Co-ordinated Tracking and Planning using Air and Ground Vehicles

Abraham Bachrach[1], Alborz Garamifard[1], Daniel Gurdan[2], Ruijie He[1], Sam Prentice[1], Jan Stumpf[2], and Nicholas Roy[1]

[1] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139.
abachrac@mit.edu,agf@mit.edu, ruijie@mit.edu, prentice@mit.edu, nickroy@mit.edu

[2] Ascending Technologies GmbH. Graspergerstr. 8, 82131 Stockdorf Germany.
daniel.gurdan@asctec.de, jan@asctec.de

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 such technical challenge was the design and operation of a micro air vehicle (MAV) capable of flying the necessary distance and carrying a sensor payload for localizing 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 that could co-ordinate with the MAV state estimates and 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 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. 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. Finally, we provide results of our system's performance during the mission execution.

2 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 28 cm rotor-tip to rotor-tip and weighs 142 grams without the navigation electronics, camera or communication hardware. The vehicle is shown in figure 1. The total flight time of the vehicle is

10-12 minutes, with maximum speed of 10 m/sec, depending on wind conditions, temperature, etc..

Fig. 1. Our six-rotor helicopter with bird's-eye video camera. The helicopter is 28cm in diameter and weighs 142g without the navigation electronics, camera or communication hardware.

The navigation system consists of a 60MHz Philips ARM microprocessor, μ-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 waypoint first to within 15m accuracy, and then to within 2.5 m accuracy. If the waypoint error is not reduced from 15 m to 2.5 m in 30 seconds, the control software assumes that external factors (i.e., wind) are interfering and holds the current position. 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 receive telemetry at 40 Hz.

The camera sensor is a Black Widow KX141 480 line CCD camera with 90° field-of-view. Additionally, we use a Black Widow TD240500TX 2.4 GHz 500 mW transmitter, and a YellowJacket YJS24 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.

3 Object Detection and Tracking

The first phase of the MAV '08 mission involved visual surveillance of the field to identify obstacles and mines, followed by tracking the guard vehicle. Once an object was located in an image, the known position of the MAV from GPS and a calibrated camera model were then used to geolocate the object (assuming the object was on the known ground plane). However, due to the noisy estimates of the vehicle pose, it was necessary to combine projections from multiple images to achieve a more accurate geolocation estimate. Given the minimal prior information of the appearance of the guards, obstacles and mines, we did not have enough information regarding a specific color, shape, or motion to allow general object detection. As a result, we focused on object tracking, where given an initial example of the object in an image, we could track it in successive images. 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.

3.1 Learning Object Appearance Models

To find the object in an image, the tracking problem is posed as a classification problem, where a classifier is trained in an online fashion to separate the pixels belonging to the object from the 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 [3]. Each pixel i is therefore a separate training instance consisting of 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

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

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 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 2(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 2(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. 2. (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.

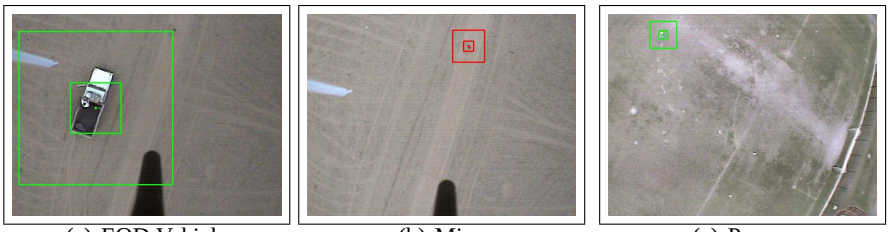
During tracking, the object appearance will vary over time; for instance, the orientation of edge features will change as objects rotate in the image. We therefore continually learn new classifiers on the incoming images. After tracking is completed on each image, the image is used as a new training 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.

3.2 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 helicopter platform.

For example, considering the EOD vehicle in figure 2(a) and figure 3(a), the tracker is able to follow both objects for the entire time that they are in the field of view, usually 10-20 seconds. This is due to the fact that the objects had distinctive appearances, which allowed the computed features to be very discriminative. In addition, the relatively large object sizes made the motion of the helicopter less significant. In contrast, tracking the mine in figure 3(b) and the walking person in figure 3(c), is more challenging. Tracking the mine was particularly difficult due to the extremely small size, and non-distinct circular shape. Similarly, the person is very small in the image, although relatively distinct; as a result, the motion of the helicopter makes the tracker lose track almost immediately without the ego-motion estimation.

As a result, we use a motion model coupled with Bayesian filtering to update the object position estimate. This allows us to more robustly estimate the object position in the image by making use of an ego-motion estimate to bias the motion update. This ego-motion estimate proved to be very important as it was able to compensate for the unpredictable motion of the camera, which would have otherwise caused the tracker



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

Fig. 3. 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.

to get lost. The motion estimate is computed using the Pyramidal-Lucas-Kanade optical flow implementation available in OpenCV [2]. Optical flow computes a set of displacements for features in the image, which we then cluster using expectation-maximization to identify the single largest flow direction, and then compute the affine transform that best explains the apparent motion.

We can then use the affine transformation 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

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

$p(x_{t-1}|z_{0:t-1})$ 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 a model of how the object moves, which we assume to be Gaussian motion with some fixed variance. In contrast to more conventional filtering techniques such as the Kalman filter [5], the particle filter is useful for modeling the non-linear sensor and motion models and the non-Gaussian noise distributions. The motion of the MAV is particularly non-linear, and large swings of the MAV generally cause very large displacements of the object in the image.

Returning to figure 3(b) and 3(c), when tracking the person, we were able to maintain the track for over 2 minutes requiring human intervention only once when the person went out of the frame for a couple seconds. This was made possible by the motion model provided by optical flow, helped by relatively stable hovering of the helicopter. Similarly, when tracking the mine in figure 3(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 movement of the helicopter.

4 Ground Vehicle Planning

Given the ability of the MAV to estimate the guard position and trajectory, the second challenge was to be able to plan a 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 cover points, 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.

Our motion planner therefore begins with a discretization of the planning area. We use a regular grid, and assume 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 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 examine 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.

TIME-STATE A*

The TIME-STATE-A* algorithm, developed by Fraichard [4], represents the state of the GV as both a position and time. In order to account for the guard vehicle, we extrapolate the 2-D space 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. In addition to the four motion commands, we add a PAUSE action that only changes the time variable by the same constant amount as motion commands. Longer pauses can be achieved by executing PAUSE twice. 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.

Notice that the input to A* are now states with an explicit time variable, and that this algorithm includes as the input 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.

Algorithm 1 : WINDOWED TIME-STATE A* (WTS-A*)

Require: $\mathbf{x}_{start}, \mathbf{x}_{goal}, \mathbf{x}_{guard}, t_{max}, t_{window}$
1: $\pi_{approx} \leftarrow$ A*$(\mathbf{x}_{start}, \mathbf{x}_{goal})$
2: $\{\hat{\pi}^i\} \leftarrow$ DIVIDE$(\pi_{approx}, t_{window})$
3: $t \leftarrow 0$
4: **for** $\hat{\pi}^i \in \{\hat{\pi}^i\}$ **do**
5: $\quad \mathbf{x} \leftarrow \hat{\pi}^i[1]$
6: $\quad \mathbf{x}' \leftarrow \hat{\pi}^i[end]$
7: $\quad \pi_{tail} \leftarrow$ A* $((\mathbf{x}, \mathbf{x}_{goal}, t), (\mathbf{x}', \cdot, t_{max}))$
8: $\quad$ **if** $\pi_{tail} ==$ **null then**
9: $\quad\quad$ **return null**
10: $\quad$ **end if**
11: $\quad \pi \leftarrow \pi + \pi_{tail}$
12: $\quad t \leftarrow t + length(\pi_{tail})$
13: **end for**
14: **return** π

The complete algorithm is shown in Algorithm 1. 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-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.

Finally, to determine if the additional complexity of planning in time and space can be avoided, we also examined the performance of planning only in the state space of the GV, using a conventional search process through only the state space, STATE-A*.

Figure 4 depicts the runtime and the quality 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 4-a). It is interesting that on average, WTS-A* outperformed STATE-A* in terms of runtime. On the other hand, the quality of the paths found by WTS-A* were on par with those found by TIME-STATE-A*, shown in figure 4(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.

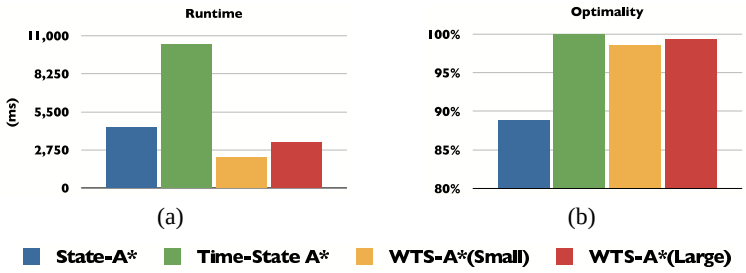


Fig. 4. Averaged results of State-A*, Time-State A*, WST-A*(Small), and WTS-A*(Large) across planning problems of different sizes.

The most interesting result occurred as we varied the number of dynamic obstacles. Figure 5 shows that as the number of dynamic obstacles increases, the extra cost of re-planning for STATE-A* dominates the cost of planning in the Time-State space (figure 5-a), indicating that as the number of obstacles increases, re-planning needed to occur more frequently. While TIME-STATE-A* has to search in a larger space, most plans found by STATE-A* are 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 5-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*).

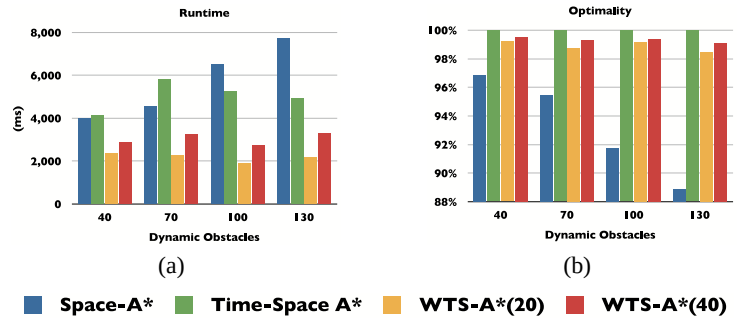
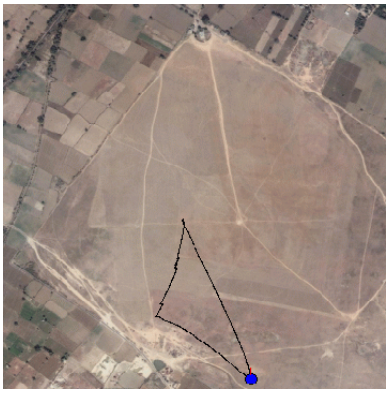


Fig. 5. Runtime and optimality results of 30 runs for State-A*, Time-State A*, WTS-A*(15), and WTS-A*(30) averaged across different numbers of dynamic obstacles.

(a) Phase 1

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

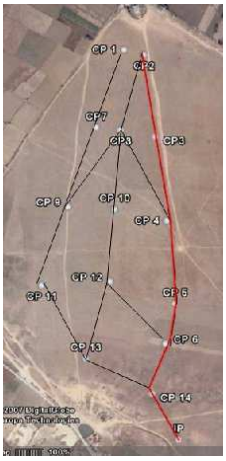


(b) Phase 2

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

(c) Phase 3

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



(d) Expected GV Path

Fig. 6. (a-c) The paths executed by the MAV. (d) The expected plan executed by the commandos and EOD vehicle.

5 Mission Performance in MAV '08

As described in section 2, 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. Figure 6(a-c) shows the actual paths flown by the MAV on each mission. Figure 6(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 replanning.

6 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 considerably 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. Intel Corporation. Open source computer vision library (opencv). http://www.intel.com/technology/computing/opencv/index.htm.
3. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
4. T. Fraichard. Trajectory planning in a dynamic workspace: a 'state-time' approach, 1999.
5. 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.