Persistent Vision-Based Search and Track Using Multiple UAVs

by

Brett Bethke

S.B., Aeronautical/Astronautical Engineering
S.B., Physics
Massachusetts Institute of Technology (2005)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2007

Author .... [illegible]
Department of Aeronautics and Astronautics
May 25, 2007

Certified by ....
Jonathan How
Professor
Thesis Supervisor

Accepted by .... [illegible]
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Persistent Vision-Based Search and Track Using Multiple UAVs

by

Brett Bethke

Submitted to the Department of Aeronautics and Astronautics
on May 25, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Unmanned aerial vehicles (UAVs) have attracted interest for their ability to carry out missions such as border patrol, urban traffic monitoring, persistent surveillance, and search and rescue operations. Most of these missions require the ability to detect and track objects of interest on or near the ground. In addition, most of the missions are inherently long-duration, requiring multiple UAVs to cooperate over time periods longer than the endurance of a single vehicle. This thesis presents a framework to enable such missions to be carried out autonomously and robustly. First, a technique for vision-based target detection and bearing determination that utilizes a video camera onboard each UAV is presented. The technique is designed to detect the presence of targets of interest in the camera video stream and determine the bearing from the UAV to the target even when the video data is noisy. Next, a cooperative, bearings-only target estimation algorithm is presented. The algorithm is shown to provide better estimates of a target's position and velocity in three dimensions than could be achieved by a single vehicle, while being computationally efficient and naturally distributable among multiple UAVs. Next, a task assignment algorithm that incorporates closed-loop feedback on the performance of individual UAVs and sensor suites is developed, enabling underperforming UAVs to be dynamically swapped out by the tasking system. Finally, flight results from several persistent, multiple-target search and track experiments conducted on MIT's Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) are presented.

Thesis Supervisor: Jonathan How
Title: Professor

Acknowledgments

I would like to thank a number of individuals for their invaluable support in completing this thesis. My advisor, Jonathan How, was extremely helpful in providing guidance and encouragement during this process. In addition, I would like to thank Kathryn Fischer for her tireless assistance and good spirits. I am grateful to Mario Valenti for his insightful advice throughout our time working together, and for his genuine concern for those around him. His friendship and wisdom helped me through many difficult times. My colleagues, Spencer Ahrens, Dan Dale, Adrian Frank, Ray He, and Jim McGrew, were and are a pleasure to work with. Their sense of humor, brilliance, and hard work made working in the lab a fun and creative experience.

I thank my family for always believing in me, and for the hard work and sacrifices they have selflessly made to give me so many opportunities to follow my dreams. Their love and encouragement is a continual source of strength. Finally, I thank Anna Massie for her unwavering support and love. My life is infinitely richer because of her.

I would like to gratefully acknowledge the Hertz Foundation and the American Society for Engineering Education for their generous support of this research. This research has been supported in part by Boeing North America, Incorporated under contract title "Vehicle And Vehicle System Health Management" and by AFOSR grant FA9550-04-1-0458.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) have attracted significant interest in recent years. Due to improvements in embedded computing, communications, sensing, and other enabling technologies, UAVs have become increasingly capable of carrying out sophisticated tasks. Because UAVs lack a human occupant and are generally simpler and less expensive than their manned counterparts, they are especially well suited to a wide range of "dull, dirty and/or dangerous" missions. Examples of such missions include traffic monitoring in urban areas, search and rescue operations, military intervention, remote weather monitoring, natural disaster relief, and border patrol.

To date, many different types of UAVs, for a number of different purposes, have been designed and used [14]. Military applications have spurred the development of a large number of UAVs for both research and in-the-field operations. The Northrop Grumman RQ-4 Global Hawk (Figure 1-1) is a long-duration UAV designed for surveillance. Equipped with Synthetic Aperture Radar, optical and infrared cameras, and other sensors, the Global Hawk can loiter over a target area for over 24 hours and provide a rich set of data to remote observers [37]. The General Atomics MQ-1 Predator (Figure 1-2) is similarly intended to fill a long-duration surveillance role, and has also been modified to carry weapons [17]. On the civilian side, NASA has experimented with a high-altitude, solar-powered UAV called Helios (Figure 1-3). The project was designed to explore technologies that might allow UAVs to cruise for weeks or months at altitudes near 100,000 feet using no consumable fuels [36].

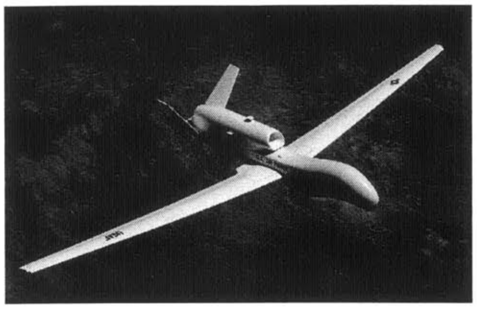
Figure 1-1: Northrop Grumman RQ-4 Global Hawk

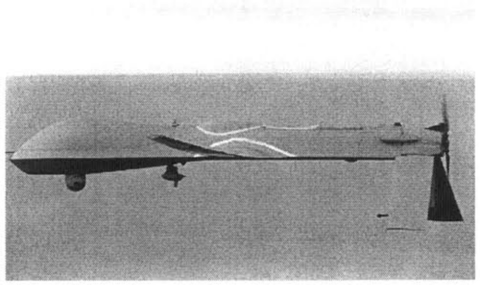
Figure 1-2: General Atomics MQ-1 Predator

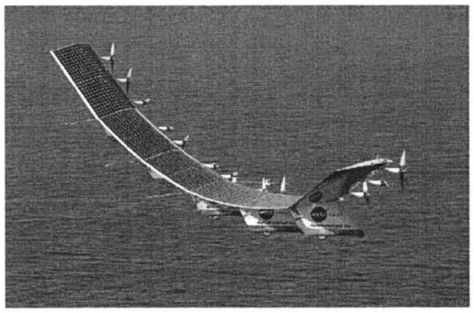
Figure 1-3: NASA Helios prototype

General Atomics has also built a civil variant of the Predator UAV, which was flown by NASA to investigate the use of UAVs in high-altitude science missions [35].

1.1 Motivation

The operation of UAV systems has traditionally required a large amount of continuous human support. Some UAVs require a human pilot to fly the vehicle by hand from a remote ground station, or at least monitor the systems of the UAV and intervene in the event of a system failure or off-nominal condition. In addition, the sensor data from a UAV is often monitored continuously by one or more people on the ground. These people interpret the data and make a plan for what the UAV should do next based on their knowledge of the mission. Thus, flying UAVs becomes a very personnel-intensive activity. Especially in missions that require multiple UAVs to perform many complex tasks, the amount of human activity and coordination required may be a limiting factor in the overall performance of the mission.

In order for a UAV system to become truly useful for such complex missions, the level of autonomy in the system must be increased. There are many aspects in the design of a UAV system where having a high level of autonomy is beneficial. Two very important aspects are:

- The ability to automatically extract useful information from the sensor data gathered by the UAVs and autonomously generate a sequence of actions based on that information that will help to accomplish the mission.

- The ability to autonomously monitor the subsystems of each UAV and decide what to do in the event of a failure or degraded performance, especially with regard to deploying or reassigning another vehicle for a vehicle which is no longer capable of carrying out its assigned duties.

A UAV system that had these abilities would be extremely useful and robust. In order to build such a system, it is necessary to understand the associated challenges. Some of these challenges are described below.

Many of the mission scenarios for UAVs require the ability to remotely detect and track objects of interest on or near the ground. For example, this ability is clearly necessary in search and rescue operations or when assisting law enforcement during a high-speed chase. In these missions, the deployment of video cameras onboard the UAVs is of particular interest due to the richness of information and real-time situational assessment capabilities that can be provided by the video stream. However, using this data in an autonomous system poses several challenges. First, vision processing algorithms must be developed to recognize the object(s) of interest in the video stream. Second, if the target is found, its state (usually position and velocity) must be estimated so that the UAV can maneuver to track the target. Especially when the target is itself maneuvering rapidly or the environment contains obstacles or terrain that can block the line of sight, it may be difficult for a single UAV to accurately estimate the state of the target. In such cases it may be beneficial to use several cooperating UAVs to track the target.

Additionally, many of the mission scenarios for UAVs are inherently long in duration, meaning that the goals of the mission cannot be fully achieved within the useful flight time of a single UAV. For example, in a forest fire monitoring scenario, it might be required to have aerial coverage of the affected areas on a continual basis for days or weeks at a time. Successful completion of the mission therefore requires multiple UAVs to be used in a cooperative fashion, so that new vehicles are ready to come on-scene when the vehicles currently performing the mission need to return to base for refueling or maintenance. Developing a system architecture that can autonomously handle such activities is a difficult problem, especially when vehicle failures and degradations are considered. These failures are more likely to occur as the mission length increases, making it very important that the system is able to handle such occurrences robustly by anticipating their effects and reacting when they occur.

This thesis addresses the development of an autonomous UAV system that uses video data from onboard cameras to perform search and track missions over long periods of time. The system incorporates vision processing algorithms to detect the presence of objects of interest in the video streams. These algorithms are designed to be robust to noise in the video data. A cooperative and distributed estimation algorithm is used to synthesize the processed data from each camera into an estimate of the position and velocity of the target object, and this estimate is used to predict the future location of the target and generate a set of tasks that will allow the system to actively track the target. A task assignment algorithm is used to assign these tasks to individual UAVs. The task assignment algorithm uses closed-loop performance feedback to adaptively respond to under-performing or low-fuel vehicles, calling in replacement vehicles if necessary. This architecture allows the system to perform the search and track mission in a robust way over long periods of time.

1.2 Literature Review

This thesis builds upon a range of work that has been done previously in vision based techniques, task assignment, and health management of UAV systems.

The use of vision for navigational purposes has been widely studied. A visual odometer was used in [38] to estimate the translational motion of a helicopter over flat ground. Another type of visual odometer, suited for use in building inspection, was presented in [13]. Several groups have studied vision-based landing techniques [6, 16, 15]. A system for autonomously following a road using vision was presented in [11]. Additionally, detection and avoidance of obstacles using forward looking cameras on a UAV was demonstrated in [47].

Vision based techniques have also been used for environmental urban monitoring and data collection. The problem of remotely detecting forest fires using a group of UAVs was studied in [25, 24, 7]. The feasibility of using a UAV to collect information about traffic flows and parking lot utilization was studied in [2].

Use of vision for estimation and tracking of ground based targets has also been studied. [30] presents a method to circle a target on the ground while keeping the target in view using a gimballed camera. Another gimballed camera system was used in [29] to track a target and estimate its position. More recently, a method to use the measurements from several UAVs to cooperatively estimate and track a moving target was presented in [1].

The task assignment problem has received considerable attention in the context of UAV systems. A method for assigning search tasks to a group of UAVs using a network flow approach was presented in [5]. Task assignment in the persistent aerial denial mission was studied in [53]. A receding-horizon approach to the task assignment problem was presented in [27] and subsequently developed into a decentralized version in [28]. Another approach to decentralized task assignment that uses Voronoi partitions to determine assignments was shown in [10].

The problem of autonomous UAV health management also has been studied. A system for coping with sensor and control actuator failures was presented in [22].

[33] presented techniques for enabling 24/7 persistent surveillance operations, while [32] presented results of a full surveillance mission with fuel management and sensor failure recovery.

1.3 Objectives

The objectives of this thesis are to design and demonstrate an autonomous UAV system capable of executing long-duration, vision-based search and track missions. The success criteria in Figure 1-4 will be used to evaluate the performance of the system.

System success criteria: The system should be able to:

1. Search for and detect a target of interest, which may be moving in three dimensions, using one or more camera-equipped UAVs. This detection should be robust to noise in the video data, which may be present due to radio interference in the wireless transmission of the data or other factors.
2. Estimate the quality of the video data from each UAV, so that the system can determine if a given UAV's camera is not performing well.
3. Combine the sensor information from one or more UAVs to create an estimate of the position and velocity of the target, and use this information to predict the future position of the target.
4. Use a task assignment algorithm to assign a set of UAVs to locations that will keep the target in view, given the predicted position of the target. In addition to simply ensuring that there are UAVs assigned to the proper locations, the task assignment algorithm should account for the health state of the vehicles, swapping out underperforming vehicles and those that must return to base for refueling.
5. Continue the search and track mission for an extended period of time. Here *extended* means a time span longer than the flight time of individual UAVs, thus necessitating the swapping out of vehicles in order to accomplish the mission.

Figure 1-4: System success criteria

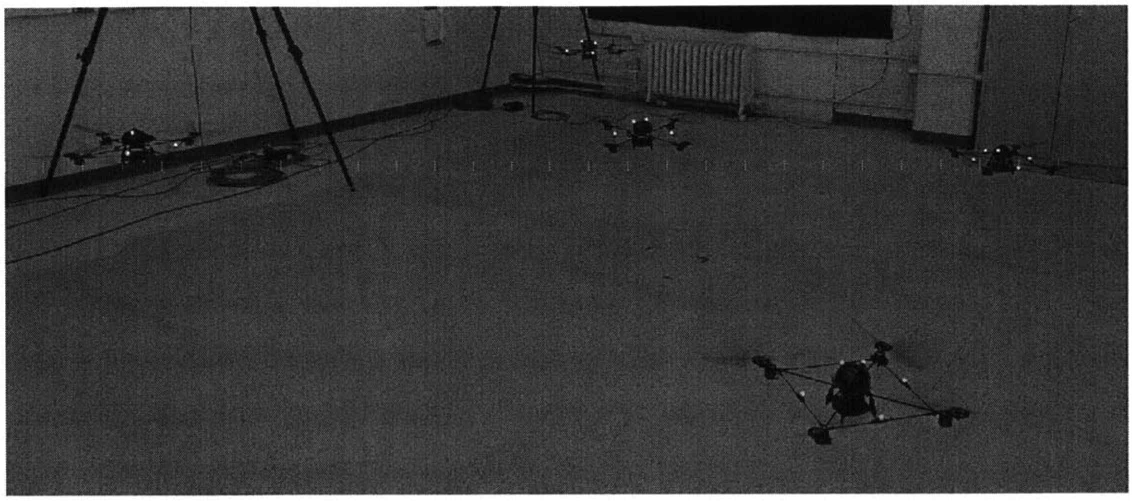
Figure 1-5: Five vehicle coordinated flight test on MIT's RAVEN

1.4 Overview of MIT RAVEN Research Platform

The MIT Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) was used as the hardware testbed for this research. A brief overview of RAVEN is presented here; for more details see [31] and [34].

The core of the testbed consists of a Vicon motion capture system [50] that is used to provide highly accurate measurements of the positions of numerous ground and aerial vehicles within a flight space measuring approximately 6x8x5 meters. Two Vicon cameras are shown in Figure 1-8. This positioning information is distributed in real-time to a number of processing computers that run the controllers for each vehicle in the system. The control commands are then sent to the vehicles over a R/C wireless link, closing the control loop and stabilizing the vehicles.

Each vehicle control computer in the system can receive commands, such as waypoint, takeoff, and landing commands, over the network, allowing multiple vehicles to coordinate their actions. These commands may be generated autonomously by higher-level software, such as a task assignment algorithm. They may also be generated by a human operator through a 3D interface (Figure 1-6) that allows the operator to easily visualize data from the system and issue commands.

The system allows many different types of vehicles to be flown and tested in a

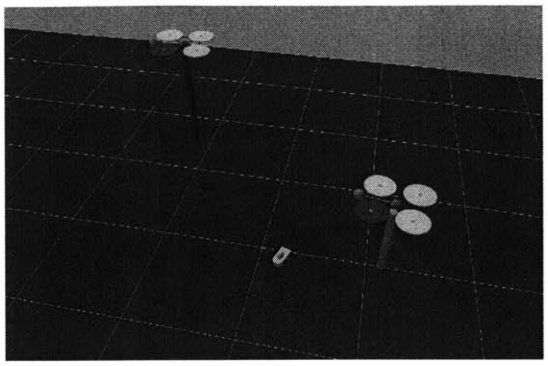
Figure 1-6: RAVEN operator interface

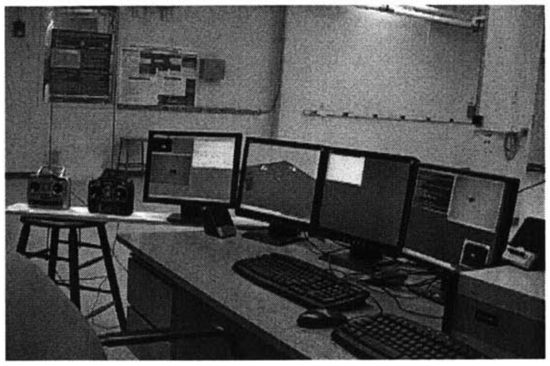
Figure 1-7: Operator station

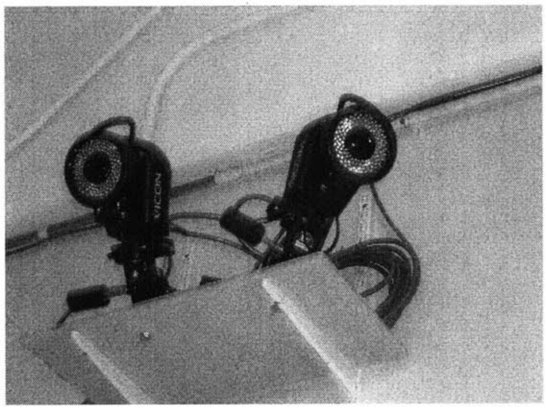
Figure 1-8: Two Vicon cameras

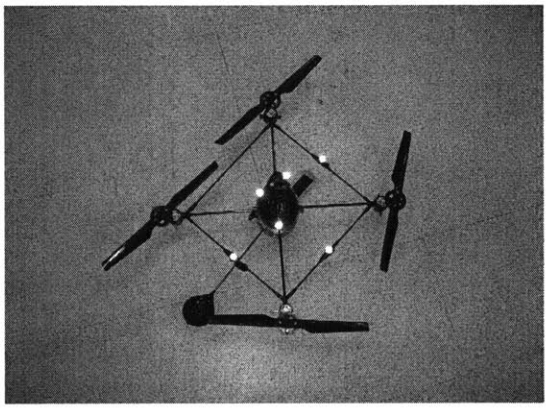
Figure 1-9: Draganflyer V Ti Pro RC helicopter

controlled environment. To date, a number of different vehicle types have been used in the system. This research used the Draganfly Innovations [8] Draganfly V Ti Pro quad-rotor helicopter as the primary vehicle (Figure 1-9). Figure 1-5 shows five such vehicles hovering autonomously in the RAVEN. The vehicles for this research were outfitted with the Draganfly SAVS camera system, which enables the video to be received on the ground by a wireless connection. The system also supports a number of ground vehicles, such as those shown in Figure 1-10. These ground vehicles were used as target vehicles in a number of experiments.

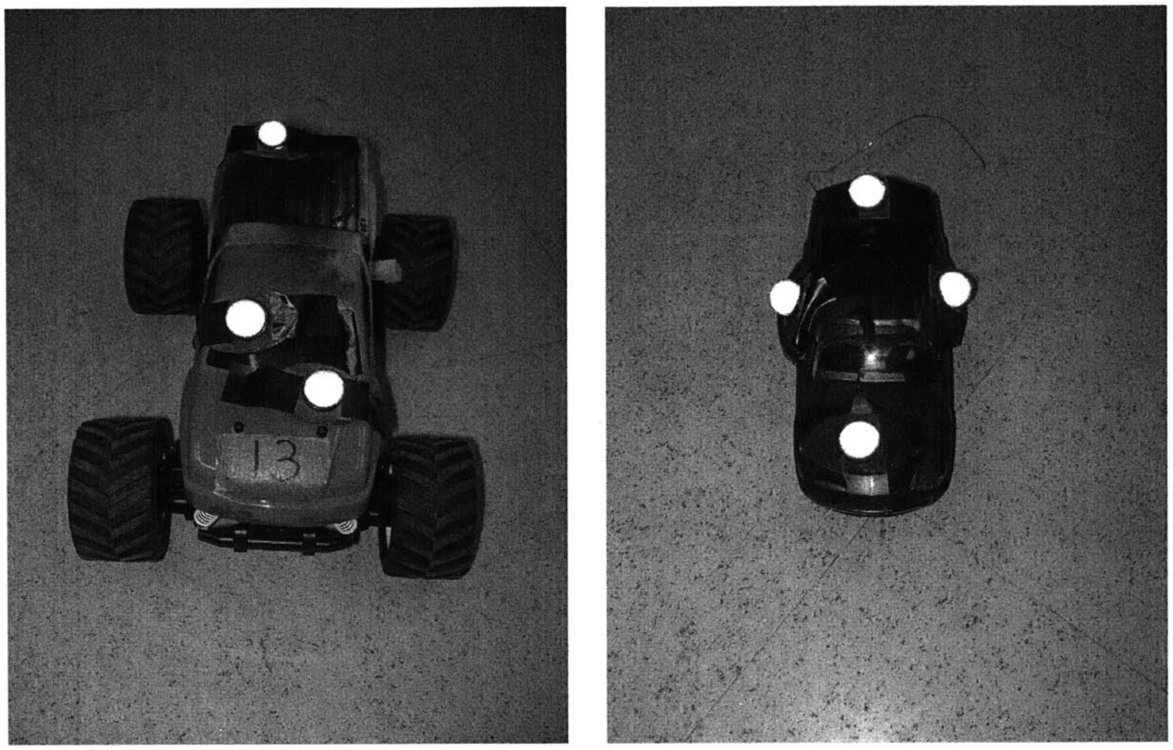

Figure 1-10: RC ground vehicles

1.5 Approach

This thesis is structured as follows. Chapter 2 presents a technique for vision-based target detection and bearing determination that utilizes a video camera onboard each UAV. Chapter 3 develops a cooperative and distributed estimation algorithm that uses the bearing information generated by the vision system to estimate the position and velocity of the target. Chapter 4 presents a task assignment algorithm that incorporates closed-loop feedback on the performance of individual UAVs and sensor suites, enabling under-performing UAVs to be dynamically swapped out by the tasking system. Chapter 5 presents a number of increasingly complex flight results, culminating with presentation of data from a fully integrated search and track mission. Finally, Chapter 6 concludes the thesis with a summary and and shows that the performance of the autonomous system meets the criteria specified in Figure 1-4.

Chapter 2

Robust Vision-Based Target Detection and Bearing Determination

Clearly, an important aspect of the overall cooperative search and track mission is the sensing equipment and processing algorithms that the UAVs use in order to detect objects of interest. Vision-based methods (i.e., methods that use the data from one or more video cameras mounted onboard each UAV) are attractive for this purpose for several reasons. First, the necessary hardware that must be carried onboard the UAV, which usually consists of the video camera itself and either a video processing unit onboard or a wireless transmitter to relay the video signal to a ground station, is readily available in a number of different sizes, weights, and configurations suitable for use even on small, payload-limited UAVs. Researchers have used a number of such small camera systems for the purpose of obtaining onboard video from small UAVs [15, 16, 1, 30]. Another advantage is that once an object of interest is located in the camera image, the direction to the object in physical space (also referred to as the *bearing*) can be determined using a pinhole model of the camera.

For this research, the Draganfly Innovations SAVS wireless camera system (Figures 2-1 and 2-2) was selected due to its light weight, low power consumption, and ease of integration with existing hardware. The system consists of a boom-mounted

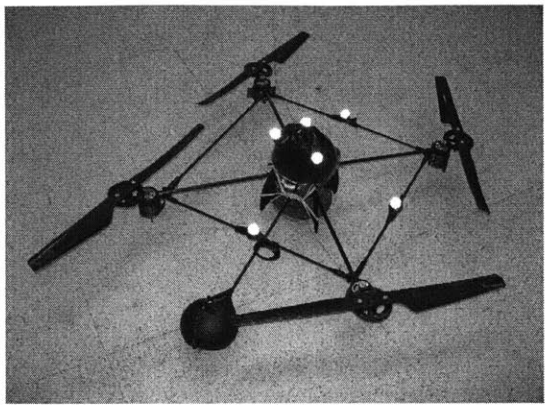
Figure 2-1: Draganflyer SAVS RC helicopter [8]

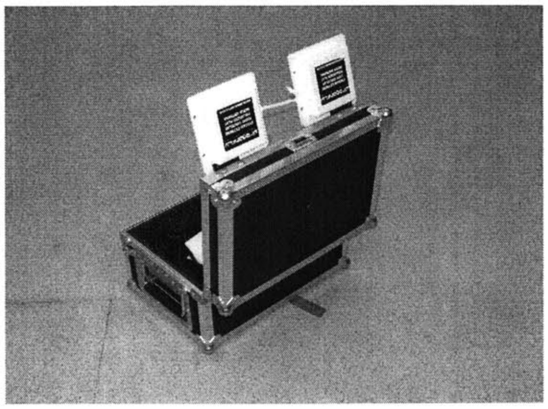
Figure 2-2: Draganfly SAVS receiver

camera, onboard transmitter and power circuitry, and a ground based receiver. Using the video stream from the camera, it was desired to build a vision-based system to detect objects of interest, such as the small RC vehicles shown in Figure 2-3, and estimate the bearing from the camera to the object.

In order to accomplish these goals, there are several tasks that need to be accomplished. First, a computer hardware and software configuration to transfer the video stream from the onboard camera into a computer for image processing needs to be established. Since the objective of the vision system is to provide timely target bearing information for tracking purposes, real-time considerations are important in the selection of the computer setup. Second, an image processing algorithm for detecting the position, in image coordinates, of objects of interest in individual camera frames must be designed. Finally, accurate calibration models of the camera must be developed so that image coordinates can be translated into target bearing information in physical space.

A very important design consideration in the image processing system is that noise and other undesirable image characteristics are very often present in the video stream. This is especially true with many of the camera systems that can be used on small, lightweight UAVs, since these cameras must be designed to be light, consume little power, and often must broadcast their video signals back to a ground station over a noisy wireless link. Examples of real images from the cameras demonstrating the type

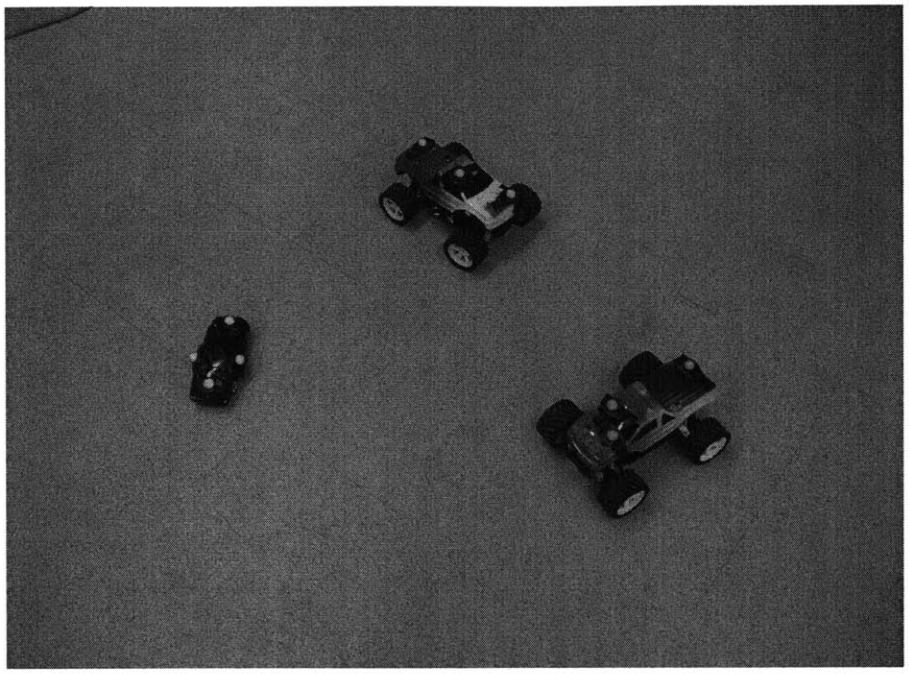

Figure 2-3: RC vehicles used as targets for the vision tracking system

of nose that appears are shown in Figure 2-4. In order for the overall vision tracking system to function reliably, it must be designed to account for these characteristics. Furthermore, it is desired to estimate the quality of the video signal at any given point in time, since this information is useful to the tasking system for determining whether a given UAV has a sufficiently well-functioning camera to be able to carry out its mission. The following sections will explain how the overall system is designed and how noise is accounted for in various parts of the processing.

2.1 Hardware and Software Configuration

The hardware setup necessary to capture onboard video from the UAV consists of an onboard camera, ground based receiver, video capture card, and PC. The camera onboard the UAV transmits a wireless video stream to a diversity receiver on the ground. The diversity receiver outputs an analog, composite video signal, which is connected to a LifeView FlyVideo 3000FM video capture card (Figure 2-5) installed in a dual-processor, 64-bit AMD Opteron PC running Gentoo Linux [18] in native

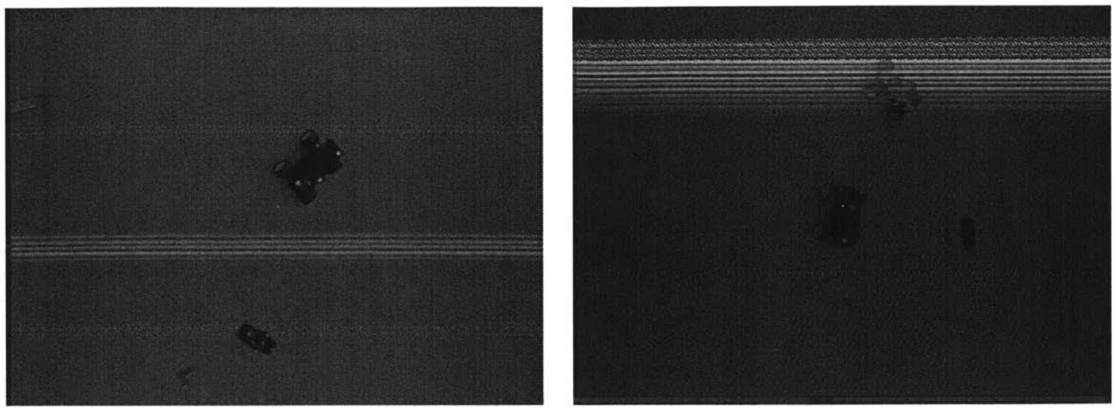

Figure 2-4: Examples of noisy images from the camera system

64-bit mode.

Gentoo was selected for the vision processing PC due to its flexibility, good support for 64-bit architectures, and ability to compile system packages with optimizations for the particular hardware configuration in use, helping to improve the real-time performance of the system. The FlyVideo card was found to be a good choice for use with linux. The card is supported under the `saa7134` driver in the linux kernel. The module should be loaded into the kernel using the `modprobe` command as follows:

```
# modprobe saa7134 card=2 tuner=39 gbuffers=2
```

The `card` and `tuner` options are necessary so that the card is properly detected and configured by the driver. In addition, it was determined that the `gbuffers` option plays an important role in the real-time performance of the card. According to the module documentation, `gbuffers` controls the number of internal capture buffers the driver maintains, with a range of 2-32. Setting this number too high resulted in high latencies, so it is desired to set `gbuffers` as small as possible.

Proper functioning of the module can be tested by running the command `dmesg` after loading the module. If the card and module are both working, `dmesg` should show messages similar to the following.

```
saa7133[0]: found at 0000:01:0a.0, rev: 16, irq: 169,
 latency: 64, mmio: 0xfc6ff000
saa7133[0]: subsystem: 5169:0138, board: LifeView Fly
```

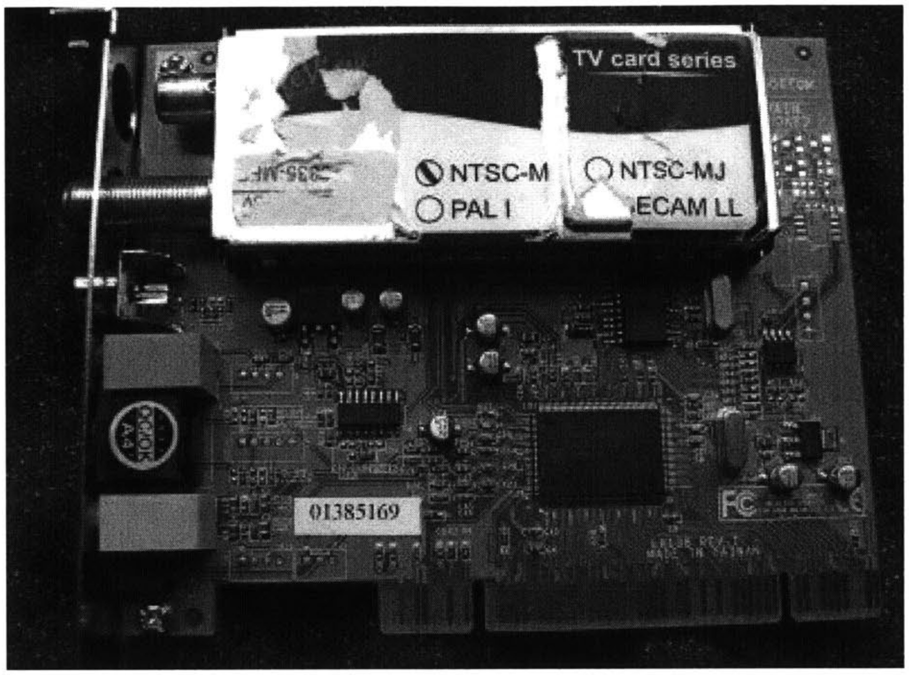

Figure 2-5: LifeView FlyVideo 3000FM video capture card [26]

```
 VIDEO3000 [card=2,insmod option]
saa7133[0]: board init: gpio is 39900
...
saa7133[0]: registered device video0 [v4l2]
saa7133[0]: registered device vbi0
saa7133[0]: registered device radio0
```

For testing purposes, the output from the video capture card can be easily viewed using a program such as `tvtime` [49].

The Intel OpenCV computer vision library [21] is used as the main software library to request images from the video driver and perform the image processing steps. OpenCV interfaces natively with the Video4Linux2 API supported by the `saa7134` driver and provides a large set of very useful utilities for image processing.

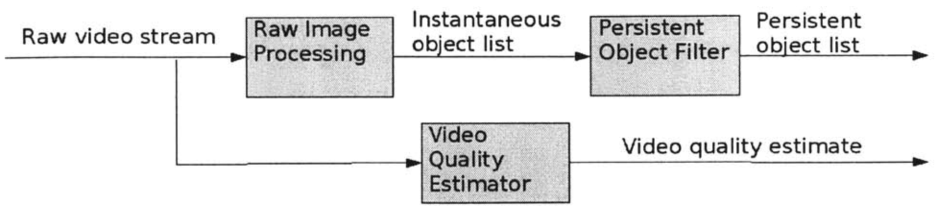

Figure 2-6: Image processing system schematic

2.2 Image Processing System

The function of the image processing system is to generate image coordinates (u, v), where u and v are measured in pixels, of all objects of interest in the camera video stream. For the purposes of this research, the objects of interest were various types of radio-controlled cars and trucks (examples shown in Figure 2-3). The system was also required to track flying objects, such as the Draganfly UAV shown in Figure 2-1. These objects were to be detected against a relatively uniform background (in this case, a white tile floor).

The requirements of the image processing system were to reliably provide the locations of the objects under a range of different lighting conditions and independently of the particular camera being used, while being robust to noise in the video stream. Also, the system had to be capable of running in real-time, so computation time of the overall image processing algorithm was important. Finally, the system should estimate the quality of the video signal.

Given these requirements, the image processing system was broken into three parts, as shown in Figure 2-6. The Raw Image Processing component searches for objects in individual, instantaneous frames. These objects are then passed to the Persistent Object Filter, which tracks the presence of the same object across multiple frames, filtering out objects which appear for only a very short time. This is necessary since noise may introduce false targets that appear for only one or two frames. Finally, a Video Quality Estimator determines how much noise is present in the video signal.

2.2.1 Raw Image Processing

An algorithm for detecting the presence of target objects in the raw camera images was designed. The algorithm is shown in pseudo-code in Figure 2-7 and discussed in detail below.

```
function processRawFrame():
    image1 = getRawFrameFromCamera();
    image2 = convertToGrayscale(image1);
    image3 = downsample(image2);
    image4 = histogramEqualize(image3);
    image5 = convertToBinaryImage(image4, thresholdValue);
    array instantaneousObjects[] = findBlobs(image5);
    return instantaneousObjects;
end function;
```

Figure 2-7: Raw Image Processing Algorithm

First, a raw image from the camera (Figure 2-8) is captured, converted to grayscale, and downsampled (Figure 2-9). In the current implementation, the raw image size is 640x480 pixels, and the downsampled image is 320x240 pixels. Downsampling allows the overall algorithm to run much faster, since the downsampled image contains only one quarter the number of pixels as the original. Furthermore, since the image processing algorithms used only look for large-scale features of the images, the full resolution is not needed to reliably detect the targets of interest.

Next, histogram equalization is performed on the image (Figure 2-10). This step serves to adjust the contrast in the image, which is important since experimental use of the cameras revealed that different cameras often have dramatically different contrast levels. In addition, the contrast level of an individual camera can change as the battery powering it depletes, causing the overall image to darken. Finally, different lighting conditions also affect the contrast level. By performing histogram equalization, the intensity level of objects of interest will remain much more constant over time and across different cameras and lighting conditions, allowing for more robust object detection.

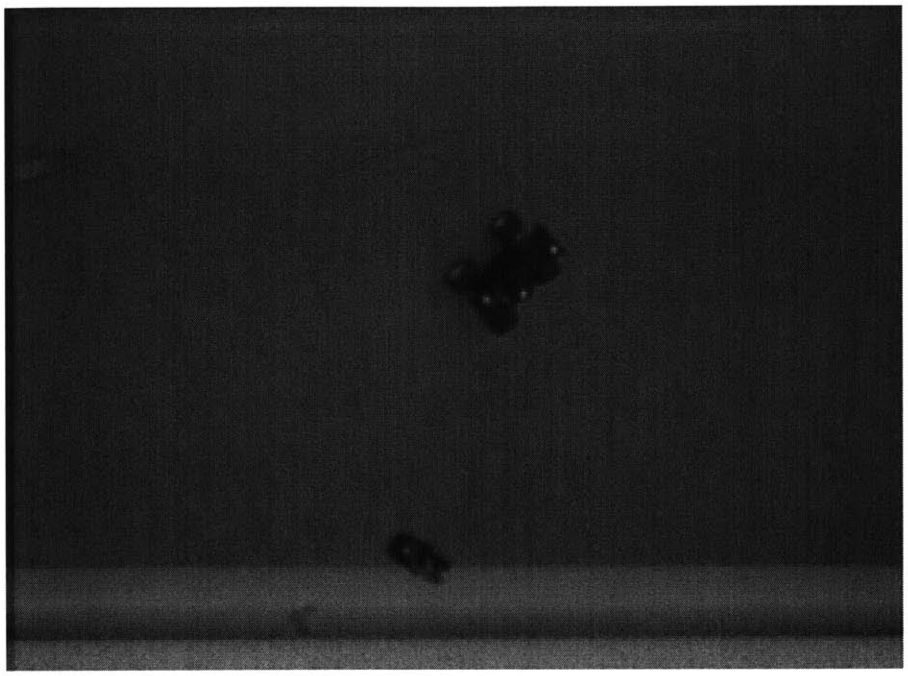

Figure 2-8: Raw image processing step 1: raw image

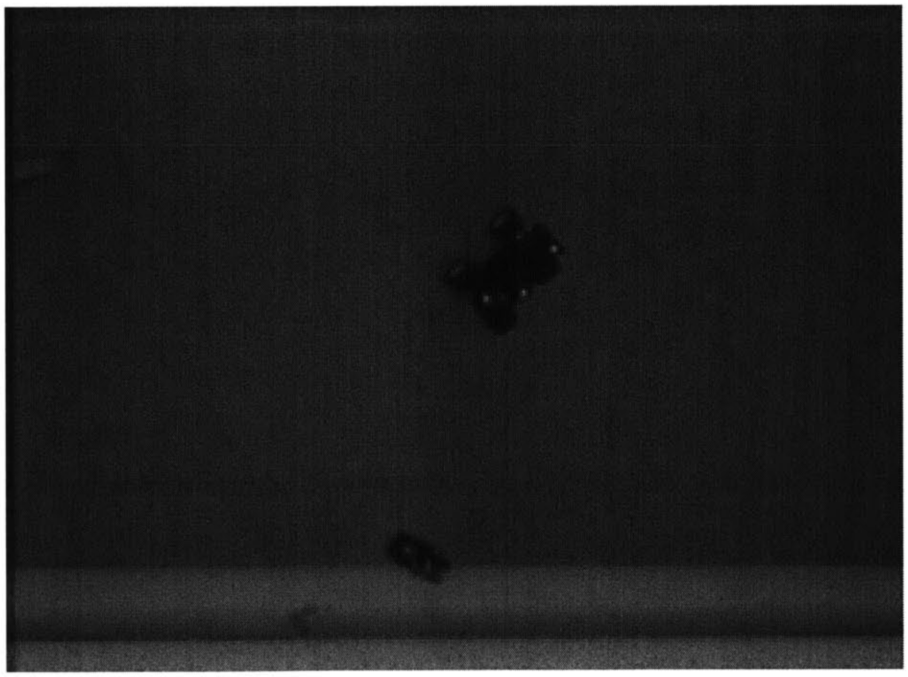

Figure 2-9: Raw image processing step 2: downsampling and conversion to grayscale

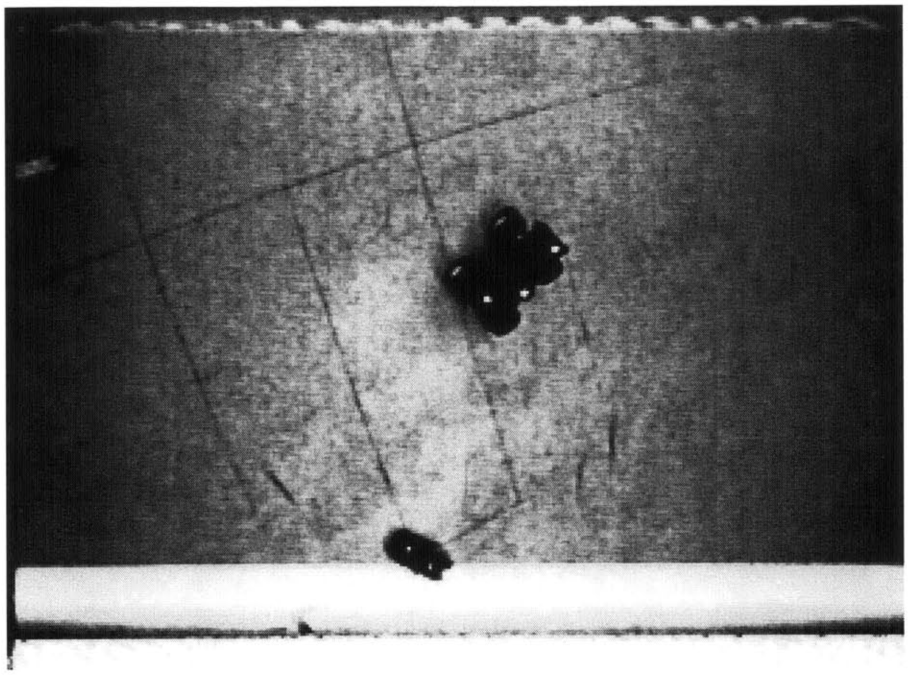
Figure 2-10: Raw image processing step 3: histogram equalization

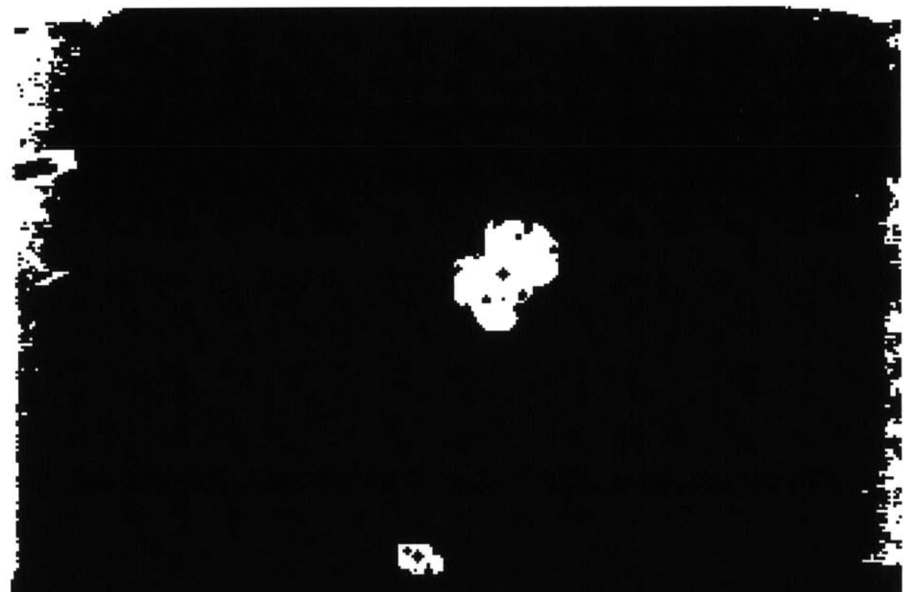

Figure 2-11: Raw image processing step 4: conversion to binary image

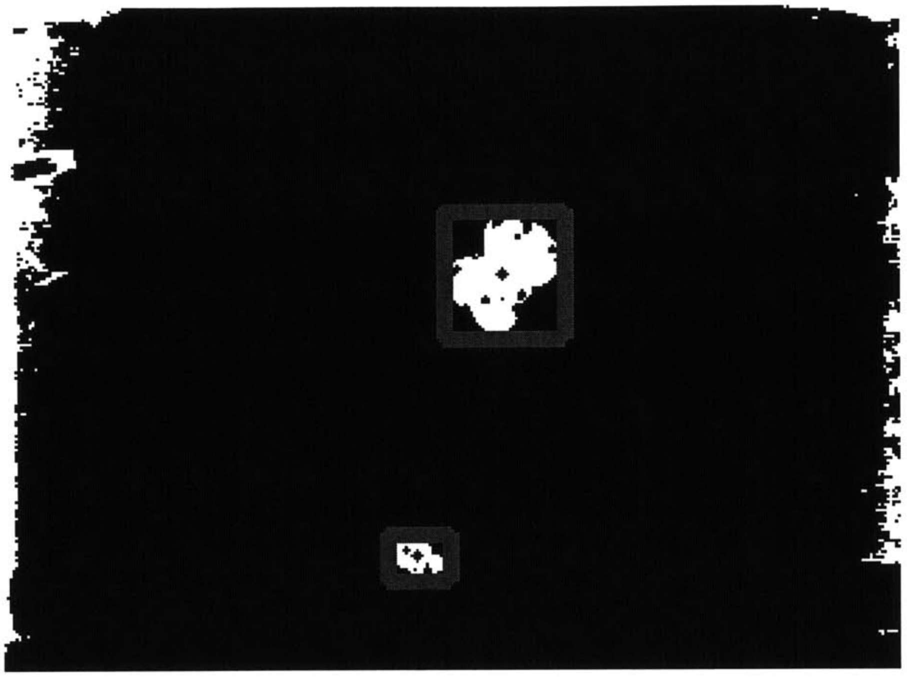

Figure 2-12: Raw image processing step 5: blob location determination

Next, the image is converted to a binary image (i.e. an image where each pixel can have only one of two values, 1 or 0) by applying a threshold (Figure 2-11). Here the assumption that the targets of interest are darker than their surroundings is used. Since the image contract was normalized in the previous step, it is possible to experimentally determine a threshold value that differentiates the target from the background, independent of the camera, battery state, and lighting conditions. It is clear that the target objects appear as continuous regions in the image.

Finally, the number, size, and location of all such continuous regions ("blobs") in the image are found using the OpenCV Blob Extraction Library [20]. At this point, the raw image processing is finished (Figure 2-12).

2.2.2 Persistent Object Filter

As mentioned above, the objects found by the Raw Image Processing algorithm (`instantaneousObjects` in Figure 2-7) may contain a number of false targets introduced by noise in the image. Since the noise is highly uncorrelated from one frame to the next, these false targets generally appear very briefly in the video stream, while real targets persist over many frames. Therefore, a filter was designed to remove transient objects while retaining ones that appear consistently. The filter is described in pseudo-code in Figure 2-13.

The algorithm maintains a dynamic list P of objects which have been seen in previous frames. At each filtering step, the objects in P are compared with those that have been detected in the current frame (set C). If an object seen previously is similar enough to one in C, its frame count number is incremented, up to a maximum number $maxFrameCount$. If it is not similar to any object in C, its count is decremented. At the end of the filtering step, objects whose frame counts have dropped to zero are removed, while those whose counts are above a threshold are returned as persistent targets.

Results of the filtering algorithm are shown graphically in Figures 2-14 and 2-15. The figures show a false object resulting from noise that is rejected by the filter (red box with no green circle), while several true objects are consistently tracked even when noise obscures them for several frames (red boxes with green circles). This is possible because the filter remembers that the object was there, even if it disappears temporarily.

2.2.3 Video Stream Quality Estimation

Detecting the quality of the video signal can be accomplished by designing a filter that amplifies the type of noise typically seen in the signal. As seen in Figure 2-4, the noise is characterized by rapidly alternating horizontal dark and light bands. Detection of these bands is possible by convolving the incoming image I with a kernel K designed to find horizontal edges, such as

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{2.1}$$

```
set $C = \emptyset$; (The set of objects in the current frame)
set $P = \emptyset$; (The set of persistent objects)
function persistentObjectFilter():
    for $p$ in $P$ do:
        $p.foundThisFrame$ = **false**; (Mark $p$ as "not found")
    end for;
    $C = processRawFrame()$; (Find objects in the current frame)
    for $c$ in $C$ do:
        $found$ = **false**;
        for $p$ in $P$ do:
            if ($\|c.x - p.x\| < \epsilon_x$) and ($|c.A - p.A| < \epsilon_A$) do:
            (Determine whether $c$ is similar to $p$ in terms of location in the
            image $x$ and area $A$)
                $p.foundThisFrame$ = **true**; (Mark $p$ as "found")
                $p.frameCount = min(p.frameCount + 1, maxFrameCount)$;
                (Add 1 to the running frame count of $p$, up to a maximum count)
                $found$ = **true**;
                break; (Since $p$ was determined, there is no need to continue
                examining the other elements of $P$)
        end for; (Ends for $p$ in $P$)
        if $found$ == **false** do:
            $c.frameCount = 1$; (Initialize the frame count for $c$)
            $P.push(c)$; (If $c$ is not found anywhere in $P$, append $c$ to $P$)
        end if;
    end for; (Ends for $c$ in $C$)
    for $p$ in $P$ do:
        if $p.foundThisFrame$ == **false** do: (If $p$ was not found in this frame...)
            $p.frameCount = p.frameCount - 1$;
            (...subtract 1 from the running frame count of $p$)
        end if;
        if $p.frameCount == 0$ do: (If the frame count of $p$ is zero...)
            $P.pop(p)$; (...remove $p$ from $P$)
        end if;
    end for;
    return $Q = \{q \in P : q.frameCount \geq frameCountThreshold\}$;
    (Return all objects that have appeared for at
    least $frameCountThreshold$ frames)
end function;
```

Figure 2-13: Persistent Object Filter Algorithm

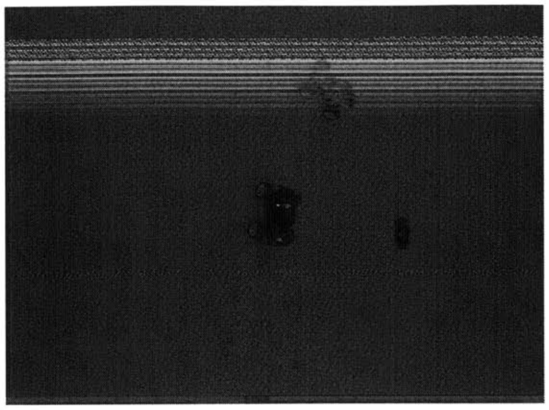

Figure 2-14: Raw image

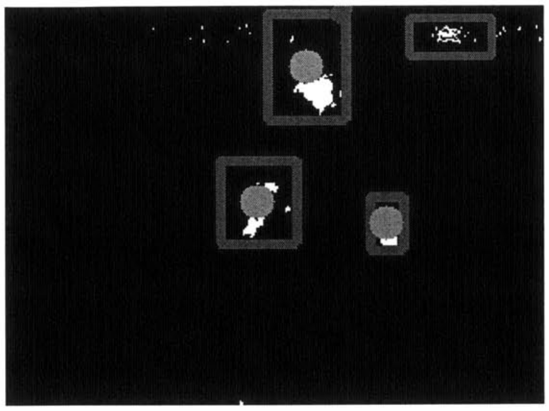

Figure 2-15: Processed image with detected and filtered objects (green circles)

The processed image P is then given by

$$P = I \otimes K \tag{2.2}$$

where $\otimes$ denotes the convolution operation. If a metric $d(P)$ over the space of processed images is defined as

$$d(P) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} |P_{ij}| \tag{2.3}$$

where n_x and n_y are the width and height of the images and P_{ij} is the value of the pixel located at (i, j) in the image, then $d(P)$ for a noisy image will be large, while $d(P)$ for a clear image will be small. Thus, $d(P)$ is a quantitative measure of the quality of a single image.

The video stream consists of a sequence of incoming images $\{I_1, I_2, I_3, ...\}$. For the image I_t captured at time t, the processed image P_t and the metric $d(P_t)$ is computed. A running average over the previous k frames can be calculated as

$$D(t) = \frac{1}{k} \sum_{\tau=t-k}^{t} d(P_\tau) \tag{2.4}$$

$D(t)$ is then a good overall metric of the health of the video system: when $D(t)$ is large, it indicates that many of the recently received images have been noisy; when it is small, it indicates that the images have on average been clear. Note that since

$D(t)$ is computed at each frame, it represents a continually updated estimate of the video system health. $D(t)$ can be passed to the tasking system, which can then make informed decisions based on knowledge of the current quality of each vehicle's video stream.

2.3 Bearing Determination

The persistent object filter in Figure 2-13 returns a list Q of persistent objects in the video stream. Each element of Q contains the location of the object in image coordinates. In order to be of use for estimation and tracking of the object, each set of image coordinates must now be converted to a unit-length bearing vector $\hat{\mathbf{d}}$ from the camera to the object in physical space (see Figure 2-16). The bearing vector is a result of the composition of three successive rotations, which are represented as quaternions $\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}$:

- $\mathbf{q_1}$ is the rotation from the fixed global frame to the UAV body frame. This is determined from the UAV's inertial sensors.
- $\mathbf{q_2}$ is the rotation from the UAV body frame to the camera frame, which is necessary to account for since the camera may not be aligned with the body frame of the vehicle. For example, the camera may be canted downward at an angle to provide a better field of view of objects on the ground. This rotation must be carefully measured prior to flight.
- $\mathbf{q_3}$ is the rotation due to the target's position in the image. This is determined from knowledge of the camera model parameters and the image coordinates (u, v) of the target.

The quaternions are represented using the standard quaternion notation

$$\mathbf{q} = (q_w, q_x, q_y, q_z)^T = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} \tag{2.5}$$

Once $\mathbf{q_1}$, $\mathbf{q_2}$, and $\mathbf{q_3}$ are known, the bearing vector $\hat{\mathbf{d}}$ can be found. This is ac-

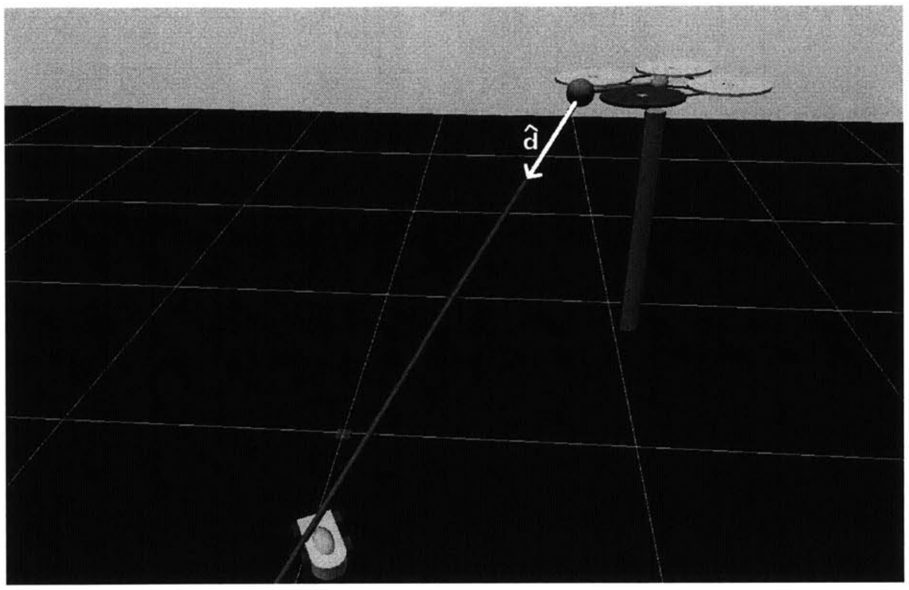

Figure 2-16: Bearing vector $\hat{\mathbf{d}}$

complished by first calculating the total rotation quaternion $\mathbf{q_t}$ by composing the individual rotations:

$$\mathbf{q_t} = \mathbf{q_3} \star \mathbf{q_2} \star \mathbf{q_1}$$

Here, $\star$ represents quaternion multiplication. Next, the 3x3 rotation matrix equivalent to $\mathbf{q_t}$ is calculated using the conversion formula

$$\mathcal{R}(\mathbf{q_t}) = \begin{pmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_xq_y - 2q_zq_w & 2q_xq_z + 2q_yq_w \\ 2q_xq_y + 2q_zq_w & 1 - 2q_x^2 - 2q_z^2 & 2q_yq_z - 2q_xq_w \\ 2q_xq_z - 2q_yq_w & 2q_yq_z + 2q_xq_w & 1 - 2q_x^2 - 2q_y^2 \end{pmatrix}$$

(here the $\mathbf{t}$ subscript has been dropped for clarity). Finally, the bearing vector is calculated by multiplying the unit vector $\hat{\mathbf{x}}$ by $\mathcal{R}(\mathbf{q_t})$:

$$\hat{\mathbf{d}} = \mathcal{R}(\mathbf{q_t})\hat{\mathbf{x}} = \mathcal{R}(\mathbf{q_t})(1, 0, 0)^T \tag{2.6}$$

In this case, $\hat{\mathbf{x}}$ is be chosen to be consistent with the particular coordinate frames in use. Particularly, $\hat{\mathbf{x}}$ in the UAV body frame is rotated by $\mathbf{q_2}$ to the central axis of the camera.

The next three sections describe how each of $\mathbf{q_1}$, $\mathbf{q_1}$, and $\mathbf{q_1}$ are calculated. Many of the calculations involve conversion from Euler angles ϕ (roll), θ (pitch), and ψ (yaw) to quaternions. The necessary conversion formula is presented here for reference.

$$\begin{aligned}
\mathbf{q}_w &= \cos(\psi/2)\cos(\theta/2)\cos(\phi/2) - \sin(\psi/2)\sin(\theta/2)\sin(\phi/2) \\
\mathbf{q}_x &= \sin(\psi/2)\sin(\theta/2)\cos(\phi/2) + \cos(\psi/2)\cos(\theta/2)\sin(\phi/2) \\
\mathbf{q}_y &= \sin(\psi/2)\cos(\theta/2)\cos(\phi/2) + \cos(\psi/2)\sin(\theta/2)\sin(\phi/2) \qquad (2.7) \\
\mathbf{q}_z &= \cos(\psi/2)\sin(\theta/2)\cos(\phi/2) - \sin(\psi/2)\cos(\theta/2)\sin(\phi/2)
\end{aligned}$$

2.3.1 Determination of $\mathbf{q_1}$

$\mathbf{q_1}$ represents the rotation from the fixed global frame to the UAV body frame. The UAV's inertial sensors may provide this information directly, or they may give the rotation of the UAV in terms of Euler angles ϕ_b (roll), θ_b (pitch), and ψ_b (yaw). If this is the case, $\mathbf{q_1}$ is calculated using Equations 2.7.

2.3.2 Determination of $\mathbf{q_2}$

$\mathbf{q_2}$ represents the rotation from the UAV body frame to the camera frame. This rotation must be found through the use of a calibration procedure. It is important that this procedure be fast to perform, especially since the particular hardware used in this research, the Draganfly SAVS camera system, uses rubber bands to secure the camera to the UAV. The camera may sometimes be jostled from its position in the course of handling the vehicle and thus require calibration on a regular basis.

For calibration, the rotation is first represented by Euler angles ϕ_c, θ_c, and ψ_c, where ϕ_c and θ_c are the azimuth and elevation angles of the camera, respectively, and ψ_c is the rotation of the camera about its centerline ("yaw"). The UAV is placed on a stationary stand, and three targets are place in view of the camera. One of the targets is moved until it is directly in the center of the camera image. Comparing the computed rays $\hat{\mathbf{d}}_i$ with the true positions of the objects using the 3D visualization tool, the angles are adjusted by hand until the rays intersect the true positions of the

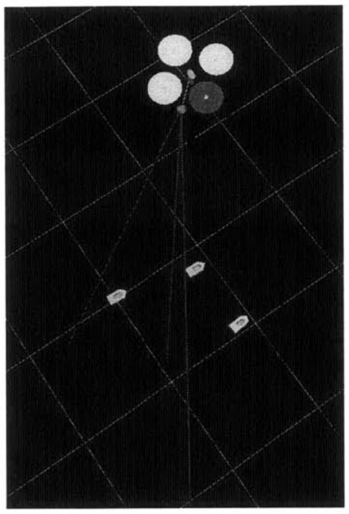

Figure 2-17: Pre-calibration misalignment

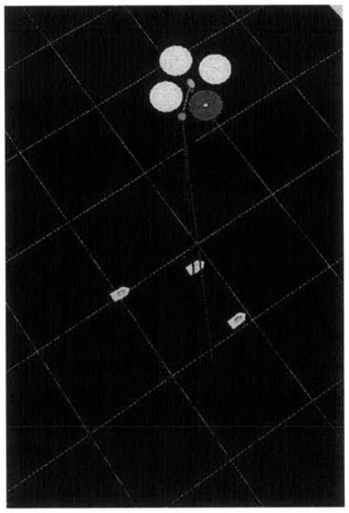

Figure 2-18: Aligning camera centerline

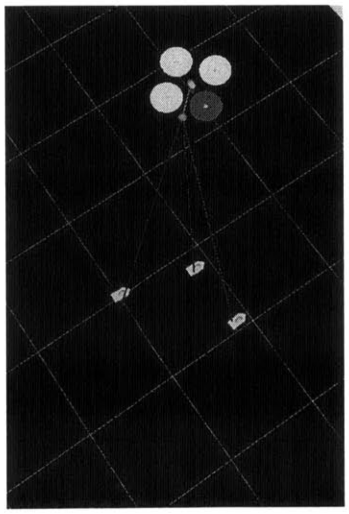

Figure 2-19: Adjusting camera yaw. Calibration finished

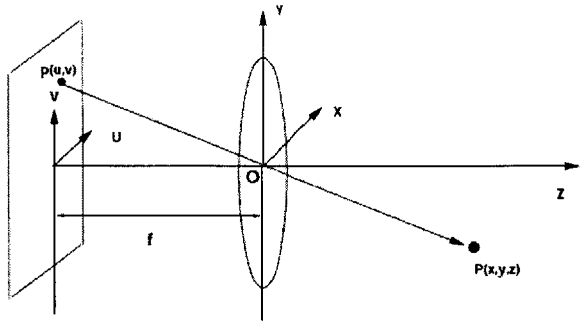

Figure 2-20: Pinhole camera model

objects (Figures 2-17,2-18,2-19). Once the angles are found, $\mathbf{q}_3$ is calculated using Equation 2.7.

2.3.3 Determination of $\mathbf{q}_3$

The camera is modeled as a pinhole camera (see Figure 2-20. In this model, light from an object located at $P(x, y, z)$ follows a straight line from the object, through a small aperture located at the origin, to the point $p(u, v)$ on the camera image plane. The camera is parameterized by the distance from the aperture to the focal plane. This distance is known as the *focal length* and is denoted by f in Figure 2-20. Once f is known, the unit direction vector $\overrightarrow{OP}$ can be found using simple geometry, since the direction is characterized by the Euler angles ϕ_i (elevation) and θ_i (azimuth), which are given by

$$\begin{aligned} \phi_i &= \arctan\left(\frac{u}{f}\right) \\ \theta_i &= \arctan\left(\frac{v}{f}\right) \end{aligned}$$

Once ϕ_i and θ_i are found, they can be converted to quaternion form using Equations 2.7, with $\psi_i = 0$. At this point, $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$ have all been found, and the bearing vector $\hat{\mathbf{d}}$ can be computed using Equation 2.6.

Chapter 3

Cooperative Bearings-Only Estimation

The most important outputs of the vision processing system presented in the previous chapter are the estimated bearing vectors from each camera to the target. In order for the UAV autonomous system to be able to track the target, this information must be used to generate an estimate of the position and velocity of the target in three dimensions.

The problem of estimating the state of a target given only bearing measurements from a single observation vehicle is known as the *bearings-only estimation* problem and has been studied extensively [52, 45, 23]. Figure 3-1 shows the general setup for the two-dimensional bearings-only tracking problem, where the observation vehicle is in blue and the target vehicle is in green. Geometrically, it is clear that it is not possible to determine the state of the target using a single observation (since the target may line anywhere along the observed line-of-sight). Thus, the problem is to somehow combine multiple observations to form an estimate of the target state. In its standard form, the bearings-only estimation problem is nonlinear because the observed state variables are the measured angles to the target (θ in Figure 3-1). For this reason, researchers have investigated using nonlinear filtering techniques, such as Extended Kalman Filters and Unscented Kalman Filters [12, 43, 3], to determine the state.

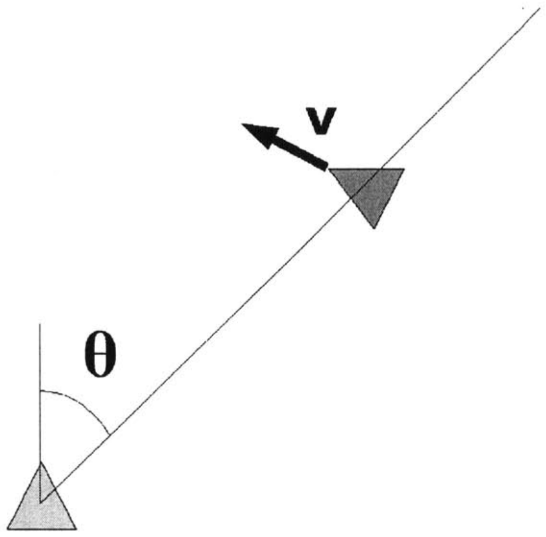

Figure 3-1: The bearings-only estimation problem. Observation vehicle: blue. Target vehicle: green

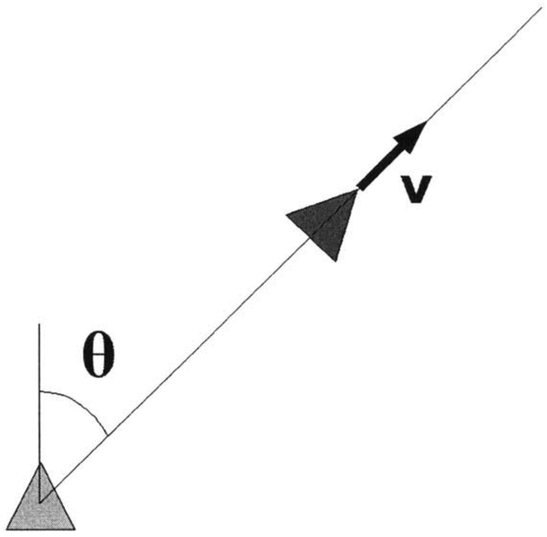

Figure 3-2: Non-observable motion in the bearings-only estimation problem

There is an inherent observability problem when bearings-only estimation is attempted with a single observer. The problem is illustrated in Figure 3-2. Geometrically, the problem is simple to understand: when the target vehicle's motion is purely radial (i.e. along the line-of-sight of the observation vehicle), the measurement θ will be constant even though the target is moving. Thus, the observer is unable to detect radial motion of the target vehicle, regardless of the number of observations taken.

One approach to solving the observability problem is to require the observing vehicle to execute maneuvers in order to obtain vantage points where the target vehicle's motion is not purely radial [44, 9, 48, 4, 39]. This approach can work very well, especially if the observing vehicle is very maneuverable, or the target vehicle is moving slowly or at nearly constant velocity. However, if the target is maneuvering rapidly, it may be difficult for the observing vehicle to complete the necessary maneuvers quickly enough to improve its estimate.

This research presents a different approach that eliminates the observation problem by simultaneously combining the measurements of several observation vehicles. Figure 3-3 illustrates the basic idea. By working together, the two observation vehicles can unambiguously determine the location of the target, even if the target is maneuvering aggressively. Furthermore, this is accomplished without requiring the observation vehicles to move at all; the system is fully observable at all times. Thus, the problem of designing and executing potentially complex trajectories for the observation vehicles is completely eliminated, freeing the observation vehicles to follow other types of trajectories that may be better suited to the mission at hand. For example, target tracking becomes simpler to accomplish because the observation vehicles only need to calculate trajectories that keep the target in the field of view of the camera, without attempting to simultaneously meet the constraints of designing an observable trajectory as well.

In addition to solving the observability problem, the cooperative approach has other key advantageous. Multiple UAVs provide redundancy, allowing for continued tracking even when individual vehicles experience failures. Such failures may include malfunctions of the camera system or mechanical problems that necessitate the vehicle returning to base for servicing. Furthermore, the presence of obstructions in the environment may temporarily block the field of view of a UAV as it attempts to observe the target. Using multiple UAVs with different lines-of-sight increases the probability that the target will remain observable to the group of UAVs, even when individual vehicles' lines-of-sight are blocked. Finally, the cooperative UAV vision tracking problem can be reformulated as a linear estimation problem. Using the observed bearings of the target from each UAV, an estimate of the three-dimensional target position can be obtained by solving an optimization problem. The solution to this problem can be found very efficiently in time that is linear in the number of observation vehicles. This estimate is then used as a measurement input to a simple linear Kalman filter whose state is the position and velocity of the target.

This chapter presents a vision-based estimation and tracking algorithm that exploits cooperation between multiple UAVs in order to provide accurate target state estimation and allow good tracking of the target without the need for observation vehicles to execute maneuvers to gain better vantage points. The method uses an optimization technique to combine the instantaneous observations of all UAVs, allowing for very rapid estimation. Furthermore, the algorithm can be naturally distributed

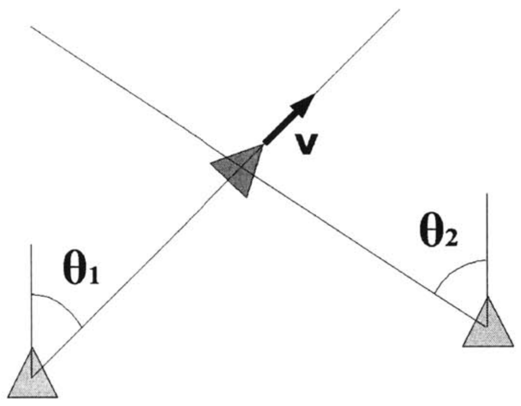

Figure 3-3: Use of multiple observation vehicles to eliminate the observability problem

among all participating UAVs with very modest communication bandwidth requirements and is computationally efficient, making it well suited to implementation on real-time applications. In addition, although the algorithm is designed for use with multiple observing vehicles, we show that it can also be used in the case where only a single vehicle is available.

3.1 Problem Formulation and Nomenclature

The statement of the estimation problem is as follows. There is a set $\mathcal{V} = \{V_1, V_2, \ldots\}$ containing one or more vehicles, each equipped with a camera and vision processing subsystem that generates estimated bearing measurements to the target at regular, discrete intervals $T_1, T_2, \ldots, T_k, \ldots$. Denote the set of discrete times by $\mathcal{T}$. Each measurement consists of a pair of vectors $(\hat{\mathbf{x}}_{v,k}, \hat{\mathbf{d}}_{v,k})$, where the subscripts $v \in \mathcal{V}$ and $k \in \mathcal{T}$ denote the observing vehicle and time the measurement was taken, respectively. $\hat{\mathbf{x}}_{v,k}$ denotes the estimated location of vehicle v at time k (as given by the UAV's onboard navigation sensors), and $\hat{\mathbf{d}}_{v,k}$ denotes the estimated bearing from v to the target (as given by the video processing system) at that time.

It is assumed that the location and bearing vectors are not perfectly known, since sensor noise or other factors may cause uncertainty in both the knowledge of the location of the UAV and the bearing to the target. To capture this uncertainty, we define

$$\mathbf{x}_{v,k} = \hat{\mathbf{x}}_{v,k} + \delta\mathbf{x}_{v,k}, \quad \mathbf{x}_{v,k} \in \mathbb{R}^3 \tag{3.1}$$

where $\mathbf{x}_{v,k}$ is the true location of the UAV, $\hat{\mathbf{x}}_{v,k}$ is the estimated location, and $\delta\mathbf{x}_{v,k}$ is a random variable that captures the uncertainty in the UAV's position. The probability distribution of $\delta\mathbf{x}_{v,k}$ is assumed to be known. Similarly, we define

$$\mathbf{d}_{v,k} = \hat{\mathbf{d}}_{v,k} + \delta\mathbf{d}_{v,k}, \quad \mathbf{d}_{v,k} \in \mathbb{R}^3 \tag{3.2}$$

where again $\mathbf{d}_{v,k}$ is the true bearing to the target, $\hat{\mathbf{d}}_{v,k}$ is the estimated bearing, and $\delta\mathbf{d}_{v,k}$ represents uncertainty in the bearing vector. For convenience, since the length of $\hat{\mathbf{d}}_{v,k}$ is unimportant (only the direction matters), we shall assume $\hat{\mathbf{d}}_{v,k}$ is unit length

$$|\hat{\mathbf{d}}_{v,k}| = 1 \tag{3.3}$$

Also, assume that

$$\delta\mathbf{d}_{v,k}^T\hat{\mathbf{d}}_{v,k} = 0 \tag{3.4}$$

This assumption is reasonable given that uncertainty in $\hat{\mathbf{d}}_{v,k}$ is most naturally characterized by uncertainty in the angles from the camera to the target. Again, assume that the probability distribution of $\delta\mathbf{d}_{v,k}$ is known. In addition, we assume that the uncertainty $\delta\mathbf{d}_{v,k}$ is small:

$$|\delta\mathbf{d}_{v,k}| \ll |\hat{\mathbf{d}}_{v,k}| = 1 \tag{3.5}$$

This is reasonable since the uncertainty in direction is likely to be largely a function of camera calibration, which can be measured relatively easily and accurately.

Finally, a weight $w_{v,k}$ is associated with each measurement. This weight may be used to account for differences in the quality of each vehicle's measurement at particular times (i.e., differences in video quality as estimated by the vision subsystem).

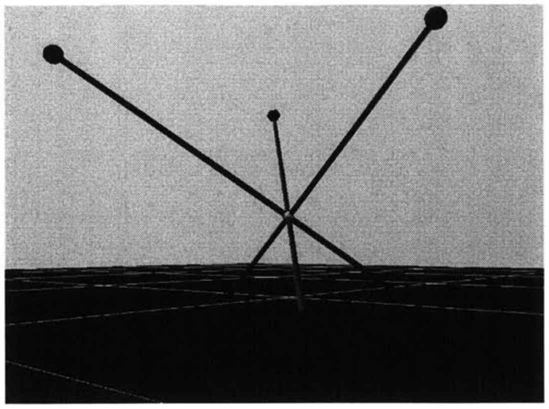
Figure 3-4: With perfect measurements, the observation rays intersect in a single point

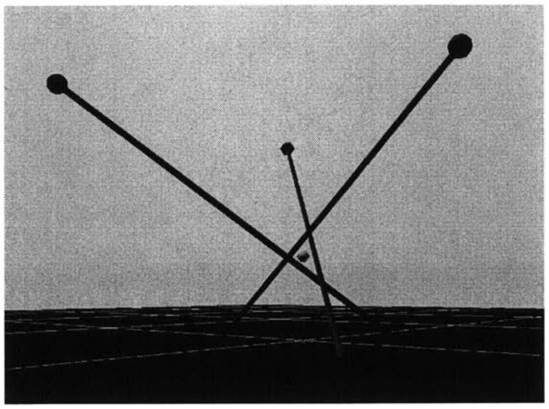
Figure 3-5: With real (noisy) measurements, the observation rays do not intersect at all

3.1.1 Discussion

Note that given the true quantities $\mathbf{x}_{v,k}$ and $\mathbf{d}_{v,k}$, the target must lie along the ray

$$\mathbf{l}_{v,k}(\lambda_{v,k}) = \mathbf{x}_{v,k} + \lambda_{v,k}\mathbf{d}_{v,k}, \qquad \lambda_{v,k} \geq 0 \tag{3.6}$$

If the measurements $(\hat{\mathbf{x}}_{v,k}, \hat{\mathbf{d}}_{v,k})$ from the UAVs were perfect - in other words, if $\delta\mathbf{x}_{v,k} = \delta\mathbf{d}_{v,k} = 0$ so that $\mathbf{x}_{v,k} = \hat{\mathbf{x}}_{v,k}$ and $\mathbf{d}_{v,k} = \hat{\mathbf{d}}_{v,k}$ - then the problem of estimating the target location would be simple, because all of the rays would intersect in a single point (see Figure 3-4). To calculate the position of the target in this case, all that is required is to calculate the intersection of any pair of rays. It is interesting to note also that in this case, the addition of more measurements does nothing to improve the estimate, since the intersection is known exactly with only two measurements.

Of course, the real measurements obtained will not be perfect. Instead, each measurement will be slightly inaccurate, resulting in a picture like the one shown in Figure 3-5. In this case, it is unlikely that any of the rays will intersect perfectly. Despite this, it is intuitive from looking at Figure 3-5 that good information about the location of the target should be obtainable from the measurements, and that in this case, having more measurements is beneficial in improving the estimate of the target position.

To incorporate the information from all of the measurements into the estimate, we would like to calculate the "pseudo-intersection" of all of the rays. This problem can be formulated as an optimization problem and solved very efficiently. The method to do this is presented in the next section.

Before presenting the solution to the pseudo-intersection problem, however, the question of which particular measurements $(\hat{\mathbf{x}}_{v,k}, \hat{\mathbf{d}}_{v,k})$ to use at each time step K needs to be addressed. The objective is to select the measurements whose pseudo-intersection will yield the most accurate estimate of the *current* position of the target (i.e. the position of the target at time K). At the same time, it is clear geometrically that at least two such measurements must be used in order to determine the pseudo-intersection. Therefore, if multiple vehicles are present, our strategy will be to use only the most current measurement from each vehicle. Specifically, the measurement set $\mathcal{M}_{multiple}$ is selected as

$$\mathcal{M}_{multiple} = \{(\hat{\mathbf{x}}_{v,k}, \hat{\mathbf{d}}_{v,k}) : v \in \mathcal{V},\ k = K\} \tag{3.7}$$

If, however, only a single vehicle is present, at least one measurement from a previous time step must be used in addition to the current measurement:

$$\mathcal{M}_{single} = \{(\hat{\mathbf{x}}_{v,k}, \hat{\mathbf{d}}_{v,k}) : v = V_1,\ k \in \{K-n, K-n+1, \dots, K\}\} \tag{3.8}$$

Here, care must be taken in selecting how many previous measurement(s) n to use. Selection of many measurements (large n) will provide greater spatial separation between the measurements (since the observation vehicle must be moving in the single vehicle case). However, this will also increase the effective phase lag of the estimator since many older measurements are being used, resulting in a time-delayed estimate of the position of the target. Conversely, selecting fewer measurements reduces the lag of the estimator but may provide small spatial separation, resulting in poor estimates of the target position. Therefore, the design of the estimator for the single vehicle case contains a fundamental performance tradeoff that is absent in the multiple vehicle case. Thus, we expect the overall performance of the multi-vehicle estimator to be

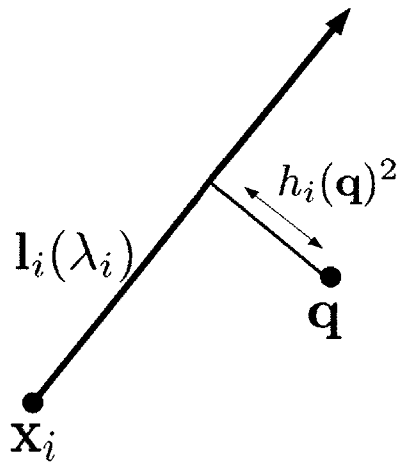

Figure 3-6: $h_i(\mathbf{q})^2$ is the shortest distance from $\mathbf{q}$ to $\mathbf{l}_i(\lambda_i)$

superior to that of the single vehicle estimator. This prediction will be confirmed in experimental results to be presented later in this chapter.

3.2 Pseudo-Intersection Optimization

The pseudo-intersection problem is to estimate the true position of the target, $\mathbf{q}$, given the set of measurements $\mathcal{M}$. The problem can be formulated as an optimization problem by choosing an appropriate objective function. In this case, a reasonable choice of objective function is

$$E(\mathbf{q}) = \sum_{i \in \mathcal{M}} w_i h_i(\mathbf{q}) \tag{3.9}$$

where $h_i(\mathbf{q})$ is the square of the minimum distance from $\mathbf{q}$ to the ray $\mathbf{l}_i(\lambda_i)$:

$$h_i(\mathbf{q}) \;=\; \min_{\lambda_i} ||\mathbf{q} - \mathbf{l}_i(\lambda_i)||^2 = \min_{\lambda_i} ||\mathbf{q} - (\mathbf{x}_i + \lambda_i \mathbf{d}_i)||^2 \tag{3.10}$$

Figure 3-6 illustrates the relationship between $\mathbf{q}$, $h_i(\mathbf{q})$, and $\mathbf{l}_i(\lambda_i)$. Minimizing $h_i(\mathbf{q})$ with respect to λ_i yields the result

$$h_i(\mathbf{q}) = \mathbf{q}^T\mathbf{q} - 2\mathbf{q}^T\mathbf{x}_i + \mathbf{x}_i^T\mathbf{x}_i - (\mathbf{d}_i^T\mathbf{q} - \mathbf{d}_i^T\mathbf{x}_i)^2 \tag{3.11}$$

Substituting this result into Eq. 3.9 and minimizing $E(\mathbf{q})$ with respect to $\mathbf{q}$ yields the equation that the optimal estimate must satisfy:

$$\mathcal{A}\mathbf{q}^{\star} = \mathbf{b} \tag{3.12}$$

where

$$\begin{aligned} \mathcal{A} &= \sum_{i=1}^{n} w_i \left(I - \mathbf{d}_i \mathbf{d}_i^T\right) && (3.13) \\ \mathbf{b} &= \sum_{i=1}^{n} w_i \left(\mathbf{x}_i - (\mathbf{x}_i^T \mathbf{d}_i)\mathbf{d}_i\right) && (3.14) \end{aligned}$$

See Appendix A for a complete derivation of the above formulae.

Now, $\mathcal{A}$ and $\mathbf{b}$ cannot be calculated by the algorithm directly because only the the noisy measurements $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{d}}_i$ are known. To compensate for these errors, $\mathcal{A}$ and $\mathbf{b}$ are expanded by substituting Eqs. 3.1 and 3.2 into Eqs. 3.13 and 3.14. After dropping second-order terms and grouping the known and unknown terms, the equations become

$$\begin{aligned} \mathcal{A} &= \hat{\mathcal{A}} + \delta\mathcal{A} && (3.15) \\ \mathbf{b} &= \hat{\mathbf{b}} + \delta\mathbf{b} && (3.16) \end{aligned}$$

where

$$\begin{aligned} \hat{\mathcal{A}} &= \sum_{i=1}^{n} w_i \left(I - \hat{\mathbf{d}}_i \hat{\mathbf{d}}_i^T\right) && (3.17) \\ \delta\mathcal{A} &= -\sum_{i=1}^{n} w_i (\delta\mathbf{d}_i \hat{\mathbf{d}}_i^T + \hat{\mathbf{d}}_i^T \delta\mathbf{d}_i) && (3.18) \\ \hat{\mathbf{b}} &= \sum_{i=1}^{n} w_i \left(\hat{\mathbf{x}}_i - (\hat{\mathbf{x}}_i^T \hat{\mathbf{d}}_i)\hat{\mathbf{d}}_i\right) && (3.19) \\ \delta\mathbf{b} &= \sum_{i=1}^{n} w_i (\delta\mathbf{x}_i - (\hat{\mathbf{x}}_i^T \delta\mathbf{d}_i)\hat{\mathbf{d}}_i - (\delta\mathbf{x}_i^T \hat{\mathbf{d}}_i)\hat{\mathbf{d}}_i - (\hat{\mathbf{x}}_i^T \hat{\mathbf{d}}_i)\delta\mathbf{d}_i) && (3.20) \end{aligned}$$

Note that $\hat{\mathcal{A}}$ and $\hat{\mathbf{b}}$ are known terms, because they involve only quantities that are measured directly. $\delta\mathcal{A}$ and $\delta\mathbf{b}$ are random variables because they involve the uncertain quantities $\delta\mathbf{x}_i$ and $\delta\mathbf{d}_i$. The optimal estimate can now be written as

$$\mathbf{q}^\star \approx \mathcal{A}^{-1}\mathbf{b} = (\hat{\mathcal{A}} + \delta\mathcal{A})^{-1}(\hat{\mathbf{b}} + \delta\mathbf{b}) \tag{3.21}$$

The error term $\delta\mathcal{A}$ is small ($\delta\mathcal{A} \ll \hat{\mathcal{A}}$), since it is composed of terms involving the small errors $\delta\mathbf{d}_i$. Expanding the matrix inverse function in a Taylor series around $\hat{\mathcal{A}}$ gives

$$(\hat{\mathcal{A}} + \delta\mathcal{A})^{-1} \approx \hat{\mathcal{A}}^{-1} - \hat{\mathcal{A}}^{-1}\delta\mathcal{A}\hat{\mathcal{A}}^{-1} \tag{3.22}$$

Thus, Eq. 3.21 becomes

$$\begin{aligned} \mathbf{q}^\star &\approx \hat{\mathcal{A}}^{-1}\hat{\mathbf{b}} + \hat{\mathcal{A}}^{-1}\delta\mathbf{b} - \hat{\mathcal{A}}^{-1}\delta\mathcal{A}\hat{\mathcal{A}}^{-1}\hat{\mathbf{b}} - \hat{\mathcal{A}}^{-1}\delta\mathcal{A}\hat{\mathcal{A}}^{-1}\delta\mathbf{b} && (3.23) \\ &\approx \hat{\mathbf{q}}^\star + \hat{\mathcal{A}}^{-1}\delta\mathbf{b} - \hat{\mathcal{A}}^{-1}\delta\mathcal{A}\hat{\mathcal{A}}^{-1}\hat{\mathbf{b}} && (3.24) \end{aligned}$$

where

$$\hat{\mathbf{q}}^\star = \hat{\mathcal{A}}^{-1}\hat{\mathbf{b}} \tag{3.25}$$

is the optimal estimate that can be calculated from the measurements. The error $\delta\mathbf{q}^\star$ in the estimate is

$$\delta\mathbf{q}^\star = \hat{\mathcal{A}}^{-1}\delta\mathbf{b} - \hat{\mathcal{A}}^{-1}\delta\mathcal{A}\hat{\mathcal{A}}^{-1}\hat{\mathbf{b}} \tag{3.26}$$

Since the probability distributions of the random variables $\delta\mathbf{x}_i$ and $\delta\mathbf{d}_i$ are known, the covariance of $\delta\mathbf{q}^\star$ can be calculated. This covariance is needed in order to implement the Kalman filter, discussed below.

Eq. 3.25 demonstrates that the optimal estimate $\hat{\mathbf{q}}^\star$ can be computed in time that is *linear* in the number of measurements to the object n. $\hat{\mathcal{A}}$ and $\hat{\mathbf{b}}$ can be constructed in linear time since they are sums over all rays. Once $\hat{\mathcal{A}}$ and $\hat{\mathbf{b}}$ are known, Eq. 3.25 can be solved in constant time by inverting the 3 x 3 matrix $\hat{\mathcal{A}}$. Since the entire process runs in linear time with respect to n, this method is very computationally efficient. Note that if there is only a single vehicle, $n = 1$, the matrix $\hat{\mathcal{A}}$ is singular and Eq. 3.25 cannot be solved. In this case, a single vehicle would have to make an additional assumption about the location of the target, such that it is located on the ground ($z = 0$), in order to calculate a solution. In all other cases, however, $\hat{\mathcal{A}}$ is invertible as long as the observed direction vectors $\hat{\mathbf{d}}_i$ are not all parallel to each other. As long as the observation points $\hat{\mathbf{x}}_i$ are not in the same location (which cannot happen since the UAVs cannot occupy the same physical point in space) or on diametrically opposite sides of the target (which is unlikely for a ground target since one observation point would be under the ground; for an aerial target, the observing vehicles need to coordinate to ensure they avoid this situation), a solution can be found.

3.3 Kalman Filter Design

Once the estimate $\hat{\mathbf{q}}_k^\star$, taken at time k, is known, it can be used as the measurement into a simple linear Kalman filter based on the assumed dynamics of the target vehicle [3]. This research uses a system model with state vector

$$\mathbf{X} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T \tag{3.27}$$

The discrete time system dynamics are then given by

$$\begin{aligned} \mathbf{X}_{k+1} &= A\mathbf{X}_k + \mathbf{v}_k && (3.28) \\ \mathbf{Y}_k = \hat{\mathbf{q}}_k^\star &= C\mathbf{X}_k + \delta\mathbf{q}_k^\star && (3.29) \end{aligned}$$

where $\mathbf{v}_k$ is the process noise and $\delta\mathbf{q}^\star$ is the measurement noise. The process noise covariance $\mathbf{v}_k$ is assumed to be known based on the dynamics of the target and the environmental disturbances present. The covariance of $\delta\mathbf{q}^\star$ is found as discussed above. A and C are given by

$$A = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{3.30}$$

where Δt is the sampling rate of the filter. Using these dynamics, a linear Kalman filter can be easily designed and implemented. This filter can be run on each UAV; the only requirement is that the UAVs communicate their locations $\mathbf{x}_i$ and estimation directions $\mathbf{d}_i$ to each other. Since each of these quantities is a three dimensional vector, this method requires only six numbers to be transmitted by each UAV, making it well suited for environments where communication bandwidth is limited.

3.4 Estimator Performance

Several flight experiments were carried out to test the relative performance of the non-cooperative, single-vehicle estimation approach and the cooperative, multi-vehicle estimation approach. The experiments were conducted by running the estimation algorithms in parallel with a logging process that collected raw position data of the target from the Vicon motion capture system. The Vicon measurements were then used as the truth data against which the estimator data was compared.

In all of the experiments, the performance of the cooperative approach was superior to that of the non-cooperative approach, as expected. In addition, the assumptions made in the derivation of the cooperative estimation algorithm (in particular, the smallness of the error terms in Eq. 3.5) were experimentally verified by the overall performance accuracy of the method. In general, the cooperative estimator was able to give the position of the target to within 0.04m at all times. This is a strong result, especially given that the target vehicle itself was approximately 0.30m long.

3.4.1 Experiment 1: Stationary Target

For the first experiment, a stationary target was used. For the single-vehicle, non-cooperative estimation, one quadrotor was commanded to move at constant velocity of approximately 0.2m/s the vicinity of the target in order to provide the spatially separated measurements required by the single-vehicle estimation process (see Figure 3-7). For the multi-vehicle, cooperative estimation, two quadrotors were commanded to hover at an altitude of 1.25m at stationary locations in view of the target (see Figure 3-8).

Results of the experiment confirmed that both estimators were able to estimate the position of the target in the stationary target case, though the performance of the cooperative estimator was significantly better than that of the non-cooperative estimator. Scatter plots showing the estimation error for the non-cooperative and cooperative estimators are shown in Figures 3-9 and 3-10, respectively. From the figures, it is clear that the cooperative estimator is able to provide a much more precise and consistent set of measurements; the standard deviation of the estimation error in the cooperative case is about an order of magnitude smaller than that in the non-cooperative case (see Table 3.1). The small bias in the cooperative estimate is due to uncertainty in the orientation of the camera since it is attached to the UAV with a vibration-dampening rubber band attachment system that may shift slightly during flight. With a more rigid attachment, the camera could be calibrated more accurately and the results of the cooperative estimate would be even better. Furthermore, the data presented for the non-cooperative estimator represents the best-case scenario where the tracking vehicle is moving constantly. Data taken while the tracking vehicle was hovering is shown for reference in Figure 3-11. This data clearly shows that the non-cooperative estimator struggles to give accurate estimates when the tracking vehicle is not moving much (i.e. when in a hover, where the only vehicle motion is due to small wind disturbances). Note that the scale in Figure 3-11 is increased by a factor of about 15 as compared with Figures 3-9 and 3-10. This poor performance is due to the lack of spatial separation in the measurements obtained.

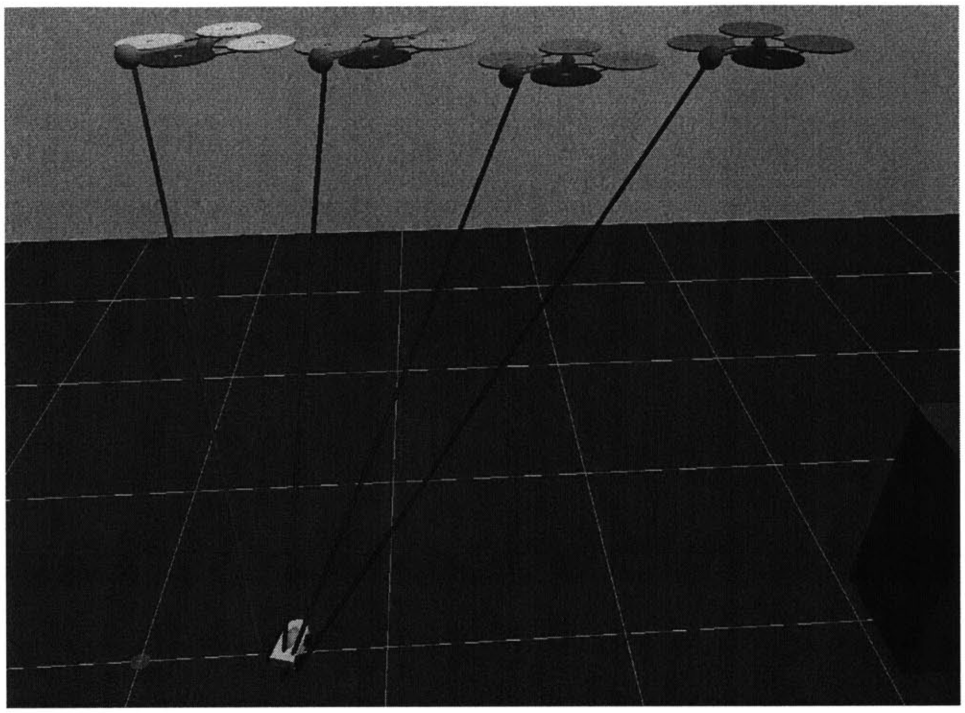

Figure 3-7: Non-cooperative estimator vehicle configuration for Experiment 1

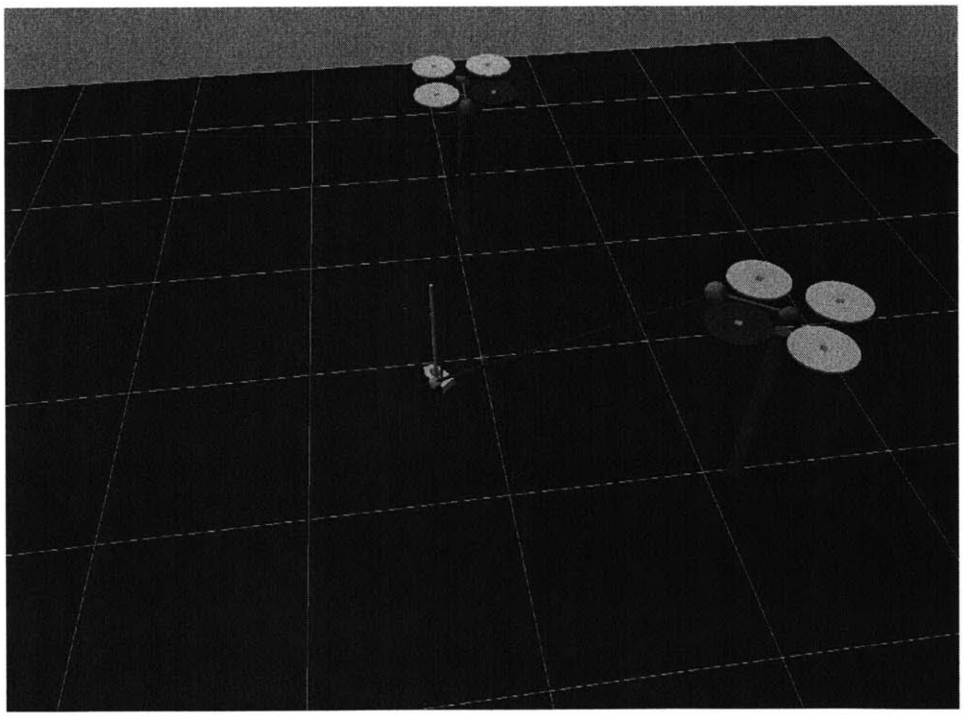

Figure 3-8: Cooperative estimator vehicle configuration for Experiment 1

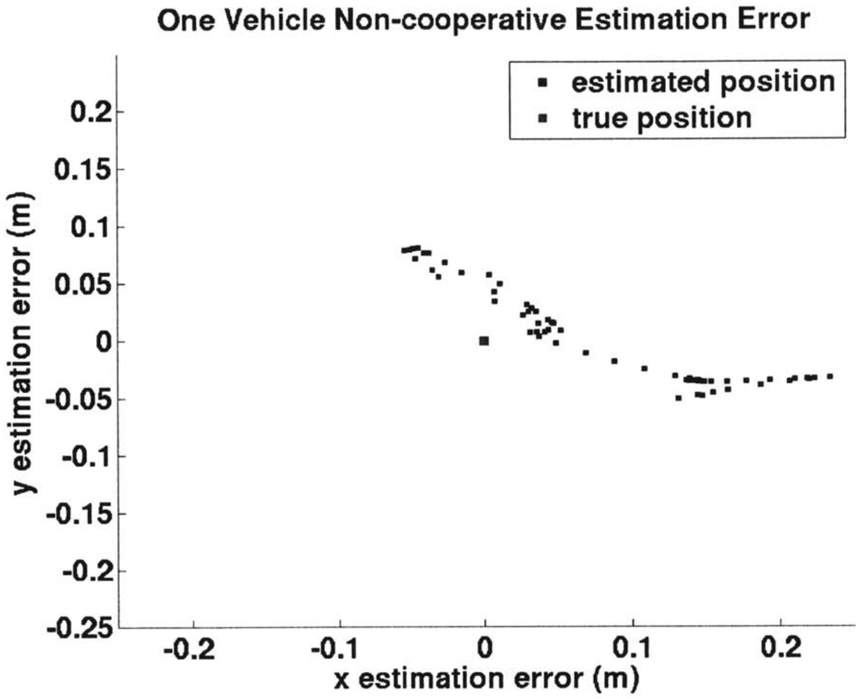

Figure 3-9: Non-cooperative estimator results for Experiment 1. Mean estimation error: x = 0.0795 m, y = 0.0390 m. Standard deviation: σ_x = 0.0852 m, σ_y = 0.0425 m.

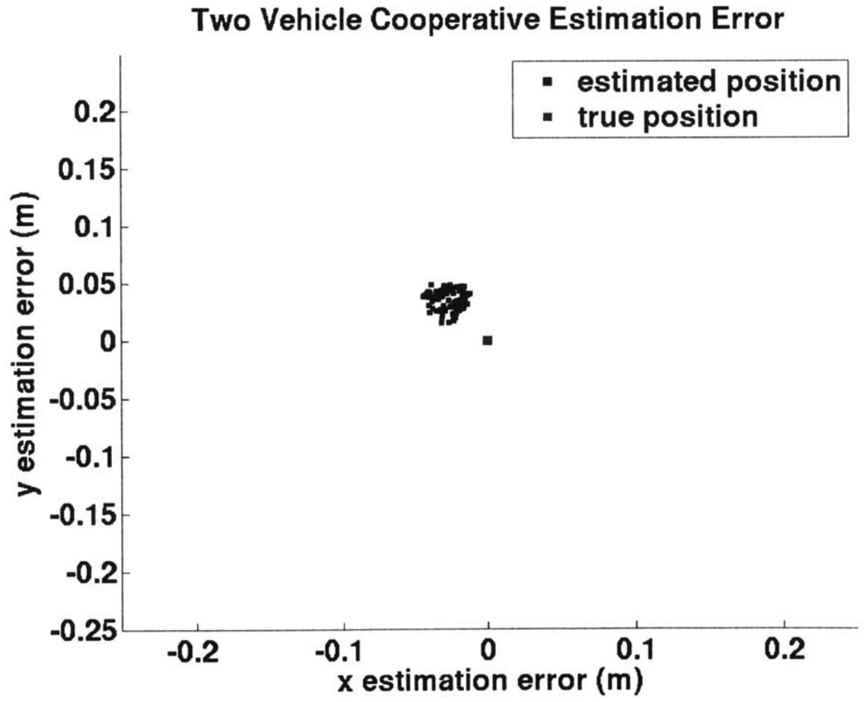

Figure 3-10: Cooperative estimator results for Experiment 1. Mean estimation error: x = -0.0265 m, y = 0.0368 m. Standard deviation: σ_x = 0.0082 m, σ_y = 0.0083 m

Table 3.1: Mean estimation errors and standard deviations for Experiment 1

	$\overline{x}$ (m)	$\overline{y}$ (m)	$\overline{z}$ (m)	σ_x (m)	σ_y (m)	σ_z (m)
Non-cooperative	0.0795	0.0390	0.0162	0.0852	0.0425	0.1397
Cooperative	-0.0265	0.0368	0.0201	0.0082	0.0083	0.0074

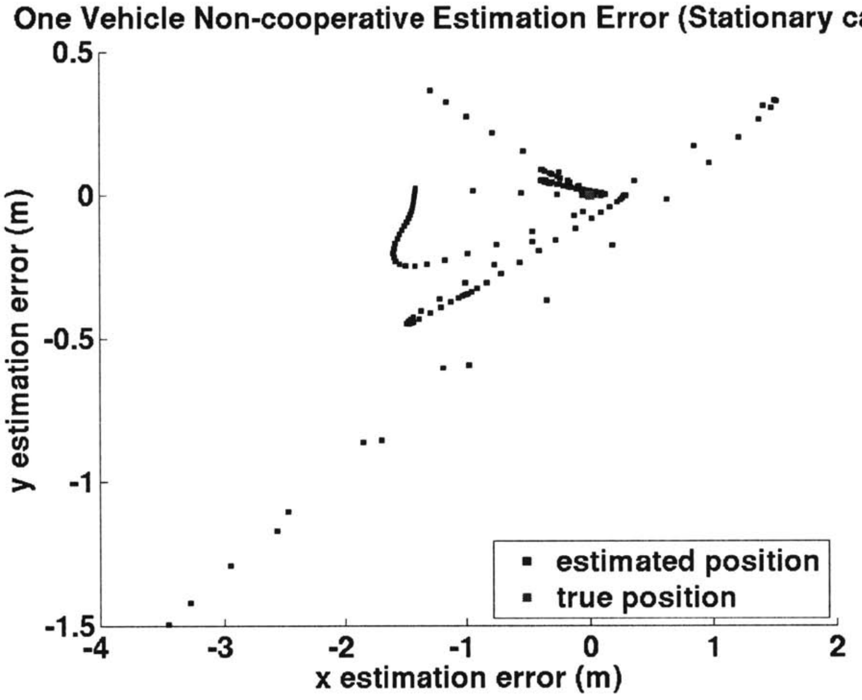

Figure 3-11: Non-cooperative estimator results for Experiment 1 (stationary observing vehicle case). Mean estimation error: x = -0.6339 m, y = -0.1280 m. Standard deviation: σ_x = 0.8718 m, σ_y = 0.3069 m

3.4.2 Experiment 2: Maneuvering Target

The second experiment examined the estimators' performance in tracking a vehicle which was maneuvering on the ground. For this experiment, the cooperative estimator was run using observations from two quadrotors commanded to move in order to keep the estimated location of the target in view of both vehicles (see Figure 3-12). The non-cooperative estimator was run in parallel, using observations from only one of the quadrotors.

Results of Experiment 2 are shown in Figure 3-13. The figure shows the estimated trajectory of the maneuvering target from both of the estimators, along with the target's true trajectory. It is clear that in the maneuvering target case, the non-cooperative estimator gives very poor results; its estimate tends to oscillate wildly back and forth in the radial (non-observable) direction. The problems encountered by the non-cooperative estimator worsen when the target vehicle makes a sharp turn, due to the fact that the estimator must use time-delayed data to make its estimate. Overall, tracking the target reliably using the non-cooperative estimator would be very difficult or impossible in this case.

Meanwhile, the cooperative estimator tracks the target vehicle almost perfectly throughout its entire trajectory, including during the sharp turning maneuver. The average estimation error during this test for the cooperative estimator was 0.0382m and the standard deviation was 0.0087m. Comparing these results with the stationary case, we see that the performance of the cooperative estimator is almost identical for stationary and moving targets, which the performance of the non-cooperative estimator degrades sharply as discussed above. Since the cooperative estimator has no inherent observability problems, maneuvering targets pose no problem, unlike in the non-cooperative case. This property makes the cooperative estimation approach attractive for use in difficult tracking problems where the target is free to maneuver.

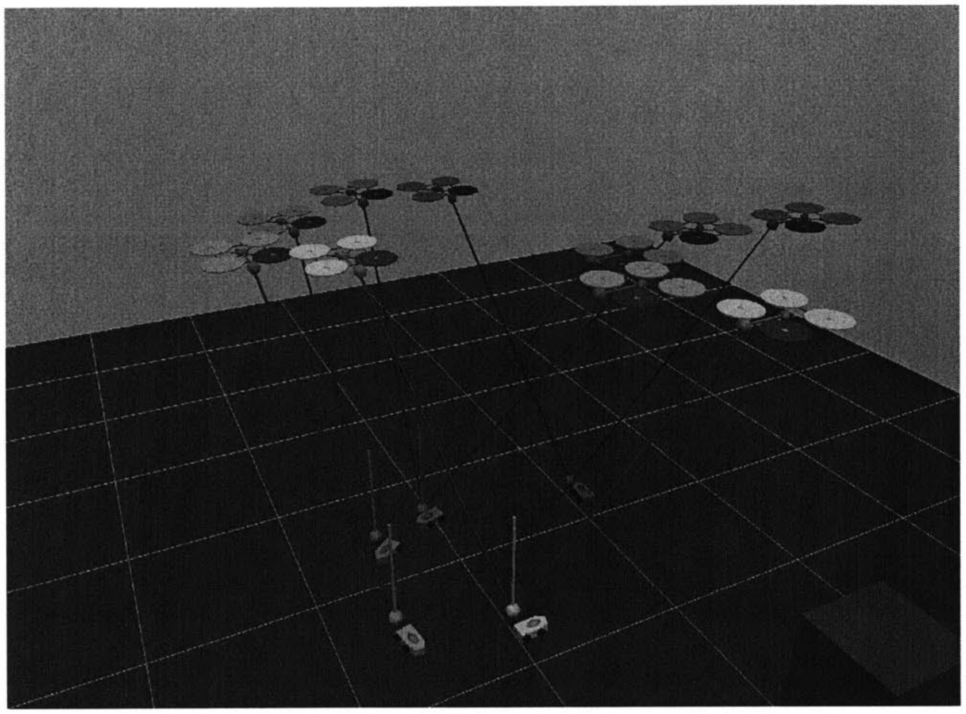

Figure 3-12: Setup for Experiment 2

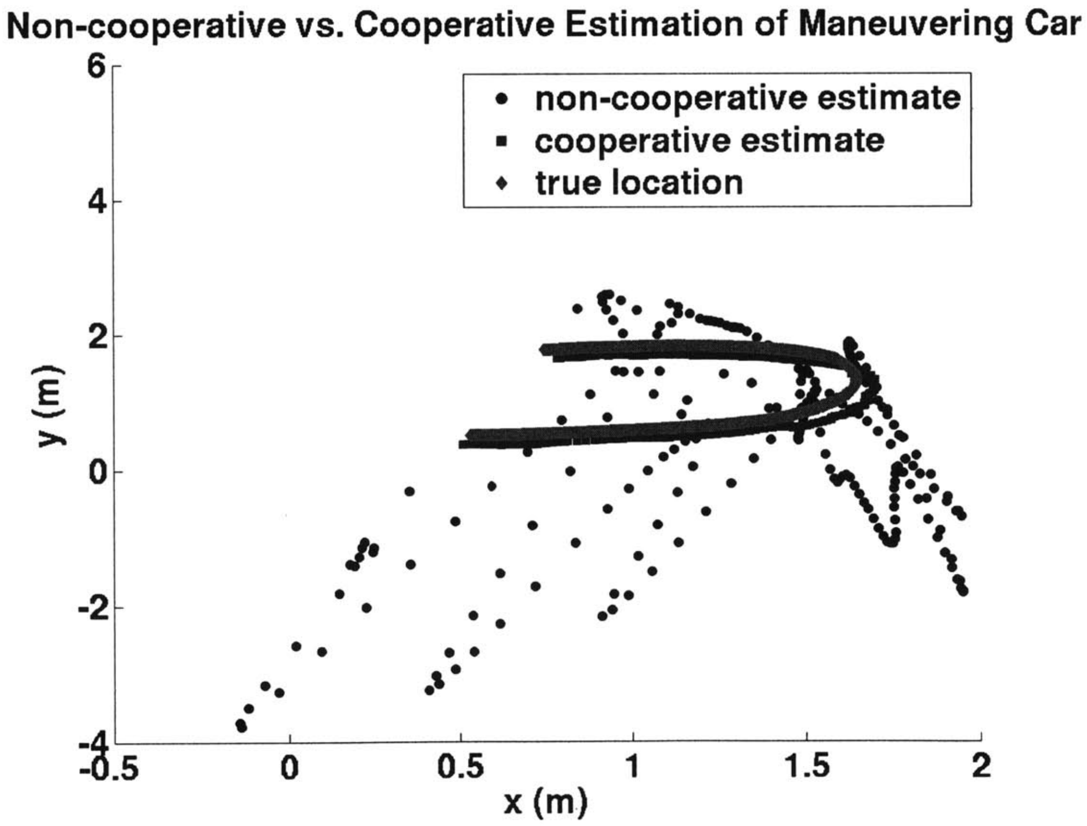

Figure 3-13: Results for Experiment 2

Chapter 4

Health Management Feedback in Task Assignment

The problem of simultaneously controlling and coordinating the actions of multiple autonomous agents in a dynamic environment is very complex. For this reason, proposed methods for solving the overall problem typically involve formulating several smaller sub-problems, each of which is simpler and therefore easier to solve [32]. One such solution architecture is shown in Figure 4-1. In this architecture, there are a number of components that work together to achieve the overall goals of the mission. The Mission Planning component is the highest level in the system. It keeps track of the mission objectives and generates *tasks*, which are discrete actions whose completion will aid the overall accomplishment of the mission. Examples of tasks include searching for, identifying, or tracking an object of interest. The Mission Planner provides the list of tasks to the Task Assignment component, which decides which of the available vehicles should perform each task based on information about the tasks and the capabilities of the vehicles. Once the assignments have been made, they are sent to the Trajectory Designer, which plans feasible trajectories for each vehicles. The output of the Trajectory Designer is a sequence of waypoints for each vehicle to follow. These waypoints are sent to the Vehicle Controllers, which compute the actual controls needed to follow the waypoint plans.

Inherent in each of the components in the architecture are a set of interconnected

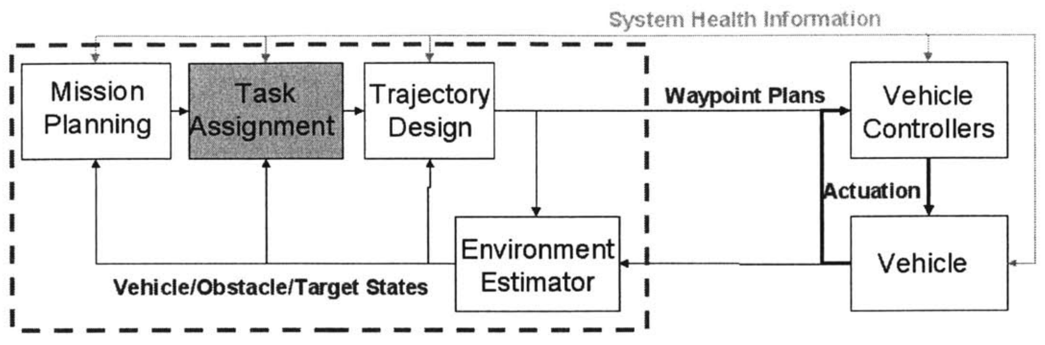

Figure 4-1: Overall autonomous mission system architecture

models that are used to make predictions of future system behavior. For example, the controller contains a model of the control input dynamics of the vehicle, while the task assignment component contains a model of the performance each vehicle can be expected to produce if assigned to a given task. In the most general sense, system actions are selected by searching for actions that lead to desirable predicted outcomes as given by the system models. Clearly, the performance of the system therefore depends heavily on the accuracy of these models.

One strategy for improving the accuracy of the system models is based on designing feedback loops that adjust the models according to sensor data and other observations gathered as the system is running. The amount, type, and quality of feedback information that each component receives plays a large role in how effectively the system can deal with dynamically changing factors in the environment, mission objectives, and state of the vehicles. Intuitively, feedback is necessary anywhere there is uncertainty in the system, so that the initial plan of action made by each of the components in the system can be modified if disturbances occur. In the complex environments and problems under consideration, there are many different sources of disturbances which occur at all levels of the architecture and should be accounted for. Of the many types of disturbances that may be present in the system, those which pertain to the current capabilities and status of the vehicles themselves are particularly interesting and important. The question of how to incorporate feedback loops that address these effects is generally referred to as *health management.* Examples of health management problems at each of the levels in the system are:

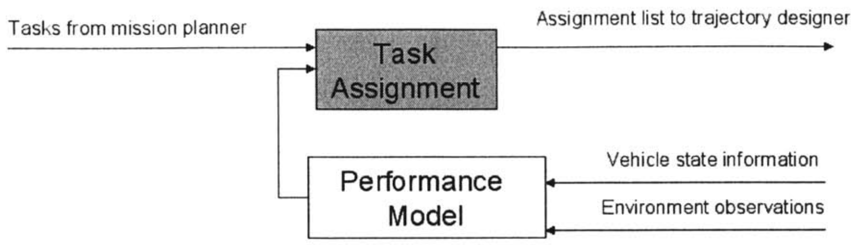
Tasks from mission planner
Task Assignment
Assignment list to trajectory designer
Performance Model
Vehicle state information
Environment observations

Figure 4-2: Incorporation of health state feedback in the Task Assignment component

- At the control level, it may be desirable to track the status of the control actuators during flight and feed this information to the controller. Use of this information may allow the controller to avoid overly aggressive command inputs when one of the actuators is determined to be near the point of failing.

- At the trajectory design level, information about the overall health of the control actuators may allow the trajectory designer to generate dynamically feasible trajectories that are adjusted for the current agility of the vehicle. For example, knowledge of a failing actuator may cause the trajectory designer to generate a path that is smoother and requires less actuation.

- At the task assignment level, knowledge of the vehicles' current sensor and fuel states may allow the task assignment algorithm to dynamically adjust to conditions that make particular vehicles unable to perform certain tasks. For example, a vehicle whose onboard camera fails is unable to perform surveillance tasks, but could still be useful for serving as a communications relay.

- At the mission planning level, feedback about the long-term maintenance needs of the vehicles performing the mission may allow the mission planner to make informed decisions about how long the vehicles should be allowed to perform tasks before returning for maintenance.

This chapter focuses on the health management problem at the task assignment level, developing a feedback mechanism for the performance model used by the task assignment algorithm. To date, the task assignment problem has been studied extensively [41, 42, 19, 46, 10]. However, most of the work done to date has used only a static vehicle performance model, making it difficult for these approaches to adapt to unexpected changes, such as sensor failures, during the course of the mission. The goal of this chapter is to develop a feedback loop that uses health state information to update the performance model in real-time. A general schematic for incorporating health feedback into the task assignment problem is shown in Figure 4-2. By taking this approach, previous work that has already developed task assignment algorithms can be leveraged and extended without requiring modification of the existing algorithm; its performance can be improved only by improving the quality of information used to make assignments.

4.1 Task Assignment Problem Definition

Formally, we define a task as a tuple (w_i, p_i), where w_i is the location of task i and p_i is the priority (or value) of task i. The tasks may be known to the planning system beforehand, but more commonly, they are generated in real-time as the mission progresses. For example, during a search and track mission, new tasks are generated at the predicted future locations of the target as new information about the target velocity and position is acquired. A task is called *active* if it has not been performed yet, and the set of currently active tasks is denoted by W.

The set of vehicles available to perform tasks is denoted by $V = \{v_1, \ldots, v_n\}$, where n is the total number of available vehicles. The vehicles originate from a base location x_{base}.

Given the above definition, the task assignment problem is to compute a mapping

$$T : V \rightarrow W$$

which assigns a task for each vehicle to visit. The goal is to compute the map T which minimizes the total weighted service time over all the tasks:

$$\min_T \sum_{(w_i, p_i) \in W} p_i \; t_i$$

where t_i is the wait time before task i is performed by any vehicle.

4.2 Selection of Performance Model

The selection of the performance model incorporating health state information about the vehicle is clearly an important aspect of the feedback design. The particular details of the model depend on the mission problem in question and vehicle hardware being used. However, there are a number of classes of general features that may be appropriate to include in a performance model:

Vehicle translational dynamics At the level of the task assignment problem, the vehicle dynamics are usually abstracted as being first-order with a maximum speed v_{max}. This abstraction allows the task assignment algorithm to capture important aspects of the vehicles' performance (in particular, how long they can be expected to take in order to reach a particular task), while being sufficiently simple to allow computational tractability. Recall that the trajectory planning and control levels below the task assignment level are responsible for carrying out those lower-level functions, allowing this simplification to be made. Note also that this is the model used in most of the previous work on task assignment.

Propulsion system state The vehicle propulsion system may be abstracted as an entity that provides the ability to move at the maximum speed v_{max} above. Health feedback about the propulsion system may dynamically modify v_{max} to reflect the state of the system. For example, knowledge of a failing motor may cause v_{max} to decrease from its nominal value.

Fuel state Knowledge of the fuel state of the vehicle is important in order to be able to estimate the remaining useful flight time of the vehicle. The performance model should include an estimator that performs the remaining flight time calculation based on fuel remaining, average fuel consumption rates, and perhaps other environmental factors. Use of this information allows the task assignment algorithm to safely make assignments while ensuring that vehicles can return to base before running out of fuel.

Sensor states The current performance level of any sensing systems onboard the vehicle should be included in the model if they are required to carry out tasks. For example, if an onboard camera is to be used for a surveillance task, the state of the camera (quality of the video signal, etc) should be accounted for in the model.

Communication system state Communication with other vehicles is often a requirement to enable vehicles to coordinate their actions with each other or relay messages to a distant ground station. Accounting for a vehicle's current estimated transmission and reception distances may allow the tasking system to avoid sending a vehicle to a location where it will be out of communication range.

4.3 Example: Modification of RHTA to Include Health Feedback

For the purposes of illustration, an example of incorporating a simple health feedback loop in the Receding Horizon Task Assignment (RHTA) algorithm developed in [27] is presented here.

Briefly, the RHTA algorithm works as follows (for more details, see Algorithm 2.3.1 in [27]). Given the set of tasks W and distances $d(i,j)$ between tasks, RHTA enumerates all possible task sequences, or *petals*, P_{vj} up to a specified length n_c. The cost of each petal is estimated as

$$S_{vp} = \sum \lambda^{T_{ip}} s_{wd} \tag{4.1}$$

where T_{ip} is the time task i is completed in petal p, s_{wd} are the task values, and λ is a time discount factor. Given the values of all the petals S_{vp}, RHTA solves the following optimization problem to select the optimal petal for each UAV:

$$\max J = \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} S_{vp} x_{vp} \tag{4.2}$$

$$\text{subj. to} \quad \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} A_{vpi} x_{vp} \leq 1, \quad x_{vp} \in \{0, 1\} \tag{4.3}$$

$$\sum_{p=1}^{N_{vp}} x_{vp} = 1, \ \forall \, v \in \{1, ..., N_v\} \tag{4.4}$$

Here, x_{vp} is a binary variable which is equal to 1 if the p^{th} petal is selected and 0 if not, and A_{vpi} equals 1 if task i is visited by vehicle v in petal p and 0 otherwise.

In the example, health state information is represented by adding a fuel state to the vehicle model. In this case, the fuel model is straightforward:

- The vehicle's fuel level f_i decreases at a constant rate k_{fuel} anytime the vehicle is flying.

- If f_i reaches zero before the vehicle refuels, the vehicle crashes and is lost.

- In addition, the occurrence of failures is modeled as a Poisson process with time intensity ρ_f; when a failure occurs, the rate of fuel burn increases to $k_{fuel,failure} > k_{fuel}$. Thus, this failure mode increases the rate at which fuel is burned (and thus decreases the time a vehicle can complete tasks).

The RHTA algorithm was extended to the health information embedded in the fuel state to the vehicle model. This was accomplished by including an estimate of each vehicle's operational radius, which is defined here as $r_i \equiv v_{max} \frac{f_i}{k_{fuel}}$. The quantity r_i represents the maximum distance a vehicle can fly given its current fuel state, before running out of fuel. This information can be used to effectively prune the list of petals that RHTA considers in order to ensure that the vehicle can always safely return to base before its fuel is exhausted. More specifically, the following pruning criterion was added to the RHTA algorithm:

For every petal under consideration, reject the petal if

$$L_i + d(w_{n_c}, x_{base}) \geq r_i$$

Here, $d(w_{n_c}, x_{base})$ represents the normal Euclidean distance between the last waypoint in the petal and the base, and

$$L_i = d(v, w_1) + \sum_{j=2}^{n_c} d(w_{j-1}, w_j)$$

is the total length of the petal. The pruning criterion rejects a petal if the length of the petal plus the distance from the terminal waypoint w_{n_c} to base is greater than the current operational radius of the vehicle. This ensures that the vehicle only visits waypoints that allow it to return safely to base.

With this extension, RHTA will assign a vehicle to return to base when every possible permutation of waypoints is rejected by the pruning criterion. Thus, this method provides a simple rule that determines when a vehicle should return to base for refueling since it cannot safely service any of the remaining tasks. Note that this method can create some problems if the above rule is followed too strictly since too many vehicles may be sent back to base unnecessarily (i.e. when they still have large operational radii) if there are few or no active tasks. This problem can be solved by inserting artificial *loiter tasks* $(w_{\text{loiter}}, p_{\text{loiter}})$ into W. These tasks are treated in the same way as real tasks by the RHTA algorithm, but their purpose is to force the vehicles to remain in advantageous areas.

4.4 Simulation Results

A multi-vehicle mission simulation was developed to test the performance of the nominal RHTA algorithm against the extended algorithm with health feedback. The simulation includes a base location and a number of vehicles (20 vehicles were used in the following tests), as well as a mechanism to randomly generate tasks and vehicle failures. The simulation runs RHTA to repeatedly assign tasks to vehicles.

Two performance metrics are calculated in the simulation: the average time it took to service each task; and how many vehicles were lost during the mission (where vehicle loss occurs when a vehicle runs out of fuel before returning to base).

In the first test, RHTA in its original form was run. The simulation test was as follows. All 20 vehicles started at the base location. Tasks were generated randomly, and as they appeared in the system, they were assigned to vehicles using RHTA. As the vehicles flew toward their assigned targets, they were subject to randomly generated vehicle failures which decreased their remaining flight time as described above. In its original form, RHTA does not account for these failures, so the vehicles would continue toward their assigned targets, even though they might run out of fuel and crash before they arrived. If this occurred, the original form of RHTA would then assign a new vehicle to carry out the task of the crashed vehicle. From Figure 4-3, it can be seen that the nominal performance of RHTA results in an average service time of 21.3 sec, and a vehicle loss rate of 25%.

In the second test, the modified form of RHTA was run, where the modified form of RHTA *proactively* recalls failed vehicles to base while quickly reassigning a new, healthy vehicle to the task, using the idea of the operational radius discussed above. From Figure 4-4, it is clear that that the modified RHTA provides a smaller average service time due to its proactive reassignment behavior. The reduction in service time is about 18%, which is fairly significant considering that the speed of the vehicles has not been changed, only the way they are assigned. In addition, the vehicle loss rate is significantly reduced because failed vehicles are returned to base instead of continuing toward their assigned tasks. Note that vehicle loss can still occur if a failure occurs sufficiently far from base so that the vehicle is unable to travel the distance to base even if it begins returning immediately.

Flight tests of the modified RHTA algorithm were also carried out using real hardware on the MIT Real-time indoor Autonomous Vehicle test ENvironment (RAVEN). These results are presented in Chapter 5.

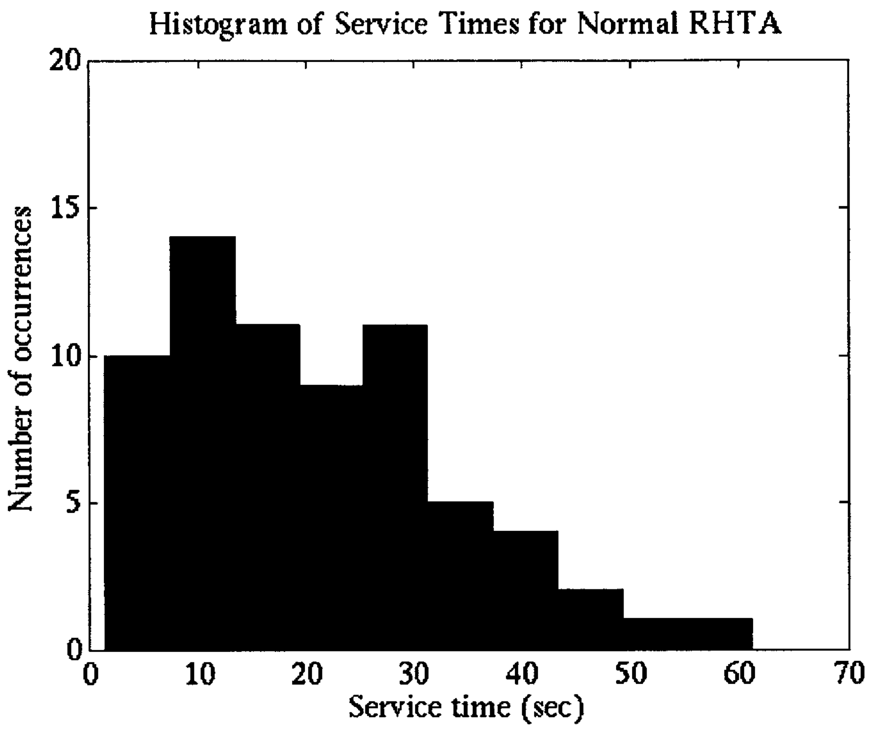

Figure 4-3: Simulation results (normal RHTA) – Median service time: 18.8 sec, Average service time: 21.3 sec, Vehicles lost: 5 of 20 (25%)

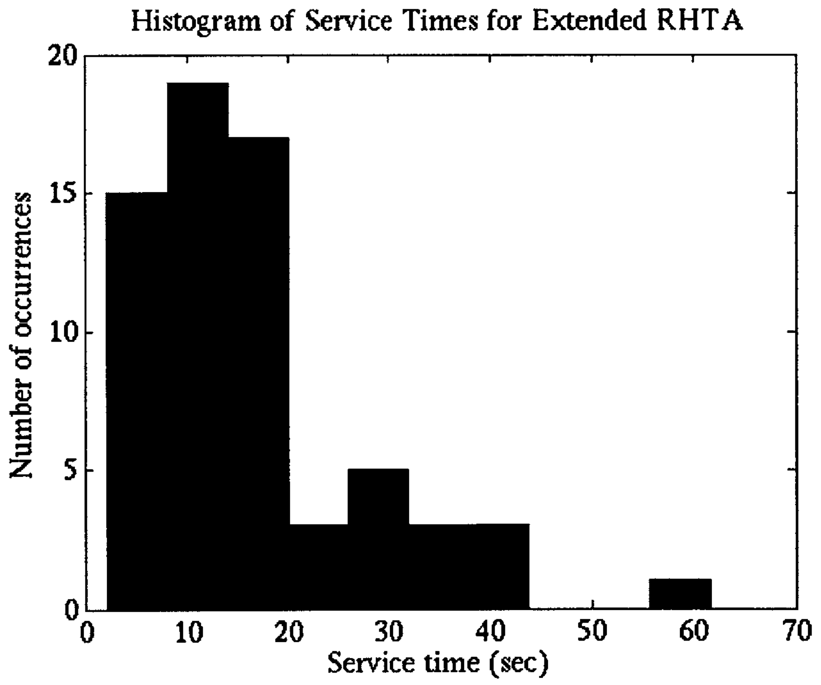

Figure 4-4: Simulation results (extended RHTA) – Median service time: 14.0 sec, Average service time: 17.4 sec, Vehicles lost: 1 of 20 (5%)

Chapter 5

Experimental Results

The main goal of this thesis is to demonstrate capability to carry out long-duration, multiple-target search and track missions. To achieve this goal, a number of flight experiments were carried out on the MIT RAVEN platform. These experiments were designed to test the performance of the cooperative vision-based estimation system and the task assignment health management feedback in a number of real flight conditions.

5.1 Real-Time 3D Operator Interface

In order to carry out the desired flight experiments, a real-time, 3D operator interface for the RAVEN system was developed. The operator interface was designed to fulfill a number of functions:

Data visualization The interface provides the ability for a human operator to easily visualize the state of the experiment both in real-time and during post-processing analysis of data collected. Because the interface communicates with the rest of the RAVEN system in the same way as the vehicle controllers, adding or subtracting data to be displayed is a simple process.

Environment awareness The interface can display additional features of the flight environment, such as obstacles and flight boundaries.

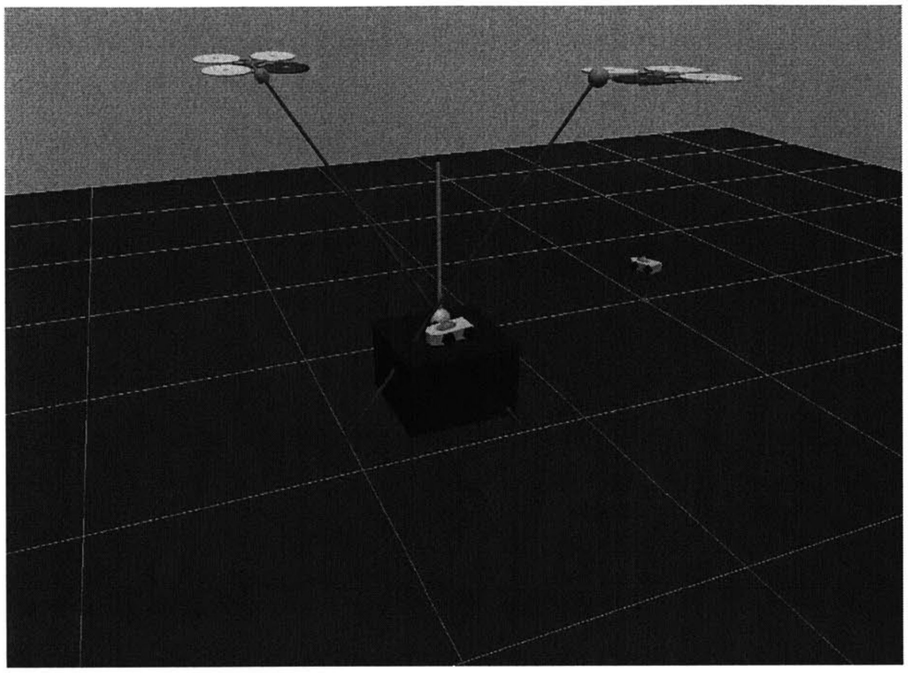

Figure 5-1: Operator interface showing flying vehicle and ground vehicle locations, vision system target vectors (red lines) and position estimate (yellow sphere with vertical column), and environmental obstacles (blue box)

Vehicle control The interface provides functionality to perform low-level control of any of the vehicles in the experiment via a number of straightforward point-and-click commands. These commands include take-off, land, and fly-to-waypoint. Because the interface shows the full state of the experiment, it is easy for a human to avoid collisions and other undesirable events if controlling the vehicles manually.

Experiment control The interface also provides high-level, easily customizable control of the experiment being conducted. For example, during a search and track mission, commands can be built in to allow a human operator to begin the test, stop the test, and log data of interest.

Flight area safety When being used in a complex experiment, the operator interface allows a human to function as a "safety pilot" for the experiment. The human operator can monitor the progress of the experiment as it runs. If the experiment is proceeding normally, the human can allowing it to run fully autonomously, while if a situation develops that could result in a crash, the human can press a single button to stop the test and/or land the vehicles.

The operator interface was developed in the Python programming language [40]. The Visual Python 3D programming module was used to implement the 3D graphics in the interface [51]. A screenshot of the operator interface is shown in Figure 5-1, which demonstrates a number of features of the software, including visualization of vehicle states as well as vision tracking data and environmental obstacles.

5.2 Flight Experiments

5.2.1 Cooperative Tracking of a Ground Vehicle

Results validating the performance of the cooperative, vision-based estimation algorithm for estimating the state of both stationary and maneuvering vehicles on the ground were presented in Chapter 3. Further experiments were carried out to incorporate active tracking into the estimation problem, so that the observing vehicles would move along with the target vehicle so as to keep the target in view at all times. In these experiments, two UAVs were commanded to detect a ground vehicle and estimate its state, consisting of its location x_{target} and velocity v_{target}. Figure 5-2 shows a ground vehicle with its estimated position and velocity as calculated by the vision system. Using the state information, the predicted location of the target a short time Δt in the future was calculated using the simple formula

$$x_{predicted} = x_{target} + v_{target}\Delta t \tag{5.1}$$

For these experiments, Δt was chosen to be 1 second. This time was chosen since the target vehicle was able to maneuver and change its velocity on a time scale of several seconds, so 1 second provided a good balance between allowing the tracking vehicles to move to an advantageous position and not attempting to predict too far in advance, when the vehicle would have a chance to maneuver away.

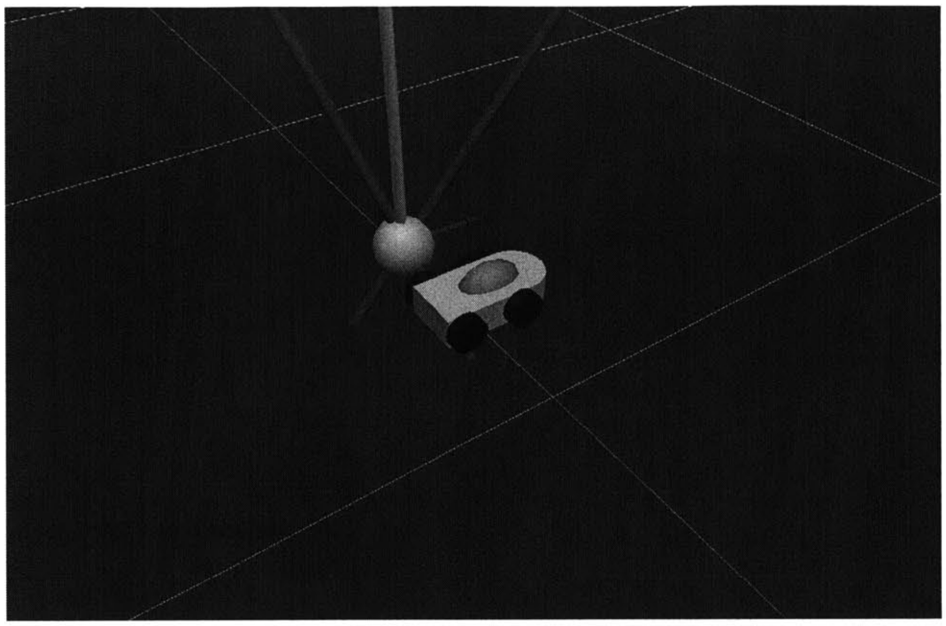
Figure 5-2: Estimation of target vehicle position (yellow sphere) and velocity (red arrow) for use in active tracking

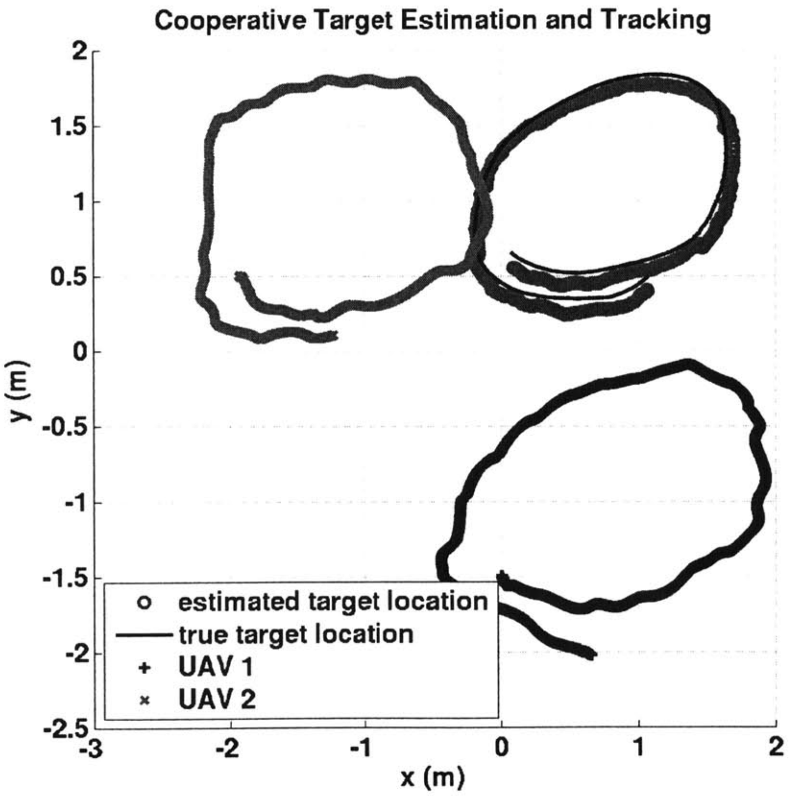

Figure 5-3: Results of cooperative estimation and active tracking of a ground vehicle

Once the predicted location $x_{predicted}$ was known, the two observing vehicles were commanded to fly to positions that would keep $x_{predicted}$ in their field of view. In this case, the positions were chosen to keep one UAV two meters south of the $x_{predicted}$ and the other UAV two meters west of the $x_{predicted}$.

Results of a typical tracking experiment are shown in Figure 5-3. The results show that the UAVs were able to estimate the position of the ground vehicle well (within about 5cm) even while they were moving cooperatively in order to keep the vehicle in the field of view of both UAVs. Note that in this experiment, the ground vehicle moved enough that it would have been outside the field of view of both UAVs at times had the UAVs not moved along with it.

5.2.2 Cooperative Tracking of a Flying Vehicle

To further extend the results of Chapter 3, more experiments were carried out with the goal of tracking a flying vehicle using the same estimation algorithm. The flying vehicle case is interesting for a number of reasons:

- Since a flying vehicle is not constrained to move along the surface of the earth, techniques that involve estimating the position of the vehicle using a single measurement combined with knowledge of the height of the local terrain cannot be applied. In comparison, the cooperative approach presented in this thesis can be used without modification in the flying vehicle case.

- Many types of flying vehicles are inherently more maneuverable than ground-based vehicles, making it more difficult to estimate their position and velocity.

- Since flying vehicles are free to move in three dimensions, the problem of tracking a flying vehicle (in other words, moving the observation vehicles along with the target to keep it in view) is more difficult. In the case of flying vehicles observing a ground vehicle, the flying vehicles may simply fly high above the ground vehicle, making it easy to keep the target in view.

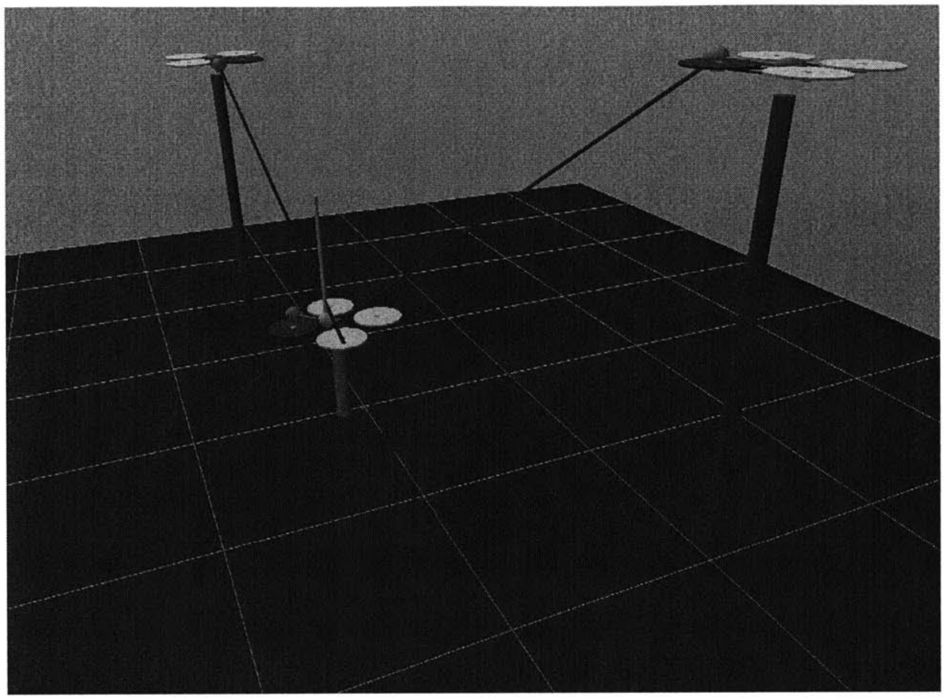

Figure 5-4: Cooperative tracking of a flying vehicle

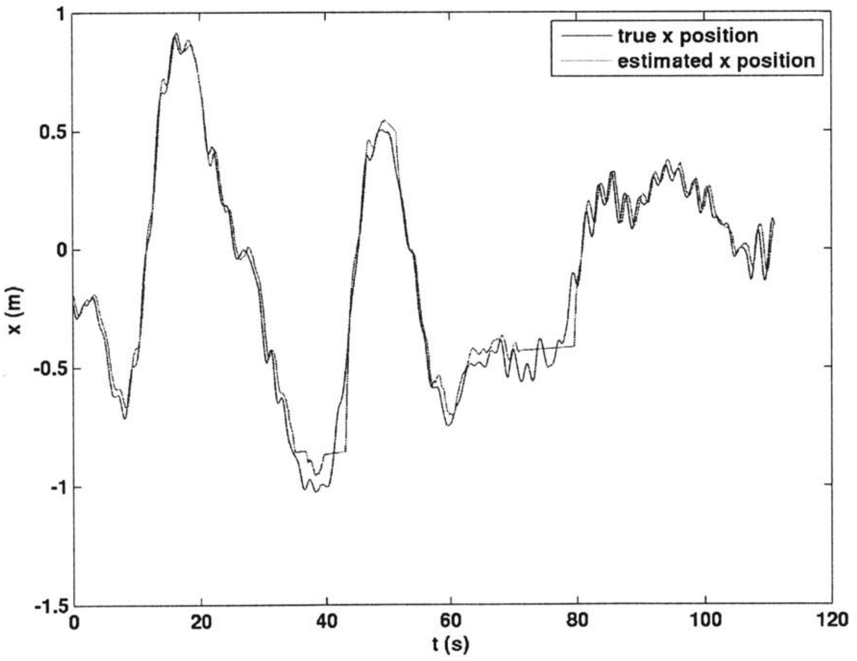

Figure 5-5: Flying vehicle tracking results. Mean x estimation error: 0.0312m. Standard deviation: $\sigma_x = 0.0192$m

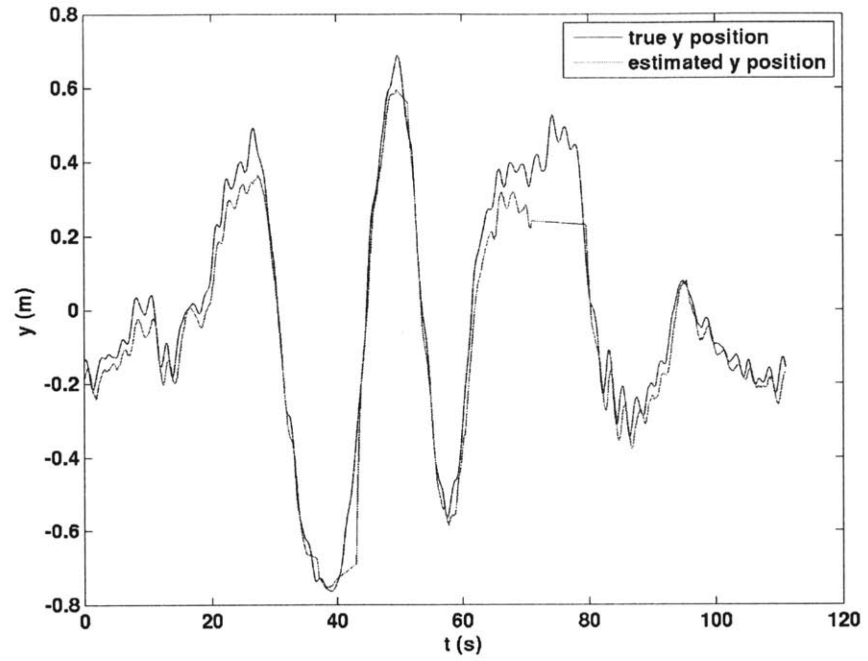

Figure 5-6: Flying vehicle tracking results. Mean y estimation error: -0.0501m. Standard deviation: $\sigma_y = 0.0417$m

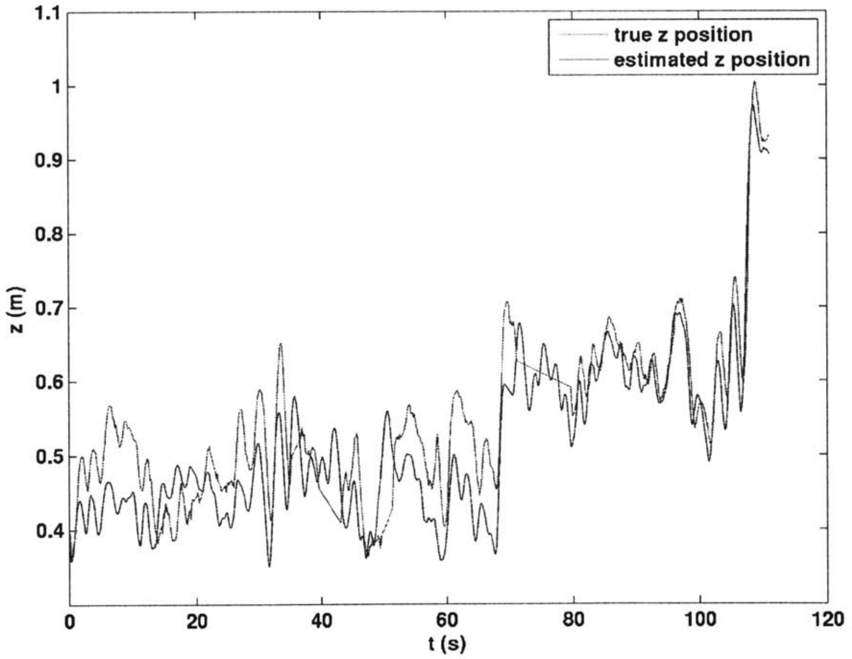

Figure 5-7: Flying vehicle tracking results. Mean z estimation error: 0.0557m. Standard deviation: $\sigma_z = 0.0318$m

In this experiment, two observing quadrotors were commanded to track a target quadrotor that was maneuvering in three dimensions in the flight area. Figure 5-4 shows the experiment setup. Results of the experiment are shown in Figures 5-5, 5-6, and 5-7. The data reveals that the two observing quadrotors were able to track the target quadrotor in all three dimensions with a high degree of accuracy, even while the target quadrotor was maneuvering. Overall, the mean estimation error and variance were 0.0811m and 0.0558m, respectively. Note that during this test, there were two brief time periods around $t = 40s$ and $t = 75s$ where the target quadrotor moved to a location that could not be tracked by the other quadrotors due to safety constraints that prevent the vehicles from moving outside a designated flight area in order to prevent collisions with walls and other obstacles. Thus, the estimator could not provide data during those periods, resulting in the straight line segments in Figures 5-5, 5-6, and 5-7.

5.2.3 Area Search

Another experiment was carried out to verify that a single quadrotor could search for and detect multiple target objects distributed over a large flight area. For this experiment, two R/C trucks were positioned in separate parts of the RAVEN flight space, and a single quadrotor was commanded to fly a search pattern through the space and detect as many object as it could.

Results of the experiment are shown in Figure 5-8. The figure shows a time-lapse history of the vehicle's progress as it moves around the search area; lighter shaded quadrotor images are more recent. The two trucks were detected by the quadrotor at different times throughout the search, confirming the system's ability to find the targets of interest in the search area.

5.2.4 Persistent Search and Track

A final set of experiments incorporating all aspects of the work presented thus far was conducted to demonstrate a complete, fully autonomous, persistent search and

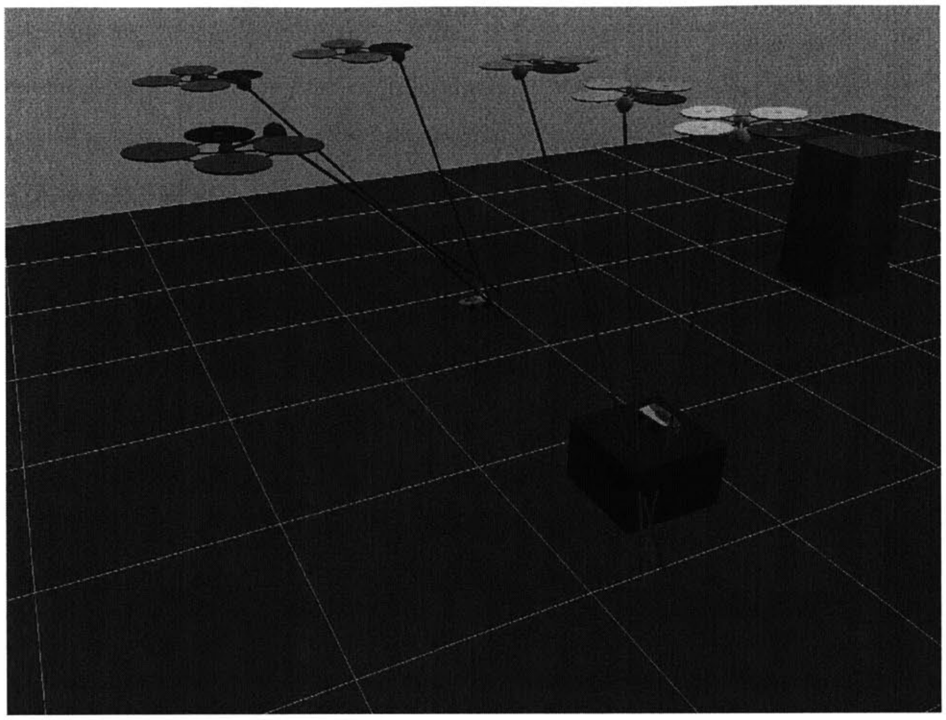

Figure 5-8: Area search: detecting multiple targets

track mission on the RAVEN platform. In these experiments, the mission goals were to search for, detect, estimate, and track an unknown number of ground vehicles in a predefined search region. Furthermore, the mission was to be carried out over a period of time longer than the flight endurance of the UAVs being used (around 12 minutes), necessitating the coordination of multiple UAVs coming in and out of the flight area as required to maintain coverage. Finally, active health monitoring was a requirement in order to detect and adapt to potential vehicle camera failures during the test.

In order to carry out the mission, the vision estimation system was combined with the modified RHTA algorithm presented in Chapter 4. Furthermore, the RHTA tasking system was interfaced to an autonomous mission system (see [32]) that employed battery monitors to estimate the time of flight remaining for each UAV in the search area and handled requests by the tasking system to activate vehicles for use in the search or tracking activities.

The experiment setup is shown in Figure 5-9. Three UAVs are initially stationed

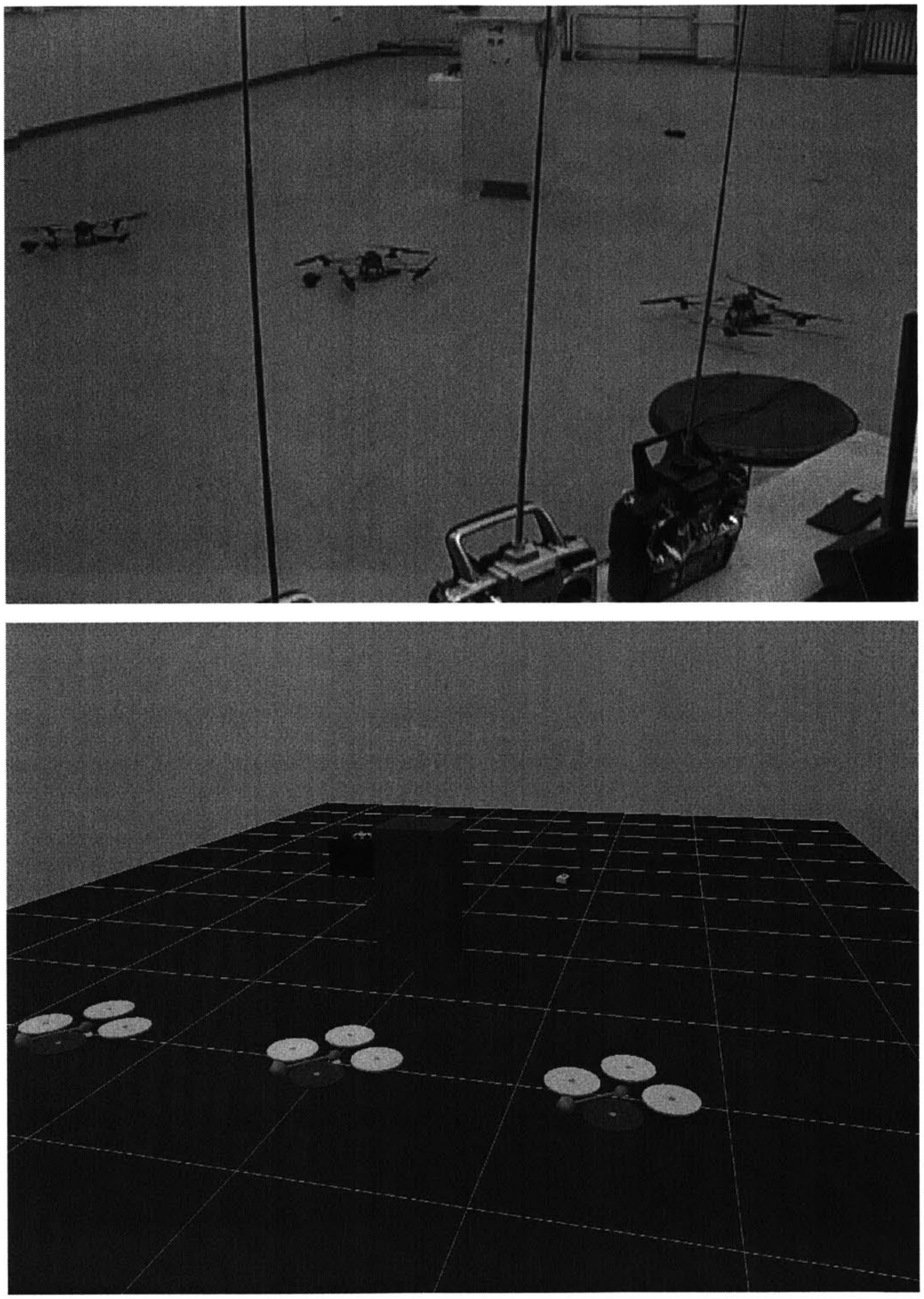

Figure 5-9: Persistent search and track mission setup

at their base location at the far north end of the flight area, while two ground vehicles are positioned at random locations in the southern region. For these experiments, one of the vehicles was positioned on top of a box, while the other was located on the ground and could be remotely controlled to move. Note that the existence of a concrete pillar in the center of the flight area creates an obstacle that the system must account for in planning safe trajectories for the vehicles.

The progression of the mission is as follows:

1. At the beginning of the test, the tasking system requests a single UAV from the mission system.

2. Once the requested UAV is airborne, the tasking system commands this UAV to begin an area search (as described in Section 5.2.3 and shown in Figure 5-8). During this initial detection phase, the UAV keeps track of how many distinct targets it has detected so far and stores them in a target list. The detection phase lasts for two minutes.

3. After the detection phase ends, the tasking system requests another UAV from the mission system.

4. Once the second UAV is airborne, the system enters the tracking phase. The tasking system commands the second UAV into the search area so that there are now two UAVs in the area. Together, these two UAVs sequentially visit each location in the target list found during the detection phase. The UAVs spend one minute at each location before moving on to the next. If there is a target at the given location when the UAVs arrive, they track the target by moving with it using the strategy outlined in Section 5.2.1. Additionally, although the tracking logic is designed to prevent collisions between the vehicles, a potential function based method is used to ensure an additional level of safety. If a UAV comes too close to another UAV or an obstacle in the environment, it is repelled away by seeking to move to an area of lower potential.

5. At any point in the mission, the tasking or mission systems may determine that

a particular UAV needs to return to base. The reason for this may be either that the UAV is getting low on battery lifetime remaining, or that the UAV's camera has failed or is performing poorly. In either case, when a return-to-base condition is detected, the tasking system send a sequence of waypoints to the UAV to command it back to base. Once at the base location, the mission system lands the UAV and schedules any necessary refueling or maintenance. At the same time, another UAV is launched and sent to the search area. In this manner, the mission is able to continue as UAVs cycle in and out.

6. The mission continues until a preset mission time expires or the human operator stops the mission. For these experiments, the mission time was 20 minutes.

Several results from the experiment are shown in Figures 5-10–5-13. Figure 5-10 shows the detection phase of the mission, where the first UAV has detected the presence of two ground vehicles. Figure 5-11 shows an early segment of the tracking phase, where the two UAVs were estimating the position of the eastern ground vehicle. Finally, Figures 5-10 and 5-11 show a time-lapse of later parts of the tracking phase in which the two UAVs were actively tracking the western ground vehicle, which was moving.

At several points during the mission, UAVs were successfully changed out due to low battery states. In addition, a simulated camera failure during the tracking phase of the mission resulted in the failed vehicle returning to base and a replacement vehicle being sent out. Due to these correct system responses, the goals of the overall mission were able to be accomplished continuously over the course of the mission.

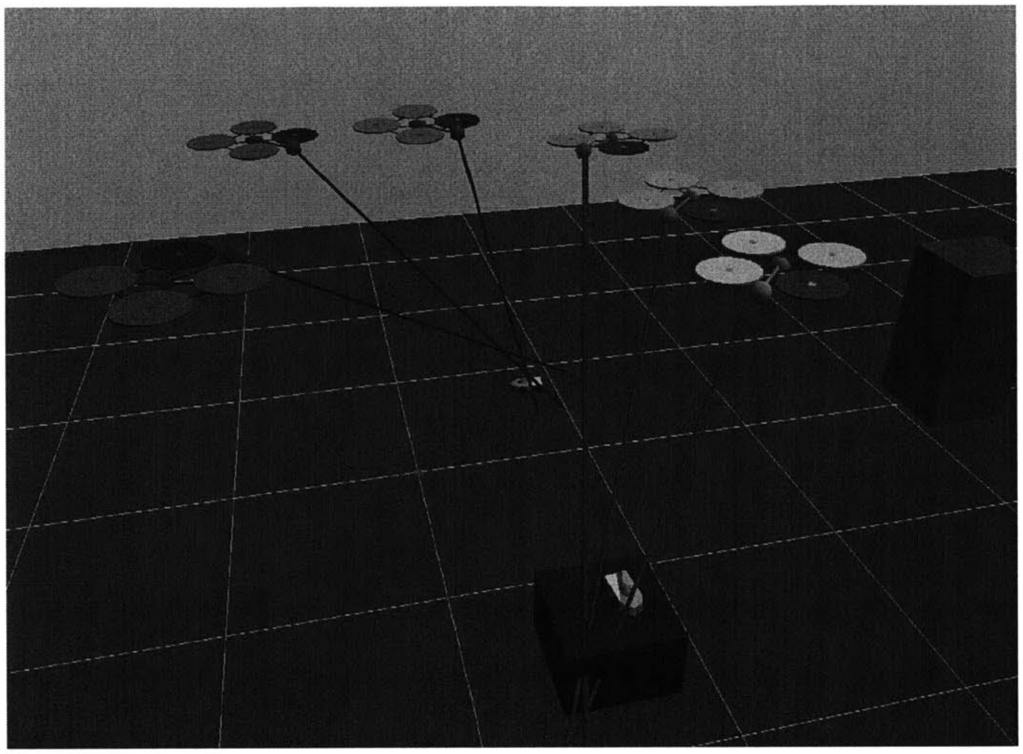
Figure 5-10: Persistent mission, detection phase

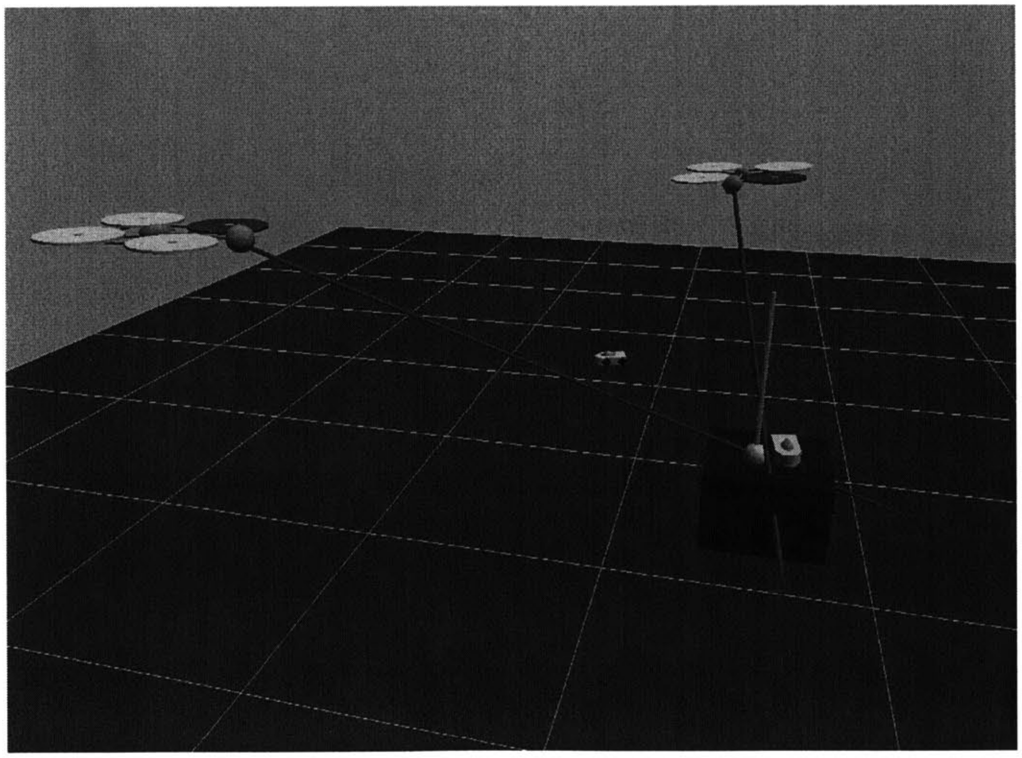
Figure 5-11: Persistent mission, estimation of a stationary ground vehicle

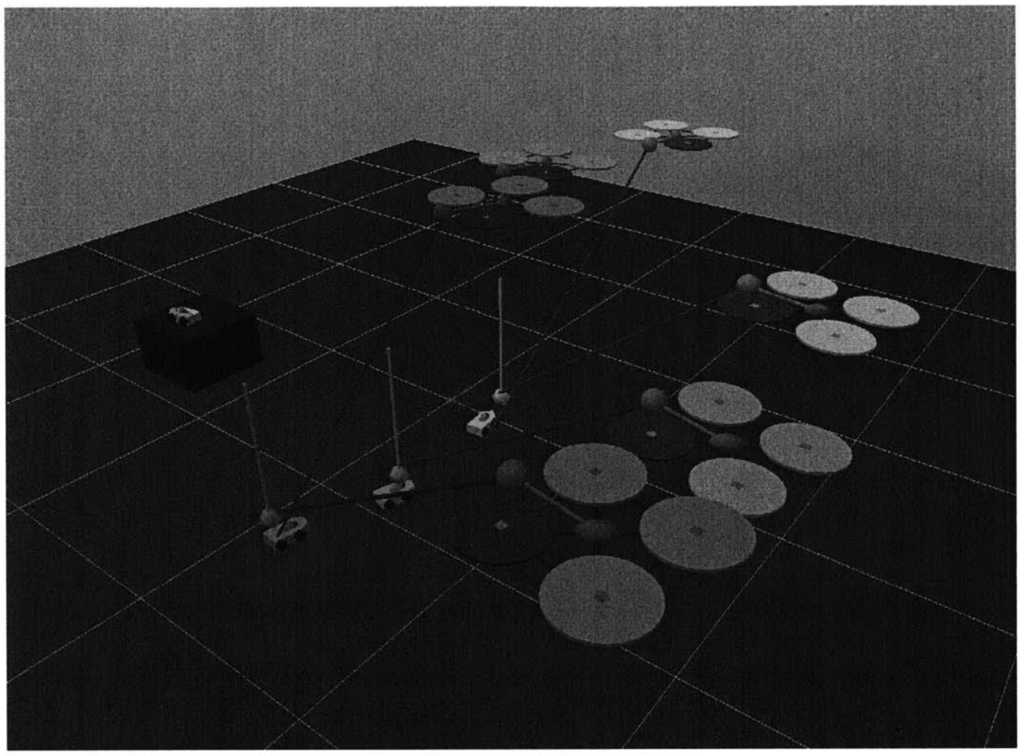
Figure 5-12: Persistent mission, tracking of a moving ground vehicle

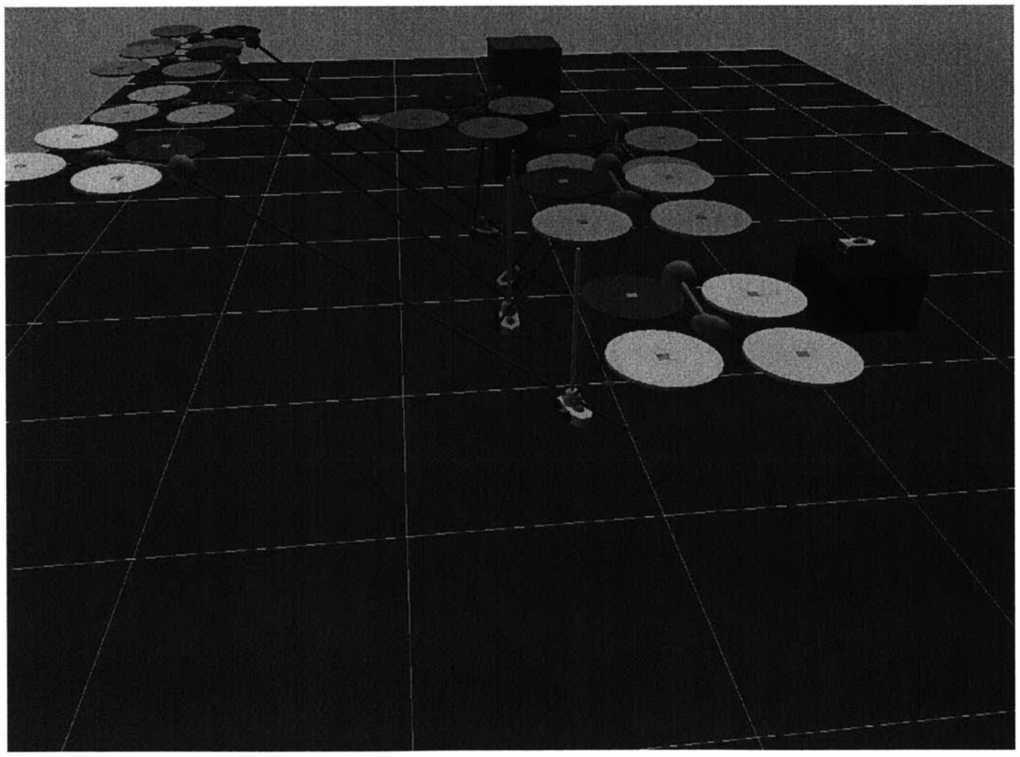
Figure 5-13: Persistent mission, tracking of a moving ground vehicle

Chapter 6

Conclusion

6.1 Performance Criteria Summary

This thesis has presented a framework for carrying out persistent search and track missions using multiple, cooperating UAVs. These missions present a number of challenges, and Figure 1-4 listed the criteria used to judge the success of the framework in addressing these challenges. We now address these criteria and show that the work presented in this thesis satisfies each of them.

Search for and detect a target of interest, which may be moving in three dimensions, using one or more camera-equipped UAVs. This detection should be robust to noise in the video data, which may be present due to radio interference in the wireless transmission of the data or other factors.

Chapter 2 presented a method for achieving this objective through the use of on-board cameras and a number of video-processing techniques. In particular, the video stream is processed using a persistent object filter as a strategy for dealing with noise in the video stream. Tests indicated that the filter effectively detects targets even when they are temporarily obscured by noise, while at the same time rejecting false targets introduced by the noise. The ultimate output of the video-processing system is a bearing vector to the target from each camera-equipped UAV.

Estimate the quality of the video data from each UAV, so that the system can determine if a given UAV's camera is not performing well.

Chapter 2 also presented a method based on convolving the incoming images with a noise-detection kernel to estimate the quality of the video stream, which is able to detect poorly-functioning camera or receiver equipment.

Combine the sensor information from one or more UAVs to create an estimate of the position and velocity of the target, and use this information to predict the future position of the target.

Chapter 3 presented a cooperative estimation algorithm that uses the bearing vectors generated by the video-processing system to provide accurate estimates of the position and velocity of the target vehicle. The cooperative approach presented was shown to have a number of advantages over the non-cooperative approach, including greater robustness to failures, better tracking ability of maneuvering targets, and the lack of a fundamental observability problem present in the non-cooperative approach. Using the information provided by the estimator, the future location of the target could be predicted.

Use a task assignment algorithm to assign a set of UAVs to locations that will keep the target in view, given the predicted position of the target. In addition to simply ensuring that there are UAVs assigned to the proper locations, the task assignment algorithm should account for the health state of the vehicles, swapping out underperforming vehicles and those that must return to base for refueling.

Chapter 4 presented a framework for integrating health management information into the overall mission architecture of an autonomous UAV system. The health management information can be specialized to the type of vehicle and sensors in use. In the flight experiments presented in Chapter 5, the health state was selected to be the flight time remaining and current camera performance of each UAV. The flight experiments utilized the target state estimate to predict the future location of the target and command the tracking UAVs to move, ensuring that the target remained in the field of view of the UAVs. This tracking worked well even in the case of a rapidly maneuvering, flying target. Furthermore, the health management framework functioned as expected, commanding a UAV that had experienced a simulated camera failure to return to base and sending a replacement UAV to continue the mission.

Continue the search and track mission for an extended period of time. Here *extended* means a time span longer than the flight time of individual UAVs, thus necessitating the swapping out of vehicles in order to accomplish the mission.

The flight experiments in Chapter 5 were conducted over the span of several flight times of a single UAV. The experiments demonstrated the successful swap-out of vehicles that were running low on fuel, allowing the mission objectives to continue to be accomplished as new vehicles entered the mission area.

6.2 Future Work

The framework and techniques developed in this thesis have led to the successful demonstration of a long-duration search and track mission under laboratory conditions. There are a number of ways that this work could be extended for eventual use in a real, in-the-field autonomous UAV system. First, the image processing algorithms used could be extended to deal with the background clutter that would be present in video data from a UAV flying above a forest in a search and rescue mission, for example. Second, more health state data could be incorporated into the health management framework. This would be especially valuable in cases where the UAVs in use were larger and more sophisticated, and therefore had more health state variables to track. Finally, the computing and communications hardware used for these laboratory experiments could be modified to be carried onboard the UAVs, which would be important for implementation in a system designed to be used over large distances where communication bandwidth between UAVs and their ground station is limited. Further investigation of these issues could lead to highly autonomous, robust UAV systems that can reliably perform very useful, real-world missions over long periods of time with little or no human intervention.

Appendix A

Derivation of the Pseudo-Intersection Formulae

The full derivation of the optimization problem presented in Section 3.2 is given in this Appendix.

A.1 Problem Statement

There is a set $\mathcal{M}$ of observations, each consisting of an observation location $\mathbf{x}_i \in \mathbb{R}^3$, a unit-length direction vector to the target $\mathbf{d}_i \in \mathbb{R}^3$, and a weight $w_i \in \mathbb{R}$. Given $\mathcal{M}$, find the point $\mathbf{q} \in \mathbb{R}$ that minimizes the objective function

$$E(\mathbf{q}) = \sum_{i \in \mathcal{M}} w_i h_i(\mathbf{q}) \tag{A.1}$$

where $h_i(\mathbf{q})$ is the square of the minimum distance from $\mathbf{q}$ to the ray $\mathbf{l}_i(\lambda_i)$:

$$h_i(\mathbf{q}) \quad = \quad \min_{\lambda_i} ||\mathbf{q} - \mathbf{l}_i(\lambda_i)||^2 = \min_{\lambda_i} ||\mathbf{q} - (\mathbf{x}_i + \lambda_i \mathbf{d}_i)||^2 \tag{A.2}$$

A.2 Solution

First, the definition of $h_i(\mathbf{q})$ in Equation A.2 is expanded using a change of variable $\mathbf{p}_i \equiv \mathbf{q} - \mathbf{x}_i$:

$$\begin{aligned} h_i(\mathbf{q}) &= \min_{\lambda_i} ||\mathbf{q} - (\mathbf{x}_i + \lambda_i \mathbf{d}_i)||^2 && \text{(A.3)} \\ &= \min_{\lambda_i} ||\mathbf{p}_i - \lambda_i \mathbf{d}_i||^2 && \text{(A.4)} \\ &= \min_{\lambda_i} (\mathbf{p}_i - \lambda_i \mathbf{d}_i)^T (\mathbf{p}_i - \lambda_i \mathbf{d}_i) && \text{(A.5)} \\ &= \min_{\lambda_i} \mathbf{p}_i^T \mathbf{p}_i - 2\lambda_i \mathbf{d}_i^T \mathbf{p}_i + {\lambda_i}^2 \mathbf{d}_i^T \mathbf{d}_i && \text{(A.6)} \end{aligned}$$

Minimizing $h_i(\mathbf{q})$ with respect to λ_i gives

$$\frac{dh_i(\mathbf{q})}{d\lambda_i} = -2\mathbf{d}_i^T \mathbf{p}_i + 2\lambda_i \mathbf{d}_i^T \mathbf{d}_i = 0, \qquad \text{(A.7)}$$

so the optimal value $\lambda_i^\star$ is

$$\lambda_i^\star = \frac{\mathbf{d}_i^T \mathbf{p}_i}{\mathbf{d}_i^T \mathbf{d}_i} = \mathbf{d}_i^T \mathbf{p}_i \qquad \text{(A.8)}$$

since $\mathbf{d}_i^T \mathbf{d}_i = 1$. Substituting $\lambda_i^\star$ back into the expression for $h_i(\mathbf{q})$ gives

$$\begin{aligned} h_i(\mathbf{q}) &= ||\mathbf{p}_i - \mathbf{d}_i^T \mathbf{p}_i \mathbf{d}_i||^2 && \text{(A.9)} \\ &= (\mathbf{p}_i - \mathbf{d}_i^T \mathbf{p}_i \mathbf{d}_i)^T (\mathbf{p}_i - \mathbf{d}_i^T \mathbf{p}_i \mathbf{d}_i) && \text{(A.10)} \\ &= \mathbf{p}_i^T \mathbf{p}_i - 2(\mathbf{d}_i^T \mathbf{p}_i)^2 + (\mathbf{d}_i^T \mathbf{p}_i)^2 \mathbf{d}_i^T \mathbf{d}_i && \text{(A.11)} \\ &= \mathbf{p}_i^T \mathbf{p}_i - 2(\mathbf{d}_i^T \mathbf{p}_i)^2 + (\mathbf{d}_i^T \mathbf{p}_i)^2 && \text{(A.12)} \\ &= \mathbf{p}_i^T \mathbf{p}_i - (\mathbf{d}_i^T \mathbf{p}_i)^2 && \text{(A.13)} \\ &= (\mathbf{q} - \mathbf{x}_i)^T (\mathbf{q} - \mathbf{x}_i) - (\mathbf{d}_i^T (\mathbf{q} - \mathbf{x}_i))^2 && \text{(A.14)} \\ &= \mathbf{q}^T \mathbf{q} - 2\mathbf{q}^T \mathbf{x}_i + \mathbf{x}_i^T \mathbf{x}_i - (\mathbf{d}_i^T \mathbf{q} - \mathbf{d}_i^T \mathbf{x}_i)^2 && \text{(A.15)} \end{aligned}$$

Note that the derivative of $h_i(\mathbf{q})$ with respect to $\mathbf{q}$ is

$$\frac{dh_i(\mathbf{q})}{d\mathbf{q}} = 2\mathbf{q} - 2\mathbf{x}_i - 2(\mathbf{d}_i^T \mathbf{q} - \mathbf{d}_i^T \mathbf{x}_i)\mathbf{d}_i \qquad \text{(A.16)}$$

Now, the optimality condition for $\mathbf{q}^\star$ can be found by differentiating $E(\mathbf{q})$ (Equation A.1) with respect to $\mathbf{q}$ and substituting the derivative formula in Equation A.16.

$$\begin{aligned}
\frac{dE(\mathbf{q})}{d\mathbf{q}} &= \frac{d}{d\mathbf{q}} \sum_{i \in \mathcal{M}} w_i h_i(\mathbf{q}) && \text{(A.17)} \\
&= \sum_{i \in \mathcal{M}} w_i \frac{dh_i(\mathbf{q})}{d\mathbf{q}} && \text{(A.18)} \\
&= \sum_{i \in \mathcal{M}} w_i (2\mathbf{q} - 2\mathbf{x}_i - 2(\mathbf{d}_i^T \mathbf{q} - \mathbf{d}_i^T \mathbf{x}_i)\mathbf{d}_i) && \text{(A.19)}
\end{aligned}$$

Therefore, $\mathbf{q}^\star$ satisfies

$$\sum_{i \in \mathcal{M}} w_i (\mathbf{q}^\star - \mathbf{x}_i - (\mathbf{d}_i^T \mathbf{q}^\star - \mathbf{d}_i^T \mathbf{x}_i)\mathbf{d}_i) = 0 \qquad \text{(A.20)}$$

Rearranging Equation A.20 yields

$$\begin{aligned}
\left(\sum_{i \in \mathcal{M}} w_i \right) \mathbf{q}^\star - \left(\sum_{i \in \mathcal{M}} w_i \mathbf{d}_i \mathbf{d}_i^T \right) \mathbf{q}^\star &= \left(\sum_{i \in \mathcal{M}} w_i \mathbf{x}_i \right) - \left(\sum_{i \in \mathcal{M}} w_i \mathbf{d}_i \mathbf{d}_i^T \mathbf{x}_i \right) && \text{(A.21)} \\
\left(\sum_{i \in \mathcal{M}} w_i \left(I - \mathbf{d}_i \mathbf{d}_i^T \right) \right) \mathbf{q}^\star &= \sum_{i \in \mathcal{M}} w_i \left(I - \mathbf{d}_i \mathbf{d}_i^T \right) \mathbf{x}_i && \text{(A.22)}
\end{aligned}$$

Here, $\mathbf{d}_i \mathbf{d}_i^T$ denotes the vector outer product. In this form, it is clear that Equation A.22 is a linear system in $\mathbf{q}^\star$,

$$\mathcal{A}\mathbf{q}^\star = \mathbf{b} \qquad \text{(A.23)}$$

where

$$\mathcal{A} = \sum_{i \in \mathcal{M}} w_i \left(I - \mathbf{d}_i \mathbf{d}_i^T \right) \qquad \text{(A.24)}$$

and

$$\mathbf{b} = \sum_{i \in \mathcal{M}} w_i \left(I - \mathbf{d}_i \mathbf{d}_i^T \right) \mathbf{x}_i \qquad \text{(A.25)}$$

Therefore, the optimal solution $\mathbf{q}^\star$ is given by

$$\mathbf{q}^\star = \mathcal{A}^{-1}\mathbf{b} \tag{A.26}$$

Bibliography

[1] B. Bethke, M. Valenti, J. How. Cooperative Vision Based Estimation and Tracking Using Multiple UAVs. In *Proceedings of the Conference on Cooperative Control and Optimization*, Gainesville, FL, January 2007.

[2] B. Coifman, M. McCord, R.G. Mishalani, M. Iswalt, Y. Ji. Roadway traffic monitoring from an unmanned aerial vehicle. *Intelligent Transport Systems, IEEE Proceedings*, 153:11–20, 2006.

[3] N. Gordon B. Ristic, S. Arulampalam. *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, Boston, MA, 2004.

[4] C. Jauffret, D. Pillon. Observability in passive target motion analysis. *IEEE Trans. Aerospace and Electronic Systems*, 32:1290–1300, 1996.

[5] C. Schumacher and P. R. Chandler and S. J. Rasmussen. Task allocation for wide area search munitions via iterative network flow. In *Proceedings of the 2002 AIAA Guidance, Navigation and Control Conference*, Monterrey, CA, August 2002.

[6] C. Sharp, O. Shakernia, and S. Sastry. A Vision System for Landing an Unmanned Aerial Vehicle. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, volume 2, pages 1720–1727, 2001.

[7] D. Casbeer, S. Li, R. Beard, R. Mehra, T. McLain. Forest Fire Monitoring With Multiple Small UAVs. In *Proceedings of the American Control Conference*, Portland, OR, April 2005.

[8] Draganfly Innovations Inc. Available at http://www.rctoys.com, 2007.

[9] E. Fogel, M. Gavish. Nth-order dynamics target observability from angle measurements. *IEEE Trans. Aerospace and Electronic Systems*, AES24:305–308, 1988.

[10] E. Frazzoli, F. Bullo. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. In *IEEE Conference on Decision and Control*, December 2004.

[11] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, R. Sengupta. Vision-Based Road-Following Using a Small Autonomous Aircraft. In *Proceedings of the 2004 IEEE Aerospace Conference*, Big Sky, MT, March 2004.

[12] E. Wan, R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, Alta, Canada, October 2000.

[13] F. Caballero, L. Merino, , A. Ollero. A visual odometer without 3D reconstruction for aerial vehicles. Applications to building inspection . In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[14] G. Goebel. In the Public Domain: Unmanned Aerial Vehicles. http://www.vectorsite.net/twuav.html, April 2006.

[15] G. Tournier. Six Degree of Freedom Estimation Using Monocular Vision and Moire Patterns. Master's thesis, Massachusetts Institute of Technology, 2006.

[16] G. Tournier, M. Valenti, J. P. How, and E. Feron. Estimation and control of a quadrotor vehicle using monocular vision and moire patterns. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, August 2006.

[17] General Atomics Aeronautical Systems. Predator Unmanned Aerial Vehicle. Available at http://www.ga-asi.com/products/predator.php, 2007.

[18] Gentoo Foundation, Inc. Gentoo Linux. Available at http://www.gentoo.org/, 2007.

[19] A. E. Gil, K. M. Passino, and A. Sparks. Cooperative Scheduling of Tasks for Networked Uninhabited Autonomous Vehicles. In *Proceedings of the 2003 IEEE Conference on Decision and Control*, pages 522–527, Maui, HI, December 2003.

[20] Intel Corporation. OpenCV Blob Extraction Library. Available at http://opencvlibrary.sourceforge.net/cvBlobsLib, 2007.

[21] Intel Corporation. OpenCV Computer Vision Library. Available at http://www.intel.com/technology/computing/opencv/, 2007.

[22] J. Boskovic, R. Mehra. An Integrated Fault Management System for Unmanned Aerial Vehicles. In *2nd AIAA Unmanned Unlimited Conf. and Workshop and Exhibit*, San Diego, CA, September 2003.

[23] J.P.L. Cadre, O. Tremois. Bearings-only tracking for maneuvering sources. *IEEE Trans. Aerospace and Electronic Systems*, 34:179–193, 1998.

[24] L. Merino, F. Caballero, J. R. Martinez de Dios, A. Ollero. Cooperative Fire Detection using Unmanned Aerial Vehicles. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[25] L. Merino, F. Caballero, J. R. Martinez de Dios, J. Ferruz, A. Ollero. A Cooperative Perception System for Multiple UAVs: Application to Automatic Detection of Forest Fires. *Journal of Field Robotics*, 23:165–184, 2006.

[26] LifeView Inc. FlyVideo 300FM Video Capture Card. Available at http://www.lifeview.com/usa/html/products/PCITV/FlyVideo3000-w1.htm, 2007.

[27] M. Alighanbari. Task assignment algorithms for teams of UAVs in dynamic environments. Master's thesis, Massachusetts Institute of Technology, 2004.

[28] M. Alighanbari, J. How. Decentralized Task Assignment for Unmanned Aerial Vehicles. In *Decision and Control, 2005 and 2005 European Control Conference*, December 2005.

[29] M. Campbell, M. Wheeler. A Vision Based Geolocation Tracking System for UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, August 2006.

[30] M. Quigley, M. Goodrich, S. Griffiths, A. Eldgredge, R. Beard. Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[31] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, August 2006.

[32] M. Valenti, B. Bethke, J. How, D. Pucci de Farias, J. Vian. Embedding Health Management into Mission Tasking for UAV Teams. In *American Controls Conference*, New York, NY, June 2007.

[33] M. Valenti, D. Dale, J. How, and J. Vian. Mission Health Management for 24/7 Persistent Surveillance Operations. In *Submitted to the 2007 AIAA Guidance, Control and Navigation Conference*, Myrtle Beach, SC, August 2007.

[34] MIT Real-time indoor Autonomous Vehicle test ENvironment (RAVEN). RAVEN home page. Available at http://vertol.mit.edu/, 2007.

[35] National Aeronautics and Space Administration. Altus II Unmanned Aerial Vehicle. Available at http://www.nasa.gov/centers/dryden/news/FactSheets/FS-058-DFRC.html, 2007.

[36] National Aeronautics and Space Administration. Helios Prototype. Available at http://www.nasa.gov/centers/dryden/history/pastprojects/Erast/helios.html, 2007.

[37] Northrop Grumman Corporation. Global Hawk Unmanned Aerial Vehicle. Available at http://www.northropgrumman.com/unmanned/, 2007.

[38] O. Amedi, T. Kanade, and K. Fujita. A visual odometer for autonomous helicopter flight. *Robotics and Autonomous Systems*, 28:185–193, 1999.

[39] P. Shar, X.R. Li. A practical approach to observability of bearings-only target tracking. *Proc. SPIE*, 3809:514–520, 1999.

[40] Python Programming Language. Available at http://www.python.org/, 2007.

[41] R. W. Beard and T. W. McLain and M. A. Goodrich and E. P. Anderson. Coordinated Target Assignment and Intercept for Unmanned Air Vehicles. *IEEE Transactions on Robotics and Automation*, 18:911–922, 2002.

[42] A. Richards, J. Bellingham, M. Tillerson, and J. How. Coordination and Control of Multiple UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, CA, August 2002.

[43] S. Julier, J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997.

[44] S.C. Nardone, A.G. Lindgren, K.F. Gong. Fundamental properties and performance of conventional bearings-only target motion analysis. *IEEE Trans. Automatic Control*, 29:775–787, 1984.

[45] S.C. Nardone, M.L. Graham. A closed form solution to bearings-only target motion analysis. *IEEE Journal of Oceanic Engineering*, 22:168–178, 1997.

[46] C. Schumacher, P. R. Chandler, S. J. Rasmussen, and D. Walker. Task Allocation for Wide Area Search Munitions with Variable Path Length. In *Proceedings of the 2003 American Control Conference*, pages 3472–3477, Denver, CO, June 2003.

[47] T. McGee, R. Sengupta, K. Hedrick. Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[48] T.L. Song. Observability of target tracking with bearings-only measurements. *IEEE Trans. Aerospace and Electronic Systems*, 32:1468–1471, 1996.

[49] tvtime SourceForge project. Available at http://tvtime.sourceforge.net/, 2007.

[50] Vicon Company. Vicon Motion Capture Systems. Available at http://www.vicon.com/, 2007.

[51] Visual Python 3D Programming Module. Available at http://www.vpython.org/, 2007.

[52] V.J. Aidala, S.E. Hammel. Utilization of modified polar coordinates for bearings-only tracking. *IEEE Trans. Automatic Control*, AC-28:283–294, 1983.

[53] Y. Liu and J. B. Cruz, Jr. and A. G. Sparks. Coordinating networked uninhabited air vehicles for persistent area denial. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.