7(b) is a sample measurement. Our sensor model $p(z_t|\mathbf{p}_t)$ can therefore be characterized as follows,

$$z_t(\mathbf{q}_t)|\mathbf{p}_t = \begin{cases} 1 + \epsilon_z & \mathbf{q}_t = \mathbf{p}_t, \\ 0 + \epsilon_z & \text{otherwise}, \end{cases} \qquad \epsilon_z \sim N(0, \sigma_z), \tag{4}$$

where $z_t(\mathbf{q}_t)$ is the response of the classifier at pixel $\mathbf{q}_t$. Equation 4 essentially predicts that the classifier will respond with a 1 at the predicted location $\mathbf{p}_t$ in the image, and 0 everywhere else, where the measurements have Gaussian noise ϵ_z. The model is clearly approximate since the noise is not Gaussian (and measurements can never exceed 1), but the Gaussian model worked well experimentally.

It is computationally expensive to run the classifier on the entire image. Hence, we only run the classifier in the vicinity of the current particle filter mean estimate, and assume that the object has a minimal likelihood of being at all other locations in the image. Additionally, we smooth the classifier responses z_t across the image using a Gaussian blur operator to obtain a spatially smooth likelihood map, and each particle is given a weight equal to the value in the Gaussian-blurred probability image at its location in the image. Although this Gaussian smoothing creates minor correlations between image pixels, we continue to assume that the likelihood of object detection at each pixel is independent; experimentally the Gaussian smoothing of the classifier responses led to more robust object tracking even with this independence assumption, and more closely matched our Gaussian model of the classifier.

The motion model $p(\mathbf{p}_t|\mathbf{p}_{t-1})$ is equal to the ego-motion estimated from optical flow with additive Gaussian noise

$$\mathbf{p}_t|\mathbf{p}_{t-1} = \begin{bmatrix} 1 & \mathbf{p}_{t-1} \end{bmatrix} \Delta + \delta_{\mathbf{p}}, \qquad \delta_{\mathbf{p}} \sim N(0, \sigma_{\mathbf{p}}) \tag{5}$$

Algorithm 3 presents the complete Agile Ensemble Tracking algorithm. For clarity, although the algorithm is presented as if the images are all given to the algorithm at the start, on the real system, the images are actually processed in real-time as they are streamed from the vehicle.

In contrast to more conventional filtering techniques such as the Kalman filter (Kalman, 1960), the particle filter is better at modeling the non-linearities in the sensor and motion models. In contrast to ground vehicles and fixed-wing aircraft that generally have stable attitudes, the attitude of the MAV rotorcraft is particularly dynamic and non-linear. Frequent attitude changes of the MAV would cause very large object displacements in the image.

Table I(b) illustrates the benefits of the particle filter. Using the modified motion model, we were able to maintain a track of the person in Figure 8(b) for over 2 minutes, requiring human intervention only once when the person left the frame for a few seconds. In contrast, a much higher rate of human intervention to reacquire lost tracks was required when the original (non-optical-flow-based) motion prediction was used.

5.3. World Location Object Tracking

Given the position of a tracked object in an image, but without knowing the distance between the object and the camera, we would normally not be able to compute the global co-ordinates of the object. However, for the MAV'08

[3] Intel Corporation. Open Source Computer Vision Library (OpenCV). http://www.intel.com/technology/computing/opencv/index.htm

(a) EOD Vehicle (b) Person (c) Guard Vehicle

Fig. 8. Examples of the variety of objects tracked. (a) The EOD vehicle for mine disposal. (b) A walking person. (c) The guard vehicle circling the hostage building. (a) was relatively easy to track, but (b) and (c) required a better motion prediction model.

No optical flow, no retraining	0 Hz (0)
No retraining	0 Hz (0)
Keep first 3 classifiers	0 Hz (0)
Full retraining	0 Hz (0)

(a) 250 frames, 17 seconds

No opt. flow, no retrain.	0.140 Hz (21)
No retraining	0.040 Hz (6)
Keep first 3 classifiers	0.027 Hz (4)
Full retraining	0.007 Hz (1)

(b) 2683 frames, 150 seconds

No opt. flow, no retrain.	0.39 Hz (21)
No retraining	0.26 Hz (14)
Retain first 3 classifiers	0.28 Hz (15)
Full retraining	0.30 Hz (16)

(c) 1000 frames, 54 seconds

TABLE I

PERFORMANCE COMPARISON FOR THE AGILE ENSEMBLE TRACKING ALGORITHM, COMPARING THE EFFECT OF RETRAINING, AND THE OPTICAL FLOW BASED MOTION MODEL. THE FREQUENCY OF REQUIRED TRACK RE-INITIALIZATIONS IS SHOWN, WITH THE TOTAL NUMBER OF ERRORS IN PARENTHESES

Algorithm 3 : AGILE ENSEMBLE TRACKING

Require: T video frames $I_1 \dots I_T$, initial object bounding box r_1

1: Learn initial strong classifier H_1 from I_1 and r_1 using ADABOOST
2: **for** $I_t = I_2 \dots I_T$ **do**
3: Compute ego-motion estimate Δ from I_{t-1} to I_t
4: Propagate image space particle locations using Δ
5: Use H_{t-1} to update the likelihood of each particle and perform importance sampling
6: Use filter's maximum likelihood estimate as prediction of rectangle r_t
7: Compute H_t using ADABOOST ONLINE UPDATE
8: **end for**

competition, we knew that the MAV would be observing the objects from a large height, relative to height variations in the terrain. This knowledge allowed us to assume that the object was located on a flat ground plane. We could therefore recover the object's location in global co-ordinates using the camera's intrinsic properties (focal length, center of projection, distortion, and the rigid transformation from the camera image plane to the vehicle's body center), and the MAV's GPS position and attitude. The camera parameters and camera transformation are obtained using a standard least-squares calibration process.

Due to noise in the sensors, the location and attitude of the MAV are not perfectly known. Unfortunately, small errors in attitude can lead to substantial errors during the projection from image co-ordinates to world co-ordinates. We therefore apply a second level of Bayesian filtering to maintain a cleaner estimate of the target's location in global co-ordinates. For simplicity, we again use a particle filter, though given our models here a Kalman filter would have also been applicable. Our motion model $p(x_t|x_{t-1})$ assumes that the particles can be propagated with Gaussian noise. A common approach is to estimate the target velocity with the filter, improving the noise in the measurements with a stronger model bias. However, in practice, the image space measurements were of too high variance to provide accurate position and velocity estimation; we therefore estimated only the position over time. Similarly, our sensor model $p(z_t|x_t)$ assumes that the output of the image-space filter is corrupted by additive Gaussian noise when it is projected to world-coordinates. Essentially, in the world model, we are using the standard technique of a very strong model bias, coupled with Gaussian noise, to smooth the high variance estimates from image space.

In the image-space filter, we generally assume that the noise associated with the motion model is large and therefore place more weight on the sensor measurements. In contrast, when tracking in global co-ordinates, we place more weight on the motion model and model the image to world co-ordinate projections as very noisy measurements. With these parameters, the filter implicitly averages over more measurements when estimating the target's location on the ground plane, resulting in a more accurate estimate.

5.4. Tracking Analysis

Human intervention is still required to ensure that the object is continuously being tracked, to potentially restart the tracker when it fails, and to initialize the tracker when new

objects of interest appear. We evaluated the tracker under different configurations, including with and without the motion prediction given by optical flow, with and without retraining, as well as retaining different numbers of original classifiers. We tested the object tracker on very different targets across a wide variety of scenes, measuring the number of times that the estimate of the object's location diverged from hand-labeled, ground-truth data.

The easiest object tracking problem was the EOD vehicle, shown in Figure 8(a). This data set contained 17 seconds of video, for a total of 250 frames.[4] Due to the large vehicle size, crisp features and stable hover of the MAV, we obtained good performance for all tracker configurations. As Table I(a) reveals, even with the non-optical-flow motion model, or the online retraining of the classifier, the tracker never lost the vehicle after initialization. In addition, retaining different numbers of the original classifiers had no effect on the tracker's performance for this target.

Tracking the walking person, shown in Figure 8(b), was much more challenging due to the small size of the person in the image. Nevertheless, by taking advantage of the ego-motion estimation, the AET algorithm was still able to achieve excellent performance. As Table I(b) demonstrates, optical flow played an important role in keeping the tracking estimate on target. In addition, adapting the object appearance over time led to improved tracking. Although the appearance of the person moving around the field was relatively constant, the background changed dramatically when the person moved from the green grass to the gray dirt patches. Retraining and adapting the classifier therefore ensured that the classifier was able to maintain enough discrimination between the person and the background to continue tracking accurately.

Finally, we evaluated the tracker performance in tracking the guard vehicle in the MAV '08 competition. Due to the mission profile, the MAV observed the bank building from a distance with the camera pointed forward, rather than hovering directly above the bank building. With a forward-pointing camera, image changes between frames due to the MAV motion became more pronounced. In addition, as shown in Figure 8(c), the hedges surrounding the bank building were exactly the same color and similar shape as the guard vehicle. As a result, the tracker lost track of the guard vehicle far more often than in the other data sets we tested on.

In this data set, the camera motion, rather than changes in guard vehicle appearance, was the major factor that resulted in the tracker becoming lost. The guard vehicle was moving slowly enough that its motion should have had a negligible effect. Instead, from watching the video of the guard vehicle, there were several situations where the pitching and rolling of the MAV caused abnormally large inter-frame motion. In some of these cases, the optical flow was able to estimate and compensate for this ego-motion. In others, however, the optical flow computation failed to compensate for the camera motion, and many of these large inter-frame motion coincided with the tracker losing track of the vehicle. As a result, retraining the classifiers actually reduced performance slightly, since newer classifiers in the ensemble were trained on bad data as the tracker began to get lost, thereby creating a positive feedback cycle from which the tracker could not recover. While it is clear that the optical flow plays an important role in keeping the tracking on target, the optical flow algorithm may be unable to capture the full camera motion in some domains, resulting in the classifier becoming lost.

Fundamentally, to solve the tracking problem in the face of potentially large inter-frame camera motion, more sophisticated object detection is needed. Once the ensemble-based tracker loses the target, there is no way to recover by using a local appearance-based tracker that is learned online, since any corruption of the current object estimate will be propagated forward. Subsequent classifiers would then get corrupted. As a result, an object detector with higher-level learned invariants is needed to recover from object tracker failures in the general case.

6. Ground Vehicle Planning

Given the MAV's ability to estimate the guard's position and trajectory, the third challenge was to plan a trajectory for the commandos to reach the hostage building without being detected by the guard. Additionally, when mines were detected by the MAV, we needed to plan a trajectory for the EOD vehicle to reach the mines without being detected. We treated these problems symmetrically as a motion planning problem for generic ground vehicles.

Traditional motion-planning algorithms are based on search strategies through a discretized state space. Although the specific MAV '08 planning problem focused on generating routes between the cover points (marked $A1, \ldots, F2$ in Figure 1a), we sought a general purpose motion planner, one that is flexible to unexpected guard motion and would allow us to express a wider range of trajectories.

Our motion planner makes a number of assumptions based on the initial problem description, though not all of these assumptions were required for the actual MAV '08 competition. The planner assumes a discretized planning area, a regular grid, and assumes that the vehicle can move from a grid cell x to any of the 4-connected neighbors. We assume that each motion incurs a cost, and that the planner's objective is to find the lowest cost sequence of states from the start to the goal without being detected by the guard. The guard has a 360° field-of-view with finite range, and a prior map of the environment reveals which obstacles would occlude the ground vehicle from the guard. Additionally, the planner assumes that the guard's current position is known, and that a model of the guard dynamics allows us to predict the guard's position in the future. While a deterministic model of the guard's motion is unrealistic, we did not have access to a reasonable stochastic model of guard motion. As a result, following Bertsekas (1995), we used open-loop feedback control in which the planner assumes a deterministic model and replans after each action. This form of planning under uncertainty relies on very fast replanning but has been shown to converge under reasonably mild assumptions.

[4] We typically received data from the vehicle at 15 Hz, but this number varied depending on the characteristics of the local RF field.

Algorithm 4 : STATE-A*

Require: $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$, $\mathbf{x}_{guard}$

$\pi \leftarrow$ A*$(\mathbf{x}_{start}, \mathbf{x}_{goal})$
$i \leftarrow$ COLLIDE$(\pi, \mathbf{x}_{guard})$
while $i > 0$ **do**
  MARK_BLOCKED$(\pi[i])$
  $\pi_{tail} \leftarrow$ A*$(\pi[i-1], \mathbf{x}_{goal})$
  if $(\pi_{tail} == \textbf{null})$ **then**
    return null
  end if
  $\pi \leftarrow \pi[0 : i-1] + \pi_{tail}$
  $i \leftarrow$ COLLIDE$(\pi, \mathbf{x}_{guard})$
end while
return π

The planner must incorporate the guard's temporal behavior to generate detection-free paths for the ground vehicle. Temporal constraints typically require planning in both space and time, but doing so leads to substantial computational complexity, especially given the large map size. Instead, we examined three different strategies for planning with respect to the guard dynamics, identifying a strategy that scales well with minimal loss in planner performance.

6.1. STATE-A*

To determine if the additional complexity of planning in time and space could be avoided, we first examined the performance of planning only in the state space of the ground vehicle. The STATE-A* algorithm discretizes the state space and searches for a plan π from the start position $\mathbf{x}_{start}$ to the goal $\mathbf{x}_{goal}$, both given in GPS co-ordinates. The plan π consists of an ordered list of states $\pi = \{\mathbf{x}_{start}, \ldots, \mathbf{x}^i, \ldots, \mathbf{x}_{goal}\}$.

In order to avoid detection by the guard, STATE-A* forward-simulates the plan with the guard starting at position $\mathbf{x}_{goal}$. As we simulate the ground vehicle moving to the next state in the plan, the guard position is predicted using the known guard dynamics. Grid cells that are within the guard's field-of-view are marked as dynamic obstacles. The planner then tests to see if a detection (failure) would result, and any state that is predicted to result in a detection is inserted into the map as a static obstacle. A new plan is generated, and the process is repeated until a plan that avoids guard detection is found, or until all possible plans have been tried. Algorithm 4 shows the STATE-A* in detail. The COLLIDE subroutine simulates the ground vehicle motion along the plan π, and the MARK_BLOCKED subroutine modifies the map for future re-planning.

The STATE-A* approach is expected to be computationally efficient compared to time-state search processes, as the branching factor in the search is limited to changes in the position of the ground vehicle, rather than changes in both time and position. However, this computational saving restricts the plan space, since the search process cannot take advantage of pause actions (without a time variable, a pause action would appear to have no effect). As a result of the restricted plan space, the planner may not be able to find efficient or robust plans that actually exist.

6.2. TIME-STATE A*

The TIME-STATE A* algorithm, developed by Fraichard (1999), represents the state of the ground vehicle with both a position and time co-ordinate. In order to account for the guard, the 2-dimensional space is extrapolated into the time domain, creating a 3-dimensional cost map (or "cube"), where each cell represents a separate (x, y, t) co-ordinate. All actions are assumed to have the same, constant duration Δt. In addition to the four motion commands, we add a PAUSE action that causes the vehicle to stop in place for an amount of time Δt. Longer pauses can be achieved by executing PAUSE repeatedly. As before, we search through the cube using standard A*, but limit the actions from every cell (x, y, t) to its 5-connected neighbors at the next time step $t + 1$, i.e. $(x, y, t+1)$, $(x-1, y, t+1)$, $(x+1, y, t+1)$, $(x, y-1, t+1)$ and $(x, y + 1, t + 1)$. The Manhattan distance between the robot's current position and the final goal in the 2-dimensional space is used as the heuristic; the Manhattan distance is known to be admissible, consistent and is a standard heuristic for 2-D search problems. Algorithm 5 shows the TIME-STATE A* in detail.

Algorithm 5 : TIME-STATE A*

Require: $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$, $\mathbf{x}_{guard}$, t_{max}

$\pi \leftarrow$ A* $((\mathbf{x}_{start}, \mathbf{x}_{guard}, 0), (\mathbf{x}_{goal}, \cdot, t_{max}))$
return π

Notice that the input to A* called from within TIME-STATE A* now includes states with an explicit time variable and a maximum time, t_{max}, in order to prevent infinite search depth resulting from multiple PAUSE actions. We have abused the notation slightly by stating that the goal state of the A* process is $(\mathbf{x}_{goal}, \cdot, t_{max})$, indicating that the guard can be in any position for the search goal state.

By modeling time explicitly during the search process, the TIME-STATE A* algorithm can express a wider variety of plans than STATE-A* by incorporating plans that deliberately wait for the guard to move. Additionally, the search incorporates knowledge of the guard more accurately by including the changing guard position as part of the search in the state-time domain. However, the computational cost of increasing the size of the state space (an additional time dimension) may affect the planner's ability to find good plans in a reasonable amount of time.

6.3. WINDOWED TIME-STATE A* (WTS-A*)

Since the search grows exponentially with the search depth, one alternative approach is to reduce t_{max}, including only plans that have a maximum length t_{max} in the search. However, this may significantly reduce the planner's ability to find good plans when plans need to be longer than t_{max}, which is likely across a 1 km distance. Instead, we examine an intermediate approach of iterating TIME-STATE A* search in a limited time window, building off techniques in the cooperative search literature that restrict the search window (*e.g.* WHCA* (Silver, 2005)) to reduce the overall computational demands of the search process. WHCA* performs cooperative search

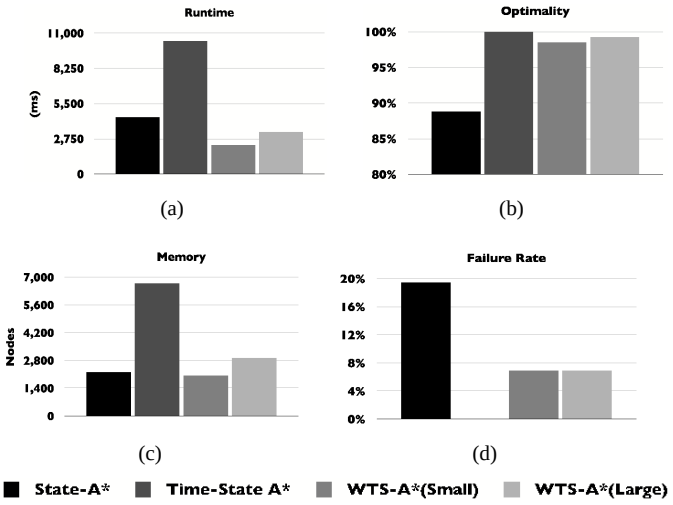


Fig. 9. Average runtime, optimality, memory and failure rate of State-A*, Time-State A*, WTS-A*(Small), and WTS-A*(Large) across planning problems with sizes: $30 \times 30, 70 \times 70$, and 100×100, each over 30 runs.

up to a fixed limited horizon and uses an abstract heuristic for the remaining search to full depth. In contrast, we propose an alternative form of windowed search by first using the abstract heuristic to compute a path, before dividing it into smaller sub-plans and performing TIME-STATE A* search within that space.

Algorithm 6 : WINDOWED TIME-STATE A* (WTS-A*)

Require: $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$, $\mathbf{x}_{guard}$, t_{max}, t_{window}

$\pi_{approx} \leftarrow$ A*$(\mathbf{x}_{start}, \mathbf{x}_{goal})$
$\{\hat{\pi}^i\} \leftarrow$ DIVIDE$(\pi_{approx}, t_{window})$
$t \leftarrow 0$
for $\hat{\pi}^i \in \{\hat{\pi}^i\}$ **do**
  $\mathbf{x} \leftarrow \hat{\pi}^i[1]$
  $\mathbf{x}' \leftarrow \hat{\pi}^i[end]$
  $\pi_{tail} \leftarrow$ A* $((\mathbf{x}, \mathbf{x}_{guard}, t), (\mathbf{x}', \cdot, t_{max}))$
  if $\pi_{tail} ==$ **null then**
    return null
  end if
  $\pi \leftarrow \pi + \pi_{tail}$
  $t \leftarrow t + length(\pi_{tail})$
end for
return π

Algorithm 6 presents our complete algorithm. First, an approximated plan is computed using STATE-A*, ignoring the guard position. This plan is then divided into sub-plans according to a window size, and for each start and end state of the sub-plan, the plan between these states is regenerated using TIME-STATE A*. Notice that the t variable is used to maintain the time required to execute each sub-plan $\hat{\pi}^i$, so as to ensure a proper connection between each section of the path.

6.4. Simulation Results

In order to determine the performance of these algorithms, we evaluated the three algorithms in a series of random map environments by varying a number of parameters. In particular, we varied the size of the map, the percentage of the map that was blocked by static obstacles, as well as the number of single-occupancy dynamic obstacles that maneuvered in the environment. Table II summarizes the 24 map settings used for simulations. All algorithms were tested on each setting with 30 randomly generated maps. For each run, $start$ and $goal$ states were positioned randomly on the two opposite sides of the grid world, while $guard$ positions were randomly generated and moved to an empty neighbor cell on each time step. Table III summarizes the various window sizes (t_{window}) and maximum time (t_{max}) used in each domain. Additionally, we measured the memory usage and the failure rate of each algorithm. A failure occurred if the algorithm failed to find the existing path to the goal.

	Static obstacles	
Map Size	20 %	30 %
30×30	$\{20, 40, 60, 80\}$	$\{20, 30, 40, 50\}$
70×70	$\{40, 70, 100, 130\}$	$\{20, 40, 60, 80\}$
100×100	$\{50, 80, 110, 140\}$	$\{10, 30, 50, 70\}$

TABLE II
NUMBER OF DYNAMIC OBSTACLES USED FOR EACH MAP SETTING.

	Window Size (t_{window})		
Map Size	Small	Large	t_{max}
30×30	15	30	60
70×70	20	40	327
100×100	25	50	667

TABLE III
WINDOW SIZES AND MAXIMUM TIME PARAMETERS USED IN VARIOUS MAP SIZES.

Figure 9 depicts the averaged runtime (a), quality (b), memory usage (c), and failure rate (d) of the resulting plan for STATE-A*, TIME-STATE A*, and WTS-A* with different window sizes. As expected, on average, TIME-STATE A* was the most time consuming algorithm, as shown in Figure 9(a). Interestingly, WTS-A* on average outperformed STATE-A* in terms of runtime. On the other hand, the quality of the paths found by the WTS-A* were on par with those found by TIME-STATE A*, shown in Figure 9(b). The plan performance found by the WTS-A* was within 97% of the optimal plan (found by TIME-STATE A*), while STATE-A* suffered a drop around 12% from the optimal. Figure 9(c) depicts the maximum memory used by each algorithm in terms of the number of identical visited nodes. While this graph resembles the running time of the corresponding algorithms, with TIME-STATE A* taking up the most memory, the memory usage of STATE-A* is on par with the small window version of WTS-A* and less than the large window version of WTS-A*. This highlights the fact that although STATE-A* had to re-plan more often, it searched through a more compact space. Figure 9(d) shows the average failure rate of the various methods. As expected, STATE-A* suffered the most because of its substantial search

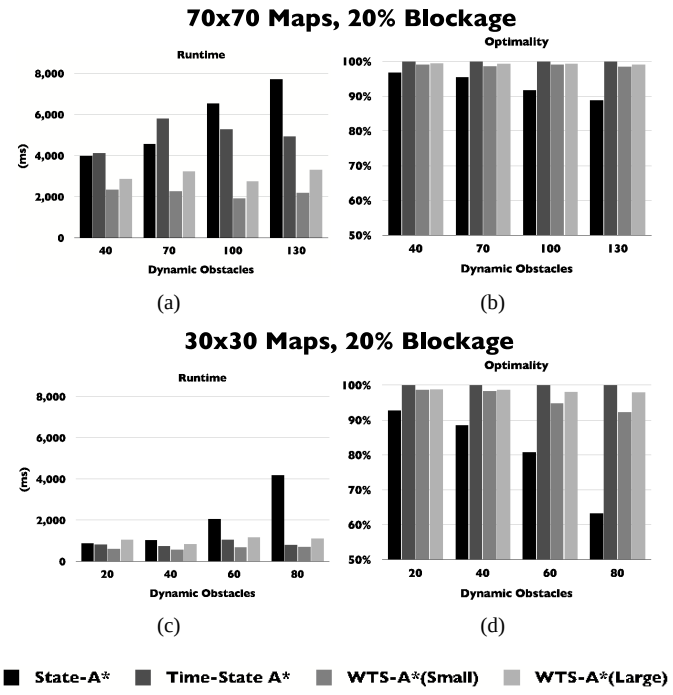


Fig. 10. Runtime and optimality results of 30 runs for State-A*, Time-State A*, WTS-A*(Small), and WTS-A*(Large) averaged across different numbers of dynamic obstacles in map sizes of 70×70 (a,b) and 30×30 (c,d) with 20% blockade.

space restriction. While WTS-A* had a much lower failure rate, it still failed to find the existing path in fewer than 7% of maps. This is due to the fact that WTS-A* assumes that each windowed path in the state space has a valid translation into the time-state space, which is not always true. TIME-STATE A* on the other hand is complete, which translates to a 0% failure rate.

Similar results were observed across different map sizes and obstacles quantity, although some additional observations can be made. Figures 10(a) and 10(b) depict the runtime and performance results of State A*, Time-State A*, and WTS-A* with window sizes of 20 and 40 in a map of size 70. As the number of dynamic obstacles increases, the extra cost of re-planning for STATE-A* dominated the cost of planning in the time-state space, as shown in Figure 10(a), indicating that as the number of obstacles increased, re-planning needed to occur more frequently. Even though TIME-STATE A* had to search in a larger space, most plans generated by STATE-A* were infeasible, leading to STATE-A* incurring a longer runtime than TIME-STATE A*. Eventually after 100 obstacles, this re-planning cost dominated the planning process in the larger space. The side-effect of such excessive re-planning can be observed in Figure 10(b), where the optimality of STATE-A* drops rapidly with increasing obstacles. Although TIME-STATE A* is guaranteed to find the optimal solution, it incurs a heavy computational cost. In contrast, WTS-A* achieve the best of both worlds: their running time is less than both that of STATE-A* and TIME-STATE A*, while the quality of the plans found remains high (about 98% of the optimal TIME-STATE A*).

In very small maps, the cost of re-planning is even more apparent. Figures 10(c) and 10(d) illustrate the runtime and performance results of all algorithms for maps of size 30. For any number of dynamic obstacles, STATE-A* and WTS-A* with a large window size exceeded the runtime of TIME-STATE A*. Since the size of this map was small, the number of possible paths to the goal was limited. TIME-STATE A* found the optimal path by a complete search through the search space, while both STATE-A* and WTS-A* had to perform a number of re-planning operations. These results suggest the applicability of using TIME-STATE A* for small search spaces with dynamic obstacles.

7. MISSION PERFORMANCE IN MAV '08

As described in section 3, the goal of the mission was to guide commandos across a field to a remote building. Our vehicle has a top speed of 10 m/sec, and the battery provides a total flight time of 10-12 minutes. We therefore divided the mission into multiple phases of mine detection, mine disposal and guard surveillance. Between each phase of the mission, we planned to return the MAV to the launch point to replace the battery.

The goal of phase 1 was to identify potential mine locations and begin guard vehicle estimation before having the MAV return to recharge. Unfortunately, once the guard position and trajectory were identified, the amount of energy required to return to the ingress point was underestimated, and the vehicle was lost after 710 seconds, having traveling 1.75 km. We were prepared to lose the vehicle in the field and therefore had multiple vehicles at the ingress point.

The goal of phase 2 was to identify additional mine locations, coordinate with the EOD vehicle to perform mine disposal and to begin execution of the commando plan. During this phase, the mine shown in Figure 11 was geo-located and successfully disposed of, and the MAV returned to the ingress point for recharging. Additionally, the commandos continued executing their planned motion towards the hostage building.

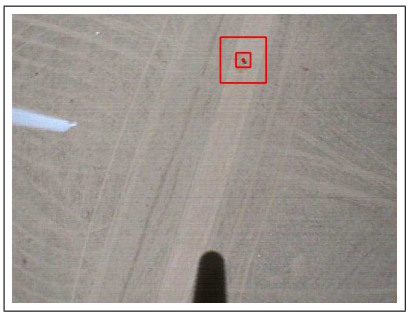
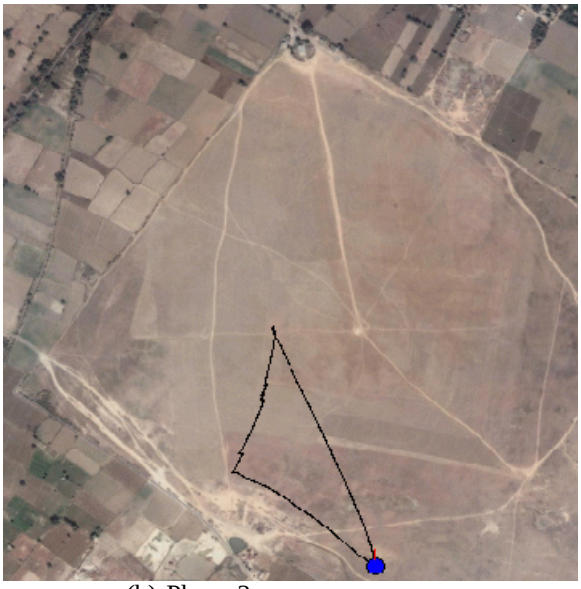


Fig. 11. A mine discovered during phase 2 embedded in a route between covered positions.

The goal of phase 3 was to finish identifying mine locations, finish disposing of remaining mines and re-acquire the guard trajectory before finishing the commando mission. A deliberate decision was taken by the human operators to abandon the vehicle in the field and avoid the time of the return trip to the ingress point for recharging, in order to provide additional time to complete the commando mission in the 40 minutes.

(a) Phase 1

Maximum height:	35.7 m
Distance traveled:	1759.2 m
Total flight time:	710.0 secs

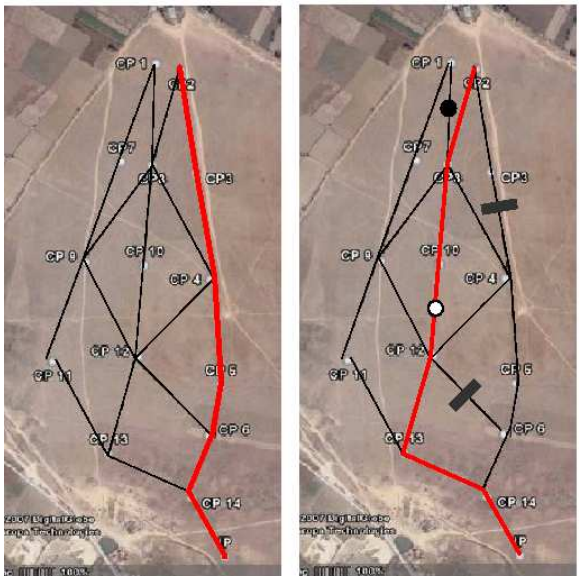


(b) Phase 2

Maximum height:	13.0 m
Distance traveled:	1247.2 m
Total flight time:	621.1 m

(c) Phase 3

Maximum height:	28.8m
Distance traveled:	1290.5m
Total flight time:	644.7 m



(d) Expected Ground Vehicle Path (e) Actual Path

Fig. 12. (a-c) The paths executed by the MAV. (d) The initial and actual plan executed by the commandos and EOD vehicle.

Figure 12(a-c) shows the actual paths flown by the MAV on each mission. Figure 12(d) shows both the initial plan of the ground vehicle computed using the WTS-A* algorithm, as well as the actual executed plan during the mission. In the final mission scenario, the guard vehicle motion was extremely deterministic and did not require much variation in the timing constraints so the timing information is not shown in the image. The path from cover point to cover point took 3 minutes and reliably avoided detection. The actual path taken by the vehicles changed from this expected path to the futtock (midline) path based on detected mines and obstacles information, and the resultant re-planning.

In general, we were very pleased with the performance of the MAV and the co-ordination between the tracked ground targets and the planned trajectories for the EOD vehicle and commandos. In particular, during phase 2 of the mission we were able to compensate for a temporary loss of GPS on the EOD vehicle; we repurposed the MAV temporarily to geo-

locate the EOD as it disposed of a mine. We flew a total of 4296.9m in 40 minutes, detected two mines and two ground obstacles and successfully disposed of the only mine along the planned trajectory.

8. Conclusion

This paper described critical hardware and software components of a combined micro air vehicle and ground vehicle system for performing a remote rescue task, as part of the MAV '08 competition organized by the US and Indian governments. While our system performed to our satisfaction and was awarded "Best Mission Execution", there are a number of key technical questions that remain unsolved before coordinated air and ground systems can become commodity technologies.

First, while the object detection and tracking system helped the human operators considerably in geo-locating objects, more work remains to be done in learning appearance-based methods and compensating for large camera motions to generate robust autonomous object detection and tracking. Second, there has been considerable amount of work in planning under uncertainty for multi-agent systems. Without an explicit stochastic model of the agent uncertainty, these planning algorithms were not useful, but in the future we plan to extend the planner to incorporate deliberate sensing actions at appropriate points in time, to allow the environmental dynamics to be learned, and robust planning under uncertainty algorithms to be used. Finally, the overall mission specification provided by the organizers allowed very simple task planning and rigid task execution. However, to allow more flexibility in planning surveillance, tracking and trajectory execution between the air and ground vehicles, we expect that more intelligent task planning will be required in the future.

9. Acknowledgements

The authors wish to thank the anonymous reviewers for their constructive discussions. The support of the Army Research Office under the MAST CTA is gratefully acknowledged. Ruijie He was supported by the Republic of Singapore Armed Forces. Abraham Bachrach was supported by Aurora Flight Sciences and the Air Force Office of Scientific Research under contract # F9550-06-C-0088. Alborz Geramifard was supported by the Boeing company, and NSERC. Samuel Prentice and Nicholas Roy were supported by the National Science Foundation Division of Information and Intelligent Systems under grant # 0546467. This project was supported by the Office of the Dean, School of Engineering and the MIT Air Vehicle Research Center (MAVRC) and their support is gratefully acknowledged. Col. Peter Young, Prof. Jonathan How, Spencer Ahrens and Brett Bethke provided additional support in the development of the vehicle and their support is gratefully acknowledged. The development of the vehicle vision system was supported by the Air Force Office of Scientific Research as part of the Defense University Research Instrumentation Program, "Cyber Flight Cage" contract # FA9550-07-1-0321. Electronic design of the vehicle was supported by Klaus-Michael Doth and his support is gratefully acknowledged.

References

Avidan, S. (2007). Ensemble tracking. *IEEE transactions on pattern analysis and machine intelligence* 29(2): 261–271.

Bagnell, J. and Schneider, J. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Bertsekas, D. (1995). Dynamic programming and optimal control .

Bouabdallah, S., Siegwart, R., and Caprari, G. (2006). Design and control of an indoor coaxial helicopter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Casbeer, D., Beard, R., McLain, T., Li, S., and Mehra, R. (2005). Forest fire monitoring with multiple small UAVs. In *Proceedings of the 2005 American Control Conference*.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Fraichard, T. (1999). Trajectory planning in a dynamic workspace: a 'state-time space' approach. *Advanced Robotics* 13(1): 75–94.

Furukawa, T., Bourgault, F., Lavis, B., and Durrant-Whyte, H. (2006). Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.-M., Hirzinger, G., and Rus, D. (2007). Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Hoffmann, G., Huang, H., Waslander, S., and C.Tomlin (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*.

Jaillet, L. and Simeon, T. (2004). A PRM-based motion planner for dynamically changing environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Javed, O., Shafique, K., and Shah, M. (2002). A hierarchical approach to robust background subtraction using color and gradient information. In *IEEE Workshop on Motion and Video Computing*.

Kalman, E., Rudolph (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering* 82(Series D): 35–45.

Ng, A., Kim, H., Jordan, M., and Sastry, S. (2004). Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*.

Porikli, F. (2006). Achieving real-time object detection and tracking under extreme conditions. *Journal of Real-Time Image Processing* 1(1): 33–40.

Quigley, M., Goodrich, M., Griffiths, S., and Eldredge, A.

(2005). Target acquisition, localization, and surveillance using a fixed-wing mini-UAV. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Schapire, R. (2003). The boosting approach to machine learning: An overview. *Lecture Notes in Statistics* 149–172.

Silver, D. (2005). Cooperative pathfinding. In *Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Tisdale, J., Ryan, A., Kim, Z., Tornqvist, D., and Hedrick, J. (2008). A multiple UAV system for vision-based search and localization. In *Proceedings of the 2008 American Control Conference*.

van den Berg, J. and Overmars, M. (2004). Roadmap-based motion planning in dynamic environments. Technical Report UU-CS-2004-020, Department of Information and Computing Sciences, Utrecht University.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.* 38(4).