3-2004

Swarm Based Implementation of a Virtual Distributed Database System in a Sensor Network

Wen C. Lee

SWARM BASED IMPLEMENTATION

OF A VIRTUAL DISTRIBUTED DATABASE

SYSTEM IN A SENSOR NETWORK

THESIS

Wen Chian Lee

AFIT/GCE/ENG/04-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCE/ENG/04-06

SWARM BASED IMPLEMENTATION OF A VIRTUAL DISTRIBUTED DATABASE SYSTEM IN A SENSOR NETWORK

THESIS

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Wen Chian Lee, BS

March 2004

AFIT/GCE/ENG/04-06

SWARM BASED IMPLEMENTATION OF A VIRTUAL DISTRIBUTED DATABASE SYSTEM IN A SENSOR NETWORK

Wen Chian Lee, BS

Approved:

Dr Gary B Lamont (Chairman)	12 MAR 04 date
Dr (LtCol) Michael L Talbert (Member)	12 Mar 04 date
Dr Gilbert L Peterson (Member)	12 MAR 04 date
Dr Henry B Potoczny (Member)	March 12, 2004 date

Acknowledgments

I would like to express my sincere appreciation to my parents for their unwavering faith in me, their timely encouragement when I feel down, and their continuous support. I also thank my brother, whose ingenious humor always brings more than a smile on my face.

My special gratitude goes to Mary Jane McCormick, Office Manager of the Electrical and Computer Engineering Department. Her kindness and considerations have ensured that every step of my experience at AFIT both inside the classroom and outside has been smooth. Her seemingly all-knowing and ubiquitous ability has been a blessing. I would not have completed my graduate education at AFIT without her. I would like to thank my classmates, in particular, Daniel Swayne and his family, Jamie Carsten, Dan Newberry, Duane Sorgaard, and Wendell Fox. Together, because of them, even the bitterly cold winter at Ohio became bearable. At AFIT, a school where even the walls started to bleed at the ninth week into the quarter (Winter03 Blg640), I found fellow classmates were valuable assets in this academic challenging environment.

In addition, let me extend my gratitude to the technical support professionals, Jim Gray and Dave Doak, for their assistance in dealing with various technical issues; especially Jim Gray, whose editorial help contributed to the completion of this document.

Lastly, special thanks to my thesis advisor Dr. Gary B. Lamont and committee member Dr. Michael L. Talbert.

Table of Contents

Page

Acknowledgments iv

List of Figures viii

List of Tables ix

Abstract xi

I. Introduction 1

1.1 Sensors on UAVs 3

1.2 Ad Hoc Network 5

1.3 Research Goal and Objectives 6

1.4 Thesis Outline 8

II. Background 10

2.1 From Data Collection to Information Retrieval 10

2.2 Distributed Database on the UAV Communication Network 18

2.3 Why Distributed Database? 19

2.4 Distributed Processing 24

2.5 Image Query Processing Strategies 29

2.6 Summary 34

III. Query Design/Structures 35

3.1 Querying Physical Space in a Networked Environment 35

3.2 Capabilities of a Sensor Database 38

3.3 Sensor and Data Representation 39

3.4 Networked Sensor Database Design 42

3.5 Summary 47

IV. Implementation/Detailed Design of Experiments 49

4.1 Virtual Database 49

4.2 Query Semantics 51

4.3 Performance Measures 54

4.4 Experiment Methodology 57

4.5 Other Sensor Scenarios 65

4.6 Summary 66

V. Results and Analysis 67

5.1 Type of Query 67

5.2 Database Size 69

5.3 Query Results in Space and Time 71

5.4 Summary 76

VI. Conclusions and Future Work 78

6.1 Summary and Conclusions 78

6.2 Research Contribution 80

6.3 Future Work 81

Appendix A. Decentralized Sensor Fusion – Target Tracking and Target Recognition 83

Appendix B. Application of the COUGAR Sensor Database Project on Unmanned Aerial Vehicles 87

Appendix C. Simulation Input 91

C.1 Sensor Database Simulator Interface 91

C.2 Swarm Simulator Settings 92

Appendix D. Experiment Settings and Output 94

D.1 Query Complexity Test 94

D.2 Data Load Test 95

D.3 Effectiveness Test 96

Bibliography 101

List of Figures

Figure Page

Figure 1 Time line on the effect of information technology in war 2

Figure 2 OODA Loop 11

Figure 3 Micro Air Vehicle- Black Widow 23

Figure 4 A mathematical model of a wireless network 27

Figure 5 Process flow for image processing in ImageMap 30

Figure 6 Feature-based multisensor data fusion system 32

Figure 7 Graph of database system response time and database size 71

Figure 8 Graph of sensor density and number of observations 73

Figure 9 Graph of sensor density and number of reporting points 73

Figure 10 Graph of number of observations and time 75

Figure 11 Graph of number of reporting points and time 76

Figure 12 Sample simulation query output 92

List of Tables

Table Page

Table 1 UAV Construction Properties. 4

Table 2 Summary of approaches for formulating image queries 33

Table 3 Existing Wireless Systems Standards 38

Table 4 Experimental variable and metrics 60

Table 5 Parameters in computing system delay 61

Table 6 Query complexity and response time 68

Table 7 Query type and response time 69

Table 8 Data store size and response time 71

Table 9 Summary of observations made in 90 time units 72

Table 10 Summary of number of positions of observations in 90 time units 72

Table 11 Summary of observations made with 75 sensors 75

Table 12 Summary of number of points of observations made with 75 sensors 75

Table 13 Parameter setting in param.txt file for swarm simulations 93

Table 14 Parameter setting for distributed sensor database simulation in query complexity test 94

Table 15 Queries used in complexity test 94

Table 16 Query inputs in data load test 96

Table 17 50 random grid points on a 20x20 grid 96

Table 18 Runs for 15 sensors with 90 time units 97

Table 19 Runs for 30 sensors with 90 time units 97

Table 20 Runs for 45 sensors with 90 time units 98

Table 21 Runs for 60 sensor with 90 time units 98

Table 22 Runs for 75 sensors with 90 time units 98

Table 23 Runs for 75 sensors with 180 time units 99

Table 24 Runs for 75 sensors with 270 time units 99

Table 25 Runs for 75 sensors with 360 time units 100

Table 26 Runs for 75 sensors with 450 time units 100

Abstract

The deployment of unmanned aerial vehicles (UAVs) in recent military operations has received much media attention. Their success in carrying out surveillance and combat missions in sensitive areas has been trumpeted. An area of intense research has been on controlling a group of small-sized UAVs to carry out reconnaissance missions normally undertaken by large UAVs such as Predator or Global Hawk. A control strategy for coordinating the UAV movements of such a group of UAVs adopts the bio-inspired swarm model to produce autonomous group behavior.

This research proposes establishing a distributed database system on a group of swarming UAVs, providing for data storage during a reconnaissance mission. A distributed database system model is simulated treating each UAV as a distributed database site connected by a wireless network. In this model, each UAV carries a sensor and communicates to a command center when queried. Drawing equivalence to a sensor network, the network of UAVs poses as a dynamic ad-hoc sensor network.

The distributed database system based on a swarm of UAVs is tested against a set of reconnaissance test suites with respect to evaluating system performance. The design of experiments focuses on the effects of varying the query input and types of swarming UAVs on overall system performance. The results show that the topology of the UAVs has a distinct impact on the output of the sensor database. The experiments measuring system delays also confirm the expectation that in a distributed system, inter-node communication costs outweigh processing costs.

SWARM BASED IMPLEMENTATION OF A VIRTUAL DISTRIBUTED DATABASE SYSTEM IN A SENSOR NETWORK

I. Introduction

The wide scale employment of remotely controlled or autonomous unmanned aerial vehicles (UAVs) is no longer a science fiction. The development in both aeronautic control technology and wireless communications have made unmanned aerial vehicles that require little human supervision flying in the sky a reality. A true story reported by Richard J. Newman on Air Force Magazine Online is:

> Important parts of Operation Iraqi Freedom were carried out by remote control.
>
> In the first week of Gulf War II, a Marine reconnaissance team near Basra reported it was surrounded by enemy troops and in need of reinforcements. The quickest way in was by helicopter, but the nearby terrain was unfamiliar.
>
> Out went an urgent request for U-2 and Predator surveillance aircraft to scout possible landing zones.
>
> Five thousand miles away, at Langley AFB, Va., USAF Capt. Bob Lyons turned to the task. He and dozens of his colleagues had been set up in 27 chilly trailers lashed together to form a distributed ground station (DGS), which monitored minute details of the war. Lyons started redirecting a U-2 that was already airborne over Iraq. The U-2 got onto the scene and snapped its first pictures a mere 20 minutes after the original call for help.
>
> Intelligence experts at Langley and another base (unnamed here, at Air Force request) quickly analyzed the photos and then transmitted them via satellite to the combined air operations center (CAOC) in Saudi Arabia. There, US planners reviewed the images and began to designate landing zones and prepare for the mission.
>
> A few minutes later, Lyons helped direct a Predator unmanned aerial vehicle to the scene of the action. Specialists looking through the UAV's camera located the Marines and scanned the ground for signs of any Iraqi activity near the potential landing zones. The UAV relayed real-time video to Langley, the CAOC, and several other posts.
>
> The long-distance linkup paid off: Two hours after the first Marine SOS, reinforcements were on their way to the LZs. [45]

The key in information warfare is to integrate information collected from all sources (intelligence, diplomatic channel, reconnaissance activities, and mass media) and to have them available to the right person at the right time.

From experiences in recent encounters on the battlefield, the Department of Defense understands the importance of information superiority. Information and the technology that allows rapid information flow are recognized to "impact every facet of military operations [73]." Figure 1 shows the impact of information technology in war over time [43]. The flow of the right information to decision makers is critical in the overall command and control. As demonstrated in the excerpt above, information technology enables raw data to travel far across distance and hierarchy to be translated into useful information for the decision maker to initiate corresponding military maneuvers.

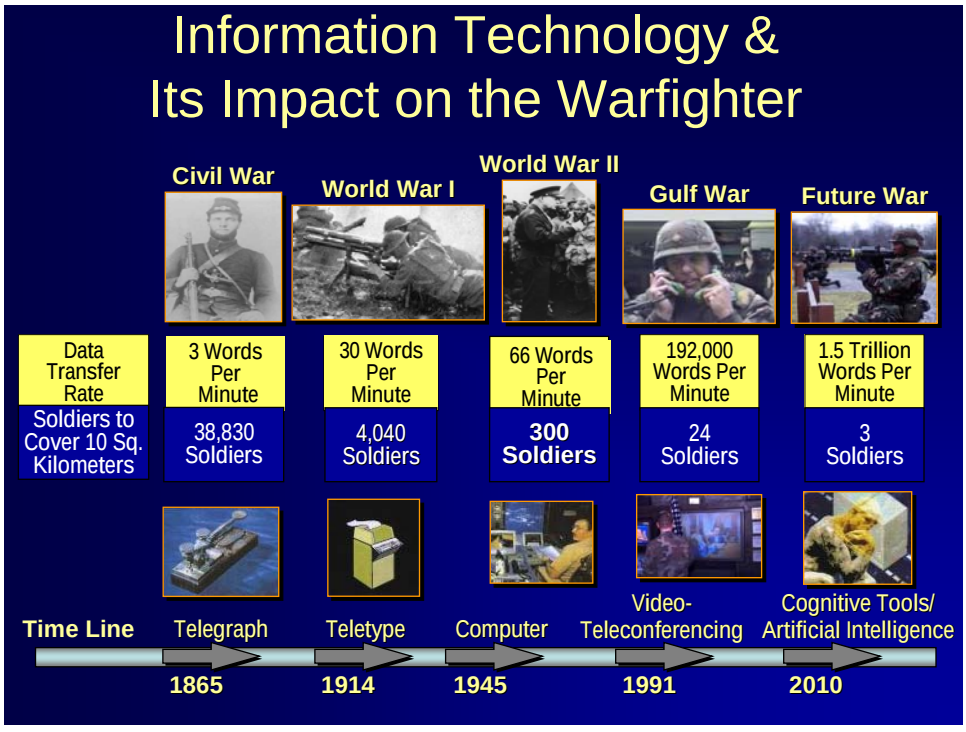

Figure 1 Time line on the effect of information technology in war

In the push to reduce casualties in the war, developing unmanned robots that perform tasks formerly done by humans is a hot topic of research. As more UAVs participate in real combat, more research [41][30][3] in this area is underway. Historically [36], two of the major military tasks for UAVs have been target identification and intelligence gathering. In these tasks, cameras or sensors for collecting data are mounted on the unmanned aerial vehicles. Some of the possible data collected are aperture radar images, infrared images, or digital images [3]. The Department of Defense spends millions of dollars pushing the technology advancement on UAV related research ranging from automatic collision avoidance, and integrating the propulsion system into the airframe for reducing size, to autonomous mission control capabilities [3]. Recently, a new direction in the research of UAV control technology has been swarming UAVs. This research builds on the concept of UAV swarms and looks into deploying a distributed database system in such a swarm network.

1.1 Sensors on UAVs

Small, wireless sensors such as MICA sensors developed by *Crossbow* [18] found applications in many areas ranging from wireless sensor networks to environmental monitoring. Each sensor is 2.25x1.25x0.25 (inch) in size and weighs 0.7oz [18]. Equipped with a multi-channel radio transceiver and various sensing capabilities, these sensors operate on battery power using 8mA of current when in active mode and less than 15μA in sleep mode [18]. In addition, MICA sensors are reprogrammmable through wireless remote control. Their small size allows them to be embedded in other equipment. They can also interact with various onboard control systems, functioning similar to the way thermostats help maintain the temperature of an office building.

Current UAVs are both scare and in high demand. The cost objective of Global Hawk and DarkStar UAVs programs were approximately $10 million apiece for a fly-ready vehicle [19]. Although the cost is only a fraction of that of a manned aircraft, inspecting the costs of an array of UAVs published by the United Stated Department of Defense UAV Roadmap Briefing in Table 1 [19] shows that UAVs are not cheap. Air Combat Command chief Gen. Hal Hornburg at an Air Force Association conference in February 2003 stated plainly that UAVs are definitely not disposable items [49]. Building numerous small, inexpensive unmanned aerial vehicles provides the solution. Multiple sensors offer adequate redundancy for target identification. These UAV sensors can be either of the same type or operate at different frequencies and resolution to complement one another. It is both intuitive and experimentally proven [35] that fused information from two sensors more reliably identifies targets than one sensor. Also because of the redundancy of sensors, failure of any sensor only marginally impacts the overall intelligence collecting ability. In addition, their reduced cost and complexity makes them easier to replace.

Table 1 UAV Construction Properties.

System	Aircraft Cost FY02 $*	Aircraft Weight, lb*	Payload Weight, lb	System Cost FY02 $	Number of Acft/System
Predator	$1,700,000	1135	450	$30,000,000	4
Pioneer	$650,000	307	75	$7,000,000	4
Hunter	$1,200,000	1170	200	$20,000,000	8
Global Hawk	$20,000,000	9200	1950	$57,000,000	1
Shadow 200	$325,000	216	60	$6,200,000	4
Fire Scout	$1,800,000	1502	200	$14,200,000	3
Dragon Eye	$35,000	3.5	1	$120,000	3

*Aircraft costs are minus sensor costs, and aircraft weights are minus fuel and payload capacities. Hard ware costs, including GFE, are used.

1.2 Ad Hoc Network

A network of UAVs is more than a wireless network of sensors. UAVs are mobile units; the relative positions of individual entity are not fixed. Conventional routings lack the flexibility offered by an ad-hoc network in several aspects. In fact, transmissions in dynamically moving sensors or swarming UAVs entail the use of an ad-hoc network. An ad-hoc network refers to a local area network with dynamic network components that form a temporary network [55]. Ad-hoc networks stand out in their ability to reconfigure and adapt to the current availability of network devices. A protocol using dynamic source routing in ad hoc networks that "adapts quickly to routing changes when host movement is frequent, yet requires little or no overhead during periods in which hosts move less frequently" is presented in [32]. Different techniques in dynamic routing of messages to mobile hosts have been suggested [32][50][77]. There are two principal reasons for using ad-hoc networks as the communication medium.

First, no established infrastructure or central administration is available. A group of UAVs can be summoned to a battlefield on a short notice where they communicate via wireless network, perform team collaboration through distributed control. In this case, no prior arrangements for communication routes are made and the nodes are highly mobile. An ad-hoc network is designed to adjust to the change of topology or the lack of defined topology among nodes.

Second, the unpredictability of the environment introduces extra challenges to the network communication. Some nodes may become temporarily unreachable due to interferences from terrain, electro-magnetic noises, or malfunctions. In addition, mobile hosts may move out of range for wireless signals. Due to the robust property of ad-hoc

networks, if a router goes down or out of reach, an ad-hoc network is able to find another path quickly.

1.3 Research Goal and Objectives

Automated tools, like sensors, act as monitoring systems constantly reading new data from the environment. This data accumulates over time and the volume of data grows rapidly when a number of sensors are deployed, especially for image data, the size of which is usually large. As the amount of data increases, handling data within the system becomes increasingly difficult. Using a database is a structured way to organize data and facilitate data searching [28], information retrieval, or pattern recognition [40]. Pattern recognition methods, such as data mining in the database, allow users to make full utilization of the data at hand.

This research imposes a distributed database on the structure of an existing UAV communication network. Because of the decentralized nature of swarms, a distributed database consisting of the individual swarming UAVs is a convenient way for managing data storage. A distributed database can be defined as

> A database that consists of two or more data files located at different sites on a computer network. Because the database is distributed, different users can access it without interfering with one another. However, the DBMS must periodically synchronize the scattered databases to make sure that they all have consistent data [72].

The database sites (UAVs) that compose the distributed database system can be physically scattered over a large area through network connections. In spite of the physical locations, they are logically considered a large single database. In a distributed database system, the application processor software at each database site is responsible for processing requests that require access to more than one site, offering the impression of a single system. In a distributed database system made of UAVs, accessing remote

database sites can be achieved through wireless network connection. The database system then combines information from multiple sites (UAVs) into a single resulting set, which is returned to the site where the query originated.

This thesis effort investigates embedding a distributed database system in the system components of a group of information-collecting unmanned aerial vehicles (UAVs). The movement of UAVs is based on the existing behavior model of swarming UAVs developed by Kadrovach [33]. It is assumed that each UAV has some processing power and local storage capability to manage the processing and storage requirements of a distributed database on top of the resources consumed by swarm and navigation control. Following the swarm model of autonomous UAVs in [33], a swarm is assumed to maintain its altitude once airborne, constraining the swarm movement in two dimensions. The goal of this research is to study the possibility of establishing a distributed database system on a swarm of sensor UAVs and what it offers both in terms of system efficiency and effectiveness (information retrieval). The following objectives are set forth to facilitate achieving this goal:

1. To develop and analyze the input query conditions and the swarm property of a distributed sensor database system and their influence on system performance.
2. To design sets of test suites for assessing system performance.
3. To examine the relative effects of variables affecting the output of the system (sensitivity analysis).

The collaboration amongst UAVs in an operational environment has not been widely employed to date. In this research effort, it is assumed that the UAV network can be

easily scaled to a large size, which is one of the properties of a distributed database discussed in chapter II. With the vision of scaling swarms of sensor UAVs to large sizes, this research takes the following approach:

- To understand the technology and issues associated in developing a distributed database system on mobile sensors.
- To develop a model of a distributed database system with rudimentary database operation support.
- To design the system model to account for major contributors of system latencies such as inter-processor communication and load and their relative contribution.
- To design and analyze results of experiments with respect to the principles stated in Chapter IV and inspect how they conform to the general expectations of a distribution database system.

1.4 Thesis Outline

The document is organized as follows: Chapter II gives background descriptions of the problem domain, pointing out application specific properties. Previous research efforts and proposals relating to the problem domain along with potential solutions are included. This provides the necessary information to understand the swarm-based distributed database system presented. Chapter III addresses the major concepts and designs of the system model. A set of system functionalities is listed along with data representation paradigms. The design model of the distributed database system on UAVs is outlined, enabling further analysis on the model. Chapter IV presents detailed implementation descriptions and discusses the process and approaches for conducting experiments. Relevant performance evaluation metrics are defined and considered for

examining the test results. Chapter V discusses the outcome of the experiments and presents the test analysis. The effects of tested variables on system performance with respect to the defined evaluation measures are inspected. Chapter VI presents a summary of the research concluding the results analysis and offers suggestions for future work.

II. Background

This chapter gives an overview of the background information for using a distributed database in a network of swarming UAVs. Related research in areas of sensor network, swarming, database data storage for efficient retrieval, and search methods for finding optimal solutions in constrained search space are discussed. The discussion focuses on the issues involved in the database aspect of airborne unmanned flying machines used for either reconnaissance or surveillance missions. The discussion then gives a closer look at distributed database systems and lists some of the application requirements and system constraints identified by previous researchers.

2.1 From Data Collection to Information Retrieval

In this information age, technology has revolutionized the structure of modern warfare. Since World War II through the military campaign in Iraq, superiority in war is not achieved in the mass –the number of soldiers in the battlefield, but through information and information-based technology. Examples of using information technology in the battlefield proliferate: autonomous GPS guided missiles, strategic planning based on satellite images of adversary territory, GPS locating devices, etc.

The popularly recognized model proposed by Col John Boyd, USAF (Ret) for Command and Control (C2) activities in the military decision making process in the theater is known as the Observe Orient Decide Act (OODA) Loop in Figure 2 [70][71]. Observation includes gathering factual data from sensors or other means in the battlefield. Knowledge and useful information is extracted from the data at hand and is supplied to the decision makers. Taking all sources of information into consideration,

decisions are made. Based on the decisions, actions are taken that directly or indirectly affect the condition of the battlefield. More information about the OODA loop concept is found in [44][47]. The surveillance activities in the battlefield again provide feedback and confirm the effects of their actions. The OODA loop repeats itself [71].

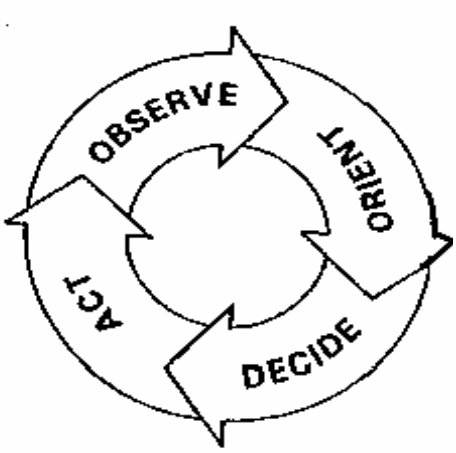

Figure 2 OODA Loop

Our discussion of sensors network and storage structure for information search falls under observe and orient in the OODA loop. The raw data from one sensor by itself does not have much meaning without putting it in context. Piecing together data from a set of sensors may reveal the layout of adversary forces. Analogously, images of an area at one moment in time may not be informative. Comparing several snapshots at different times may indicate the movement of certain objects of interest. The seemingly simple concept of integrating collected data so that information can be conveniently induced through inquiry is not trivial. A discussion of data fusion in a decentralized sensor structure for target tracking and recognition is presented in Appendix A. The next few sections contain some of the past research efforts related to the various system design

components. Also, other related research are included as appropriate for an overall understanding.

2.1.1 Sensor Network

The COUGAR project, developed in part by Cornell University, focuses on the sensor network [66]. Motivated by the insufficient support for scalability and flexibility in current sensor networks, the COUGAR project takes a distributed database approach for processing sensor data over an ad-hoc network. The system supports long-running queries for monitoring the environment as well as extemporaneous queries submitted by users [66]. The Cougar system is capable of handling simple queries requesting raw data from sensors to complex queries involving aggregate information gathered by multiple sensors. Appendix B highlights the issues covered in the COUGAR database project as applied to the dynamic UAV sensor system. The current research objectives of the COUGAR project are "to build a new distributed data management layer that scales with the growth of sensor interconnectivity and computational power on the sensors" and "cross-layer optimizations," which exploits commonly occurring patterns in query processing to preserve resources [66].

By introducing the assumption that network nodes (sensors) possess local storage capability, network traffic can be greatly reduced due to the fact that data buffering can reduce the frequency of data transfer. Instead of transmitting small pieces of data frequently across the network, large quantities can be transferred in less frequent intervals, reducing the chance of transfer collision and the loss of data. Moreover, having local storage in sensors enables sensors to keep a history of the sensor readings locally and facilitates the fast detection of either environment changes or any query regarding

recent sensor updates. In a stable environment devoid of changes, many consecutive sensor readings would be the same over a period of time. Keeping a short history of the readings makes it convenient to implement a policy at the sensor level such as storing the updates only when there are changes in sensor readings.

Similarly, augmenting sensor nodes with local processing power has several advantages. In a networked system, the computation workload can be allotted to a set of nodes exploiting parallel computation. Another potential use of the local processor is to perform pre-processing tasks locally. One simple pre-processing task that follows from the previous example of storing sensor readings only when changes occur involves comparing readings to decide if two readings are identical. More computational intensive pre-processing tasks are possible, too. For image data, images need to be stored in a certain format along with some metadata that distinguish the image to allow content-based retrieval. In this case, image-preprocessing may include computing image features and extracting identifying properties before the images are admitted to the database.

As suggested in the Cougar project, the combination of processing and storage ability at local nodes can be exploited for both conserving power consumption and eliminating network congestion [76]. Instead of transferring sensor data from each node to a central location over a large network, computations can be performed in a distributive manner at local nodes to produce intermediate results. The intermediate results of reduced size can then be processed subsequently. Aggregation functions such as average, max, or min, can use this approach. Since only the intermediate results are sent, the number of message transmitted across network is greatly decreased, which both conserves energy due to reduced transmission and mitigates network congestion. The

challenge in this approach is synchronization between nodes, which is exacerbated by the variance in network latency.

2.1.2 Swarming UAVs

Having UAVs swarm has received much attention in the DoD [12]. Swarming is an emergent behavior observed in species such as bees, ants, and geese. Swarming has been tested through millions of years of evolution and proves to be a good strategy for tasks requiring collaboration. Swarming allows individuals to interact and achieve elaborate group behavior that is not within the capability of any one individual [12]. The assumption is that a large number of the unmanned air vehicles, on the order of hundreds or thousands, equipped with wireless sensing devices can emulate social insects and swarm like flocks of birds [33].

Because swarming has some very attractive attributes to the control of a group of autonomous machines such as decentralized, implicit control, resilient to imperfection, and robust scalability, the swarming behavior is integrated to the methodology of autonomous cooperative control of UAVs [12]. Through swarming, complex and elaborate structures emerge from the interaction of a number of low-intelligent entities following a few simple rules. By emulating the swarming behavior in the controller, a group of simple, autonomous unmanned aerial vehicles can cooperate to perform the job of a highly trained human [12].

In swarming, individuals communicate and interact amongst the group. The information collected by sensors is organized and stored in a convenient manner. Another feature of a swarm is that despite the lack of global knowledge, the absence of global communication, and the presence of environmental noise, individuals are capable

of performing tasks efficiently as a group [23]. This feature of swarms is desirable for a collection of swarming UAVs since wireless communications are noisy and limited in range.

Swarming is one aspect in the domain of autonomous aerial vehicles. The subject of autonomous aerial vehicles has attracted much research interest because it encompasses and inter-relates problems across multiple disciplines. For example, scheduling problems that are combinatorial in complexity are found in the mission planning and sensor allocation tasks. In fact, a genetic algorithm model designed for the mission planning and dynamic allocation of airborne sensors has been proposed with the goal of minimizing both execution cost and time, while maximizing the fulfillment of high priority requests [61]. Particle simulation concepts also revealed their use in the modeling of swarm dynamics in [68].

Existing research efforts established both a behavior model for swarming sensors and a communication model for swarm-based sensor networks [33]. These swarm-based sensors do not just move; they communicate and move in a structured manner following a set of simple rules. While a plethora of protocols for mobile ad hoc networks exist [33], due to either the lack of scalability or the large amount of overhead in implementation, only three network routing protocols were investigated, namely, simplified Directed Diffusion, Geographical Routing, and Flood protocol. When applied to steady-state networks, simplified Directed Diffusion protocol slightly outperforms Geographical-based Routing protocol, while they both outperform the Flood protocol [33]. However, Directed Diffusion requires more system resources than the other two [33].

2.1.3 Fast Data Retrieval

For large databases, mechanisms for efficient and accurate information retrieval become mandatory. Commonly used techniques include fragmentation, caching, and partial replication. [38] discusses the effects of data storage organization, collection selection, and partial replication with replica selection on performance. The results of simulation confirm that performance is heavily impacted by data locality and data partition. It is also noteworthy that partial collection replication is not the same as caching, which though simple and fast, fails to identify similar queries relating to the same data set [38]. Another work by the authors of [38] indicates that the approach of partial replication with replica selection would increase query locality and outperform simple cashing with various configurations; "a combined approach will probably yield the best performance. [39]"

Ezeife and Barker in [21] address the fragmentation of data across individual sites for distributed object based systems. Investigations in the placement scheme of data to minimize data transfer and thus the communication delays over the network for a distributed database system mostly consider data as relations with the underlying assumption of having text-based data. Research concerning fragmentation in a distributed object-based system is rare. [21] provides algorithms for horizontal fragmentation based on class inheritance and hierarchies. The objects in the system are grouped into classes which include methods and attributes. While the expected computation time of the proposed algorithm is favorable – having polynomial order of complexity, the algorithm has its shortcomings. The algorithm assumes that the data access pattern is known a priori and organizes the data fragments based on these

predetermined patterns. The proposed algorithm has been tested on a static network; support for transparent migration of fragments in a mobile environment is still the subject of ongoing research [21].

2.1.4 Data Replication

As mentioned in Section 2.1.3, for a distributed database of moderate size to deliver satisfactory performance, a data replication mechanism must be in place. Data replication in distributed database systems refers to keeping the same data at multiple sites to reduce the communication cost for transferring data. Although data replication enables quick reference for read operations, it introduces complexities in the algorithm for write operations. In the simplest form, the read-one/write-all technique allows read operations to read from any available copy and write operations have to change all copies of the data. Updating data can be costly and may severely degenerate performance due to the delay involved in keeping all copies of the same object consistent. Innovative methods such as enhanced tree quorum algorithm and multiple tree quorum algorithm [11] were designed to decrease the operational cost while improving data availability. In general, these methods present a balance between data consistency and the time it takes to complete updates.

For replication to improve the system performance, copies of the object should be widely available when large numbers of reads are requested, while the number of replicas should be minimized when there are numerous writes. Taking this concept, Ouri Wolfson et al. in [75] designed an adaptive data replication (ADR) algorithm for replicating objects in the distributed database systems. The dynamic algorithm changes the locations and number of processors in which replicas are maintained according to the

read-write pattern of the accessing object. The replication scheme expands, contracts, and shifts depending on the usage requests in the network. Experiments show that "the communication cost of the ADR algorithm is on average between 21% and 50% lower than that of a static replication algorithm [75]." Two features that make the ADR algorithm suitable for a distributed system are: the execution of the ADR algorithm does not depend on global knowledge of other nodes but only locally collected statistics of the network traffic and some memory to keep track of status of its neighbors; the ADR algorithm is compatible and can be integrated with several existing "concurrency control and recovery mechanisms of a distributed database management system [75]."

2.2 Distributed Database on the UAV Communication Network

In the report of using UAVs in a reinforcement mission quoted in the introduction, the command center appears to be the only interface with UAV from which orders were issued and responses were sent back. Yet, the scenario can potentially be broadened to have more components involved in the command and control (C2) chain. Instead of one UAV, a group of UAVs are deployed and there's communication among the group as well as outside the group. In this case, each UAV is equipped with a sensor/camera and limited computing and storage capability. Each of the UAVs can be treated as a small database site; the ground command center, equipped with more physical resources, is another large database site. Additional sites potentially may include nearby aircraft that help exploit the locality of resources.

From the view of a communication network, a database search request can come from a bomber aircraft requesting the location of a target. The request may be routed through the command center, relayed by the satellite, to surveillance UAVs to find the

current position of such an object. On the other hand, the UAVs may want to classify the images they collect and identify objects of interest such as automated target recognition. In this case, the database is searched against the collected image. Since the searched object most likely is not on the local disk of the requesting UAV, the request is passed on to other distributed data sites. In sophisticated aerial vehicles such as UAVs, it can be reasonably assumed that UAVs periodically receive positioning information from satellites as part of the global positioning and flight control routines. The database system can take advantage of the existing communication link for navigation to distribute data and transmitting queries among nodes.

As a side note, one important use of images captured by UAVs is to facilitate automatic target detection and recognition. Borphys et. al. [6], for example, proposed an approach for long range automatic detection of targets using multi-sensor images to detect stationary or moving targets. The major task of target recognition is comparing images for patterns and searching for objects of interest. Logical entities are separated from their backgrounds using techniques of image segmentation [56]. If a match is found, the information can be transmitted to other units. Automatic target recognition, however, is the topic of many research efforts [6] [34] [48] [65] [7] [46] and is outside the range of this research.

2.3 Why Distributed Database?

In a distributed database network consisting of UAVs, each UAV acts as a small database site while the ground command center is a large data repository; other aircraft participating in the database transactions can also be treated as data repositories. The small and mobile swarming UAVs neither have nor should be assumed to have the same

amounts of resources for computation and storage as the ground command center. Because of the non-uniform distribution of resources among database sites, the distributed database system thus formed is considered a heterogeneous database.

Nonetheless, presenting the various data repositories as a single system is the responsibility of the distributed database management system software. The database management system functions similar to the middleware present in a cluster of a parallel computing system that supports the single system image (SSI) infrastructure and system availability infrastructure. It is described in [8] that "the SSI infrastructure glues together operating systems on all nodes to offer unified access to system resources." The distributed database management system is responsible for numerous essential tasks that ensure the smooth operation of the distributed database. Among them are system recovery from crashes, communication link failures; keeping track of data distribution and replication; maintaining data consistency of replicas.

2.3.1 Advantages of a Distributed Database System

As a result, the key benefits of a distributed database system include [1]

- **Transparency** Queries are submitted independent of the location of the data (location transparency) and the operation is the same for local or remote objects (access transparency). The distributed database system appears as a single system. The database system masks object migration, concurrency issues, object replication, system expansion, system failure, and system load from users (migration, concurrency, scalability, replication, failure, and performance transparency).

- **Capacity and Scalability** The amount of memory and the number of hard disk drives of a single system is limited. Having several database servers that act as a single system increases the resources available and the software and hardware from any node in the system can be shared (resource sharing). As the demand grows, more computer systems are connected with little or no upheaval to the DBMS.
- **Efficiency and flexibility** Data is stored close to the anticipated point of use. Multiple copies of the same data are made and distributed throughout the network making them readily available to requesting sites (data replication). Extensions and improvements to the system can be accommodated through both standardized and published interfaces (openness).
- **Reliability and Availability** The distributed database consists of many sites and contains duplicated data (redundancy). When one site fails, the overall system remains functional and allows data access from other sites (fault tolerance). In the face of failures, the database system is also responsible for data recovery.

Besides these benefits, the distributed database system has close correspondence to the decentralized swarming behavior and the distributed nature of the application.

The features of the distributed database system meet many of the demands of the UAV operation. First, queries submitted to the database should be allowed to enter from any site on the network – queries from any UAVs for target recognition, from the command center for pattern searching, from tanks or air combat fighters for information retrieval. No information regarding the location of the relevant data is required by the query input. Second, as the amount of image data increases or more sites are added, the

database system should to adapt to the changes without major modification to the system. Third, querying response time of the system is tested under the real time requirement of military operations. The positions of various sites may change, as is the direction of flow of query evaluation. Fourth, the nature of the task induces higher risk of encountering site failures. The loss of communication may make one or more sites temporarily unreachable. The system's robustness to failures and the ability to provide service in spite of imperfect conditions is a merit.

2.3.2 Constraints and Assumptions

Other limitations in the design of UAV network processing concern with the physical resources as suggested in the Cougar project [76].

- **Communication** The UAVs use wireless communication. Problems with the wireless links include bandwidth sharing, latency variance, transmission range, and data loss rate.
- **Power Consumption** Energy conservation is a consideration since both communication and computation on UAV sensors are powered by onboard batteries.
- **Computation** A class of UAVs known as micro air vehicles is distinguished by its small size and weight. Figure 3 shows a micro air vehicle, Black Widow, developed by AeroVironment [33][69]. Having a length of 0.5 foot and weighing 0.093 pounds, Black Widow is an aircraft designed to carry a black and white 300x240 pixels video camera for military intelligence activities [33][69]. As the sizes of UAVs diminish, resources become more constrained. One of the considerations in sensor network design is that sensors are limited in computing

power and memory sizes which confines the type of processing algorithms and the amount of results stored on a UAV.

- **Uncertainty in Sensor Readings** Physical readings from sensors have uncertainty because of the resolutions of the sensor, various environmental noises, and other surrounding effects [76].

Figure 3 Micro Air Vehicle- Black Widow

One of the features of distributed database systems is data sharing while maintaining local control. In our case, queries can be submitted to the database from any database site. A query presented to an image database presumably can be in the form of query by sample or query by features. In the former case, an example image is submitted and the result is a set of images in the database similar to the input image to a certain degree. In the latter case, the sample image is converted to a set of features representing the image, and the query is formulated accordingly. A human expert or a pattern extraction algorithm can be responsible for this conversion. In either case, for the discussion of a distributed database system studied in this document, it is assumed that such a mechanism is in place and so is a reasonable content-based retrieval algorithm.

2.4 Distributed Processing

The model of a swarm of processors establishes the basic structure of a distributed system. A distributed system affords reduced system processing time in most situations and has more resources at its disposal to meet the time sensitive requirements of real-time applications. Compared to a single-processor system, distributed processing enjoys larger storage by means of utilizing all the hard disks of the processors connected in the network. Distributed processing also enjoys the prerogative of concurrent processing without the constraint of sharing the computing time of a CPU. The sections that follow describe an algebraic model for message routing, a representation of the network connection topology, and includes a discussion of the various image processing algorithms applicable in a distributed environment.

2.4.1 Query Processing

In database systems, for a given operation, different processing strategies are evaluated to find the strategy with the lowest cost. For centralized database systems, I/O operations are the slowest operations in a uniprocessor application, thus the number of disk accesses is often a good indicator for evaluating the cost of a processing strategy. For distributed database systems, additional factors such as the cost of communication over the network and the performance gain from parallel data processing should be taken into consideration [58].

For message passing costs between processors in the network, store-and-forward routing is briefly discussed. Assume data transmission within the network uses packet routing, which is generally deemed more suitable for "networks with highly dynamic states [25]." Under packet routing, a message is broken into small parts of equal length.

For simplicity, in the formulation of a cost model described in this section, all packets of a message are assumed to follow the same path from the source to destination node. Let t_s stands for the startup time of a message transfer, which is the overhead associated with each message. r is the size of the original message in a packet; s is the additional information including error correction and sequence number for each packet. Together, $r+s$ is the total length of a packet. If it takes t_{w1} time to prepare a packet, preparing a message of size m into packets takes mt_{w1} time units. If a packet traverses l hops before reaching the destination, each hop adding t_h time to the relay, and the network transfers one word every t_{w2} unit of time, then the first packet would spend $t_h l+t_{w2}(r+s)$ time in network transmission. A message of size m has m/r packets. Because of the pipelined packet routing technique, after the first packet arrives, the rest of the packets arrive one by one at $t_{w2}(r+s)$ seconds intervals. Borrowed from [25], the network transmission cost model can be expressed by Equation 1.

Time for a message transfer

$$t_{comm} = t_s + t_{w1}m + t_h l + t_{w2}(r+s) + (m/r-1)\, t_{w2}(r+s) = t_s + t_h l + t_w m \qquad (1)$$

$$\text{where } t_w = t_{w1} + t_{w2}(1+s/r).$$

As Grama et al. points out in [25], the communication cost model suggests that there are three ways to minimize the cost, targeting toward reducing each variable in the model.

- Reduce the number of times a startup cost is incurred. The startup cost t_s is fixed for each message transfer. If multiple short messages combine into a long message, the average startup cost per unit length is reduced.

- Minimize the size of data. Keeping the data size small reduces the $t_w m$ term.
- Minimize the number of hops in data transfer. The number of intermediate nodes a packet visits affects the routing delay expressed in $t_h l$.

In addition to the communication cost, the performance gain obtained from multiprocessors and more aggregate cache space should be considered. A widely-accepted law governing parallel processing for fixed workload is Amdahl's Law, which states the performance improvement to be gained from using many processors over a serial execution by one processor is upper-bounded by the fraction the program can be executed in parallel [52]. For a program that requires T(1) to complete execution in serial mode using one processor, having a fraction *B* of the program that can only be run serially, if *N* processors are used, the parallelized execution time would be *B**T(1)+((1-*B*)*T(1))/*N*. The speedup (S), governed by Amdahl's Law, would thus be S= *N*/((*B***N*)+(1-*B*)) [42]. Gustafson's Law offers a fixed time speedup model in which the problem size is scaled with the assumption that parallel work scales with the problem [10][64]. Sun and Ni's Law generalizes both Amdahl's Law and Gustafson's Law to propose a fixed-memory model [10]. With these three models available to evaluate parallel processing applications, the model of choice depends on both the assumption of the test suites and the testing strategies adopted.

For wireless network communications, in addition to the message passing model discussed previously, it should be noted that not all routes in a wireless network incur the same latency. The network connecting the distributed database can be modeled as a weighted directed graph with no negative edge. Such a model can expresses the fact that in a wireless network, messages traversing in different directions on the same route can

have different costs. The graph consists of a set of vertices (nodes), a set of edges (arcs) and weight function *w* to map edges to weights. The weights of the edges are associated with the data transfer cost as computed by the communication cost model.

Definition [59]: Given a weighted directed graph (G(V,E),w), we define the weight of a path

$$p = (v_0, v_1, \ldots, v_k)$$

as the sum of the weights of it's constituent edges, i.e., as equation 2.

$$w(p) = \sum_{i=1}^{k} w(vi - 1, vi) \qquad (2)$$

An example of a weighted directed graph is shown in Figure 4 [26].

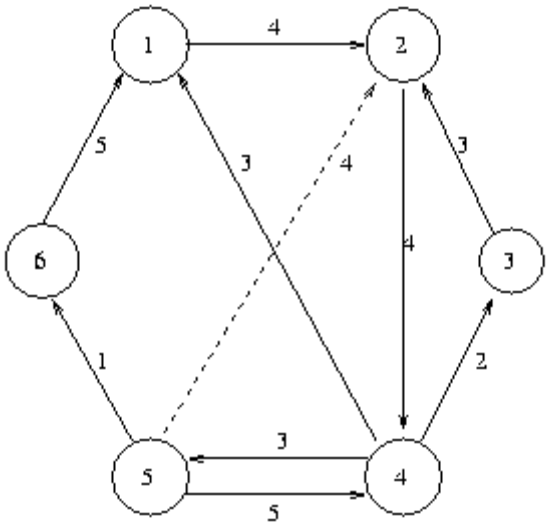

Figure 4 A mathematical model of a wireless network

The shortest-path weight δ(u, v), which is the sum of the weights along the shortest path, from u to v is expressed in equation 3 [60].

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there is a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases} \quad (3)$$

2.4.2 Parallel Image Processing

Image processing can take advantages of parallel processing in distributed processing as well. Because a high degree of locality and parallelism exist in most image processing jobs [62], parallel computing platforms provides easy mapping and becomes an economical computing option. Squyres et. al. implemented a parallel image processing software library, the Parallel Image Processing Toolkit, for obtaining speedy processing of large images [62]. The software library is built on the commonly accepted Message Passing Interface (MPI) standard; experiments using this library have shown promising.

Along the same line in providing tools for high performance applications in support of parallel image processing, a library-based software architecture is designed that makes parallel implementation transparent to developers. Seinstra et. al. give an assessment of the effectiveness of the proposed architecture in terms of performance improvement from three example applications: template matching, multi-baseline stereo vision, and line detection [57]. The results show that the architecture allows efficient application executions that are comparable to hand-coded programs (not significantly outperformed by hand-coded programs). As a consequence, application programmers are able to develop high performance applications in image processing without mastering parallel programming [57].

Parallel processing can even increases the efficiency of pattern matching tasks. Parallel algorithms that deliver low time bound complexity for one or two dimensional pattern matching can be found in the literature. Crochemore et. al. describes an alphabet-independent deterministic parallel algorithm for pattern matching in two dimensions using a concurrent-read-concurrent-write-parallel-random-access-machine (CRCW-PRAM) model [17]. The algorithm takes constant time following a preprocessing that can be bounded by O(log log m) where m is the larger of the size of either dimension of the two-dimensional pattern array. In another paper discussing the same parallel CRCW PRAM algorithm, they argue that the O(log log m) time bound is optimal for both the one and two dimensional pattern matching problem since another work has shown that the problem has a Ω(log logm) lower time bound [13].

2.5 Image Query Processing Strategies

Multiple sensors are often used in a target acquisition and recognition application providing multiple sensing sources. Pan et. al. [48] examined the use of fuzzy causal probabilistic networks in multi-sensor data fusion. Korona et. al. in [34] proposed a multi-sensor target recognition method based on logical models and feature fusion. Another paper [35] by Korona discusses the idea of fusing multi-sensor data in different frequency bands and resolutions for target recognition. In any rate, researches in multi-sensor data fusion for target identification and detection abound. When it comes to database searching and retrieval, attention should be directed to the representation of images. Unlike conventional database in which contents are text-based, a database for images or videos may contain textual annotations needs a standardized format to store the contents. Ekin et. al. [20] suggested an integrated semantic-syntactic video event

modeling that combines text and low-level video features to facilitate search and retrieval in the database. Under this model, video events are represented by graphs. While this integrated model supports the formulation of flexible queries and has been demonstrated to be effective [20], its utilization is limited. Because the goal of the model is to describe video events by capturing object-based motion features, instantiation of the model for stationary images would not have components related to motions, which degenerates the expressiveness of the graph, if such graph can be constructed. A more compelling reason that makes this modeling system unattractive is the lack of an automated way to convert video clips to the appropriate graphical representation. The authors of the paper did not directly address this problem but simply stated that video segments can be expressed as graphs following the proposed model. Another method, ImageMap, for indexing and searching similar images based on graphical representations is introduced by Petrakis et. al. in [51]. A commonly used image representation, Attributed Relational Graphs (ARGs), along with ARG editing distance functions is applied. ImageMap represents objects and regions in an image as nodes and arcs in a graph the maps the images into low-dimensionality points. For fast retrieval, the f-dimensional points are indexed by an R-tree structure. The process is illustrated by Figure 5 [51].

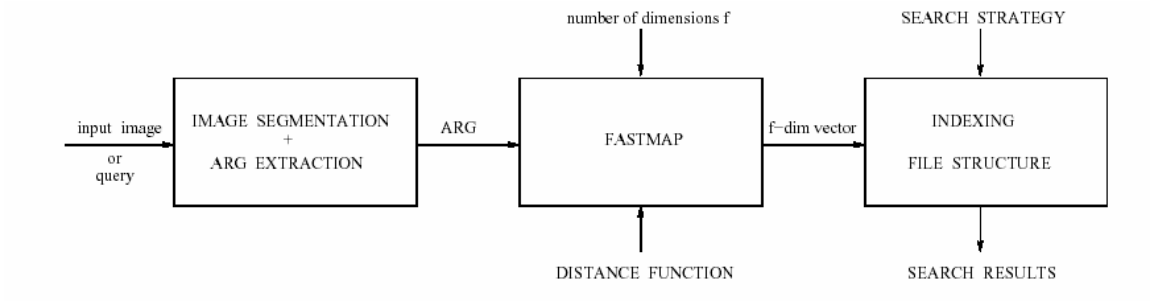

Figure 5 Process flow for image processing in ImageMap

Another aspect of database query processing is the formulation of queries. With the integrated semantic-syntactical model proposed by Ekin et. al. [20], graph patterns for queries to the database can be generated by editing the generic model through insert, delete, or duplicate parts of the model. Alternatively, users can modify existing image descriptions or use model templates for special graph patterns [20]. Not only is this approach convenient and flexible, it reduces the amount of information that needs to be transferred for a query. The description of the search criteria or a graph template is transferred to the system. Comparing this with a straight-forward query by sample approach that sends the entire sample image, this approach structures the query more succinctly and provides the possibility of further decreasing the network transfer load by only sending the modified portion of the model. Still another approach suggested by Saux et. al. in [37] aims at assisting the users to form a query by presenting an overview of the contents of the database. Using Adaptive Robust Competition (ARC) approach, [37] the database system selects the most representative image from each image category to present to the users. In addition to the purpose of query by example, the support for browsing a large image database implies that the time for image searching and retrieval is shortened due to the reduction of search space – only images belonging to the same category are considered.

Following the idea of categorizing data in the database, Ghose et. al. investigated a resilient data-centric storage scheme in wireless sensor networks [24]. In data centric storage, queries are quickly directed to sensor nodes designated for storage of the specific data type. The resilient data-centric storage model augments the capabilities by supplying duplicated data and control information across the network to enhance data

availability and hardware failure. Integrating the structure of data-centric design in the context of pattern matching produces a process flow similar to the feature-based multi-sensor data fusion system in Figure 6 [34]. The processing of data is organized as follows: The raw data describing the physical world is collected and stored at local sensor nodes. Initial processing of the raw data such as feature extraction and possibly model checking can be performed locally at each sensor node before storage. The feature sets representing sensor data are then fused to a feature vector followed by the recognition step.

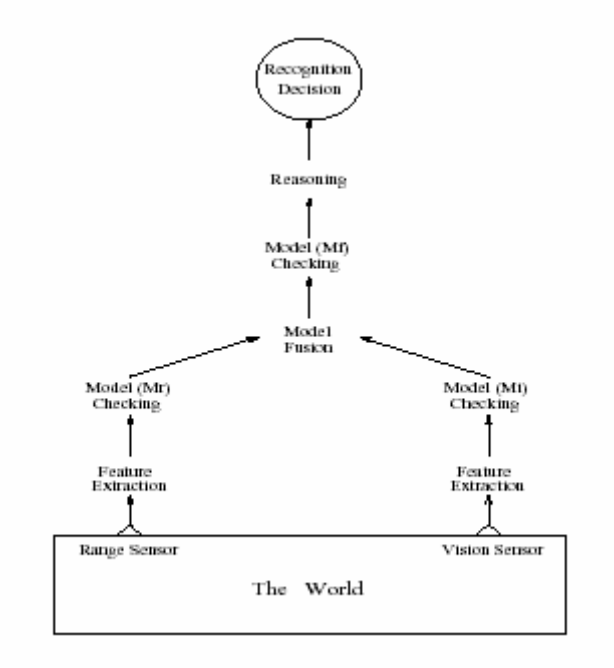

Figure 6 Feature-based multi-sensor data fusion system

Table 2 summarizes the different approaches for formulating image queries discussed so far. The approaches are compared with a base line method in which images from sensors are stored directly. Upon receiving an image query, the sample image is

compared with all images in the database using an affinity function that determines the similarity between two images quantitatively.

Table 2 Summary of approaches for formulating image queries

Approach	Advantage	Disadvantage	Image Structure
Entire example image	No preprocessing	Long retrieval time, high volume network transfer	Pixels or raw data format
Integrated Semantic-Syntactic Video Event Modeling	Flexible query construction, include semantic and syntactic information	Designed for capturing motion, degenerated model for stationary objects	Graph model with entities and relations
ImageMap	Fast search and retrieval	Computation overhead for feature extraction and mapping	f-dimensional points organized in a tree structure
Adaptive Robust Competition (ARC)	Reduced search space, have overview of the database	Computation overhead for fuzzy partitioning clusters, dimensionality reduction	Feature vector

For target recognition, two approaches are usually used –abstract-then-fuse and fuse-then-abstract [35]. In the *abstract-then-fuse* approach, target recognition is performed at each sensor node based on the local data. A global decision is then made by integrating local information. In the *fuse-then-abstract* approach, sensor data from each sensor is fused before target recognition process is performed [35]. None of the literatures mentioned in this section regarding data fusion has explicit details on how multi-sensor fusion is achieved in a distributed system. Having information from all sensors gathered at one location to perform fusion clearly contradicts the concept of distributed processing and leads to: flooding the network communication links,

overburdening one processor, creating processing bottlenecks. Hence, a distributed data fusion flow of control is of necessity for any data fusion scheme.

Because the focus of this research effort is on the utilization of distributed databases to meet the storage and search requirements of the UAV application, it is assumed that some form of image processing mechanism is embedded in the distributed system. The underlying image representation and query strategy may involve one or more or a combination of the approaches previous discussed. Images are internally represented by streams of bits; and the degree of similarities of an image to another image, according to a predetermined comparison algorithm, can be represented as a floating point number. For these reasons, in the simplified model of the database system that is described in detail in Chapter III and IV, measurements taken by the sensors are represented as floating point numbers rather than in image format. By keeping the data representation generic, it helps accommodate future expansion to suit various applications.

2.6 Summary

This chapter has covered the context in which UAVs are deployed, their significance in the information-oriented modern warfare, and the potential components and setup of a dynamic distributed UAV network. The distributed database concepts in a (UAV) sensor network and its relation to a parallel processing system have also been discussed. Research related to the techniques for distributed database systems, image processing, and parallel computing applications is identified. The information puts the UAV sensor database under study into perspective and recognizes the interrelations and variety of subjects that can be addressed in the ongoing and future work.

III. Query Design/Structures

Getting from information describing the UAV sensor network and what is involved in the system to proposing a simulation modeling this system consists of a lot of deliberations. Although many of the techniques and approaches discussed thus far are interesting, due to the scope of the research, simplifications have been made to adequately investigate the issues identified by the objectives of this research. In the process of narrowing on our focus, some decisions involve trade-offs and some topics have to be deferred for future work. The system model constructed in this research for the swarm-based distributed database places emphasis on the system performance of the distributed system and the message passing costs for query execution. As a result, the database specific inner works such as data retrieval techniques, the use of data replication, image processing mechanisms, and the concept of parallel processing, are not discussed further.

The first section of this chapter explains foundations of the high-level query design from sensor network and query in the physical world. In the sections that follow, a list of system design objectives (functionalities) is explicitly defined, along with a description of the designs for both data structures and internal representation. Several sensor database system design considerations are also discussed.

3.1 Querying Physical Space in a Networked Environment

The basic component in the distributed sensor UAV environment is the network, which ensures the connectivity of sensors on the system. The concept of *many processing units, one network* is the heart of environment monitoring and the backbone of

distributed systems. One view of querying in the physical world is rendered in this section along with the types of queries that might be submitted against such database. In practice, wireless network connections are the media in a swarm of UAVs. Some of the issues that may be encountered in a networked system are included in the discussion.

3.1.1 Networked Globe

The world today is sometimes referred to as an "information world." The immense amount of information around us can only be described as explosive and the modern way of life relies heavily on information through a slew of channels. While there are many different mediums to access news, one of the most powerful ways to access a plethora of information is through the World Wide Web, an information highway.

The keyword in all these information sharing channel we all enjoy so much every day is ***Network***. How is it that CNN has so much news? Why so much information about almost anything can be found using the World Wide Web? When information all around the world is shared via a connected network infrastructure, it becomes the power of networked collections.

Various networks, LAN, WAN, and SAN scattering around the globe, has made the earth an increasingly networked place. When it is tightly connected by the tangled networks, the globe can be viewed as one networked entity. The three-dimensional physical space can be treated as one large database containing billions of objects. The concept of addressing objects geographically, rather than using logical addressing, and defining a physical space containing data as *DataSpace* is brought forth in [29]. In this paradigm, objects can be associated with processors, on top of their inherent

characteristics. These objects can be queried to extract data pertinent to them, such as color, size, or network connectivity [29].

Like a regular database, the world described by DataSpace can be queried and monitored. Data are stored in objects. What distinguishes the paradigm of DataSpace from conventional databases is that objects are spatially located, operations are spatially driven – rooms or streets replace local area network, and the physical world is the database. Stated in another way, instead of having "physical objects become merely the artifacts of their corresponding entry in a database," in DataSpace, data are an inherent part of the objects and can only be retrieved by reaching the objects [29]. Not incidentally, this view of "the database is the network" is shared by [5] in association with querying and monitoring the environment through a device network. [5] proposes integrating query processing to the network of sensing devices.

3.1.2 Network Connection

As the backbone of a distributed system, the network mediates all communication between nodes, encompassing passing assigned workloads, reporting local data, and any other message transfer in support of distributed execution. Although wired connections offer higher capability (current transfer data rate is approximately 10Mbps) than wireless connections (around 1-2 Mbps for most wireless system), the specialized application of swarming UAVs requires wireless connections [33]. In the wireless domain, system capabilities also vary.

The properties of currently available IEEE standards for wireless technology are listed in Table 3, which is extracted from [33].

Table 3 Existing Wireless Systems Standards

	Data rate(Mbps)	Range(meters)	Frequency(GHz)
802.11	2	100	2.4
802.11b	11	100	2.4
802.11a	54	TBD	5
802.15 (Bluetooth)	<1	10	2.4
802.15 (high-rate)	20+	TBD	2.4/5
IrDA	4	1	IR

In a dynamic ad-hoc network, besides the connection media, other factors such as the network protocol and the formation of nodes in the network affect the characteristics of the networked system. From the design perspective, it is preferred that the network connection should adhere to these goals:

Minimize latency – the bandwidth should be large enough and additional overhead should be low enough to sustain the typical operations of the system.

Good Scalability – the network performance should be as linear as possible, or decreases within a commonly acceptable range, as the participating entities increase.

Good Flexibility – the system should maintain high resilience to restructuring of connecting nodes, change in query processing strategy, and functions relatively independent to upgrades/modifications to the sensor nodes over time.

3.2 Capabilities of a Sensor Database

Sensor databases typically work with applications that monitor its surroundings to detect, classify, or track physical objects. In the context of satisfying the demands of a monitoring system, a sensor database often needs to support these functions:

- Disseminate query execution plan to sensors in the network (let each sensor know what is needed)
- Compute/correlate data from sensors like performing average, max, min operations
- Allow for long-running queries that keeps a continual watch on the surroundings
- Support communication among nodes efficiently
- Allow for collection of all sensor data in the network (may include sensor value history)
- Support for ways of constructing a query and getting a report back

3.3 Sensor and Data Representation

While a large number of sensors are employed in the physical world to transform physical environments to digital data, the capabilities of sensors vary. Sensors capable of chemical detection find their use in measuring the chemical activities of an area, and temperature measuring sensors are able to report the temperature of their positions at any point in time. However, the ability to query the sensor systems allows information of interest to be extracted to facilitate other tasks. As an illustration, statistical information about the change of rainfall in an area contributes to the study of agriculture, climate analysis, or consideration for the suitability of building a water reservoir. Queries in a chemical sensor system can provide information for the analysis of the air composition in an area and evaluate the actions appropriate in a disaster situation.

In the discussion of sensor database systems in [4], it is pointed out that many designs of sensor databases lack flexibility and scalability. All sensor data is retrieved in a predefined way and is transmitted to a centralized location for processing, independent of the nature of the query. In a distributed sensor database, related data is produced simultaneously by a set of sensors according to the query execution plan. Rather than considering each sensor as tables, sensors are viewed as response-active objects and data is extracted from sensors based on the query. This approach avoids confining data extraction in a predefined manner and affords a better analogy to querying objects in a physical space – states of the object is returned upon request. The distributed query execution plan is designed to aggregate operations on sensor data to be processed concurrently in a distributed manner, thus exploiting the additional processing capability of the distributed system.

In this sensor database system, queries may access the state of sensors at a particular point in time, at a specific region, or may be concerned with the aggregates of sensors over a time window. The representation of sensors, therefore, should be able to satisfy the different types of queries. In particular, each sensor object contains methods that facilitate the search, retrieval, and modification of the sensor's history. The data representation at the minimum should include the location where the data is collected, a timestamp based on the current system time, readings of the sensor state, and a sensor id. Notice that since the sensor is not static, the sensor location, along with the timestamp needs to be maintained to completely describe the state of the collected data.

Another aspect concerning the result of the sensor database is that wireless connections are prone to be affected by the terrain, environmental noises, and other

factors. The connection of sensors to the network can be intermittent. Some sensors containing pertinent information can be out of reach because of the swarming behavior of the UAVs. It is possible to obtain different result sets in response to the same query at different times. The result of a query is represented as a set of records or an empty set, though it may not necessarily consist of responses from all of the sensors.

The design of this sensor database system is closely related to virtual sensors. *Virtual Sensors* is a software abstraction to facilitate manipulation of the sensed data. Three common sensor types are mentioned [2]: state sensors that return the current measurement from sensors, event sensors that monitor for change of states, and trigger sensors that report the current sensing value when an event is reported. These three types of sensors correspond with the querying capacity of our sensor database system above. The work of [2] indicates that *Virtual Sensors* provide both "control abstraction and data abstraction [27]". More importantly, [2] showed that using the design of *Virtual Sensors* to implement a system, the system is able to guarantee a timeframe for real-time operations and determine whether a request cannot be fulfilled via scheduling.

The subject of sensor database systems is by no means a revolutionary idea as quite much material about this concept can be found in the literature. Among them, the Cornell COUGAR system uses an object-relational database system [4]. The Cornell COUGAR system models sensors as objects of an Abstract Data Type (ADT), expresses queries in slightly modified version of SQL, and represents sensor data as time series [4]. The concept of virtual relations follows from referencing sensor ADT attributes in a query. *Virtual relations* indicates the representation of ADT functions as a table, which is not available in the database until requested and extracted from the sensors [4].

Our design of the sensor database supports the object-oriented software abstraction of objects, encapsulating sensor attributes and permitting interaction with sensors through a well-defined interface. Multiple sensor types can be supported through multiple ADTs. Scenarios in which multiple sensor types are present can be a swarm is composed of multiple types of sensors (specialized UAVs with varying duties) or a UAV carrying multiple sensors (UAVs with multiple duties). In our database model, each UAV is assumed to carry one sensor; all sensors are of the same type, that is, all sensors have the same sensing capability. The sensors can return the observations qualifying to the conditions set by a user query. A triggering condition on sensor measurements for all sensors on the network can also be defined. Once a trigger is set, new sensor measurements meeting the threshold are returned to the user until the trigger is removed.

3.4 Networked Sensor Database Design

There are a few assumptions the simulation designers of a networked distributed system made regarding the characteristics of the network and the database. As noted before, the sensor database is built upon a swarm of UAVs. It is crucial, however, to differentiate the swarming control algorithms from the algorithms for data processing in the network. Admittedly, since both the swarming control mechanism and database query processing share the same wireless network, some resource contention would occur. In practice, it would be necessary to evaluate both mechanisms to prioritize/de-conflict tasks; some messages can be piggybacked to conserve energy and bandwidth. For simplicity, this database model is oblivious to the inner workings of the swarming behavior of UAVs in so far the only information retained relating to swarm is the position of each sensor at points in time as reflected by the records stored on sensors. The model

thus assumes that system resources are available as needed. With these assumptions in place, the rest of this section examines some other considerations involved in designing this system in relation to time.

3.4.1 Cost of System Delay

When considering sending messages between two nodes over a network, the sending node needs to marshal the data, prepare message into packets, packets then traverse through nodes in the network, arrive at the receiving node, the receiver collects packets, and unmarshals the data. The reverse process takes place when replies are sent from the receiver to the sender.

From the standpoint of nodes, packets hop from the sending node, one hop at a time through an indefinite number of nodes depending on the distance between the sending and receiving nodes, to the destination. To simulate the system response time, our system calculates the cost of generating a response to a query based on this concept. Further, our system takes into account the complexity of the query and the number of records on average a sensor node has to search through before giving a result. The complexity of the query is the same for all sensors, but the average cost for query processing is evaluated sensor by sensor. For queries involving aggregate operations, it is assumed that the aggregates can be computed as the results are sent back from distributed sensor nodes. Once results from sensors are gathered at the central location, they need to be merged before presenting to the user. Assuming that records are sorted in timestamps for presentation, the time complexity for sorting is O($n \log n$) for n number of elements.

The delay incurred by each component is highly dependent on the properties of the system: the types of network, processor speed, transmission protocol, etc. The weights allocated for each factor in our rudimentary model is not specific to any system and is by no means representative of typical network or database systems. The cost of the system delay computed hereby is a combination of CPU, I/O, and network cost. Further tuning of these parameters can be done to reflect the environment of potential deployment of the system.

3.4.2 Clocks and Synchronization

The calculation of message transfer delays and query processing cost in the system involve the recording of time. In order to know the time an event occurred, events are timestamped at each process using the clock at the node where the process resides. In distributed systems, processes at different processors often need to coordinate and participate in transactions in which timestamps are used to determine the order of execution and serialize transactions. That is to say processors in distributed systems have a need to synchronize their physical clocks to know the state of each process at a certain time and to maintain the consistency of distributed data.

Timing in distributed systems is an important issue. Each computer has its own physical clock. *Clock skew* and *clock drift* are two terms used to describe the disparity of clocks. *Clock skew* refers to the difference in readings in computer clocks while *clock drift* is the variation in frequency of clocks' counts [16]. Numbers of algorithms exist for approximately synchronizing processes in distributed systems by sending messages through the network [16].

The design of our distributed sensor network includes the simulation of clock skew, which allows sensors to have different notion of what the current time is. It is assumed, on the other hand, that all observations are taken at one time increment interval in lock-steps. Hence, no clock drift exists. The sensors can later be synchronized to a global time through broadcasting. Another assumption is that the Global Positioning System (GPS) transmits timing signals to UAVs at regular intervals to maintain a universal time across the network. Our design also supports the notion of a global time and eliminates clock skew when the global time assumption is used.

3.4.3 Current Observations

Sometimes it is tempting for users to ask for the current measurements of all sensors as a *live report*. A query requesting information that is collected *right now* is unusual to a conventional database and is not specifically identified as the capabilities in the initial design of our sensor database. The closest approach to obtaining current measurements of sensors using the existing query structure is asking for records with the maximum timestamps, namely, the latest observations, assuming sensors are synchronized to a global time.

One of the concerns in requesting for current sensor readings is the clock skew problem discussed earlier. With the absence of a global clock, sensors may have different timestamps for the current time. If a global time exists, another problem arises –how is *current measurements* defined. What do we mean by *right now*? There is invariably some delay from the moment a query is issued to the moment a query is evaluated at sensors. To top it all, the delay is not the same for every sensor. A sensor that is farther from the query dissemination point would receive the query later than a

sensor that is nearby. Does current measurement mean the sensor reading at the time the query is submitted or the latest reading at each sensor as the query is evaluated? The definition for current observations must be made clear if the ability to query current measurements is to be added to the sensor database system.

The current design of the system is such that the result produced for a query is based on the contents of the database at the time the query is issued. It is very well be the case that some more observations have been made by the time the query arrived at the sensing nodes. But the returned records from the sensors would include only the records that qualify the restrictions imposed by the query at the time the query is entered.

3.4.4 Message Passing Method

Part of the assumptions of the swarming UAVs model is that a UAV moves in accordance to the behavior of its neighbors and its local information; there is no centralized control. For the sensor network, it means the whereabouts of sensors some time in the future is unknown; in that regard, even the current positions of the sensors are unknown without the aids of satellites and GPS. Some sensors could possibly go astray and move out of the reach of others. The message passing method should be robust and tolerate these scenarios while finding all the sensors that can be reached.

For the sake of simplicity, the sensors in our network pass messages (query request and results) using a hopping technique. All queries are assumed to be issued from the same location, which can be a command center. A radius is defined for the distance that can be covered by wireless without forwarding. Sensors located within the radius can be reached with one hop and are marked. For each unmarked sensor, it is

determined if it can be reached by any sensor which has been marked; if so, mark the sensor with a hop number one greater than the sensor by which it is reached. The number of hops needed to reach a sensor is kept to the minimum by always testing if a sensor can be reached by starting from the sensors marked with the lowest hops.

The number of hops needed to pass messages to a sensor is included in the calculation of system delays. By using the technique above, unmarked sensors are out of the reach by others. It is also possible that the entire swarm of sensors is out of reach because the swarm has moved too far from the command center. Additionally, some sensors can receive the query but move away during query processing and unable to send back the results. Our design of the database tolerates these situations and generates a message to the user. While it is not conventional for databases to report information about data that the database is not capable of returning, a swarm-based sensor database is distinct in that the system is built upon moving sensors. As further elaborated in Section 4.3 and 4.4, additional information supplies users with the condition under which the results are reported and assess the values of the results accordingly.

3.5 Summary

The Cougar database project suggests that the prevalence of networks and sensors, either wired or wireless, has transformed the physical world into a computational world. Still, sometimes it is not customary for people to grasp the usefulness of the ubiquitous network -- that live environmental properties can be monitored and queried over a network. After examining the concept of DataSpace and querying through networked sensors, this chapter provides a high-leveled description of the initial system design steps and considerations. Other design choices are justified and estimated from

our knowledge of the properties of the distributed system. Implementation specifics and more design issues in experiments are presented next in Chapter IV.

IV. Implementation/Detailed Design of Experiments

Following the design of the distributed sensor database, this chapter begins with presenting the details of some interesting features of the system and the rationale behind these implementation choices. By referencing similar or alternative designs in literature, the first two sections explain what makes this database system suitable to the tasks of its potential application and what makes it distinct. Discussions of low-leveled system designs and their relations to sound software engineering principles are included where appropriate. Then, measures for assessing the performance of the sensor database system model are chosen. This chapter proceeds with a discussion of the approaches for testing in order to provide a quantitative understanding of the execution of the system. The design of experiments also shows what the user should be able to expect of the system. A short rationale for alternative testing scenarios is provided.

4.1 Virtual Database

In the description of device database systems in [5], each device is represented by an Abstract Data Type (ADT) object. By using an object-relational database system, a method of an ADT can be represented as a virtual relation, a record of which consists of the input arguments to the method as well as an output parameter [5]. Query execution plans can include virtual relations. The virtual relation is not materialized until function calls invoke the corresponding method on all ADT objects and virtual records containing the results are returned [5].

Analogously, our sensor database system can be regarded as a virtual database in the sense that information in the database is not available until queries are submitted and

evaluated in the form of a set of timed records. Due to the limited life time of sensors and their local storage constraint, data that is not obtained through query is lost as the sensors die or as memory space is full and new readings overwrite old records. For this reason, an *all* command is supported in our query system to enable users retrieve all records currently stored on sensors. On receiving the *all* command, all sensors would send their entire store of records back to the command center, which consumes a lot of energy and may clog the network. Thus, this should be done sparingly only when it is necessary to preserve the current state of the entire system.

Sensors are represented as ADT objects in our sensor database system, too. The use of ADT adheres to object-oriented programming concepts and offers a number of benefits. From the point of view of software engineering, ADTs allow for building a software model that has close correspondence to application domain objects or concepts. In our case, a sensor ADT object maps to a sensor UAV and has the properties associated with a physical sensor in the application model. In specific, each sensor has a system clock, has sensing ability, and stores a history of sensor readings. Also, ADTs aids software maintainability and extensibility. If the attributes of sensors ever change in the application (change storage size or allow sensors to have other capabilities), modifications can be made with ease through ADTs. Virtual relations, though useful for accessing attributes of sensor devices through methods, is not incorporated in our design of the database. Because in our simulation model there is only one type of sensor in the sensor network and one value is measured by the sensor, the advantages of using virtual relations do not exist. Besides, embedding virtual relations as part of execution plans and evaluating virtual relations assumes a sufficiently great control, if not total control, of the

inner workings of the underlying database. This kind of low-leveled control on the database is not available in our simulation.

4.2 Query Semantics

In addition to the *all* command, our sensor database system supports long-running queries, queries involving aggregations operations, and queries consisting of comparison operators and logical operators. But what makes querying in this system easy is that users need not have knowledge of the Structured Querying Language (SQL) to construct a query, nor does the developer. Users can build a query from commonly known operators like larger-than (>), less-than (<), and (&&), or (||).

4.2.1 Query Functions

Both [5] and [2] mention the capability of sensors to detect changes of state and raises signals asynchronously. Also known as long-running queries, the ability to detect changes or events enables users to monitor an environment. [5] shows an example of using a device network for flood detection; with long-running queries, sensors can raise alarms when abnormal rainfall level is detected. In our system, a long-running query is initiated by setting a trigger condition on the sensor readings. The current implementation allows one trigger condition to be set at any point in time and permits only one condition on the threshold, e.g. one of >=, >, <=, or <. Multiple triggers taking more complex conditions can potentially be added in the future. The *trigger* command is used to instruct the database to set a triggering condition. In subsequent runs, the *removeTrigger* command allows the previously defined trigger to be removed from the database. The trigger does not have to be the first command issued to the database

system and can be set at any time. However, only the new measurements since the trigger was set are reported.

Aggregation operators that act on a collection of records are implemented in our database system. The currently supported aggregation functions are max, min, and avg. While the query evaluation is assumed to be done in a distributed manner on all sensors in parallel, the aggregation is performed at a central location. Since distributive processing of aggregation functions requires merging results in multiple steps, mechanisms responsible for keeping track of sensors need to be in place. Without a fixed topology and global knowledge of other sensors, implementing such mechanism is not trivial. Therefore, our implementation reports one result in response to the aggregation function from each sensor and carries out the final aggregation at the central location.

4.2.2 Query Implementation

The object persistence technique adopted by the simulation of the distributed sensor database is Java Data Objects. Java Data Objects (JDO) is a standard for storing and retrieving data and preserves the states of objects beyond the lifetime of the Java Virtual Machine. Implementations that conform to the JDO standard are JDO implementations. Our simulation uses FastObjects™ j1 community edition by Poet Software. One of the benefits of JDO is its portability. Applications can be written independent of data stores. Therefore, no change in application is required when moving between data stores, even when the data stores use different paradigms such as relation database versus object database [54]. When changing to a different data store, appropriate JDO implementation for the data store need to be adopted [54].

Another important benefit of JDO is that knowing SQL is not a requirement [54]. Not using SQL frees the developer from the burden of formulating SQL-styled queries from user input. It becomes easier to tailor the user query interface to the specific problem domain model than being dictated by the SQL syntax. For example, in our database system, besides the aggregation functions and the trigger for long-running queries, the users can specific queries conditioning on four variables: the x coordinate, the y coordinate, the time of data collection, and the value of the sensor reading. These queries can be expressed in simple forms to be entered to the program; some examples are:

x > 30.0 && y < 25.5

time > 5 && time < 10

(value > 46.2 || value < -14) && x > 70.0.

The first query asks for all observations made when the x coordinates of sensors are greater than 30.0 and the y coordinates are less than 25.5. The second query is interested in the sensor readings from time units 5 to 10. The third query wants sensor readings which are greater than 46.2 or less than -14 when the x coordinates of the sensors are greater than 70.0. Currently, our simulator takes in queries from command line and subsequent queries can be issued once the previous task is complete. Simulation time automatically increments as a function of query delay and the number of commands that has been issued since the system starts. Appendix C describes the input the simulator expects and shows a sample run of the simulation.

On the whole, eliminating the use of SQL enables the internals of the data store and the execution of queries being more opaque to developers and promotes data store independence. Relating to software principles, the choice of using JDO rids unnecessary complexity in program development for database operations and eliminates places for potential bugs. Through JDO, programs can be written at a high level, impervious to changes in the low-level vendor specific database infrastructure, which greatly improve the maintainability of the software.

4.3 Performance Measures

To evaluate the performance of the system of our study, some of the common measures applied for evaluating the performance of a database are presented here with an explanation of their applicability to the system under discussion.

Efficiency Efficiency measures how fast tasks can be completed. Two closely related terms are response time and throughput of the system.

- Response time is the elapsed time starting from when a task (query) is submitted until the system returns a result.
- Throughput refers to the amount of work (number of queries) the system completes in a unit of time.
- **Scalability** Scalability refers to the extensibility of the system as the number of participants grows.

Effectiveness Effectiveness measures the quality of work completed –how well the returned result matches what it is supposed to return.

.

For database systems, throughput and response time are two conventional metrics for performance [5]. When applying these metrics to the distributed database system, the query response time, for example, would be affected by the query decomposition, query optimizer, network transmission delay, data retrieval, and data merging delays, to name a few. The throughput of the database is of concern when flow of transactions is large. As [5] states, in a pipelined processing environment, the response time is determined by the longer operation in the pipeline. Since our simulation of the distributed database is neither pipelined nor multithreaded (it can not process multiple queries simultaneously), response time and throughput are associated with an inverse relation. It would be redundant to include both throughput and response time in the system evaluation. The response time of the system, in our case, is limited by the accuracy of the assumptions made in the simulation and does not include delays accrued by all components of the system. Because the time cost of processing a query, which is captured in the simulation, directly affects the response time; response time is chosen as one of the performance measures.

Response time and throughput fall under the category of efficiency. Scalability is another measure of efficiency. Scalability measures the impact on the system as more nodes are added or more jobs are submitted simultaneously. Due to the lack of a detailed communication protocol implementation, information of delays from increased network communication as typically expected when increasing the number of nodes is not available in our simulation. Besides, since the current design of the distribute database system does not support multiple users or simultaneous processing of multiple queries,

scalability loses its meaning. Thus, scalability test is not a suitable measure for our system.

Effectiveness applied to our simulation serves as a validation check for the accuracy of the sensor collection process. Also, in the applications of swarming UAVs, it is of vital interest to know how much information about the environment is reflected by the swarm sensor readings. The reliability of the sensors is built upon several factors, including sensors' resilience against outside noise, error detection/correction mechanisms in the communication channel, and data storage stability. What is more, the spread of the swarms in an area inserts bias in the system reports. Since the movement of swarms does not guarantee an even coverage of the entire surveying area, the some areas could contribute to more of the sensor readings in the database than other areas. Hardware stability such as electronic malfunctions in sensors, too, can affect the accuracy of the environment that is deduced from the swarm of sensors. When deemed appropriate, a set of experiments should be designed to collect meaningful statistics to reduce the obscure effect.

It is desired to have the collective measurements of the sensors to closely follow the characteristics of the physical world they monitor. Data in the sensor database actually represents the monitored environment in various levels of completeness. The resulted collection depends on the path taken by the UAVs in the region of interest, the density of sensors covering the region, and the duration of time sensors spent taking measurements in that region. So, the content of the sensor database is best interpreted with an understanding of the accompanying conditions as mentioned above.

4.4 Experiment Methodology

This section starts by providing an overview on the setup of the simulation and various assumptions made in the experiments. In specific, the sources of data taken by the distributed database simulator are discussed. Then, the discussion continues to cover the objectives of the experiments and the approaches used. Experimental parameters are explained while appropriate.

4.4.1 Experiment Background

The distributed database simulator does not simulate the swarming behavior of UAVs. To provide the swarming movement for a group of networked sensors, our simulation adopts the work of Brian Kadrovach on swarming UAVs [33]. Brian Kadrovach developed a simulator for his swarming UAV behavior model; the swarm simulator generates positions of each UAV in two-dimensional Cartesian coordinates at each simulation time step. A description of the parameter settings and commands used for swarm simulations is provided in Appendix C. The swarm simulation assumes that sensor UAVs fly at the same altitude, and two-dimensional coordinates are sufficient to describe the locations of each UAV on a plane. Our sensor database makes the same assumption about UAVs. Our sensor database further assumes that these sensors take measurements of certain environmental property.

For the purpose of demonstrating the environmental monitoring ability of the UAV sensor network, the sensors make temperature measurements of the environment. The simulated data for sensor readings is taken from a MM3 weather forecast model that ran for 24 hours with output at an interval of an hour. The model uses gridpoints on a

198 x 198 lambert conformal grid. All variables are taken at 500mb and sensors' temperature readings are in Kelvin (K). The gridpoints are 15km apart.

The experiments are categorized into those that are concerned with the delay in time from query submission to receiving the outcome of the query and those that consider the quality of query results.

The experiments from the first category are conducted on a scenario by scenario basis to examine the effects on database system reaction time as the swarm evolves. The different type of queries submitted to the system is a major contributor to the system delay. In general, the time spent doing IO or network communication is orders of magnitude slower than the time performing CPU processing. In the simulator, the cost factor concerning communication is deliberately set to be much higher than that representing computation to reflect the relative system delay encountered in each case. As time progresses, the contents of the database grows from empty up to a limited storage. This produces an increase in processing load. The objective of experiments concerning the system delay is to investigate the system response time corresponding to the interested parameter of study. For the experiments in this research, the type of query and the state of the database system are investigated.

Another focus of the research is to investigate and give an estimate of how close the sensor readings from the swarming UAVs approximate the real physical conditions of the environment. This is achieved by providing information about the swarm relative to an area of interest in terms of space and time. The movement of swarm is best captured using animation to show the positions of individual members as time changes. Our simulated distributed sensor database system stores the temperature readings as a group

of sensors fly over a region in certain time span. It should be noted that the original grids on which weather data is modeled and the positions of the original swarm simulation output do not coincide. While the grids from the weather data are specified in terms of longitudes and latitudes, the swarm simulation supplies UAVs with simulated coordinates. For purpose of experiments, the positions of UAVs from the swarm simulator are scaled and shifted to map onto the grids. The sensor readings are taken from the nearest grid point after the mapping. An animation of the movement of swarming UAVs within the grids is provided to show the sensor coverage of the area in time. The center of the swarm, expressed as the mean position of the swarm, at each time unit is calculated and displayed as the swarm moves.

As part of the efforts in informing the users of the quality of the query results, results for aggregate operations are returned with useful accompanying information. The pieces of accompanying information consist of the number of sensors and the total number of sensor readings involved in producing the result. In evaluating the conditions under which the sensor readings are taken, it is helpful for the decision makers to understand the reliability of the reports thereby produced.

It is sometimes difficult to find appropriate quantative metrics for a qualitative study. In an effort to quantify the expectations for a query outcome, scenarios are constructed to observe the outcome of queries by varying the scenario settings. The objective of the type of experiments involving the quality of the data reported by swarming sensors is to study the outcome of the specific types of queries and how it matches or does not match a user's expectation.

From the experiment design perspective, each set of experiment is conducted by varying the variable of study while holding other variables as close to constant as possible. Table 4 lists the parameters of study in each set of experiments performed.

Table 4 Experimental variable and metrics

Experiment	Tested Variable	Evaluation Metric
Query Complexity	Number of query conditions	Response time
Query Characteristics	Size of query result	Response time
Database Size	Number of records in the data store	Response time
Sensor spread and coverage(Section 5.3.1)	Sensor density	Effectiveness
Sensor spread and coverage(Section 5.3.2)	Time coverage of query	Effectiveness

4.4.2 Cost Function Parameters

A number of elements affect the response time of the queries as described in Section 3.5.1. An approximation of the system delays is incorporated in the simulation through a set of parameters. Some parameters are constants fixed for the duration of the simulation; some are variables that change with time and input. Table 5 lists the set of parameters used by the simulation for calculating the system response time. Each parameter is shown as a variable or constant within the simulation and whether a change in its value would affect inter-node communication cost or computation processing cos. These parameters are included to account for the major delays in the database system under consideration.

Table 5 Parameters in computing system delay

Parameter	**Type**	**Effect**
Query type (complexity)	dynamic	communication & processing
Wireless range	static	communication
Command center location	static	communication
Time of query	dynamic	processing (& communication)
Hop rate	static	communication
Central processing rate	static	processing
Data retrieval rate	static	processing

Query type: The type of the query can be separated into queries containing aggregation operator and queries without aggregation (including *all* command). Queries are submitted at run-time and can have arbitrary complexity with any number of conditions. Complexity of the query, in our simulation, is assumed to be proportional to the number of conditions comprising the query. Typically, one result from each sensor node is returned in response to an aggregation operation. In contrast, all observations have to be returned for an *all* operation. The number of messages sent is directly related to the communication cost and the processing cost at both ends of the communicating nodes.

Wireless range: The effective distance of wireless connection determines how far the swarm can go from the command center without losing contact. It also determines the number of hops- message forwarding via intermediate node- involved for query dissemination and collecting results.

Command center location: The position of the command center, from which all queries are assumed to be generated and returned, determines the hops needed.

Time of query: As time progresses, the database experiences an increase in size as more observations are made and stored (up to the storage limit). Search and retrieval time is lengthened. Potentially, depending on the queries, more qualifying results are to be returned as time goes on, which increases the communication cost.

Hop rate, central processing rate, and data retrieval rate: Hardware properties such as the characteristics of the routing node, speed of the processing unit, and each sensor node's memory structure and speed respectively affect the values of these parameters. In the following experiments, they are set to be constants.

This part of our experiments focuses on investigating the effects of varying queries and query submission time on the reaction time of the system while retaining the other parameters at constant. Because the system response time is internally calculated using a set of mathematical formulas comprising of these parameters, the same response time is returned by the simulation for the same set of parameters and swarm data. In other words, it is of little meaning to run the simulation multiple times with the same input unless the simulation is for a different set of swarm data or at different simulation time. In reality, for a stable database system, the same system delay should be expected while all environmental variables are. The variable which can vary in practice and this simulation does not include is the delay due to network traffic, which is almost unpredictable if the wireless network is shared as is often the case in commercial sectors. None of the experiments discussed are intended to validate the correctness of the simulation, but rather to observe the system performance caused by specific variables. The query response time is controlled by mathematical functions whereas the correctness of the query output is determined by the embedded database implementation.

Furthermore, since the initial input describing the UAV positions contains a predefined number of simulation time steps for swarm movements, the experiments concerning varying time steps should allow reasonably large number of time steps separations between runs to obtain noticeable performance difference. This implies that the number of experiments should be kept small. On the other hand, a sufficient number of runs should be conducted for analysis. The experiments testing system response delays follow what is typical in computation analysis and uses 5 to 6 runs [31].

4.4.3 Sensor Density and Coverage

The focus of experiments here is examining the relation between increased sensor density and the physical world the sensor database measures. Unlike other database systems where data is entered in a static environment, such as the device network described in [5], the sensors are stationary at fixed locations; sensor UAVs represent a dynamic database. Enabling sensors to move adds immense flexibility to the system. It is possible to monitor a large area and tolerate coarse granular samples of the entire area. Or, sensors can concentrate in a small area to get fine-grained data. The price that comes with increased flexibility is that it is no longer clear how good the data is – how much authority is associated with the data or how reliable the results are. Especially, swarms do not have a fixed formation. Total control of the swarm is sacrificed by allowing swarms to run autonomously.

Applications which employ swarm sensors to collecting information are often interested in not only the contents of the database but the quality of the results from a query. Databases can be queried in many ways, but not all responses carry the same significance in representing the queried environment. For illustration, if a user wants to

know the average temperature of area A from 7am to 7pm, the database would produce a result based on its contents. Suppose the swarm sensors merely skimmed over area A for one hour, say from 12pm to 1pm, and spent the rest of the day zooming area B, the response for the average temperature for area A would not represent what the user expects. In this scenario, there is no flaw in the database and the result is correct as far as the database is concerned. Regardless, the quality of the result is poor and most likely misleading. To mitigate the adverse effects, other information about the swarm sensors and the condition under which the result is generated are indispensable.

Although scalability of the database system is not one of the performance metrics, the effectiveness of the database system is. By increasing the number of sensors in the network, more sensor readings can be obtained for an area. Comparing the database comprised of these sensors and the underlying environment would serve a good indicator as to the effectiveness of the database. Despite of the proximity of the database contents and the properties of the physical world at a certain time, it is vital for users to know what information the database can be relied on to contain in order to get the most out of the queries. Effectiveness of the queries can be enhanced by understanding the movement of the swarm over time. For an increasingly large swarm, visualizing their movements is the easiest way to perceive their movement relative to space and time.

Intuitively, the density of the sensor in the queried area and the coverage of the area would impact the quality of the database report. The expected number of observations per square unit area is proportional to the density of sensors in that region. Time introduces another dimension in the number of observations vs. sensor density coverage relation. The user should have some reasonable expectations based on the

nature and characteristics of swarms to the query responses before queries are entered. Experiments in Section 5.3 are performed to study the expected effectiveness of a sensor database through querying databases under a variety of scenarios. For experimental purpose, the size of the grids is trimmed down by a factor of 10 in each dimension to allow adequate swarm coverage. Namely, a 20x20 grid is applied instead of the entire 198x198 grids.

4.5 Other Sensor Scenarios

A number of alternative scenarios can be devised in the experiment design. In the study of sensor density and coverage, the current experimental design only covers the variation of time and sensor density with respect to a random set of grid points. Similar tests cases can be built for regions on the grid. Consider a region of a grid is made of contiguous grid points; a region is in effect a two-dimensional expansion of a single grid point. Testing for observations made within a region perhaps has a more practical purpose because regions on a grid correspond to geographical areas in the physical world. However, sensor scenarios for area testing include details such as the size of the region, whether multiple areas should be selected on the grid for observation, and how the areas should be selected. The selection of these parameters often has application specific purposes and justifications should be made for their selection. It is decided that the experiment designs are to be kept generic and simple.

Based on the capability of the sensor database system, tests of validation can be performed. For example, to validate the functionality of setting and removing a long-running query for monitoring the physical environment, a potential set up would be to set a trigger and make sure at least two events meeting that triggering requirement occurs at

some time apart. Then the system can be validated by checking if the trigger is activated when the event happens and if the second event is reported once the trigger is removed. Yet, tests like these give little insight about the system but rather some confidence in the correctness of the program's execution.

4.6 Summary

Implementation characteristics are elaborated in this chapter and are compared or adopted from systems of other researches. Choices of implementation are detailed in the chapter. Also covered in the discussion are the measures for evaluating the system and justifications for their use. The designs and variables of testing for the suites of experiments to be performed are included along with other experiment alternatives. This sets up the preparation for the experiments in Chapter V.

V. Results and Analysis

This chapter presents the results of our experiments and the associated analysis. The experiments follow the design and methodology as discussed in Chapter IV. The focuses are on the performance of a distributed swarm sensor database system with respect to query responses. Analysis is carried out both quantitatively and qualitatively. The experiments in the first section deal with system delays in generating query responses due to the nature of the query. Then more experiments are performed leading to an analysis of query delays as related to database size. Finally, swarm characteristics with relation to query outputs are examined. The experiments in the first two sections follow the discussion of Section 4.4.1 and are concerned with the system delay cost. The experiments in the third section entail from Section 4.4.2.

5.1 Type of Query

The set of experiments in this section shows the effects on response time as the type of query varies. First, the correspondence between response time and query complexity is investigated. Query complexity in these experiments is defined to be proportional to the number of restricting conditions a query contains. Detailed parameter settings under which the response times are obtained are included in Appendix D. Queries are submitted to the database when the database has the same size to eliminate the effect of data volume on the response time. Similarly, the size of the returned results is controlled by carefully selecting the conditions so that the same number of observations is returned for each query. Table 6 shows the response time and the corresponding query complexity.

Table 6 Query complexity and response time

Query Complexity (# of conditions)	Simulated Response Time (ms)
1	98.62
2	98.72
3	98.82
4	98.92
5	139.02
6	139.12

From the table, it is clear that query response times become steadily longer as the number of query conditions goes up. At first glance, the cause of the sudden jump in response time when the query complexity goes from 4 to 5 conditions is unexplained. However, after tracing through the simulation, it is found that network communication delay is responsible for the sudden increase in response time. The additional delay in query processing due to query complexity leads to results being sent back at different time units and the corresponding structure of the swarm at those time units requires more time to sent data back.

Second, using the same parameter settings, the following runs intend to show the system delay due to both processing cost and communication cost in aggregation queries versus a query that returns the entire database. Again, all queries are submitted while the database is of the same size. The first three runs validate the logic embedded in the database simulator that all aggregation operations on the entire database have the same delay cost. Aggregate operations are chosen to be compared with the results of an *all*

command because both operations involve scanning through the entire record set. Thus, it can be assumed that both incur approximately the same processing cost at the sensing nodes, if the slight computation cost for finding local min, max, or average in aggregate operations is ignored. As shown in Table 7, the response time of a query consisting of only an aggregate operator is in sharp contrast to that of collecting the records of the entire database. The reason is twofold. A vast amount of data sending through network accumulates considerable communication delay which is much more than the delay incurred by sending one record back per node as an aggregate operation would. Another reason is concerned with the processing cost at the receiving end in merging data from all nodes. It is expected that any assumption and operation that affect inter-node communications outweighs those impacting the CPU processing time. Thus, the large difference in response time between an aggregation operation and an *all* operation is mainly attributed to the excessive network transmission involved in the *all* operation.

Table 7 Query type and response time

Query Type	Aggregate (avg y)	Aggregate (min x)	Aggregate (max value)	All
Response Time (ms)	8.2	8.2	8.2	98.72

5.2 Database Size

The size of the database impacts system performance both in communication time (for sending potentially more results) and in query processing cost (due to the data volume to be searched through). As an illustration, for a system of 10 sensors, each of

which making observations at the rate of 10 readings per hour, an *all* command at the end of the first hour causes 10*10=100 records to be sent back; the same *all* command submitted at the end of the fifth hour produces 500 records. In the efforts to observe the impacts of query submission time on query response delay as described in 4.4.1 for the parameter *Time of Query*, 5 runs are performed. For each run, an aggregate operation is submitted to make the effect of increased data load at each sensor node more observable. Aggregation operations mandate that every record in the sensor is processed but typically only one record is sent back from each node. Choosing aggregate operations therefore exploits the full effect of the size of the data store while ridding the query response time from unnecessary network communication cost. Besides, using aggregation has the additional benefit that the number of records that are returned is much more predictable than queries consisting of arbitrary conditions. By choosing aggregation queries, the variable contribution to the overall response time resulting from sending a different number of messages over time is reduced.

Table 8 shows the results of the simulations. The trials are designed so that at the submission time of each query, the amount of data in sensor nodes is raised by a fixed increment. Related environment settings for these runs are discussed in Appendix D. Notice in Figure 7 that the last four trials exhibit a linear increase in response time with the increased amount of data at each node. The first trial seems to be off from the trend. Upon further investigation, it is revealed that since the value for wireless range parameter is quite large, all nodes are reachable within one hop at the time trial #1 is run. After trial #1, the swarm move farther away; while the whole swarm remains connected, some

nodes require more than one hop to be researched. Because the query result is not returned until the last node is reported the gap in response time is explained.

Table 8 Data store size and response time

Trial #	1	2	3	4	5
Response Time (ms)	4.9	9.8	10.7	11.6	12.5

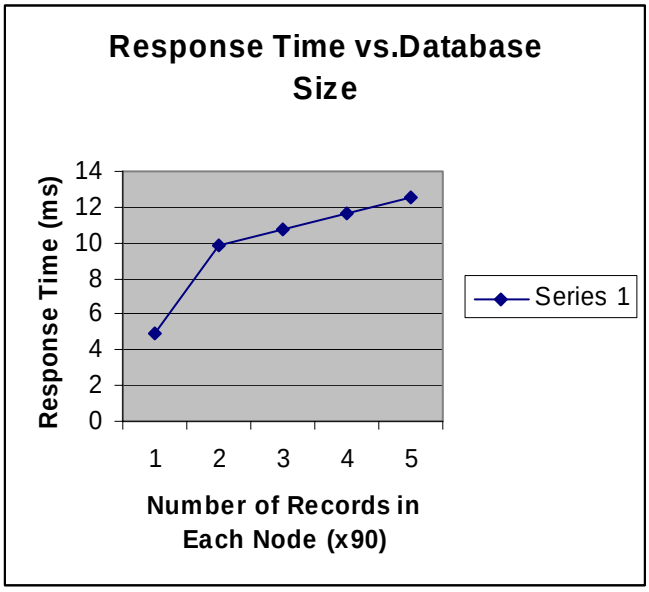

Figure 7 Graph of database system response time and database size

5.3 Query Results in Space and Time

As discussed in the experiment design, the following experiments map the swarm movements to be enclosed in a 20x20 grid. The observations are recorded for 500 time steps over a 24 hour period. At each time unit, each sensor records the temperature of the nearest grid point; that is, the sensing radius of a sensor is restricted to one grid point- the one it is closet to. This assumption simplifies the experiments. To study the expected number of observations from a query concerning specific locations on the grid, 50

random grid points are chosen. 50 points out of 400 grid points covers 12.5% of the entire swarming area and should be sufficient to examine the effect of sensor density in space and time. Appendix D.3 includes the actual grid points used and other system settings.

5.3.1 Grid Points and Sensor Density

For each randomly chosen grid point, the number of resulting sensor readings from the corresponding query is recorded. 5 sets of experiments querying the same 50 random grid points are performed with each set containing the number of sensors that is a multiple of the number of sensors in the first set. Refer to Appendix D.3 for the results of these experiments. Table 9 and Table 10 summarizes the results of each set of experiment in terms of the sum of the total sensor readings obtained and the number of reporting points, respectively. Subsequently, the summaries are put in graphs in Figure 8 and Figure 9, respectively.

Table 9 Summary of observations made in 90 time units

# of sensors	15	30	45	60	75
# of readings	61	435	821	314	12

Table 10 Summary of number of positions of observations in 90 time units

# of sensors	15	30	45	60	75
# of points	6	5	20	7	7

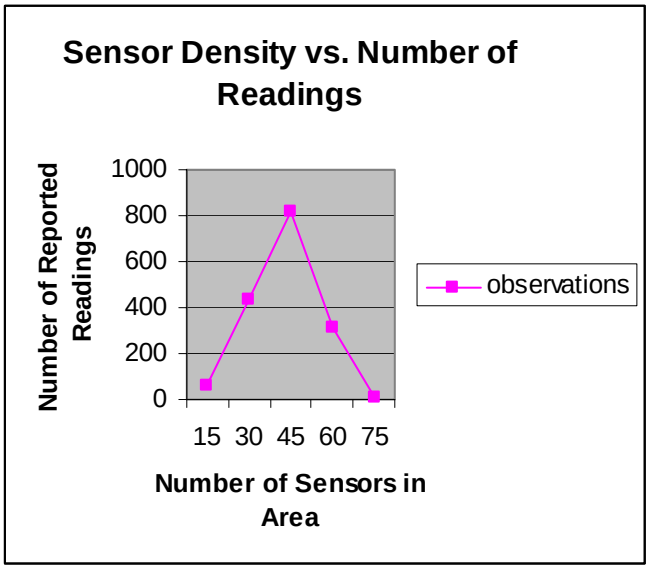

Figure 8 Graph of sensor density and number of observations

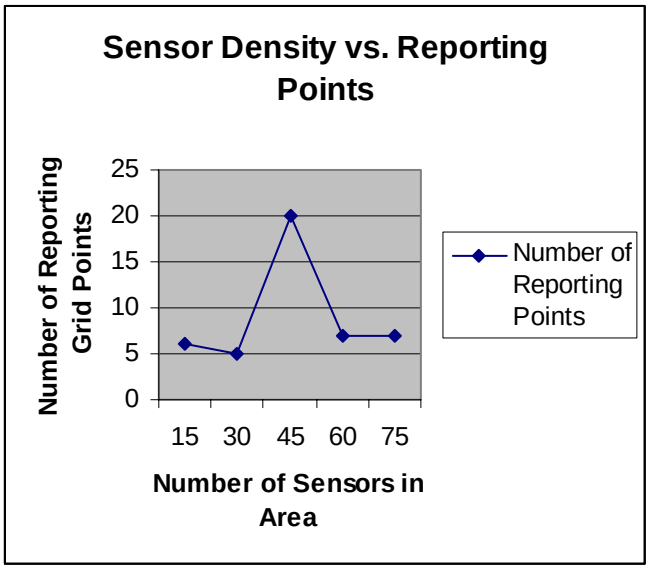

Figure 9 Graph of sensor density and number of reporting points

From the graphs, there does not appear to be any apparent trends. Although the first part of the graph in Figure 8 suggests that the number of returned sensor readings has a linear relation with the number of sensors in the area, this trend is not observed in the second part of the graph. In fact, the second part of the graph seems to obey an inverse

linear relation. An interesting point shown by the graph is that the number of sensor readings reported by the system is less when the sensor density is highest than that at any other sensor density, which is counter-intuitive. Nonetheless, it can be inferred from the graph that the structure of the swarms of size 60 and 75 are quite different from their structures at smaller sizes. The swarm structure as its size grows over 60 sensors happens not to have much overlap with the random grid points, which is quite possible considering that only 50 out of 400 grid points are selected. From Figure 9, our inference is assisted by the indication that the number of reporting points at swarm size 60 and 75 are the same, but the number of reported readings shows apparent decline from 314 to 12. This can also be an indication that the swarm of size 75 has spread apart and deviates from the grid points captured by these trials. By combining the information contained in both graphs, it suggests that the swarm hovers over a small number of points at size 30 when the lowest number of reporting points contributes to the second largest number of readings. The formation of the swarm coincides best with the locations of the selected grid points at size 45 when both the number of reporting points and number of readings are largest.

5.3.2 Time and Sensor Density

The experiments in the previous section vary the number of sensors, that is, sensor density with respect to a fixed time span (90 simulation time units). The next sets of experiments look at a fixed sensor density of 75 sensors in the region over an increasing time span. All values from the previous settings are retained with each set of experiments conducted at increasingly later points in time. The results of each run are

included in Appendix D.3. Table 11 and Table 12 summarize the query results from 50 runs. The graphs of summaries are shown in Figure 10 and Figure 11.

Table 11 Summary of observations made with 75 sensors

# time units	90	180	270	360	450
# of readings	12	573	2216	2881	3672

Table 12 Summary of number of points of observations made with 75 sensors

# time units	90	180	270	360	450
# of points	7	17	20	22	32

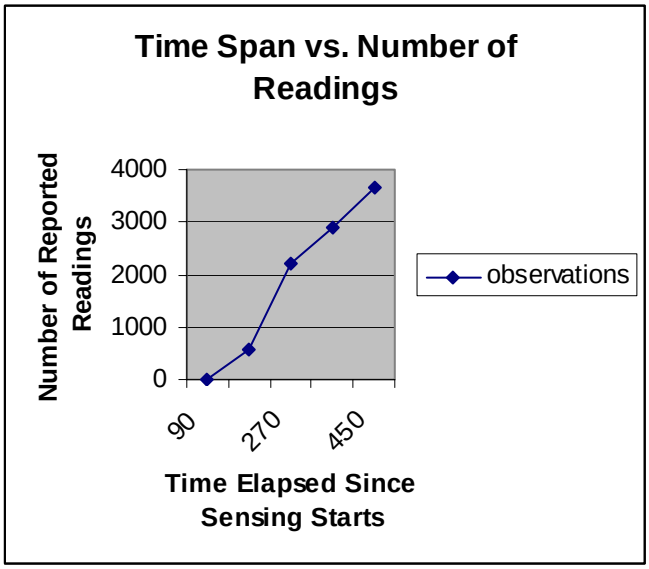

Figure 10 Graph of number of observations and time

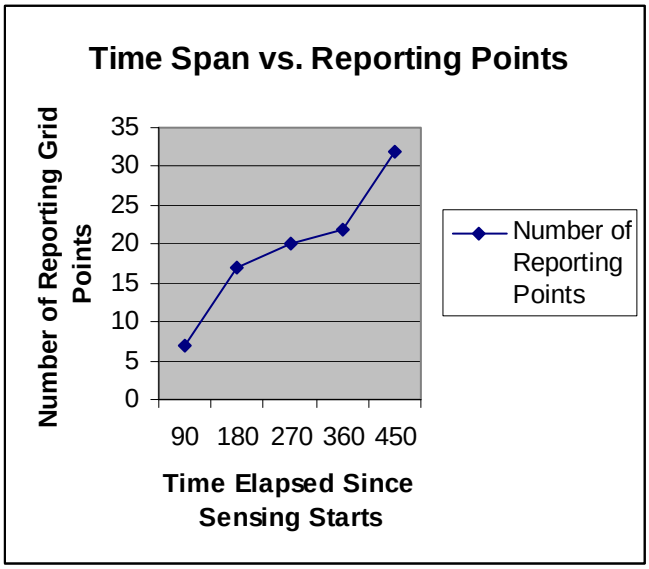

Figure 11 Graph of number of reporting points and time

Each time the length of time span is increased, it increases the time window during which any sensor can make an observation. Both graphs show increased query results as time goes on despite some fluctuations, which is expected for measurements involving statistical data at random points. Also, the dynamic changing of the swarm structure and path would have some impact on the outcome. Furthermore, it should be noted the suggestion that the number of sensor readings or number of reporting points would be proportional to sensor density in a region is based on the assumption that sensors are scattered evenly or move at random in space. This assumption is somewhat defeated when the underlying sensors follow, not random movement, but a specific type of coordinated collective behavior – swarming.

5.4 Summary

The experiments examined the effects on the system as the type of query varies, the database size grows, and more sensors are deployed. In the test of query complexity,

increased system response delay was observed as the query conditions grows, which is consistent with the system delay formulas built in the simulator. Besides query complexity, the query type affects the volume of data being transmitted in the network. As the second part of Section 5.1 revealed, the amount of data produced by a query has a significant impact on the system response time. Compared to the effects on response time caused by network transmission delay, the increase in processing time due to large data set is less significant, which can be understood given the relative speed of processor and network communication. Section 5.3 provides analyses of query results with relation to space and time. The results in Section 5.3.1 showed little indication that the expected number of sensor readings is proportional to sensor density. This result is hardly surprising given that the selected grid points are random and swarming individuals tend to move as a group. Finally, in Section 5.3.2, the time window for observation is widened gradually over trials to examine its effects on the result size. In this case, both number of observations and points of observation showed approximate linear growth with time. In all the experiments, the formation of swarm and location has a steady impact on the results, most prominently shown in the experiments in Section 5.3.1 and Figure 7. The results in Section 5.3.1 are directed more by the swarm movement than density, and the sudden increase in system delay shown in Figure 7 is explained by the change of swarm formation over time.

VI. Conclusions and Future Work

The top-level goal of this research is to examine the prospect of storing reconnaissance data from a collection of networked swarming unmanned aerial vehicles (UAVs) in a distributed database system. A discussion is presented as to what has been done in this research and how the objectives in support of the top-level goal set in Chapter I have been satisfied. Significant results and contributions of this work to the area of study of UAVs are provided, along with suggestions and recommendations for further research in this area.

6.1 Summary and Conclusions

In this research, a model has been developed that simulates a distributed database built on a swarm of sensors. Using this simulation model, a set of test suites have been designed to observe system performance. The two primary measures used to evaluate the system performance are efficiency and effectiveness. For the experiments testing efficiency in Section 5.1 and Section 5.2, query characteristics and the data load of the data store at query submission are analyzed. The input query conditions examined include queries consisting of multiple conditions of increasing number, aggregate queries, and queries requesting for the entire record set in the data store. To examine the effects of data load, another set of experiments entered queries of the same type to the database system at different simulation times. For the experiments intended to examine the effectiveness of the networked sensor database in Section 5.3, swarm properties are examined in the context of space and time. By analyzing the results, it is found that dynamic changes in swarm topology have a distinct impact on both the system response

time and query results. Inputs leading to large network transmissions affect the response latency much more than the effects resulted by increased processing load, which validates the cost function of the simulator and is consistent with our expectation. It is also revealed that the data (temperature) readings reported by the sensor network do not necessarily grow with the average sensor density in the monitored environment. But rather, an understanding of the swarm movement in space and time is necessary to facilitate effective data retrieval. In any rate, lengthening the observation window showed a positive impact on both the number of sensor readings returns and the number of grid points where readings are obtained.

In conclusion, the objectives as stated in Chapter I have been successfully accomplished through:

- Building a distributed sensor database model with built-in cost functions simulating delays caused by various factors as discussed in system model design in Chapter III and implementation details in Chapter IV.
- Constructing test cases specifically examining various query characteristics, data store attributes, and swarm properties as discussed in Section 4.4.
- Adopting evaluation metrics for analyzing the output of the system model as presented in Section 4.3.
- Exploring the effects and relative amount of influence resulted from various system components as discussed in Chapter V.

6.2 Research Contribution

Querying objects in a physical environment against a network of sensors deviates from the perceptions of a traditional database system where data is not thought of as dynamic. Understanding how networked sensors are able to be queried is the first step in constructing a sensor database model. In this research effort, a sensor distributed database system simulation model has been developed. The simulation model has been designed specifically considering the use of swarming UAVs as distributed data stores. Swarms of sizes ranging from 15 UAVs to 75 UAVs have been successfully tested in the simulation model. A number of potential system components in the networked database system are considered. Through the use of a set of test cases and evaluation metrics, this research explored the properties of such system model in a typical reconnaissance mission. The values of the simulator variable settings, though not necessarily correspond to the properties of a real system, give a convenient way of evaluating the impacts of a number of system properties on the overall performance. In all, this research builds on the ongoing study of swarming UAVs [33][23][14] and extends this swarm idea by incorporating a distributed database system to solve the data storage/search needs of the reconnaissance application in the modern network-centric warfare. Another aspect of this research is using a sensor network for environment monitoring. The Cougar project by the Cornell Database Group did an extensive study in this area [66][15][76]. As a result of their study, a sensor database supporting in-network query processing in an ad-hoc network has been developed. Although the sensor network described in the Cougar project does not account for mobile sensors, much can be learned in their work as applicable to the swarming UAV sensor network.

6.3 Future Work

As pointed out in Chapter II, the subject of a swarm of networked UAVs carrying sensors and possessing computation power and storage involve topics from multiple disciplines. Several areas of study have been identified, including internal optimization mechanisms of distributed database systems; techniques regarding the storage, search, and retrieval of image data; and the concept of utilizing parallel processing in a distributed system. A contemporary research effort at the Air Force Institute of Technology has successfully parallelized simulating swarming UAVs in a reconnaissance environment using the technique of discrete event simulation [14]. Using multiple processors for swarm simulation provides the frame work for possible development of a more realistic distributed database system model in which each processor represents one UAV.

The distributed database system simulator developed in this work while capturing the essential functions of a distributed database, has room for improvement. Some of the potential areas for improvement are:

- Event detection: the ability to have multiple criteria for environment event monitoring. [2] uses the notion of *even sensors* as a class of sensors that detect changes in sensor states asynchronously.

- Multithreading: capability for multiple queries from users can be handled. Research for processing multiple queries and optimization techniques for processing multiple queries abound [9][78][74].

- Network communication: more details can be added in simulating transmissions in the wireless network, including bandwidth considerations, estimates for energy consumption, application of a network communication protocol, and considerations for network traffic congestion [33][15].

Other details that can be included in the model are estimates for reliability of sensor (failure rate), reliability of network communication, and uncertainty in sensor readings.

Appendix A. Decentralized Sensor Fusion – Target Tracking and Target Recognition

Decentralized sensor fusion architecture differs from centralized architecture in that the filtering and some computational processing is performed at the local sensors instead of aggregate all sensor data in one place and fuses them. Even in decentralized architecture, the processed local results are further fused to form a global estimate in a central unit. Pucar and Norber [53] proposed an algorithm for decentralized sensor fusion and target tracking using extended Adaptive Forgetting through Multiple Models (AFMM) with a decentralized Kalman filtering scheme.

Since target recognition is closely related to target tracking, this paper intends to better understand the implications of the decentralized sensor fusion algorithm to target recognition and the underlying assumptions. This paper starts with a short overview of the algorithm proposed by Pucar and Norber, followed by a closer examination of the operating considerations and how they differ from those for target recognition. Then, the paper concludes with a discussion of the applicability of adopting the technique for decentralized sensor fusion to target recognition.

Two features proposed by Pucar and Norber as an extension to AFMM algorithm are support and alert of the sensor. To better understand support and alert, consider that a global estimate is obtained from sensor fusion and filtering. If that central estimate is fed back to the local sensors, the local sensors can benefit from this knowledge by either using the central estimate to reinforce the tracking ability of the sensor if the estimate of the local sensor is inferior, or altering the local sensor to focus its signal processing on

the indicated area if the local sensor did not detect the tracks contained in the central estimate. This approach could be defected by negative feedbacks and result in degradation. An essential part of the algorithm proposed in [53] is to contain the undesirable feedback while allowing local sensors benefit from the central estimate. This is accomplished through probabilistically including the central estimate in the filtering algorithm of the local sensor. When the global estimate is transferred back to the local sensors, the global track estimate is checked against the local sensor measurements. If the global estimate is better, the probability associated with that sequence increases; otherwise, it is lessened. Eventually, the sequence containing an estimate inconsistent with target measurements is terminated.

A crucial assumption in the tracking model is that a number of sensors, possibly of different type, "observe the same dynamic system (target) [53]" and use the same target model. This is an over-simplifying assumption. Realistically speaking, although a collection of sensors can be deployed at the same region to observe a dynamic object, not all sensors may be able to detect that object of interest due to limitations in the field of view of the sensor and the individual sensor location. The sensors can certainly be placed such that all sensors are able to observe the same target, provided the location of the target is known a priori, which is rarely the case, given that the target can move. The scenario is much more complicated when multiple targets exist. As a result, it is not clear which sensors have a specific target in view, making it difficult to distinguish noises from targets based on the sensor measurements. It is especially true for target recognition where the appearance of an object taken from different views can greatly impact the ability of identifying the object.

Also, an implicit assumption of the sensor fusion algorithm is that the sensors have a full view of the target. [53] states that when either the target dynamics or the measurement processes are nonlinear, the sensors do not have the same model of the target. In such situation, the algorithm depends on the assumption that since all sensors are tracking the same target the difference in the local state estimates (which directly affect the transition matrix) of each sensor is approximately close. However, this assumption does not necessarily hold when sensors only have limited partial data of the target. As the technology progresses, the sizes of sensors become smaller. It is the trend of the future to have a larger number of small sensors. Logically, most sensors only have a partial image of the target; a challenge of sensor fusion would be to produce the big picture from the measurements of local sensors while reducing the effect of noise. A target recognition or target tracking algorithm is then applied to the integrated big picture to sift out the objects of interest. The proposed decentralized sensor fusion algorithm does not address this possibility. On the other hand, the techniques of data association and target association may prove to be useful here. Track initiation and target association can be applied to supply extra information to existing tracks or targets to *glue* together the missing information. Track initiation is a process in which when a target in local measurement is not confirmed to be associated to any existing track, a decision is made for whether it is a new target or noise. Track update modifies the existing picture by updating the track or the targets when applying in the context of target recognition.

The main difference in decentralized sensor fusion for target recognition and target tracking is that target recognition does not need to keep track of the movement of the target or predict the most probable movement though knowing the most likely

movement of the target would help the placement of sensors to gather measurements to better facilitate identification. With the exception of having a feature comparison/target identifying algorithm in place, the process for target recognition is very similar to target tracking. The discussion above indicates two concerns that should not be overlooked when considering applying the extended AFMM and Kalman filter scheme to target recognition. It is doubtful that the existing tracking algorithm would accommodate these additional needs without considerable amount of efforts. Nonetheless, the approaches of fusing data from decentralized sensors serves as a valuable reference for future development in this area.

Appendix B. Application of the COUGAR Sensor Database Project on Unmanned Aerial Vehicles

The COUGAR system developed by the Cornell Database Group at Cornell University is targeted toward the investigation of a distributed database infrastructure on a network of small sensors having certain (e.g. seismic, acoustic, or temperature) sensing capabilities. The system is designed to be scalable and fault-tolerant, at the same time providing flexible data access [66]. The COUGAR sensor database project addresses a number of concerns that need to be considered for the design of a distributed database system for unmanned aerial vehicles (UAVs) in a surveillance mission. As a working system comprising of query processing over ad-hoc sensor networks [66], the COUGAR system represents one feasible solution to the sensor network data management problem.

Because of the highly similarity in the problem domain, it is helpful to understand and evaluate the challenges faced and the system architecture of the COUGAR system. Nevertheless, this paper is not a summary of the COUGAR system; rather, this paper takes the important points identified by the COUGAR project and describes their relevance to the problem of data management using distributed database on UAVs. This paper also points out how the characteristics inherent in our problem domain, that is, distributed database system on UAVs, affect our considerations.

The main reason for advocating *in-network storage and processing* in a sensor network is due to the fact that with the current state of technology the power consumption for communication in networks is much more than performing computation within nodes

[66]. The previous approach of sending sensor data offline for querying and analysis is both not cost-effective and unattractive in terms of system flexibility. While the statements above hold true in the UAV operations, when handling multiple images of high resolution the onsite storage capacities (amount of storage on the sensor or the UAV on which the sensor is mounted) may be pressed to the limit. Other image processing operations and possibly some rudimentary target recognition steps are reasonably expected to be taken onboard. They all contend for the already limited system resources on a network node.

The type of sensors considered by the COUGAR project is "nodes communicating via wireless multi-hop RF radio powered by small batteries [66]." Sensors of this type are subject to some physical constraints, as identified in the COUGAR project: communication, power consumption, computation, uncertainty in sensor readings [66]. In general, the same constraints apply to surveillance UAVs. The good news is that the sensors on UAVs do not have to operate by themselves; they are connected to a relatively large vehicle (compared to the size of the sensors themselves). The unmanned aircrafts have existing communication channel for navigation and fly control. Communication between aircrafts on the same network is most likely available as well. The communication needed for the distributed data management system can use the existing communication channel or piggyback on other communication data to reduce the overhead. Similarly, the resource constraints for sensor's power consumption and computation capability can be relaxed if the computing and energy capacities of the aircraft is included in the consideration. Even so, the aircrafts have still limited

capacities. Today, the flight duration of typical UAVs ranges from one to forty-eight hours with payloads from two to one thousand nine hundred and sixty lbs [22].

Though the additional resources on UAVs cannot help the uncertainty in sensor readings, the number and mobility of the UAVs can. Despite that there are invariably uncertainties inherent in the physical limitation of the sensors, for example, the image resolution; having multiple UAVs scout an area eliminates the problem of a bad sensor placement. A particular property of UAVs is that they are mobile – a property that is not explicitly stated in the sensor network investigated by the COUGAR project. The UAVs can make several rounds around an object, thus reducing the uncertainty. Besides, the uncertainty in sensor readings becomes less significant when automatic target recognition or some pattern matching algorithms are applied. Because fuzzy logics (or approximations) are used in pattern matching, little noises have a good chance of being glossed over by the algorithm.

From the ad-hoc network model and using the direct diffusion protocol for network communication, the sensor network in the COUGAR project is certainly a dynamic network. Still, the model of the COUGAR system does not completely satisfy the requirements of a distributed image database. Inspection of the COUGAR system processing steps indicates that the user query is submitted to the system through the Java-based GUI using graphical input or in SQL format, which is translated to XML format for query processing [15]. Queries are passed to a FrontEnd component that serves as the sensor network gateway. The FrontEnd manages the running queries and processes the tuples returned from QueryProxy, which is in charge of the exchanges of tuples among sensor nodes and communication with sensor devices and within the sensor clusters, to

produce the result [15]. The final result is shown in tabular form with the option of being sent to a MySQL database [15]. In its essence, the COUGAR system is designed to work with relational databases. Unless conventions are adapted to format image data into relations and represent images as tuples, different system architecture is needed for the UAVs.

A large part of the problem background information in the COUGAR project coincides with that of the scouting UAV problem, such as the use of a distributed system, the dynamic reconfigurable sensor network, and in network wireless communication. But a group of UAVs is more than a network of sensors. The fact that in surveillance/reconnaissance UAV operations, sensors are equipped on UAVs gives the sensors an edge over a pure sensor network. Integrating the sensors as part of the UAV allows more resources from UAVs to be accessible to sensor. On the other hand, this also introduces complexities in the sharing of resources. Another complexity in developing a distributed system for UAVs is the presentation of data. The current internal structures of the COUGAR system expect queries and sensor data to be translated to relations. To take advantage of the exiting COUGAR system, images collected on UAVs need to conform to the structure of a relation. Or a system structure suited for distributed processing of images need to be developed.

Appendix C. Simulation Input

C.1 Sensor Database Simulator Interface

To run the simulator, it is assumed that a Java Virtual Machine is installed on the platform. In the command window, enter the following:

java QueryNetwork [all] | [trigger comp_op value] | [aggregate_op property] [property condition [log_op property condition] ….]

The options for running the simulator are:

all – return all data in the system

trigger – set a long running condition on sensor measurements

value – conditioning threshold on sensor reading

aggregate_op – max, min, or avg

property – x, y, value, or time

condition – expression composed of comp_op followed by a value

comp_op – ==, >, <, >=, <=, !=

log_op – ||, &&

Optional parentheses are allowed to specify evaluation order.

Figure 12 shows a sample output of the database in response to the query (time > 5&& time < 10) && y < 18.1. The query is submitted at time units 20 of simulation time and the wireless range is set to 156.8.

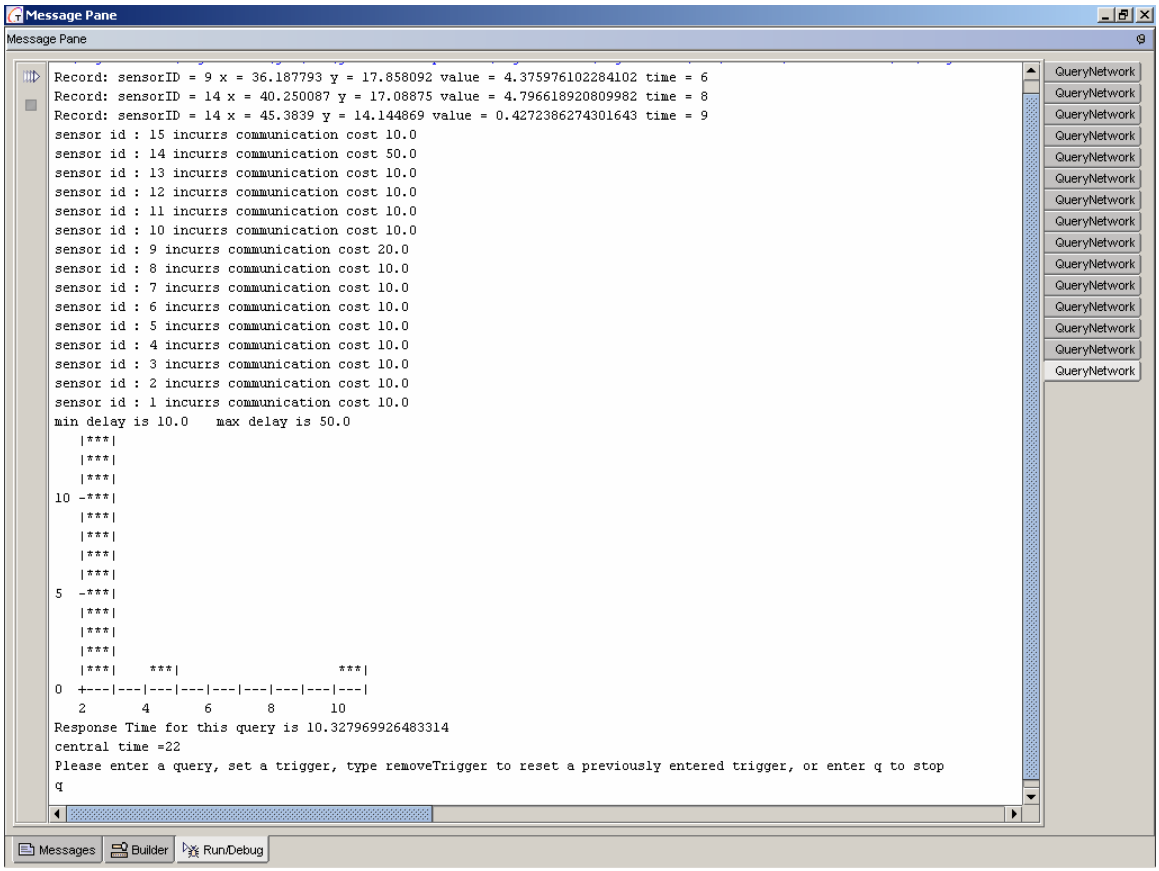

Figure 12 Sample simulation query output

C.2 Swarm Simulator Settings

Several runs were conducted to collect the UAV positions of swarms of various sizes. The switch settings used to run the simulation with 15 sensors is

swarm /h w.swh /i 500 /s 3454 /b yes

where /h flag specifies the output file, /i flag indicates the number of time steps to simulate, /s flag gives a seed for the internal random number generator, and /b specifies that a boundary is used. An additional file containing swarm configuration data is used in the swarm simulation. The parameters values for the swarm movements input to the database simulator are included in Table 13. The parameter that is responsible for the

size of the swarm is Popsize, which is varied from 15 to 75 at an interval of 15 to produce the swarms included in the experiments in Chapter V. Refer to Appendix F of [33] for a complete list of command line switches available and information about the parameter file.

Table 13 Parameter setting in param.txt file for swarm simulations

Region 3000 2000
Popsize 15
Scale 50.0
CZone 10.0
Dir 0.0
Seed 5216
Type 1
Velocity 1.0
Turn 5.0

Appendix D. Experiment Settings and Output

D.1 Query Complexity Test

The values of simulation parameters used in the trials for discussing query complexity in Section 5.1 are listed in Table 14.

Table 14 Parameter setting for distributed sensor database simulation in query complexity test

Parameter	**Value**
Wireless range	142.0
Command center location	(0,0)
Time of query	20
Hop rate	2.0
Central processing rate	1/170
Data retrieval rate	1/100

The swarm data contains 15 sensors in the swarm network making observations over 500 time units. Only data of the first 20 time units are in the database when the queries are submitted; that is, it is assumed that queries are initiated at time unit 20. For each run trial, the number of conditions in the query is varied. The query input for each trial is as shown in Table 15.

Table 15 Queries used in complexity test

Trial#	**Query**
1	x < 145.5

2	x < 145.5 && x > 27.1
3	x < 145.5 && x > 27.1 && y > -61.1
4	(x < 145.5 && x > 27.1 && y > -61.1&& y < 110.2)
5	(x < 145.5 && x > 27.1 && y > -61.1&& y < 110.2)||value <=59
6	(x < 145.5 && x > 27.1 && y > -61.1&& y < 110.2)||(value <=59&&value>=30)

D.2 Data Load Test

The tests for data load are conducted with the same parameter settings as in Table 14 with the exception of wireless range, which is set to 666.9. The value for wireless range is determined experimentally from the locations of the swarm and their distances to the command center at different points in time. The range of the wireless connection is chosen as to prevent any nodes from moving out of the range. Generally, keeping all nodes within reach is not a requirement; the data of those nodes not connected in the network is simply lost. Nevertheless, the value for wireless range is deliberately chosen to be large enough for all nodes to be able to report back in order to produce meaningful results and remove special conditions in evaluating the system response times.

Timing is imperative in this set of experiments. If observations are taken at a fixed time interval, which is one of the assumptions in the sensor data simulation, the time a query is entered should be evenly apart. To ensure that the database is augmented by the same amount between each trial, the same number of simulation time unit has to elapse before the next query can be submitted. Specifically, since there are 500 time steps from the swarm simulation and 5 trials are performed, each query is presented to the

database system 90 time units apart to allow extra time for query processing. Table 16 gives the queries entered for each trial.

Table 16 Query inputs in data load test

Trial#	**Query**
1	max x
2	min x
3	max y
4	min y
5	min value

D.3 Effectiveness Test

For the effectiveness experiments, response time is not considered, which renders most of the parameter settings irrelevant. Two parameters, the wireless radius and the length of time since sensors start storing data; still play a part in the outcome of these experiments. The radius of wireless range is set to 668.99 to guarantee all sensor observations can be reported. Time span is an experimental variable that changes with the trial. A random number generator is used to generate 50 grid points, listed in Table 17, to be monitored for observations. The grid points fall in the range of [0-19, 0-19] inclusive, corresponding to a 20x20 grid.

Table 17 50 random grid points on a 20x20 grid

(0,13)	(9,15)	(12,16)	(19,10)	(0,11)	(13,13)	(9, 0)	(14,13)	(15,4)	(16,10)
(13, 9)	(10,17)	(14,13)	(18,9)	(8,6)	(16,12)	(7,3)	(3,6)	(3,10)	(7,4)

(12,11)	(5, 17)	(4,11)	(1,14)	(11,19)	(18,1)	(5,19)	(14,16)	(8,12)	(17,17)
(16,14)	(12,13)	(13,11)	(14,15)	(12,15)	(11,13)	(5,4)	(11,6)	(12,19)	(4,15)
(3,1)	(12,3)	(17,15)	(4,17)	(6,8)	(13,9)	(5,17)	(14,0)	(5,2)	(8,9)

Tables 18 to Table 22 show the sizes of the returning result sets when querying each of these randomly generated grid points against the database. The trial number corresponds to the order of the grid points. Each query is submitted at the same time unit – time 90.

Table 18 Runs for 15 sensors with 90 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	0	0	0	0	0	0	0	0
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	0	0	0	0	4	0	0	0	0	21
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	0	0	0	0	0	0	0	0	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	0	0	0	0	0	0	8	0	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
# of readings	0	0	0	0	22	0	0	0	2	4

Table 19 Runs for 30 sensors with 90 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	0	0	0	0	60	0	0	0
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	0	0	0	0	0	0	196	3	0	0
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	0	0	0	0	0	0	0	0	0	0

Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	0	0	0	0	0	0	44	0	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
	0	0	0	0	10	0	0	0	132	0

Table 20 Runs for 45 sensors with 90 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	5	0	0	155	0	71	0	14
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	22	2	71	0	3	4	0	0	0	0
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	0	0	0	0	0	0	0	41	91	2
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	93	0	40	89	74	7	0	4	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
	0	0	7	0	4	22	0	0	0	0

Table 21 Runs for 60 sensor with 90 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	0	0	0	0	0	0	154	4
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	15	0	0	0	0	0	0	0	0	3
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	0	0	0	0	0	3	0	0	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	0	0	0	0	0	0	0	63	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
	0	57	0	0	0	15	0	0	0	0

Table 22 Runs for 75 sensors with 90 time units

Trial#	1	2	3	4	5	6	7	8	9	10

# of readings	0	0	0	0	0	0	0	5	2	1
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	41	0	5	0	0	0	0	0	0	0
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	0	0	0	0	0	0	0	0	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	0	0	0	0	0	0	0	0	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
	0	5	0	0	0	41	0	0	0	0

Instead of varying the number of sensors, for the experiments shown in Table 23 to Table 26, 75 sensors are used. For each set, queries are submitted at increasing time units.

Table 23 Runs for 75 sensors with 180 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	4	0	0	25	0	76	64	56
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	100	0	76	7	0	0	0	0	0	0
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	0	0	0	0	0	0	0	3	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	0	9	10	10	7	11	0	8	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
# of readings	0	7	0	0	0	100	0	0	0	0

Table 24 Runs for 75 sensors with 270 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	4	0	0	231	0	498	534	63
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	111	0	498	26	0	5	0	0	0	0
Trial#	21	22	23	24	25	26	27	28	29	30

# of readings	8	0	0	0	0	0	0	3	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	14	16	27	14	7	11	0	14	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
	0	21	0	0	0	111	0	0	0	0

Table 25 Runs for 75 sensors with 360 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	4	2	0	328	0	559	619	63
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	120	0	559	28	0	5	0	0	0	0
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	8	0	0	0	0	0	0	39	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	15	55	70	185	16	13	0	14	0	0
Trial#	41	42	43	44	45	46	47	48	49	50
	0	45	14	0	0	120	0	0	0	0

Table 26 Runs for 75 sensors with 450 time units

Trial#	1	2	3	4	5	6	7	8	9	10
# of readings	0	0	4	2	2	328	0	559	1306	66
Trial#	11	12	13	14	15	16	17	18	19	20
# of readings	129	1	559	28	0	20	2	0	0	0
Trial#	21	22	23	24	25	26	27	28	29	30
# of readings	13	2	0	0	2	0	0	39	0	0
Trial#	31	32	33	34	35	36	37	38	39	40
# of readings	23	55	81	185	16	16	2	16	3	1
Trial#	41	42	43	44	45	46	47	48	49	50
	0	65	15	1	0	129	2	0	0	0

Bibliography

[1] Dr. Alonso, Eduardo. The City University. Distributed Systems. http://www.soi.city.ac.uk/~eduardo/ds/ds-motivation.ppt

[2] Armstrong, Edwin and Nigel Hardy. Real-Time Virtual Sensors.

[3] Barr, Larine. Dull, Dirty and Dangerous. Next generation of UAVs hover on the horizon Military Aerospace Technology Online Edition www.mat-kmi.com

[4] Bonnet, Philippe, Johannes Gehrke, and Praveen Shshadri. Towards Sensor Database Systems. In Proceedings of the Second International Conference on Mobile Data Management. Hong Kong, January 2001.

[5] Bonnet, Philippe, Johannes Gehrke, and Praveen Shshadri. Querying the Physical World.

[6] Borphys, D., P. Verlinde, C. Perneel and M. Acheroy. Multi-level Data Fusion for the Detection of Targets using multi-spectral Image Sequences. Optical Engineering, 37(2), 1998.

[7] Bradski, Gary and Stephen Grossberg. Fast Learning VIEWNET Architectures for Recognizing 3-D Objects from Multiple 2-D Views.

[8] Buyya, Rajkumar. High Performance Cluster Computing Architectures and Systems Volume 1, Prentice Hall PTR, 1999, pp22.

[9] Charkravarthy, U. and J. Minker. Processing multiple queries in database systems. Database Eng., vol. 5, no. 3, pp.38-44, Sept. 1982.

[10] Chen, Guihai. Lecture 10 through 14: Performance Evaluation. cs.nju.edu.cn/~gchen/teaching/fpc/lecture10-14.ps accessed Mar 2, 2004.

[11] Chung, Soon M.. Enhanced Tree Quorum Algorithm for Replica Control in Distributed Database Systems.

[12] Clough, Bruce. UAV Swarming? So What Are Those Swarms, What Are The Implications, and How Do We Handle Them? Air Force Research Laboratory, Control Automation.

[13] Cole, Richard, Maxime Crrochemore, Zvi Galil, Leszek Gasieniec, Ramesh Hariharan, S. Muhukrishnan, K. Park and W. Rytter. Optimally Fast Parallel Algorithms for Preprocessing and Pattern Matching in One and Two Dimensions. Proc. 34th Annual IEEE Symposium on Foundations of Computer Science, pp.248-258, 1993.

[14] Corner, Joshua. Swarming reconnaissance using UAVs in a parallel discrete event simulation. March 2004.

[15] Cougar: The Sensor Network is the Database http://www.cs.cornell.edu/boom/2002sp/extproj/www.cs.cornell.edu/database/cougar/default.htm accessed Mar 4, 2004.

[16] Coulouris, George, Jean Dollimore, and Tim Kindberg. Distributed Systems Concepts and Design. Third Edition, 2001. Addison-Wesley. pp.387-409.

[17] Crochemore, Maxime, Leszek Gasieniec, Ramesh Hariharn, S. Muthukrishnan, and Wojciech Rytter. A Constant Time Optimal Parallel Algorithm for Two-Dimensional Pattern Matching. SIAM Journal on Computing Vol. 27, No. 3, pp. 668-681, June 1998.

[18] Crossbow Technology Inc. Smarter Sensors in Silicon. MICA2 Wireless Measurement System. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-04_A_MICA2.pdf accessed Feb 29, 2004.

[19] Defense Link U.S. Department of Defense. UAV Roadmap Briefing. March 17, 2003. Unmanned Aerial Vehicles RoadMap. http://www.acq.osd.mil/usd/uav_roadmap.pdf, accessed Feb 29, 2004.

[20] Ekin, Ahmet, A. Murat Tekalp, and Rajiv Mehrotra. Integrated Semantic-Syntactic Video Event Modeling For Search and Retrieval.

[21] Ezeife, C.I. and Ken Barker. A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System.

[22] FAS Intelligence Resource Program. Unmanned Aerial Vehicles (UAVs) http://www.fas.org/irp/program/collect/uav.htm accessed Mar 4, 2004.

[23] Gaudiano, Paolo, Benjamin Shargel, Eric Bonabeau, and Bruce T. Clough. Swarm Intelligence: a New C2 Paradigm with an Application to Control of Swarms of UAVs

[24] Ghose, Abhishek, Jens Grosskloags, and John Chuang. IEEE Distributed Systems Online. Mobile Data Management. Resilient Data-Centric Storage in Wireless Sensor Networks. Ds online exclusive, November 2003. http://dsonline.computer.org/0311/f/jen.htm accessed Mar 2, 2004.

[25] Grama, Ananth, Anshul Gupta, George Karypis, and Vipin Kumar. Introduction to Parallel Computing, Second Edition, Pearson Education Limited 2003, pp54-59.

[26] Graphs. L. Allison Computer Science. Weighted Directed Graph. http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Graph/

[27] Hardy, N.W., H.R. Nicholls and J.J. Rowland. The design of sensing commands in the InFACT assembly machine. In: Proc. 23rd Int. Symp. Industrial Robots (ISIR '92).

[28] Hunter, Jane and Zhimin Zhan. An Indexing and Querying System for Online Images Based on the PNG Format and Embedded Metadata. http://archive.dstc.edu.au/RDU/staff/jane-hunter/PNG/paper.html accessed Mar 8, 2004.

[29] Imielinski, Tomasz and Samir Goel. DataSpace: Querying and Monitoring Deeply Networked Collections in Physical Space. IEEE Personal communications, October 2000.

[30] Institute for Defense Analyses. Technology Assessments – Materials. Low-Cost Unmanned Aerial Vehicles. Core Research Areas. http://www.ida.org/IDAnew/Research/materials.html accessed Mar 1, 2004.

[31] Jain, Raj. The art of computer Systems Performance Anayalsis. John Wiley & Sons, Inc., 1991.

[32] Johnson, David B., David A. Maltz Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile Computing, Vol 353. Kluwer Academic Publishers, 1996.

[33] Kadrovach, Brian A. A Communications Modeling System for Swarm-based Sensors. September 2003.

[34] Korona, Zbigniew and Mieczyslaw M. Kokar. Model Based Fusion for Multisensor Target Recognition. SPIE Proceedings Vol. 2755, paper #2755-19, pp178-189, 1996.

[35] Korona, Zbigniew and Mieczyslaw M. Kokar. Multiresolutional Multisensor Target Identification.

[36] Krock, Lexi. Spies that fly. Timeline of UAVs. NOVA Science programming on air and online. PBS Online. http://www.pbs.org/wgbh/nova/spiesfly/uavs.html accessed Feb 29, 2004.

[37] Le Saux, Bertrand and Nozha Boujemaa. Unsupervised Categorization for Image Database Overview. INRIA, Imedia Research Group. http://pc-erato2.iei.pi.cnr.it/lesaux/papers/lesaux-visual02.pdf accessed Mar 3, 2004.

[38] Lu, Zhihong and Kathryn S. McKinley. The Effect of Collection Organization and Query Locality on Information Retrieval System Performance and Design.

[39] Lu, Zhihong and Kathryn S. McKinley. Partial Collection Replication versus Caching for Information Retrieval Systems.

[40] McLees, Lea. Learning From Experience: New Pattern Recognition & detection Technique May Help Radiologists Analyze Digital Mammograms. Jun 11, 1997.http://gtresearchnews.gatech.edu/newsrelease/MAMMOG.html

[41] Melville, Reid and Miguel Visbal. High Fidelity Analysis of UAVs Using Nonlinear Fluid/Structure Simulation. US Air Force, Air Force Research Laboratory, Air Vehicles Directorate. DoD Challenge Projects. http://www.hpcmo.hpc.mil/Htdocs/Challenge/FY03/18.html accessed Mar 1, 2004.

[42] Michalove, Aaron. Amdahl's Law, Speedup. http://home.wlu.edu/~whaleyt/classes/parallel/topics/amdahl.html

[43] Mills, Robert. Information Technology & Its Impact on the Warfighter. CSCE 525 Lecture Slides. 01-Intro to IW.ppt

[44] MindSim Corporation, 2000. Decision Making. http://www.mindsim.com/MindSim/Corporate/OODA.html accessed Mar 1, 2004.

[45] Newman, Ricard J. AIR FORCE MAGAZINE ONLINE JOURNAL OF THE AIR FORCE ASSOCIATION. War From Afar. August 2003, Vol.86, No. 8. http://www.afa.org/magazine/Aug2003/0803war.asp

[46] Olson, Clark F. and Daniel P. Huttenlocher. Automatic Target Recognition by Matching Oriented Edge Pixels.

[47] OODA Loop and Maneuver Warfare. http://prodevweb.prodev.usna.edu/SeaNav/ns310/Web%20Documents/ppt%20docs/Boyd_OODA.ppt

[48] Pan, Heping, Nickens Okello, Daniel McMichael and Matthew Roughan. Fuzzy Causal Probabilistic Networks and Multisensor Data Fusion.

[49] Peck, Michael. National Defense Magazine. Pentagon Unhappy About Drone Aircraft Reliability Rising mishap rates of unmanned vehicles attributed to rushed deployment. May 2003. fhttp://www.nationaldefensemagazine.org/article.cfm?Id=1105 accessed Mar 1, 2004.

[50] Perkins, Charles E., Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, pp234-244, 1994.

[51] Petrakis, Euripides G.M., Christos Faloutsos, and King-Ip (David) Lin. ImageMap: An Image Indexing Method Based on Spatial Similarity.

[52] Prabhu, Gurpur M.. Computer Architecture Tutorial. Amdahl's Law. http://www.cs.iastate.edu/~prabhu/Tutorial/title.html accessed Mar 2, 2004.

[53] Pucar, Predrag and Par Norberg. Decentralized sensor fusion and support using multiple models. Proceedings of SPIE vol 3068. Signal Processing, Sensor Fusion, and Target Recognition VI.

[54] Roos, Robin M. Java™ Data Objects Addison-Wesley 2003, pp1-8.

[55] searchMobileComputing.com Definitions. ad-hoc network http://searchmobilecomputing.techtarget.com/sDefinition/0,,sid40_gci213462,00.html accessed Feb 29, 2004.

[56] Seetharaman, Guna S.. A Genetic Coding Structure and Algorithms for Image Segmentation, 1996.

[57] Seinstra, F.J., D. Koelma, J.M. Geusebroek, F.C. Verster and A.W.M. Smeulders. Efficient Applications in User Transparent Parallel Image Processing.

[58] Silberschatz, Abraham, Henry F. Korth, and S. Sudarshan. Database System Concepts, Fourth Edition, McGraw-Hill Higher Education 2002, pp735.

[59] Single-Source Shortest Paths http://www.cs.umbc.edu/~lomonaco/s03/641/slides/single-source-shortest-paths.pdf

[60] Sing-Source Shortest Paths http://www.cs.umsl.edu/~sanjiv/cs278/lectures/sssp.pdf

[61] Soliday, Stephen W. A Genetic Algorithm Model for Mission Planning and Dynamic Resource Allocation of Airborne Sensors, March 19, 1999. citeseer.ist.psu.edu/soliday99genetic.html accessed Mar 2, 2204.

[62] Squyres, J., A. Lumsdaine, B McCandless, and R. Stevenson. Cluster-Based Parallel Image Processing. Journal of Parallel and Distributed Computing, 1996. 74.

[63] Stillger, Michael and Myra Spiliopoulou. Genetic Programming in Database Query Optimization, 1996.

[64] Strohmaier, Erich. Lecture 14: Performance Modeling of Parallel Applications. UT CS594 April 21, 1999. www.cs.utk.edu/~dongarra/WEB-PAGES/lect14.ps accessed Mar 2, 2004.

[65] Tackett, Walter Alden. Genetic Programming for Feature Discovery and Image Discrimination

[66] The Cornell Database Group. COUGAR: The Network is The Database hhttp://www.cs.cornell.edu/database/cougar/ accessed Mar 1, 2004.

[67] Toth, Paola and Daniele Vigo. The Vehicle Routing Problem pp1.

[68] Trahan, Michael W., John S. Wagner, Keith M. Stantz, Perry C. Gray, and Rush Robinett. Swarms of UAVs and Fighter Aircraft. The proceedings of the Second International Conference on Nonlinear Problems in Aviation and Aerospace, Volume 2, pp745-752, 1998.

[69] UAV forum. Black Widow. http://www.uavforum.com/vehicles/developmental/blackwidow.htm accessed Mar 2, 2004.

[70] United Press International. Commentary: Outside View: Looping OODA Loops. http://quickstart.clari.net/qs_se/webnews/wed/aj/Uoutsideview-lind.RpCO_Da5.html accessed Mar 1, 2004.

[71] US Marine Corps. MCDP 6 Command & Control. http://www.tpub.com/content/USMC/mpdpub6/css/mpdpub6_71.htm accessed Mar 1, 2004.

[72] Webopedia. Distributed database. http://www.webopedia.com/TERM/D/distributed_database.html accessed Mar 1, 2004.

[73] Weldon, Curt. Information Superiority for the 21 st Centurry Battlefiedl, March 20, 1997. 1997 Congressional Hearings Intelligence and Security. http://www.fas.org/irp/congress/1997_hr/h970320w.htm accessed Mar 5, 2004.

[74] Wolf, J., J. Turek, M. Chen, and P. Yu. The optimal scheduling of multiple queries in a parallel database machine. Technical Report RC 18595 (81362) 12/17/92, IBM, 1992.

[75] Wolfson, Ouri, Sushil Jajodia, and Yixiu Huang. An adaptive Data Replication Algorithm.

[76] Yao, Yong and Johanne Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. ACM SIGMOD Volume 31, Issue 3 (September 2002) pp9-18.

[77] Yates, R., C. Rose, S. Rajagopalan, and B. Badrinath. Analysis of a Mobile-Assisted Adaptive Location Management Strategy. Mobile Networks and Applications, Vol. 1 #2, pp 105-112, 1996.

[78] Yoo, Song Bong, Phillip C.-Y. Sheu. Evalution and Optimization of Query Programs in an Object-Oriented and Symbolic Information System. IEEE Trans. Knowl. Data Eng, vol. 5, no. 3, pp479-495, 1003.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
23-03-2004	**Master's Thesis**	Mar 2003 – Mar 2004

4. TITLE AND SUBTITLE
SWARM BASED IMPLEMENTATION OF A VIRTUAL DISTRIBUTED DATABASE SYSTEM IN A SENSOR NETWORK

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Lee, Wen Chian

5d. PROJECT NUMBER
If funded, enter ENR #2001001

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
Wright Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCE/ENG/04-06

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/ Information Directorate
Attn: Bob Ewing, Ph.D.
Air Force Research Laboratory Comm: (937) 255-6653x3592
WPAFB, OH 45433 e-mail: Robert.Ewing@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR/Software and Systems

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The deployment of unmanned aerial vehicles (UAVs) in recent military operations has received much media attention. Their success in carrying out surveillance and combat missions in sensitive areas has been trumpeted. An area of intense research has been on controlling a group of small-sized UAVs to carry out reconnaissance missions normally undertaken by large UAVs such as Predator or Global Hawk. A control strategy for coordinating the UAV movements of such a group of UAVs adopts the bio-inspired swarm model to produce autonomous group behavior. This research proposes establishing a distributed database system on a group of swarming UAVs, providing for data storage during a reconnaissance mission. A distributed database system model is simulated treating each UAV as a distributed database site connected by a wireless network. In this model, each UAV carries a sensor and communicates to a command center when queried. Drawing equivalence to a sensor network, the network of UAVs poses as a dynamic ad-hoc sensor network. The distributed database system based on a swarm of UAVs is tested against a set of reconnaissance test suites with respect to evaluating system performance. The design of experiments focuses on the effects of varying the query input and types of swarming UAVs on overall system performance. The results show that the topology of the UAVs has a distinct impact on the output of the sensor database. The experiments measuring system delays also confirm the expectation that in a distributed system, inter-node communication costs outweigh processing costs.

15. SUBJECT TERMS
Distributed Database, Swarm UAVs

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Gary B. Lamont
U	U	U	U	119	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565 x4118; e-mail: Gary.Lamont@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18