Co-ordinated Tracking and Planning using Air and Ground Vehicles

Abraham Bachrach and Alborz Geramifard and Daniel Gurdan and Ruijie He and Sam Prentice and Jan Stumpf and Nicholas Roy

***Abstract*—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, including the design and operation of micro air vehicles (MAVs), using the MAVs to geolocate and track ground targets, and planning the motion of ground vehicles to reach the hostage location with detection.**

In this paper we describe our solutions to these technical challenges. Firstly, we summarize the design of our micro air vehicle, focusing on the navigation and sensing payload. Secondly, we describe the vision and state estimation algorithms used to track ground features through a sequence of images from the MAV, including stationary obstacles and moving adversaries. We examine different variants of an adaptive tracking algorithm and report the performance with respect to different target types. Thirdly, we describe the planning algorithm used to generate motion plans to using target information from the MAV, to allow the ground vehicles to approach the hostage building undetected by adversaries tracked from the air. We examine different variants of standard search algorithms that allow us to plan efficiently 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) capable of flying the necessary distances and carrying sensor payload to localize the hostages. The second technical challenge was the design and implementation of vision and state estimation algorithms to detect and track ground adversaries guarding the hostages. The third technical challenge was the design and implementation of robust planning algorithms for using the co-ordinated MAV state estimates to generate tactical motion plans for ground vehicles to reach the hostage location without detection by the ground adversaries.

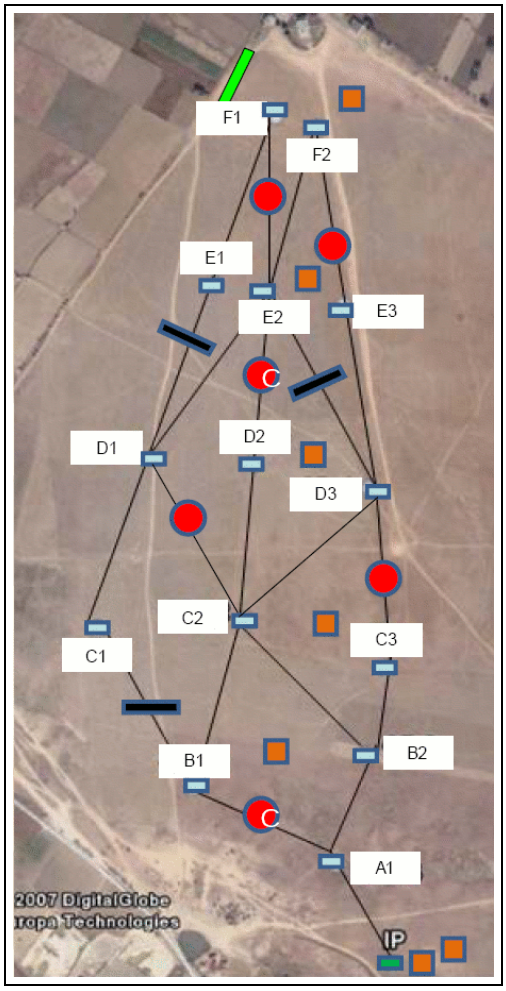
In this paper we describe our solutions to these technical challenges. Firstly, we summarize the design of our micro air vehicle, focusing on the navigation and sensing payload. Secondly, we describe the vision and state estimation algorithms used to track ground features through a sequence of images from the MAV, including stationary obstacles and moving adversaries. Specifically, we used an adaptive algorithm that learned to discriminate image features corresponding to the target from the background, coupled with standard Bayesian filtering to track the object in a global co-ordinate system from image to image. Thirdly, we describe the planning algorithm used to generate motion plans to allow the ground vehicles to approach the hostage building undetected by adversaries tracked from the air. In order to plan with respect to the changing position of the ground adversaries, we examine different variants of standard search algorithms that allow us to plan efficiently and react to unexpected or modeled changes in the position of the ground adversary. Finally, we provide results of our system's performance during the mission execution.

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

2. The MAV '08 Mission

The MAV '08 mission was a hostage-rescue scenario, in which commandos must be guided across a field to a remote building. The hostage building was guarded by a moving adversary; to allow the commandos to reach the building undetected, an estimate of the guard vehicle and its field of view was required. As the guard vehicle moved, the commandos were able to take advantage of known covered positions 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, otherwise, the commandos remained hidden. Further complicating the problem, 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. Once detected and geolocated, the mines could be disposed of using an explosive ordinance disposal (EOD) vehicle. Finally, the commandos were required to reach the hostage building in 40 minutes from the start of the mission, including all surveillance, mine disposal and guard tracking.

In order to obtain the position of the guard vehicle, detect the route blockages and geolocate the mines, aerial surveillance was essential. However, the MAV '08 rules dictated a maximum size of air vehicle of 30cm. Our approach to the mission was to use a set of rotorcraft to survey the field, search for mines and obstacles, and also maintain a position



(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 light blue squares are cover points for the commandos, the red dots are mine locations and the black 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 light blue cover positions can be seen faintly.

estimate of the guard vehicle. Figure 1(a) shows a map of the field, containing the known covered positions (A1, ..., F2). The red dots are mine positions and the black bars are route blockages (these positions were not provided during the competition). The ingress point for MAV launch, commando and EOD vehicle entry is shown at the bottom right 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 1km field is shown in figure 1(c).

3. 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 2. The total flight time of the vehicle is 10-12 minutes, with maximum speed of 10 m/sec, depending on wind conditions, temperature, etc.

The navigation system consists of a 60MHz Philips ARM microprocessor, u-blox GPS receiver, compass, IMU and pressure sensor. The ARM microprocessor integrates the IMU and GPS measurements to provide a consistent state estimate at 1000 Hz. The on-board software accepts waypoints in the GPS (world) co-ordinate frame and uses PID control to achieve the desired position. The height estimate is relative to the position of the vehicle on take-off. The waypoint controller attempts to achieve the desired position initially with 15m 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 the 30 seconds, the control software assumes that external factors (i.e., wind) are interfering and ends the attempt. In this way, we are guaranteed some baseline level of performance (15m), and the vehicle will attempt to achieve a higher level of accuracy without excessive time delays.

The vehicle additionally carries a Digi 900MHz Xtend RF module operating at 100 mW. We communicate with the MAV with a USB-serial converter to the Xtend base station; the bandwidth is such that we typically can get telemetry at 40

Fig. 2. Our six-rotor helicopter with bird's-eye video camera. The helicopter is 29cm in diameter and weighs 142g without the navigation electronics, camera or communication hardware.

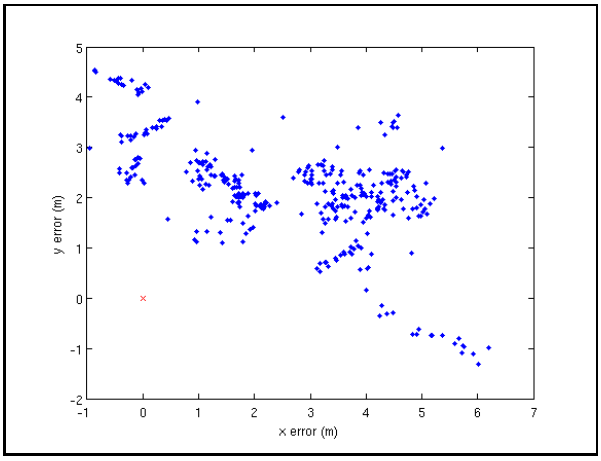


Fig. 3. Target geolocation estimates for an obstacle with a known location. The true location is mapped to $(0, 0)$.

Hz. The vehicle is configured to use the digital data link as the primary communication mechanism. If the digital data link is lost, then the vehicle throttles back to 30% and attempts to land safely. This can be over-ridden with an auxiliary 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 camera sensor is a Panasonic KX141 480 line CCD camera with a 90° field-of-view lens. Additionally, we use a LawMate TM-240500 2.4 GHz 500 mW transmitter, and a 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 from directly forward to straight down. 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 $5m$.

4. Object Detection and Tracking

The first phase of the MAV '08 mission involved surveying the field, identifying obstacles and mines, and then beginning to track the guard vehicle, leading to the second challenge of identifying the positions of targets on the ground. Our approach was to locate objects in the image, then use the known position of the MAV from GPS and a calibrated camera model to geolocate the objects, assuming the objects were on the known ground plane. However, due to noisy estimates of the vehicle pose and fast vehicle dynamics, it was necessary to combine projections from many successive images to achieve a more accurate geolocation estimate. For example, when we analyzed the geolocation estimates for an object with known location, we see in Figure 3 that the individual estimates deviated from the ground-truth by up to 6m. On the other hand, the mean of the measurements deviated by only 3.3m. This residual error was likely due to errors in the calibration of the transformation between the body coordinates and the camera coordinates.

We were given minimal prior information of the appearance of the guards, obstacles and mines and therefore did not have enough information regarding a specific color, shape, or motion to allow general object detection of any of the target objects. As a result, focused on object tracking, relying on a human operator to detect the initial appearance of each object in the scene, and then tracking the object in successive frames. To accomplish this, we used a modified version of the classifier-based adaptive ensemble tracker, developed in [1]. While this approach did not allow completely autonomous operation, it significantly reduced the amount of attention required from the operator.

4.1. Learning Object Appearance Models

Once an initial estimate of the target object is identified by a human operator in an image, we pose the tracking problem as a classification problem, where a classifier is trained in an online fashion to separate pixels which belong to the object from background pixels. To train the classifier, we assume that the object is localized within a known $n \times n$ 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 [6]. Each pixel i is therefore a separate training instance consisting of a d-dimensional feature vector $\mathbf{x}_i$ and a label y_i. AdaBoost requires a weak classifier, which in this algorithm is implemented as a separating hyperplane $\mathbf{h}$, such that

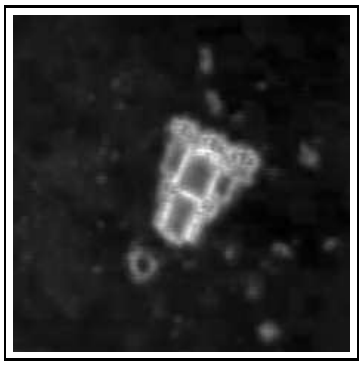
$$\hat{y}(\mathbf{x}_i) = \text{sign}(h^T \mathbf{x}_i) \quad (1)$$

where $\hat{y}(\mathbf{x})$ is the classifier output label for instance $\mathbf{x}$. The separating hyperplane for a set of examples is computed using weighted least squares given a training data set consisting of

pixel features and labels, $\{\mathbf{x}_i, y_i\}$. We then boost to learn an ensemble of classifiers $h_1, \ldots, h_n$ with associated weights $\alpha_1, \ldots, \alpha_n$. In addition, we train a separate ensemble of classifiers for each of w image scales in order to capture the distinctive appearance characteristics at different scales. Finally, we can 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. The output of the classifier is a new image where each pixel represents the probability that a given pixel belongs to tracked object.

Figure 4(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 4(b) shows the response of the classifiers to the same image after training. Notice that the classifiers have the most response along the sharply distinct color boundaries.

(a) Original Image



(b) Ensemble Filter Response

Fig. 4. (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 car.

During tracking, the object appearance will vary over time; for instance, the orientation of edge features will change as objects rotate in the image. We therefore continually learn new classifiers on the incoming images. After tracking is completed on each image, the image is used as a new training instance. The k best classifiers are retained, and $n - k$ additional classifiers are trained, again using boosting. In order to ensure that this retraining of the classifier does not cause the original concept to become lost over time, we also investigated a model in which m of the original n classifiers are kept, regardless of their weight. This ensures that at least some of the classifiers where trained with labels that were known to be correct.

4.2. Object Tracking

In [1], a mean-shift tracker is applied to the probability image to update the estimate of the object location, in which a region of the image is classified and the maximum likelihood pixel in the image is assumed to be the new center of object. While this approach works quite well for relatively stationary cameras, we found that the mean-shift approach was not able to handle the fast motion of the MAV platform.

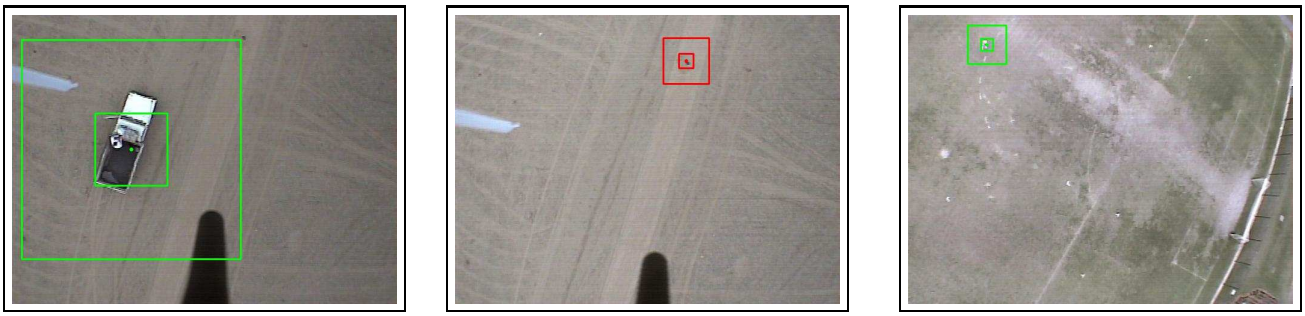
For example, the tracker was regularly able to follow the EOD vehicle (figures 4(a) and 5(a)) for the entire duration of its time in the field of view, usually 10-20 seconds. This good performance was due to the fact that the EOD vehicle had a distinctive appearance, leading to computed features that were very discriminative. In addition, the large object sizes made the relative motion of the MAV less significant. In contrast, tracking the mine in figure 5(b) and the walking person in figure 5(c), was more challenging. Tracking the mine was particularly difficult due to its extremely small size and non-distinct circular shape. Although the person had a very distinct appearance in the image, its small size relative to the motion of the MAV in the image caused the tracker to lose the tracked person almost immediately without an ego-motion estimation.

As a result, we modified the tracking algorithm to use a motion model coupled with Bayesian filtering to update the object position estimate. The tracker can more robustly estimate the object position in the image by using the ego-motion estimate to bias the motion update. This ego-motion estimate is essential for compensating for unpredictable motions of the camera, which would otherwise cause the tracker to get lost. The motion estimate is computed using the Pyramidal-Lucas-Kanade optical flow implementation available in OpenCV [5]. The optical flow algorithm computes a set of displacements for features in the image, which are clustered using expectation-maximization to identify the single largest flow direction. The flow direction is then used to compute the affine transform that best explains the apparent motion.

The affine transformation is used as a motion model and the ensemble tracker as the sensor model, in order to more accurately estimate the object trajectory. We use a particle filter to implement the probabilistic estimate $p(x_t|z_{0:t})$, where x_t is the location of the object in the image at time t, $p(x_t|z_{0:t})$ is the probability of the object at the location after having received measurements $z_{0:t}$, such that

$$p(x_t|z_{0:t}) = \alpha p(z_t|x_t) \int_{X_{t-1}} p(x_t|x_{t-1}) p(x_{t-1}|z_{0:t-1}) dt, \tag{2}$$

where $p(x_{t-1}|z_{0:t-1})$ is the object distribution on the previous time-step, and $p(z_t|x_t)$ is our sensor model (the likelihood of detecting the object at position z_t given the object is at x_t). $p(x_t|x_{t-1})$ is the model of how the object moved in the image, which we assume to be Gaussian motion with mean given by the optical flow algorithm and some fixed variance. In contrast to more conventional filtering techniques such as the Kalman filter [13], the particle filter is useful for modeling the non-linear sensor and motion models and the non-Gaussian noise distributions. In contrast to ground vehicles and fixed-wing aircraft that have generally stable attitudes, the attitude of the



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

Fig. 5. Examples of the variety of objects tracked. (a) The EOD vehicle for mine disposal. (b) A mine embedded in a route between covered positions. (c) A walking person. (a) was relatively easy to track, but (b) and (c) required a better motion prediction model.

No optical flow, no retraining	0 errors
No retraining	0 errors
Keep first 3 classifiers	0 errors
Full retraining	0 errors

TABLE I
EOD VEHICLE (FIGURE 5A), 250 FRAMES OVER 17 SECONDS.

No optical flow, no retraining	0 errors
No retraining	0 errors
Keep first 3 classifiers	0 errors
Full retraining	0 errors

TABLE II
VEHICLE (FIGURE 4A), 88 FRAMES OVER 9 SECONDS.

MAV rotorcraft is particularly dynamic and non-linear; the frequent attitude changes of the MAV generally cause very large displacements of the object in the image.

Returning to figure 5(b) and 5(c), when tracking the person, we were able to maintain the track for over 2 minutes, requiring human intervention only once when the person left the frame for a few seconds. This good performance was made possible by the motion model provided by optical flow. Similarly, when tracking the mine in figure 5(c), given the motion model from optical flow, the tracker was able to track for over 30 seconds, only needing human intervention once due to an abrupt perturbation of the MAV attitude.

Given the tracked position of an object in the image, we can recover the position of the object in the world co-ordinates from knowledge of the intrinsic camera properties, such as camera focal length, center of projection, etc., the rigid transformation from the camera image plane to the center of body of the vehicle, and the knowledge of the MAV GPS position and attitude. We recover the camera parameters and camera transformation in a standard least-squares calibration process. However, the GPS localization and attitude of the MAV are not known perfectly and in particular, small errors in attitude can lead to substantial errors in projecting from image co-ordinates to world co-ordinates. As a result, we apply a second level of Bayesian filtering to maintain a cleaner estimate of the target location in the global coordinates. In contrast to the image-space filter, where we generally assume that the motion variance is large and emphasize the measurement model, when tracking in global coordinates, we place more weight on the motion model and model the projections from image coordinates to world coordinates as very noisy measurements. In this way, we average over many measurements to attain a more accurate estimate of the target location.

4.3. Tracking Analysis

Human intervention is still required for continuous tracking of the objects to initialize the tracker for new objects, and to potentially restart the tracker when it fails. We evaluated the tracker in a number of configurations and test cases, with and without optical flow, retaining different numbers of the original classifiers, and measured the number of times that the tracker object estimate diverged from a hand-labeled ground truth image track after the initialization step. We tested the object tracking across a wide variety of scenes with very different targets, and were able to see performance that allowed unattended tracking for extended periods of time.

The easiest object tracking problem was the EOD vehicle, shown in figure 5(a). This data set consisted of a 17 seconds of video, for a total of 250 frames[1], and led to good performance for all tracker configurations due to the large vehicle size, crisp features, and stable hover of the MAV. As table I shows, even without the motion model from the optical flow, or the online classifier retraining, the tracker never lost the vehicle after the initialization. We also examined the effect of not retraining the classifier after each image, or retraining all but the first three classifiers (to ensure the original concepts does not get lost). Again, table I shows that retraining the classifier had no effect on the tracker performance with this target. Similarly, the tracker performed equally well in all configurations when tracking the vehicle shown in figure 4(a), with results in table II.

Tracking the walking person shown in figure 5(c) was challenging due to the small size of the person from the MAV height. However the algorithm was still able to maintain excellent performance. The MAV experienced relatively stable

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

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)

TABLE III
PERSON, 2683 FRAMES, 150 SECS. FREQUENCY OF REQUIRED TRACK REINITIALIZATIONS (TOTAL NUMBER OF ERRORS SHOWN IN PARENTHESES).

flight, which made the task easier than in other datasets, however due to the small object size, the ego-motion estimate was critical. As table III shows, the optical flow played a critical role in keeping the tracking estimate on target. In addition, it is clear that the adaptation to the object appearance led to improved tracking. As the person moved around the field, its appearance remained relatively constant, however the background changed drastically when the person moved from the green grass to the gray dirt patches. As a result, by retraining, and adapting the classifier it was able to maintain enough discrimination between the person and the background to maintain the track.

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 the camera pointed forwards, the motion of the scene in the image due to the MAV motion became more pronounced from frame to frame. In addition, as can be seen in table IV, the hedges surrounding the bank building were exactly the same color and similar shape to the guard vehicle. As a result, the tracker lost the track of the guard vehicle far more often than in the other scenes we tested on.

In this dataset, the major factor that resulted in the tracker losing track was the motion of the camera, rather than a change in appearance of the guard 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 resulting in the tracker not being able to recover. While it is clear that the optical flow plays an important role in keeping the tracking on target, it 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, one needs to include more sophisticated object detection. Once the ensemble-based tracker loses the target, there is no way to recover using an appearance-based tracker learned online, since any corruption of the current object estimate will propagate, corrupting subsequent classifiers. As a result, an object detector with higher-level appearance-based invariants is needed to recover from object tracker failures in the general case.

5. GROUND VEHICLE PLANNING

Given the ability of the MAV to estimate the guard position and trajectory, the third challenge was to be able to plan a

TABLE IV
THE GUARD VEHICLE CIRCLING THE HOSTAGE BUILDING.

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)

TABLE V
GUARD VEHICLE, 1000 FRAMES, 54 SECONDS. FREQUENCY OF REQUIRED TRACK REINITIALIZATIONS (TOTAL NUMBER OF ERRORS SHOWN IN PARENTHESES).

trajectory for the commandos to the hostage building without their being detected by the guard vehicle. Additionally, when the MAV found mines, we wanted to be able to plan a trajectory for the EOD vehicle to the mines, also without being detected. We treated these problems symmetrically as a motion planning problem for a generic ground vehicle (GV).

Standard motion planning algorithms are generally based on search strategies through a discretized state space. Although the specific planning problem in the MAV '08 problem was centered around routes between the cover points (marked $A1, \ldots, F2$ in figure 1a), we developed a general purpose motion planner that would be more flexible to unexpected guard motion and allow us to express a wide range of trajectories that may not exactly follow straight-line routes between cover points.

Based on the initial problem description, our motion planner makes a number of assumptions. Not all of these assumptions were required for the MAV '08 competition, but in some cases allowed us to address more general problems. The planner assumes a discretization of the planning area, specifically a regular grid, and assumes the GV can move from a grid cell x to any of the 4-connected neighbors. We assume that such a motion incurs a cost, and the goal is to find the lowest cost sequence of states from the start to the goal without being detected by the guard vehicle. The guard has 360° field of view with finite range, and we have a prior map of the environment giving the location of obstacles that would obstruct the guard field of view, occluding the GV from the guard. Additionally, the planner assumes that the current position of the guard vehicle is known, and there is a model of the guard dynamics that allows the guard position to predicted into the future. The planner must therefore incorporate this model of the temporal behavior of the guard in generating paths that avoid detection. The temporal constraint typically requires planning in both

Algorithm 1 : STATE-A*

Require: $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$, $\mathbf{x}_{guard}$
1: $\pi \leftarrow$ A*$(\mathbf{x}_{start}, \mathbf{x}_{goal})$
2: $i \leftarrow$ COLLIDE$(\pi, \mathbf{x}_{guard})$
3: **while** $i > 0$ **do**
4: MARK_BLOCKED$(\pi[i])$
5: $\pi_{tail} \leftarrow$ A*$(\pi[i-1], \mathbf{x}_{goal})$
6: **if** $(\pi_{tail} == \mathbf{null})$ **then**
7: **return** null
8: **end if**
9: $\pi \leftarrow \pi[0 : i-1] + \pi_{tail}$
10: $i \leftarrow$ COLLIDE$(\pi, \mathbf{x}_{guard})$
11: **end while**
12: **return** π

space and time, which can lead to substantial computational complexity. Given the large size of the map, planning in space and time may not be feasible, and so we examined three different strategies for planning with respect to the guard vehicle dynamics, to identify a strategy that scales well with minimal loss in planner performance.

I. STATE-A*

To determine if the additional complexity of planning in time and space can be avoided, we first examined the performance of planning only in the state space of the GV. The STATE-A* 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 vehicle, STATE-A* forward-simulates the plan given that the guard starts at position $\mathbf{x}_{goal}$. As the GV is simulated to move to the next state in the plan $\mathbf{x}^i$, the guard vehicle position is predicted using the current estimate of the guard motion, and the state $\mathbf{x}^i$ is tested to see if a detection (and failure) would result. Any state that is predicted to result in detection as a result of executing the plan is inserted into the map as a static obstacle, and a new plan is generated. This process is repeated until no collision with dynamic obstacles is found or the algorithm fails to find a plan. Algorithm 1 shows the STATE-A* in detail. This algorithm assumes that the internal A* algorithm has access to a cost map including obstacles. The COLLIDE subroutine simulates the GV motion along the plan π, and the MARK_BLOCKED subroutine modifies the cost 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 GV, rather than changes in both time and position. However, this computational savings also restricts the plan space, in that the search process cannot take advantage of actions such as PAUSE (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.

II. TIME-STATE A*

The TIME-STATE-A* algorithm, developed by Fraichard [7], represents the state of the GV as both a position and time. In order to account for the guard vehicle, the 2-D space is extrapolated into the time domain, creating a three-dimensional cost map (or "cube"), where each cell represents a separate (x, y, t). All actions are assumed to have the same, constant duration Δt. In addition to the four motion commands, we add a PAUSE action that only changes the time variable by the same constant amount Δt as motion commands. Longer pauses can be achieved by executing PAUSE repeatedly. We then search through the cube using standard A* as before, but limiting the actions from every cell to be the 5-connected grid cell in the next time step. (The cube is 5-connected because the legal transitions are the four motions and the PAUSE action). The Manhattan distance between the robot's current position and the final goal in the 2-D space is used as the heuristic. This algorithm again assumes that A* has access to a cost map that includes the obstacles. Algorithm 2 shows the TIME-STATE-A* in detail.

Algorithm 2 : TIME-STATE A*

Require: $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$, $\mathbf{x}_{guard}$, t_{max}
1: $\pi \leftarrow$ A* $((\mathbf{x}_{start}, \mathbf{x}_{guard}, 0), (\mathbf{x}_{goal}, \cdot, t_{max}))$
2: **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.

There is a slight abuse of notation in that the goal state of the A* process is $(\mathbf{x}_{goal}, \cdot, t_{max})$, which we use to denote a goal state of the search where the guard can be in any position. By modeling time explicitly during the search process, the TIME-STATE A* algorithm can express a wider variety of plans to incorporate plans that deliberately wait for the guard vehicle to move. Additionally, the search incorporates knowledge of the guard vehicle 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 number of actions (and therefore the branching factor of the search), and furthermore substantially increasing the state space by including time, may have a significant effect on the ability of the search process to find good plans.

WINDOWED TIME-STATE A* (WST-A*)

Since the search grows exponentially with the search depth, by reducing t_{max}, the search space can itself be reduced, only including plans of length at most t_{max}. However, this may significantly reduce the ability to find good plans when plans need to be longer than t_{max}, which is likely across a 1 km distances. We also examine an intermediate approach by iterating TIME-STATE-A* search in limited time window.

The complete algorithm is shown in Algorithm 3. 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

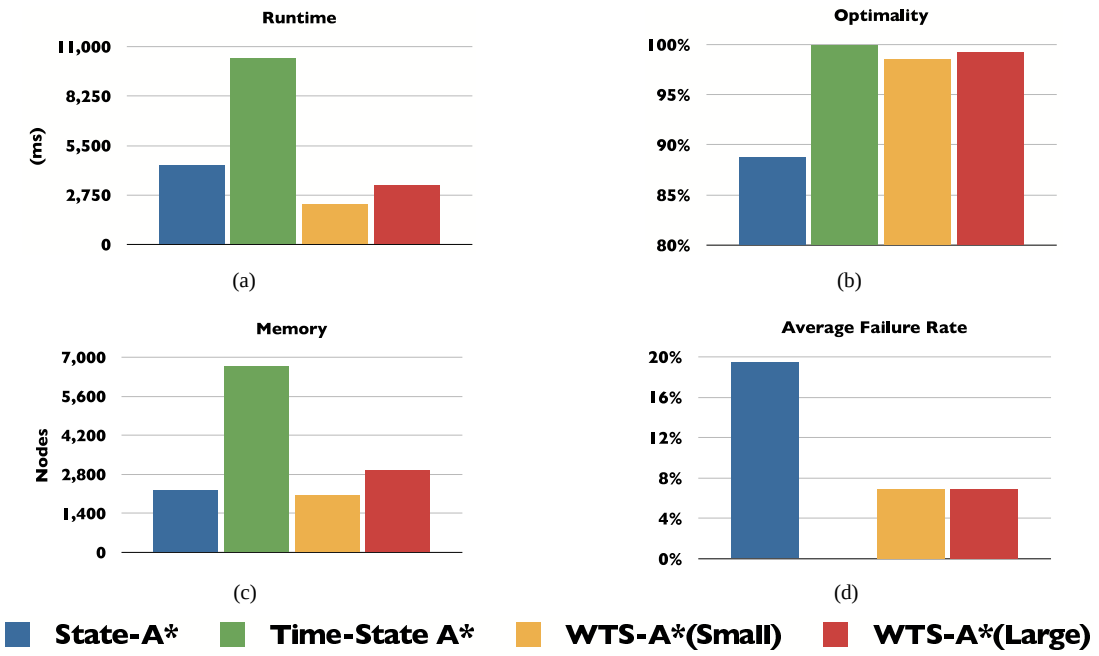


Fig. 6. Average runtime, optimality, memory and failure rate of State-A*, Time-State A*, WST-A*(Small), and WTS-A*(Large) across planning problems of different sizes.

Algorithm 3 : 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

of the sub-plan, the plan between these states is regenerated using TIME-SPACE-A*. Notice that the t variable is used to maintain the time required to execute each subplan $\hat{\pi}^i$, to ensure a proper connection between each section of the path.

5.1. 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 VI illustrates the 24 map settings used for simulations. All algorithms were tested on each setting with 30 randomly generated maps. 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 VI
RANGE OF DYNAMIC OBSTACLES FOR EACH MAP SETTING.

Figure 6 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 (figure 6-a). It is quite interesting that on average, WTS-A* 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 6-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 6-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 the lead, STATE-A* memory usage is now on par with WTS-A* with small window size and even less than WTS-A* with large window size. This highlights the fact that STATE-A* searched through more compact space, although it had to replan more often. Figure 6-d shows the average failure rate of methods. As expected STATE-A* suffered the

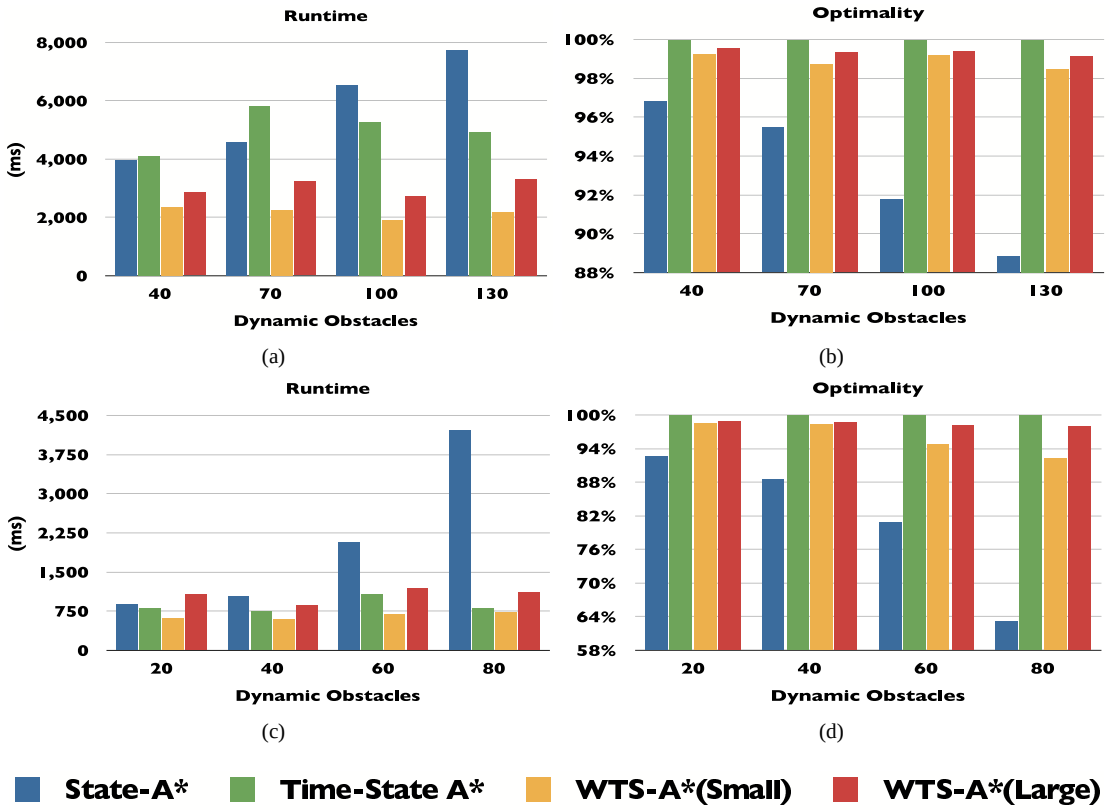


Fig. 7. Runtime and optimality results of 30 runs for State-A*, Time-State A*, WTS-A*(Small), and WTS-A*(Large) averaged across different numbers of dynamic obstacles in a map sizes of 70×70 (a,b) and 30×30 (c,d) with 20% blockade.

most because of its substantial search space restriction. While WTS-A* had a much lower failure rate, it still failed to find the existing path in less 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 into 0% failure rate.

Most domain specific results followed the averaged graphs. Although we encountered a few stark observations. Figure 7-(a,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. It can be seen that 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 (figure 7-a), indicating that as the number of obstacles increased, re-planning needed to occur more frequently. While TIME-STATE-A* had to search in a larger space, most plans found by STATE-A* were infeasible, leading to more re-planning. Eventually after 100 obstacles, this re-planning cost dominated the planning in the larger space. The side-effect of such excessive re-planning can be observed in figure 7-b. The quality of the solutions found by STATE-A* drops rapidly. TIME-STATE-A* is guaranteed to find the optimal solution. In contrast, both versions of the WTS-A* achieve the best of both worlds: their running time is less than of both STATE-A* and TIME-STATE A*, while the cost of the plans found is nearly optimal (about 98% of the optimal TIME-STATE A*).

In very small maps, the cost of re-planning was fatal even for WTS-A* with large window size. Figure 7-(c,d) illustrates the runtime and performance results of all methods in map of size 30. For any number of dynamic obstacles, STATE-A* and WTS-A* with 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. This observation suggests the use of TIME-STATE A* for small search spaces with dynamic obstacles.

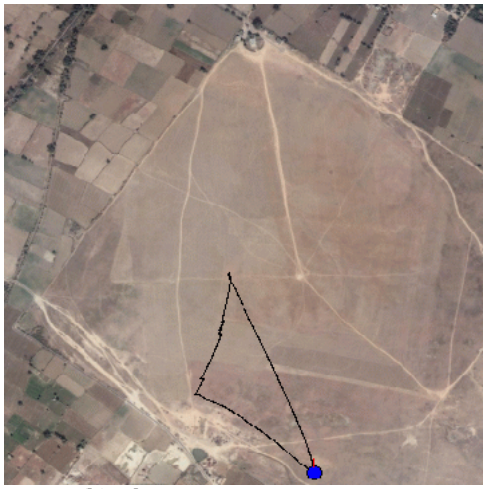
6. MISSION PERFORMANCE IN MAV '08

As described in section 2, the goal of these systems 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 returning to be recharged. 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, after traveling 1.75 km. (Not included in the flight time is the time required to launch the vehicle

(a) Phase 1

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

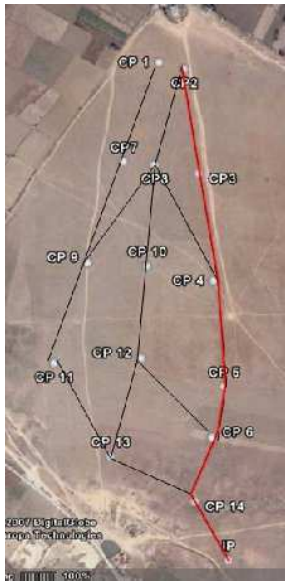


(b) Phase 2

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 GV Path

Fig. 8. (a-c) The paths executed by the MAV. (d) The expected plan executed by the commandos and EOD vehicle.

and begin the mission execution.) 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, co-ordinate with the EOD vehicle to perform mine disposal and to begin execution of the commando plan. During this phase, once mines were geolocated and successfully disposed of, followed by the MAV successfully returning to the ingress point for recharging. Additionally, the commandos continued executing their planned motion towards the hostage building.

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.

Figure 8(a-c) shows the actual paths flown by the MAV on each mission. Figure 8(d) shows the expected trajectory of the GV computed using the WTS-A* algorithm. 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, obstacles and the resultant re-planning.

We were in general 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 temporary lost GPS on the EOD vehicle; we repurposed the MAV temporarily to geolocate 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.

7. Related Work

We brought together work from the fields of robotics, computer vision, and planning to compete in the MAV '08 competition. As a result, there is much in the literature that would be considered related work. There have been many rotorcraft UAV platforms developed, including quadrotors such as the one developed in [9] and [10] which operate on the same principles as the hexrotor developed here; as well as other morphologies such as coaxial [2] and conventional helicopter platforms [14]. However, to the best of our knowledge, our hexrotor platform is the first to acheive fully autonomous outdoor flight at such a small scale, all the while carrying a useful sensor payload.

In the computer vision literature there has been a lot of work on object tracking, [20] and [15] give a good overview of the current state of the art. While there have been many successful algorithms such as background subtraction [12], mean shift tracking [4], and of course ensemble tracking [1] which our algorithm is based on, all of these algorithms make assume relatively stationary cameras, which does not hold for the camera on our vehicle. On the other side of things, there is also a lot of work on tracking in the UAV literature [8], [18], [16] and [3]; however, much of this work focuses on the abstract problem of tracking the target over time. They do not actually integrate powerful enough vision algorithms to effectively use a camera to track the objects, without making major assumptions regarding their appearance. In this work, we have integrated powerful computer vision algorithms with bayesian filtering to effectivly track the targets over time.

Path planning with moving obstacles has been a challenging problem for researchers in many fields, including robotics and navigation. Van den Berg and Overmars [19] explored the idea of using a Probabilistic Road Map (PRM) to first generate discrete points in a continuous map which takes into account only static obstacles. Given the model of the moving obstacles, the resulting points are then extended into the time dimension to calculate the optimum path. Jaillet and Simeon [11] follow a similar approach to generate the initial map based on static obstacles and then use a lazy-evaluation technique to reach a complete map in state-time space. Similarly, Fraichard considered the path planning in the state-time space [7]. Using this idea, all dynamic obstacles in the 2-D space are represented as static obstacles in the 3-dimensional space. Finally a general path planning algorithm can be used to find the 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 [17] explored the similar approach explained by Fraichard to resolve collisions in multi-agent path planning using a reservation table.

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 co-ordinated air and ground systems can become commodities.

Firstly, while the object detection and tracking system helped the human operators considerably in geolocating 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. Secondly, there has been considerable amount of work in planning under uncertainty for multi-agent systems but we have not yet taken advantage of these methods to keep the system complexity at a manageable level. However, in the future, we plan to extend the planner to incorporate deliberate sensing actions at appropriate points in time, to allow more flexible response to environmental dynamics. 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.

References

[1] Shai Avidan. Ensemble tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[2] Samir Bouabdallah, Roland Siegwart, and Gilles Caprari. Design and control of an indoor coaxial helicopter. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2930–2935, Oct. 2006.

[3] David W. Casbeer, Sai ming Li, Randal W. Beard, and Raman K. Mehra. Forest fire monitoring with multiple small uavs.

[4] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Ieee cvpr 2000 real-time tracking of non-rigid objects using mean shift.

[5] Intel Corporation. Open source computer vision library (opencv). http://www.intel.com/technology/computing/opencv/index.htm.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[7] T. Fraichard. Trajectory planning in a dynamic workspace: a 'state-time' approach, 1999.

[8] T. Furukawa, F. Bourgault, B. Lavis, and H.F. Durrant-Whyte. Recursive bayesian search-and-tracking using coordinated uavs for lost targets. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2521–2526, 15-19, 2006.

[9] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. *Robotics and Automation, 2007 IEEE International Conference on*, pages 361–366, April 2007.

[10] Gabriel M. Hoffmann, Haomiao Huang, Steven L. Waslander, and Claire J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2007. AIAA Paper Number 2007-6461.

[11] L. Jaillet and T. Simeon. A prm-based motion planner for dynamically changing environments. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2:1606–1611 vol.2, 2004.

[12] Omar Javed, Khurram Shafique, and Mubarak Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *in IEEE Workshop on Motion and Video Computing*, pages 22–27, 2002.

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

[14] Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Inverted autonomous helicopter flight via reinforcement learning. In *In International Symposium on Experimental Robotics*. MIT Press, 2004.

[15] Fatih Porikli. Achieving real-time object detection and tracking under extreme conditions.

[16] Morgan Quigley, Michael A. Goodrich, Stephen Griffiths, and Andrew Eldredge. Target acquisition, localization, and surveillance using a fixed-wing mini-uav. In *in International Conference on Robotics and Automatio*, pages 2600–2605, 2005.

[17] David Silver. Cooperative pathfinding. In *Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005.

[18] John Tisdale, Allison Ryan, Zu Kim, David Trnqvist, and J. Karl Hedrick. A multiple uav system for vision-based search and localization.

[19] Jur P. van den Berg and Mark Overmars. Roadmap-based motion planning in dynamic environments. Technical Report UU-CS-2004-020, Department of Information and Computing Sciences, Utrecht University, 2004.

[20] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), 2006.