Air Force Institute of Technology

AFIT Scholar

3-2006

A Monocular Vision Based Approach to Flocking

Brian Kirchner

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

A Monocular Vision Based
Approach to Flocking

THESIS

Brian P. Kirchner, Second Lieutenant, USAF

AFIT/GCS/ENG/06-09

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and opinions expressed in this document are solely those of the author and do not reflect those of the United States Air Force, United States Department of Defense, or the United States government.

AFIT/GCS/ENG/06-09

A Monocular Vision Based
Approach to Flocking

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

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

Brian P. Kirchner, B.S.C.S.

Second Lieutenant, USAF

March 2006

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

A Monocular Vision Based
Approach to Flocking

Brian P. Kirchner, B.S.C.S.

Second Lieutenant, USAF

Approved:

/signed/	6 Mar 2006
Dr. Gilbert L. Peterson (Chairman)	date
/signed/	6 Mar 2006
Dr. Gary B. Lamont (Member)	date
/signed/	6 Mar 2006
Dr. Barry E. Mullins (Member)	date

Abstract

Flocking is seen in nature as a means for self protection, more efficient foraging, and other search behaviors. Although much research has been done regarding the application of this principle to autonomous vehicles, the majority of the research has relied on GPS information, broadcast communication, an omniscient central controller, or some other form of "global" knowledge. This approach, while effective, has serious drawbacks, especially regarding stealth, reliability, and biological grounding. This research effort uses three Pioneer P2-AT8 robots to achieve flocking behavior *without* the use of global knowledge. The sensory inputs are limited to two cameras, offset such that the area of stereo vision is minimal, thus making stereo image analysis techniques effectively impossible, but allowing a much larger effective field of vision. The flocking algorithm analyzes these images and updates each robot's velocity vector according to the relative position, heading, and speed of its nearest neighbor. The result of this velocity update is an eventual stabilization of speed and heading, resulting in a coherent, stable flock, demonstrated in both software simulation and in hardware.

Acknowledgements

Above all, I am thankful to Jesus Christ, without whom nothing is possible. It is solely by His grace that I am able to carry on, and to Him alone do I give the credit for the opportunity to complete this thesis.

I am also thankful to my teammate, co-worker, and friend, Lt Carrie Crews. Her support has been invaluable throughout the past 18 months, whether through encouragement, advice, support, or a simple smile and witty remark.

I thank my thesis advisor, Dr. Gilbert Peterson, for his unwavering support, guidance and aid throughout this process. I also thank Don Smith, the lab technician whose job seemed never to end as he worked on the various hardware problems that plagued this research effort. Without the support of these people I would not be where I am today.

Finally, I owe a debt of gratitude to all others who have supported me throughout this process. Family, friends, classmates, instructors, thesis committee members, and others. Thank you for everything - be it large or small.

Brian P. Kirchner

Table of Contents

List of Figures

A Monocular Vision Based
Approach to Flocking

I. Introduction

God has put a secret art into the forces of Nature so as to enable
it to fashion itself out of chaos into a perfect world system.
---Immanuel Kant

One of the most rapidly growing areas of cooperative robot control in recent years has been research on flocking. Flocking, loosely defined as an approximate matching of velocity vectors, is most often seen in nature in groups of birds, fish, and some land animals such as cattle. The ability of these relatively unintelligent animals to achieve a behavior as seemingly complex as flocking without any form of communication has puzzled scientists for years and has led many researchers to try to imitate that behavior.

Another growing area in computer science is that of visual imaging, focusing on removing the dependence on humans for effectively capturing and analyzing visual images. As scientists strive to implement truly autonomous robots and systems, the ability to receive and process visual stimuli is crucial to the success of these research efforts

This research integrates the concepts of flocking and visual imaging, applying them to actual hardware and helping pave the way towards truly autonomous robots.

1.1 Rationale

Although the concept of an unmanned aerial vehicle (UAV) has been around since the beginning of the 20th century, it is only recently that the dream of an effective pilotless aircraft has been realized. The advent of the unmanned aircraft was marked by a flurry of successful missions by the US Air Force's RQ-1 Predator during Operation Desert Storm. Since then, the success has continued, and the

Predator now holds the designator MQ-1, indicating its transition from a pure reconnaissance ('R') aircraft to a multi-role ('M') aircraft, brought about by the addition of a weapon carrying capability. Although many are excited at the prospect of an entirely unmanned Air Force, some still carry inhibitions. Former Air Force Chief of Staff, General John Jumper has publicly stated that the future UAV "has to persist for long periods of time over the battlefield and be able to survive...It has to be able to defend itself...[and] be able to air refuel in order to get that persistence" [24]. The Joint Chiefs agree, having recently commissioned a Joint Unmanned Combat Air Systems (J-UCAS) program whose objective "is to develop, demonstrate and transition an affordable, lethal, survivable, and supportable unmanned combat air system to meet the operational needs of the Air Force" [3].

One potentially promising approach to achieving this "affordable, lethal, survivable, and supportable" UAV is to model the aircraft's actions after a tried and proven system – nature. Some of the earth's most beautiful, effective and elegantly simple designs are seen in nature itself. It is with this concept in mind that we develop a simple control algorithm allowing differential drive robots to achieve flocking in two dimensions. This research is a step toward the ultimate goal of applying a flocking algorithm to UAVs, providing them with the ability to move towards an objective as a group while maintaining a cohesive formation, avoiding collisions, and all the while maintaining a passive presence in the environment by using only information local to each UAV.

1.2   Problem Statement

The amount of new information available via theoretical analysis and simulations of flocking is approaching a climax. Flocking theory has been proven many times over, in many different environments, and with many variations. In addition, most simulations have a major drawback in that they simplify the problem greatly by offering a completely accurate, omniscient knowledge of all other flockmates. That is, each bird knows the exact position, speed, and heading of all others. Some simula-

2

tions limit the birds that can be "seen" to a small surrounding neighborhood and even introduce random perturbations to the data in order to simulate sensor noise, however even this has its limitations. The more interesting challenge now is to effectively implement a flocking behavior on a hardware platform. This has been done successfully, although most hardware implementations of flocking technology have relied on some sort of global knowledge (e.g., global positioning systems) to control the flocking. Furthermore, much of the work relying on visual sensory input has used binocular cameras with stereo vision. The goal of our research is to effectively capture and analyze *monocular* camera images, applying the resultant data to a control algorithm based only on local information to achieve flocking. By implementing a monocular approach to this problem, we are able to minimize the amount of visual imaging hardware required, thereby restricting the overall cost of each robot and supporting the stated goal of minimizing the per-unit cost.

1.3 Approach

The approach to this problem taken here is twofold. First, we capture and analyze an image pair received from cameras located on a Pioneer P2-AT differential-drive robot. From this analysis, we extract the change in angular offset ($\dot{\beta}$) and time to impact (τ) of the nearest robot and use this information to update the robot's velocity. This process is decomposed into several steps, discussed briefly here and in more detail in Chapter III.

The first step in this approach is capturing an image from both cameras. Next we parse each image, looking for a specific color or colors. This portion of the algorithm several capabilities, including pure color extraction, monochrome subtraction, multiple color extraction, and analysis of the image using multiple color schemes. Once the image is parsed, the results are analyzed to determine the location of the nearest object of the specified color (in our case an orange parking cone) and $\dot{\beta}$ and τ for that object. Note that calculating these values also requires previous data points. They cannot be calculated from one stand-alone image. Next, these calculations are

used to determine an angular acceleration for the robot based upon a control equation proven to eventually result in a stable "flock" [39]. The calculation results (along with select portions of the raw data) are then sent to the robot controller, and the robot's speed and turn rate are subsequently updated.

This approach is first tested in simulation, using global knowledge but constraining the simulated robots to act within the physical capabilities of the Pioneer robots. We analyze the resulting formations based primarily on the number of coherent flocks. The complete control algorithm is then applied to the Pioneers, and we observe the results. The analysis of these results is based on using recorded odometry data to examine the path of the robots as they use the applied flocking algorithm to travel through the test environment. This applied algorithm is then compared to the simulation, and we briefly discuss a modified simulation to demonstrate robustness with respect to failing robots.

1.4 Thesis Outline

Chapter II presents a brief overview of birds, flocking behaviors, and previous work in modeling those behaviors. It begins by discussing the flocking behavior that is observed in nature, the biology behind that behavior, and the reasons for it. This provides the foundation for the next section which discusses the process by which biologists and computer scientists have reproduced this phenomenon in simulations. Clearly no simulation is perfect, however, and some of the many issues that arise when modeling flocks are discussed. This chapter then concludes with a discussion of previous work in implementing flocking behaviors on a hardware platform.

In Chapter III the mathematics behind the implementation are presented, as well as the details of our methodology. This chapter begins with a discussion of the hardware and software used and discusses the software simulation used to validate that implementing flocking with our hardware was actually plausible. A step by step explanation of the image segmentation algorithm is then provided. This is followed

4

by a discussion of the robot control algorithm and the method of communication between the two portions of the program.

Chapter IV gives an overview of the testing of the developed software and analyzes the results. The results of the initial following capability testing is discussed, with a single lead robot controlled via a simple keydrive program and the following robot using the implemented flocking software to track and follow. This is compared to the results seen in the simulation. In addition, a trial run of the complete, three robot test is compared with the simulation results. Finally, we examine a modified simulation which allows robot failure.

Finally, Chapter V concludes with a summary of the results of the project, and discusses possible extensions and improvements to the project.

II. An Overview of Flocks and Vision Based Robotics

Nothing in Nature is random. ... A thing appears random only
through the incompleteness of our knowledge.
                                    --Benedict Spinoza

The idea of creating seemingly complex behaviors with a series of simple rules
is an intriguing one. Braitenberg provides one of the groundbreaking works in this
area, which discusses a series of "thought experiments" based on one- or two-wheeled
vehicles, simple sensory input, and rules regarding reactions to those inputs [9]. He
argues that even the most basic of these machines would lead an outside observer to
ascribe "intelligence" of some sort to the vehicle based upon its actions. Since this
work, many others have researched this phenomenon of imparting intelligence and
intricate behavior patterns with minimal computational power. This chapter begins
with a brief discussion of birds and flocking and then outlines some of the previous
work in the area of flocks and machine intelligence, especially that which is relevant
to our research.

2.1 Biology of Birds

As discussed in Section 1.1, one of the primary objectives of this research, as
applied to the Department of Defense, is the application of flocking techniques to
UAV technology. Due to the three dimensional nature of UAVs, the desired behavior
is most closely associated with the natural phenomenon of bird flocking. As such,
although this research deals solely in two dimensions, it uses the biological struc-
ture and capabilities of birds as a basis, with the ultimate goal of three dimensional
extension in mind.

2.1.1 Avian Eyes. The exact biological construct of avian eyes depends
on the bird's adaptation to factors such as its normal food source, most common
predators, and other criteria [25]. For example, birds that primarily forage for food
on the ground often have eyes adapted for near vision while birds that forage from
the air or that have aerial predators have more acute far vision. In general, flocking
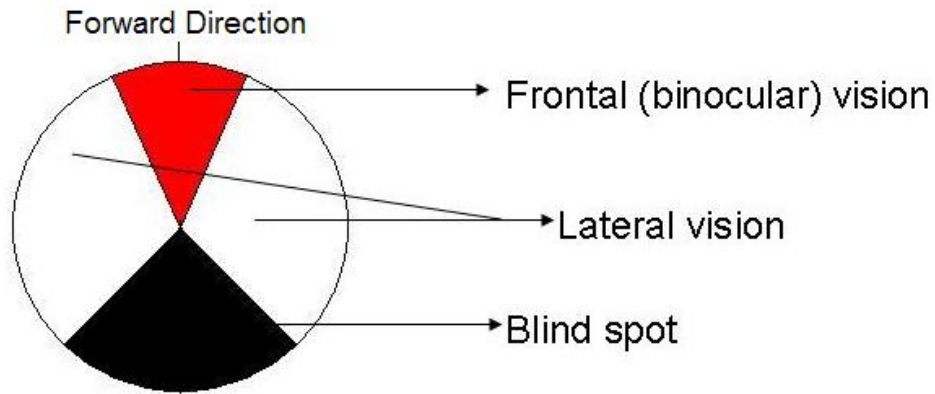
Figure 2.1:   A typical avian eye

birds have laterally aligned eyes, giving them a wide range of vision with a (usually small) overlapping area of binocular vision in the front and a blind spot in the back, as seen in Figure 2.1 [25]. There are, of course, exceptions to these generalities, such as the American Woodcock which has eyes that sit nearly on the top of its head, thus resulting in areas of binocular vision in both the front and rear. This standard physical configuration provides birds with a large field of vision, useful for self-preserving behavior such as predator avoidance and foraging.

2.2   Why Flock?

Group behaviors such as flocking, herding, and schooling are observed and studied in many species including many birds, livestock, fish, and other social animals such as wildebeests. The exact details of the grouping behavior have been shown to vary based upon factors such as speed and turn rate [40], but the basic dynamics of grouping, or flocking, remain the same.

Artificial life pioneer Craig Reynolds summarizes a bird's urge to flock as the result of evolutionary pressure from several factors including gene pool preservation, protection from predators, improvement in search area and coverage, and advantages

7

for social and mating activities [46]. That is, it is primarily for self preservation. Some birds, such as geese, also flock for aerodynamic reasons [5].

The flock structure provides many benefits to each individual member of the flock, and as such, flocking has been studied and modeled extensively. One of the most common methods of modeling flocks is by using the concept of a graph.

2.3   Modeling Flocks

2.3.1   *Graph Theory Background.*   A *graph*, $\mathbb{G}$, consists of a vertex set $\nu = \{1, 2, ..., n\}$ and a set of un-ordered pairs $E = \{(i, j) | i, j \in \nu\}$ of *edges*. The set of all edges in the graph, E, is referred to as the *edge set*. If E contains edge (i, j) then vertex i and vertex j are said to be *adjacent*. A *directed graph* is similar, but each edge, (i, j), is an ordered pair having its *tail* at i and its *head* at j. A graph is said to be *connected* if for all $(i, j) \in \mathbb{G}$, there is a path from i to j. For a directed graph, the directions of the edges are ignored for purposes of connectivity. A directed graph is then *symmetric* if $\forall i, j \in \nu, (i, j) \in E \Rightarrow (j, i) \in E$.

2.3.2   *Flocks as Graphs.*   One common method of representing a flock is as a directed graph where each bird is represented by a vertex, and an edge $(i, j) \in E$ exists if and only if j is in the *neighborhood* of i. The neighborhood of bird i, $\aleph_i$ is defined as the birds which are "visible" by bird i. The exact semantics of "visible" can vary, but Jadbabie et. al. show that if the neighborhood graph stays connected over time then all agents converge to a common heading [28]. It is also important to note that the neighborhoods are not necessarily symmetric, i.e., $i \in \aleph_j \nRightarrow j \in \aleph_i$, indicating that bird i's ability to see bird j does not necessarily mean that j is also able to see i.

2.4   Flocking Mechanics

Since birds primarily receive information about other birds through visual perception, all calculations are based upon the relative positions and velocities of nearby

birds. As first outlined by Reynolds [46] and later expanded by Lorek and White [34], the phenomenon we know as "flocking" can be simulated by employing five basic urges. These urges can be can modeled as vectors representing a desired heading, as discussed in [47], and at each time-step all vectors are calculated for each bird in the flock.

2.4.1 Flock centering. Birds exhibit a desire to be in the center of the flock. This urge is perhaps more accurately described as attraction or cohesion, since it is a result of the birds' tendency to move towards their closest neighbors. As a result, the birds in the center of the flock remain relatively stable, while those on the edge only have neighbors on one side and thus move towards the center. This behavior utilizes a simple summation of the normalized relative positions of all birds within each bird's neighborhood, weighting each relative to its influence. Thus, for each bird in the flock, i, the urge to center, $Center_i$, is given by the equation

$$\overrightarrow{Center_i} = \sum_{b \in \aleph} \overrightarrow{P_b} \times I_b \tag{2.1}$$

where $\aleph$ is the neighborhood of the bird, $\overrightarrow{P_b}$ is a normalized vector representing the relative position of bird b, and I_b is a measure of how much effect the neighboring bird, b, has on i. The value of this influence measure is discussed in more detail in Section 2.5.4. An example of the urge to center can be seen in figure 2.2[1].

2.4.2 Separation. Birds attempt to maintain a minimal distance from other birds in order to preserve a safe immediate vicinity. This natural tendency can be modeled by the equation

$$\overrightarrow{Avoid_i} = \sum_{b \in \aleph} \frac{\overrightarrow{P_b} \times I_b}{|P_b|^2} \tag{2.2}$$

---

[1]Figures 2.2-2.7 created by Bill Crowther, PhD [13]. Used with permission.

9
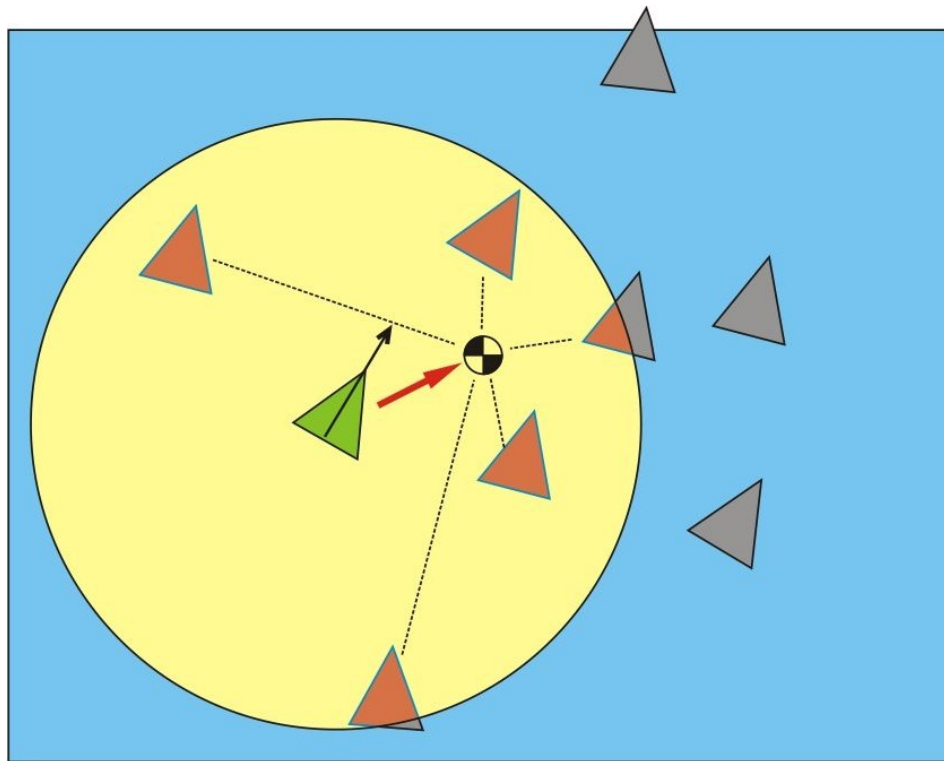
Figure 2.2:   The bird in the center of the circle represented with the green triangle
desires to move towards the center of the birds within its neighborhood.

www.manaraa.com
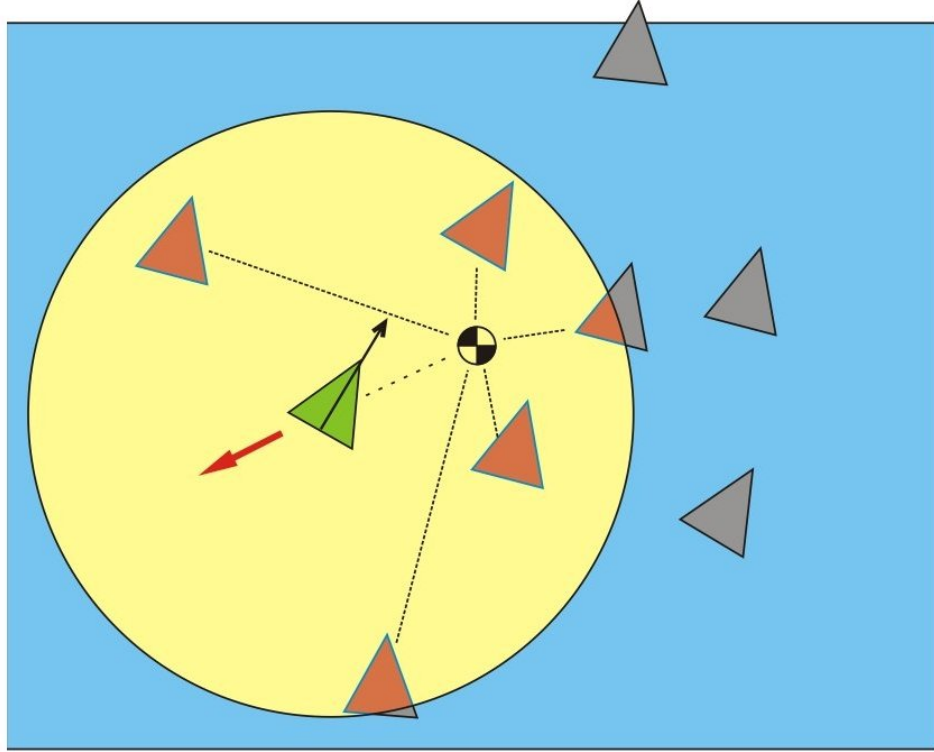
Figure 2.3:   The bird represented with the green triangle desires to move away from its neighbors to avoid overcrowding. The bird reacts the most strongly to the nearest neighbor, which exerts the most influence.

which represents a non-linear increase in the urge to move away from neighbors as the distance to them decreases. In this equation I_b and $\overrightarrow{P_b}$ are the same as above, and an example of this urge is shown in Figure 2.3.

2.4.3   Velocity Matching.   Absent other limitations or more pressing needs, a bird attempts to fly in the same direction and with the same speed as neighboring birds. This urge serves the dual-purpose of being yet another method of sustaining an obstacle-free area for flying and keeping the bird on the same (presumably beneficial) heading as its flockmates. It is computed by finding the sum over the difference between the birds current direction vector, $\overrightarrow{D_i}$, and each neighbor's direction vector, $\overrightarrow{D_b}$ as shown in Equation 2.3, again weighted to account for influence. An example is shown in Figure 2.4.

11
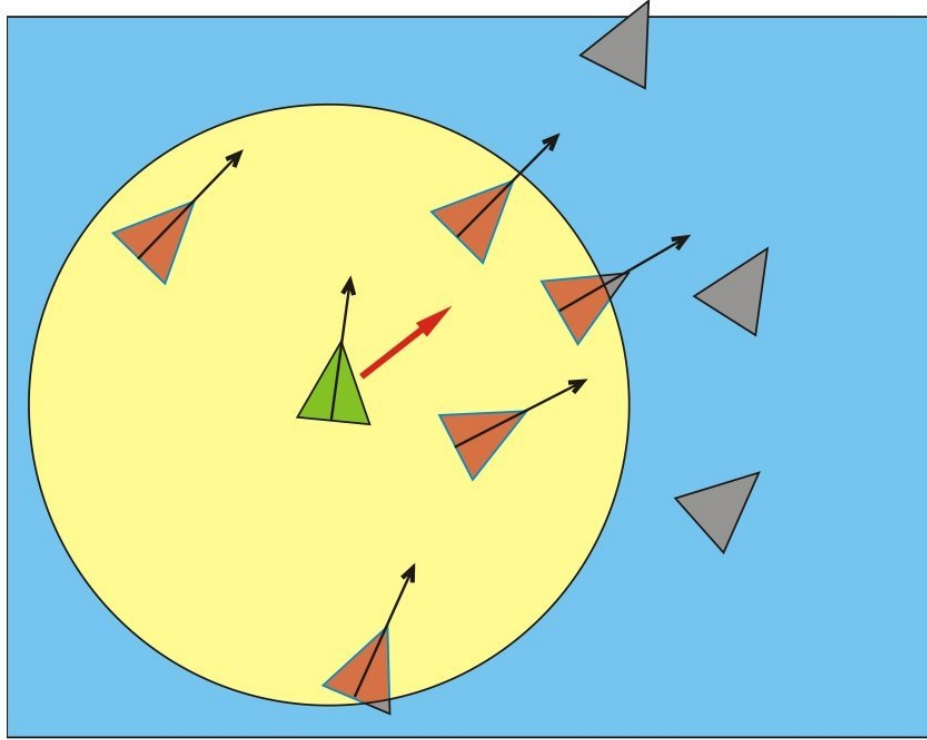
Figure 2.4:   The bird represented with the green triangle and velocity vector desires to modify its current heading and speed to match the average heading and speed of the other birds in its neighborhood.

$$\overrightarrow{Match_i} = \sum_{b \in \aleph} (\overrightarrow{D_b} - \overrightarrow{D_i}) \times I_b \tag{2.3}$$

2.4.4   Obstacle Avoidance.   Birds tend to steer away from external obstacles (including other birds). Much simulation research has been done in this area, including using object force fields or ray tracing [47], brightness gradients [35], neural networks [18, 19], and other vision-based methods [6, 26, 33]. For the purposes of this thesis effort, it is assumed that robots can only recognize or react to an obstacle, O, that is within its neighborhood, be it a flockmate, a predator, or an external object such as a tree. Furthermore, they react to the nearest obstacle only. Obstacle avoidance
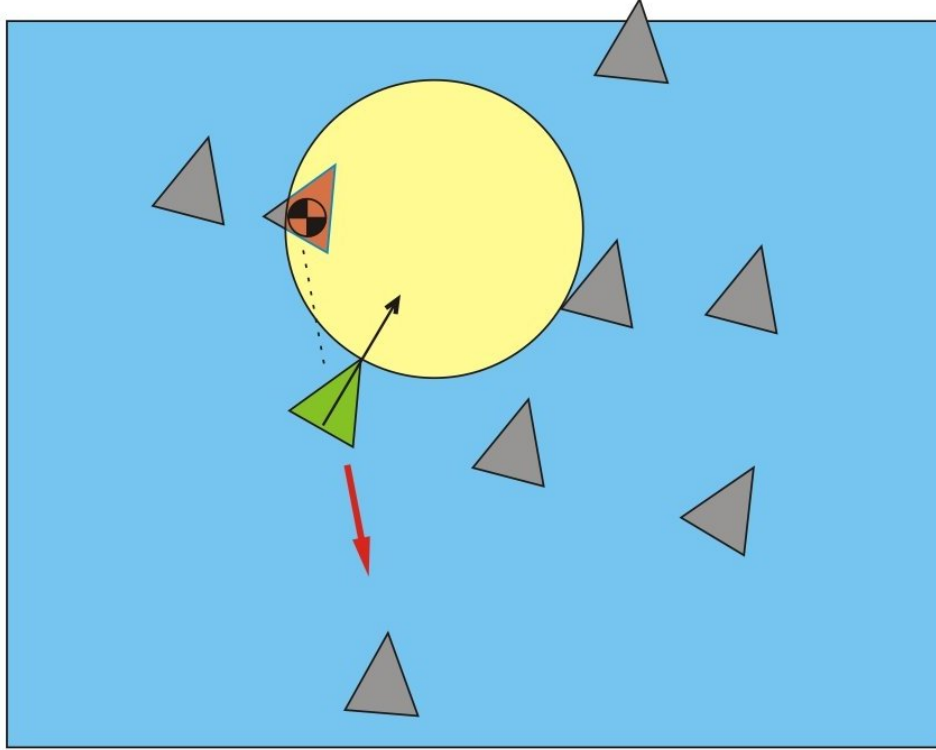
Figure 2.5: The bird represented with the green triangle and velocity vector must update its velocity to avoid the bird within its "danger zone."

is modeled for each bird, i, with the generic equation

$$\overrightarrow{Obstacle_i} = \min_{O \in \aleph} |D_O| \times I_O \tag{2.4}$$

where $\min_{O \in \aleph} |D_O|$ is the distance to the nearest obstacle and I_O is the influence of that obstacle. An example of avoiding another bird can be seen in Figure 2.5.

2.4.5 Migration (Optional). Birds in a flock may have a specific location they wish to reach or a specific direction they desire to head. For example, they may know the direction in which they are likely to find food or a new nesting place, they may be returning home, or they may be migrating. This desire is modeled by a simple vector pointing in the current direction of the goal location and is simply denoted for each bird, i, by $\overrightarrow{Migrate_i}$. Note that this urge is often overwhelmed by other more immediate urges such as avoiding a flockmate or an obstacle. Also, there is

13
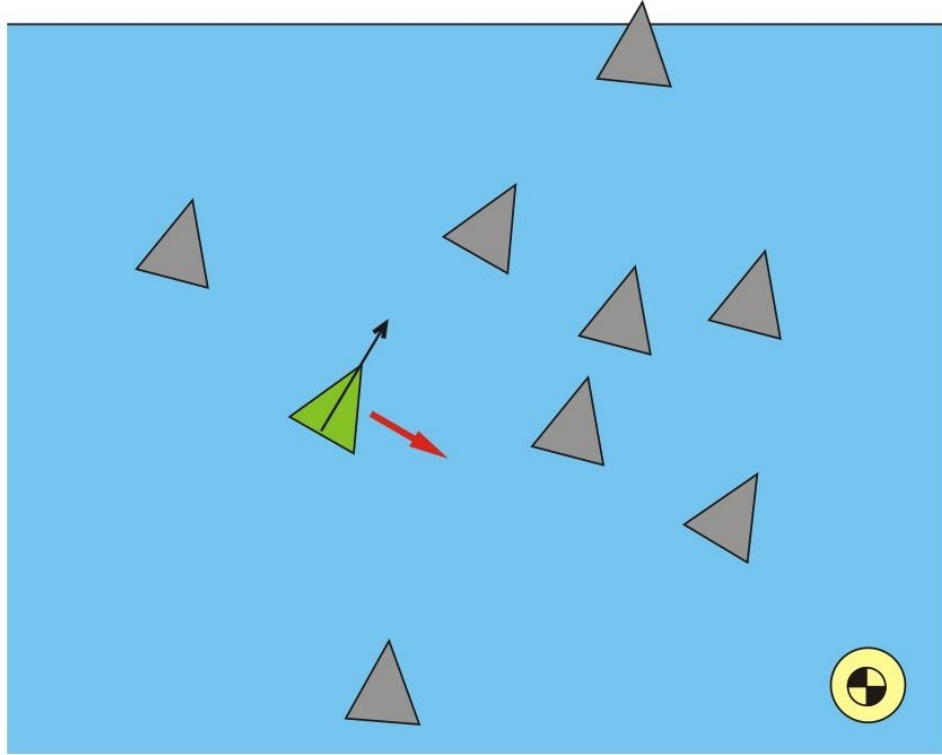
Figure 2.6:   The bird represented with the green triangle and velocity vector desires to head towards the goal in the lower right hand corner

no influence measurement in this equation. This reflects the idea that a bird's desire to migrate does not change over time. Clearly, if some model requires a changing migration urge, this equation can be modified by simply multiplying the migration urge by a scaling factor. A bird affected by the desire to migrate is shown in Figure 2.6.

2.4.6   Combining the Basic Urges.   Once each of the urges is calculated, they must be combined in some way. Even choosing a simple rule such as averaging the heading of all neighbors and adding a random perturbation can lead to, at worst, interesting behavior and at best, flocking [49]. The most basic method of combining

14
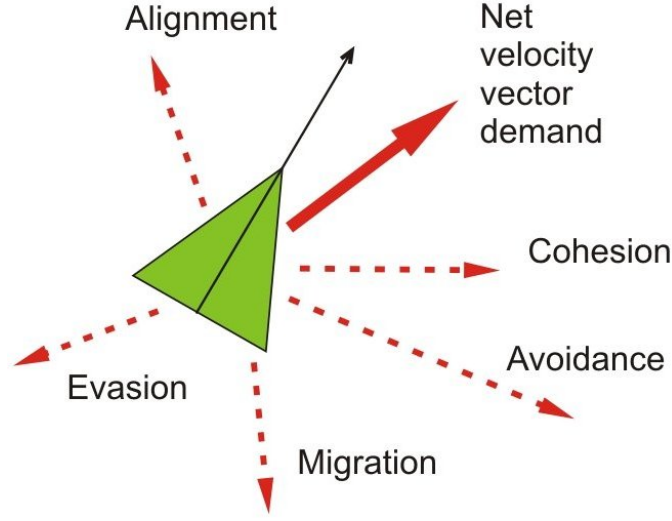
Figure 2.7: The thicker vector shows the combination (sum) of the five urges

these urges is to simply sum them, resulting in Equation 2.6.

$$
\begin{aligned}
\overrightarrow{Urge_i} &= \overrightarrow{Center_i} + \overrightarrow{Avoid_i} + \overrightarrow{Match_i} + \overrightarrow{Obstacle_i} + \overrightarrow{Migrate_i} \\
&= \sum_{b \in \aleph} \overrightarrow{P_b} \times I_b + \sum_{b \in \aleph} \frac{\overrightarrow{P_b} \times I_b}{|P_b|^2} + \sum_{b \in \aleph} (\overrightarrow{D_b} - \overrightarrow{D_i}) \times I_b \\
&\quad + \min_{O \in \aleph} |D_O| + \overrightarrow{Migrate_i} \\
&= \sum_{b \in \aleph} (\overrightarrow{P_b} \times I_b + \frac{\overrightarrow{P_b} \times I_b}{|P_b|^2} + (\overrightarrow{D_b} - \overrightarrow{D_i}) \times I_b) \\
&\quad + \min_{O \in \aleph} |D_O| \times I_O + \overrightarrow{Migrate_i}
\end{aligned}
\tag{2.5}
$$

An example of this approach is seen in Figure 2.7. Previous work, however, has operated on the realization that the simple aggregation of data does not always produce ideal results [47]. For example, given a strong urge to turn right and an equally strong urge to turn left, it is almost certainly not the best course of action to sum the impulses and maintain a course straight ahead.

Since a simple summation of the urges does not usually produce ideal results, another method of aggregation must be defined. One common method, proposed by Lorek and White, is to construct a hierarchy of urges such as the one shown in Figure

15
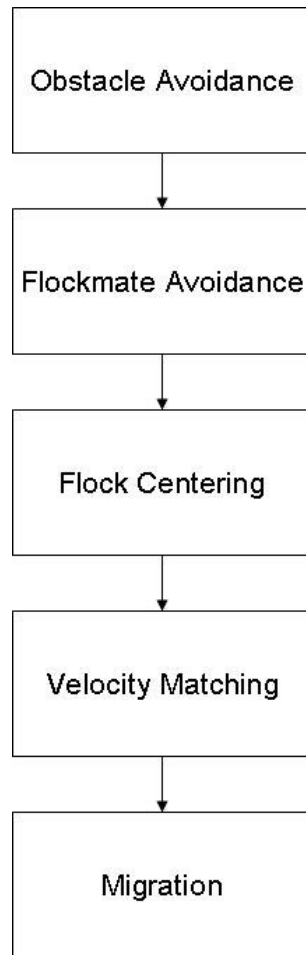
Figure 2.8:   Sample Hierarchy of Urges

2.4.6 [34]. The idea behind the hierarchical approach is that each bird evaluates each of the basic urges in a pre-specified order based upon importance. Usually, flockmate avoidance and obstacle avoidance come at the top of this list for obvious reasons. A bird's sense of self-preservation (i.e., not crashing) naturally comes above its desire to maintain a given heading. Likewise, in most cases we desire any developed physical system to preserve itself, possibly even at the cost of failing its specified task.

Once the order of importance is determined, each urge is calculated in order and optionally multiplied by a scaling factor which [34] calls an *urgency value*. This occurs for each of the basic urges and they are then added to the composite urge with the following stipulation: if at any point the desired acceleration exceeds the physical

16

limits of the bird, the calculation is terminated and the maximum velocity change in the appropriate direction is immediately executed. This means that a very strong urge to avoid a flockmate could not be balanced out by migration and centering urges. This hierarchy can also be thought of as a priority queue, or modeled as a decision tree, i.e., "if desired acceleration within physical limits, then evaluate next input." Lorek and White note that manually scaling the urgency values "can be a tedious and enervating process," but did not have time to fully explore alternatives to this method in their initial research [34].

This method of processing sensory inputs in a parallel manner as opposed to using a serial approach was first popularized by Brooks and is now know as behavior-based architecture [10]. This architecture is characterized by independent modules which serve a specific purpose (in our case alignment, cohesion, etc.) and is a significant departure from the previously used deliberative paradigm, which focuses on building an internal model of the environment and using path-planning techniques to determine subsequent actions. The deliberative architecture is extremely computationally intensive and poorly suited for most real-time tasks.

Clearly, not every flock should act the same way. Especially as technology advances, varying goals require different types of swarms. For example, one of the long-term possibilities of this technology is that it would allow "nanobot swarms" - millions of microscopic robots working together to achieve complex tasks. The National Air and Space Administration (NASA) is currently funding such a project. Their Autonomic NanoTechnology Swarm (ANTS) project "is a generic mission architecture consisting of miniaturized, autonomous, self-similar, reconfigurable, addressable components forming structures" [14]. Ideally, such a project would utilize very inexpensive, easily manufactured robots so that even losing several thousand would not have an adverse effect on their behavior. In this case we can easily forsee a behavior set that emphasizes task completion over self-protection. Mataric has begun to explore the possibility of varying behavior sets by defining and classifying various robot behaviors. By considering different methods of combining basis behaviors such

17

as following and dispersion, it is possible to classify the resulting behaviors into an overlying class of robots such as foragers or flockers [37].

2.4.7 *Resultant Behaviors.* The interaction between the birds' urges often results in some interesting group dynamics. For example, the leaders of the flock can tend to change rapidly due to flock centering and frequent changes in direction. Biological research indicates that any individual can initiate a flock movement [43]. There are some restrictions, however, as in most circumstances, the flock only reacts to birds initiating a movement towards the center. This follows from the five urges since birds on the outside edge are reluctant to split away from the flock. Even if they do, the neighboring birds are more likely to follow the flock than a single bird turning away. This behavior also is intuitive from a personal protection standpoint, since turning away from the flock means turning away from protection and a bird's best means of finding food.

The introduction of outside actors (wind, trees, visual stimuli, other birds, etc.) can also causes a large amount of flock movement. Each movement by a single bird causes an effect that first extends to its neighbors and then spreads throughout the flock. Since each bird is affected by all its neighbors, however, this tends to spread any error throughout the flock, causing some variation in heading for each individual bird, but very little overall effect on the flock [51].

Also, as illustrated by [50], birds tend to veer side to side within a flock much more often than forward or backward. This effect can be easily visualized by considering a car travelling at 65 mph as shown in Figure 2.9. Even a 10 degree change in direction causes only a one mph difference in forward velocity, while the same change causes an 11 mph difference in side to side velocity.

Other methods of modeling interaction between birds have also been studied including ad-hoc network creation [31] and evolving self-organized behavior for homogenous and heterogeneous UAV swarms [44].
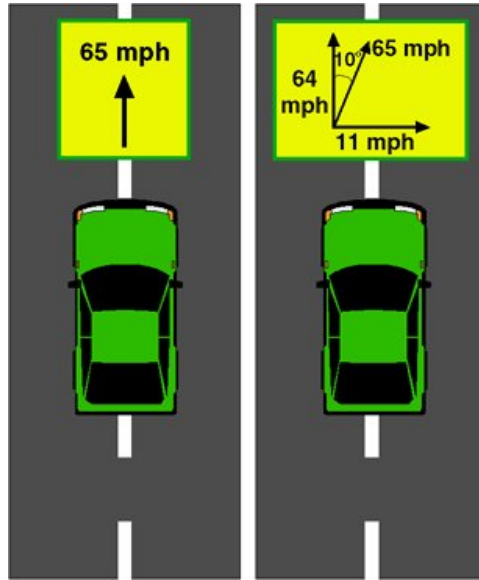
Figure 2.9:   A slight course adjustment may cause significant side-to-side movement, but usually results in a relatively minor change in forward speed [50]

2.5   Other Flocking Issues

Although many researchers have observed and modeled flocks, no one experiment or model has incorporated every known flocking behavior or phenomenon. The following subsections discuss some of the many observations and issues that should be considered when observing or implementing any flocking algorithm.

2.5.1   Size Limits.   Natural observation has not revealed any limit on the size of flocks except the limit imposed by the number of birds in the area. When an additional bird joins, the only effect is a slightly larger flock. Likewise, when two flocks near each other, they tend to merge to form one large flock, and the only other effect is a possible heading change. The impracticality of reacting to every bird in the flock indicates that a bird is affected mostly by its nearest neighbors. Reynolds suggests that birds are aware of the flock as a whole only in the broadest sense, that it exists. He contends that each bird mainly concentrates its vision on its two or three nearest neighbors. The addition of another bird to the flock, therefore, is essentially noticed by only a couple birds. Reynolds also observes that whatever algorithm is

19

used to achieve flocking in nature, it must be a constant time algorithm since a bird's decision process appears to be independent of the size of the flock [46].

2.5.2 *Fish and Dancers.* As discussed so far, the flocks of birds could just as accurately be described as schools of fish or herds of some "flocking" land animals. The major difference between these is simply a matter of vision. While birds have excellent vision, fish cannot see nearly as well and their vision is further affected by the clarity of the surrounding water. Likewise, land animals often have relatively poor vision (since they have less need to see long distances) and they have more trouble seeing past their neighbors since most of the time they essentially operate in two dimensions [46].

The enhanced vision of birds, as compared to other animals, brings about some interesting consequences. The most widely researched of these is the effect on reaction time. The birds' vision seems to provide a very quick reaction time with regard to group movement. A sudden turn, for example, propagates throughout the flock much more quickly than it should based upon measured reaction times. Potts hypothesizes that this happens for the same reason as it does in a "chorus line" of dancers, where the same phenomenon can been observed [43]. Once the movement begins its propagation through the flock, the birds can see it coming and anticipate when it will reach them, timing their reaction to occur with minimal delay and minimal effect on the overall flock coherence. In his experiments he notes that a bird's mean startle reaction time to a light flash as measured in the laboratory is 38 milliseconds. The first birds in a flock to respond to an initiator take 67 milliseconds to react. Once started, however, maneuver waves spread through the flock at a mean speed of less than 15 milliseconds [43].

2.5.3 *Too Perfect.* In nature, a bird never has complete and perfect information about its environment. It is therefore inaccurate to model objects within a perfectly defined environment. Previous simulations implementing flocking behavior

in birds have demonstrated this fact, resulting in spontaneous convergence to the center of the flock by all the birds and other non-naturally occurring behaviors [46].

Reynolds' initial experiment models this uncertainty by defining a sphere of cognizance around each simulated bird. The sensitivity to actions decreases exponentially as the distance from the bird increases. He notes that based upon the biology of a bird and its eyesight, this should not really be a sphere, but should be forward weighted. Although most birds only have 10 to 15 degrees of overlapping forward perception (and thus a small area of accurate depth perception), they are able to shift their sensitivity forward while flying and become more aware of the birds in front than of those on the sides. This weighted sensitivity is affected by speed among other factors. Reynolds, however, did not implement this feature and reports some inconsistent behavior in his results such as birds in the front of the flock becoming distracted by birds behind them [46].

2.5.4 Influence. The exact method by which birds are affected by their neighbors is not known, however it is clear by their actions that they are not affected equally by all other birds within their field of vision. Schnapauff notes that if this happens, with a spherical range of vision, birds form spherical flocks. There are a variety of ways to determine the influence which one bird has on another. Schnapauff, for example, calculates the influence of bird b, (I_b), with Equation 2.6, where C_1 and C_2 are constants and $\overrightarrow{P_b}$ is the vector indicating bird b's relative position. In this equation, C_2 is constrained to be between 0.0 and 1.0, reflecting the fact that closer birds have more influence [47].

$$\overrightarrow{I_b} = C_1 \times C_2^{\overrightarrow{|P_b|}} \tag{2.6}$$

2.6 Robotics

Much research has been done regarding flocking in the field of robotics. This section does not intend to be an exhaustive survey, but rather an overview of the re-

21

search most applicable to our project. For a more comprehensive review of cooperative robotics, the reader is referred to [4].

The most successful area of research within this field, in terms of achieving coherent flocks, uses global, or universal knowledge to achieve behaviors consistent with those of flocks found in nature. This area of research assumes that each robot has complete and accurate knowledge of the position, speed, and heading of every robot within its neighborhood. This is often accomplished via open channel broadcast communication (wireless networks, etc.) or via a central controller or central omniscient source of information. Research operating under the assumption of global knowledge has made great strides including robust obstacle avoidance and formation control [6].

Nembrini et. al. achieved flocking using a short range, omnidirectional radio robust even under noisy conditions [30]. Hayes and Dormiani also focus on the robustness aspect of flocking, allowing a robot to essentially drop out of the picture entirely and still maintain formation through an obstacle field [23]. This work does, however, rely on an overhead camera system to emulate sensory information. Jadbabaie et. al. [28] extend the work of Vicsek [49] using neighbor information with the variation of including a predefined leader. [28] proves that using the averaging algorithm defined in [49], each robot within the flock ultimately converges upon the same speed and heading as the leader. Finally, Tanner et. al. present flocking within a variety of graph topologies, all using some form of omniscient knowledge of other robots' positions and speeds [21, 22].

Other research efforts have focused more towards purely localized techniques to achieve flocking. Fredslund and Matarić [17] have developed a leader-based algorithm to form a pre-specified formation, however this requires cameras, sonars, and even uses radio communication to determine the number of robots in the flock. Martinelli et. al. are able to localize (i.e., determine the position of) multiple robots by using an Extended Kalman Filter to aggregate proprioceptive (self-sensing) and exteroceptive (outward sensing) data [36]. A team out of Sherbrooke, Canada, led

22

by Francois Michaud, has developed an approach to achieve multiple formations, including formation initialization, again using cameras, sonar, and a form of wireless communication [16].

Some other unique approaches are also present within the many variations on the flocking theme. For example, Shen et. al. use hormones as a basis for communication, expanding on Turing's reaction-diffusion model. Robots "lay" hormones which diffuse over time. Subsequent robots may or may not sense these hormones based on their proximity to the locus of the hormone [48].

2.7 Other Applications of Flocks

Although flocking technology is most naturally applied to groups of mobile robots, it also is applicable to many other areas. Even in his original paper on flocking, Reynolds recognized the potential of flocking as applied to the fields of traffic flow modeling and other particle flow research, its applicability to computer graphics in the generation of large crowds of "extras" in films, or even the animation of (as of yet) unrealistic scenarios such as spaceship traffic jams or herds of pogo sticks [46].

Other applications include research on panicking crowds and how to create a safer environment in case of an emergency [15] and creating a realistic driving simulation for teaching purposes [41]. Another area that has recent a lot of recent attention is the application of flocking techniques to the hunter/prey problem [12, 38, 45]. Finally, flocking behavior is currently being developed for use in path planning techniques [8].

2.8 Future Flocking Efforts

Great strides have been made in the area of autonomous vehicles and many of the principles and strategies underlying this research are presented within this chapter. One of the ultimate goals, however, is to implement flocking behavior in a robust, cost-effective, computationally efficient manner on a hardware platform.

No one has yet been able to completely achieve this goal, and the remainder of this document focuses on our work in continuing the effort to expand these ideas.

III. Methodology

Order is not sufficient. What is required, is something much more
complex. It is order entering upon novelty; so that the
massiveness of order does not degenerate into mere repetition; and
so that the novelty is always reflected upon a background of
system.

---Alfred North Whitehead

Our research focuses on expanding the ideas presented in Chapter II, developing a flocking algorithm based solely on local knowledge. This chapter details our implementation of this algorithm which includes capturing images from two cameras, analyzing those images, and using the information gained to adjust the velocity of our robots.

3.1 Hardware

The developed software runs on Pioneer P2-AT8 robots (one robot is shown in Figure 3.1) designed and manufactured by ActivMedia Robotics [1]. The robots come equipped with a 16 sensor sonar ring and front and rear bump sensors as well as two USB ports. The '*AT*' in the name of the robots indicates that they are equipped for all-terrain, including the ability to climb 45% grades, travel at speeds up to 0.7 mph, and to either rotate in place using all four wheels or use only the wheels on one side to turn with a 40 cm radius.

We have modified the robots to include two IEEE 1394 (Firewire) ports and a 1.6 GHz Pentium Mobile onboard processors with 1 GB of Random Access Memory (RAM). The robots support wireless communication to a host computer via an 802.11b PC card.

For sensory input, two Videre Design [2] MDCS-C cameras (as seen in Figure 3.2) are mounted on each robot as the sole input device for our flocking algorithm. The MDCS-C is a compact, low-power camera with an IEEE 1394 (Firewire) digital interface. The cameras are mounted with 3.5 mm lenses allowing an 125 degree field of view (FOV). These cameras capture images with a resolution of 640 x 480 pixels at a maximum rate of 60 images per second. These cameras are mounted on each robot

25
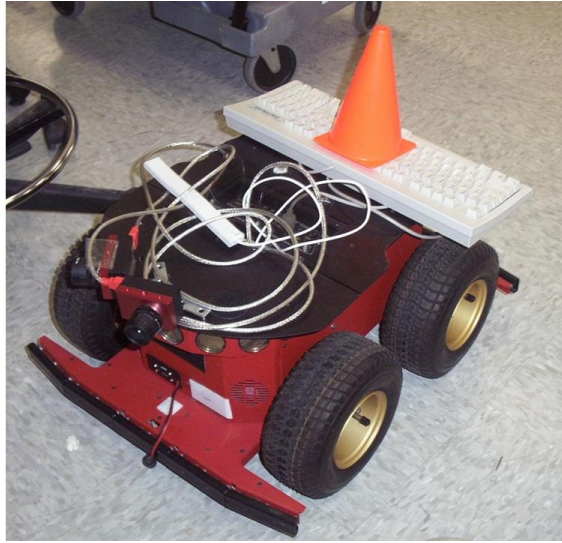
Figure 3.1:   A Pioneer P2-AT8 Robot

Figure 3.2:   An MDCS2-C Camera

26
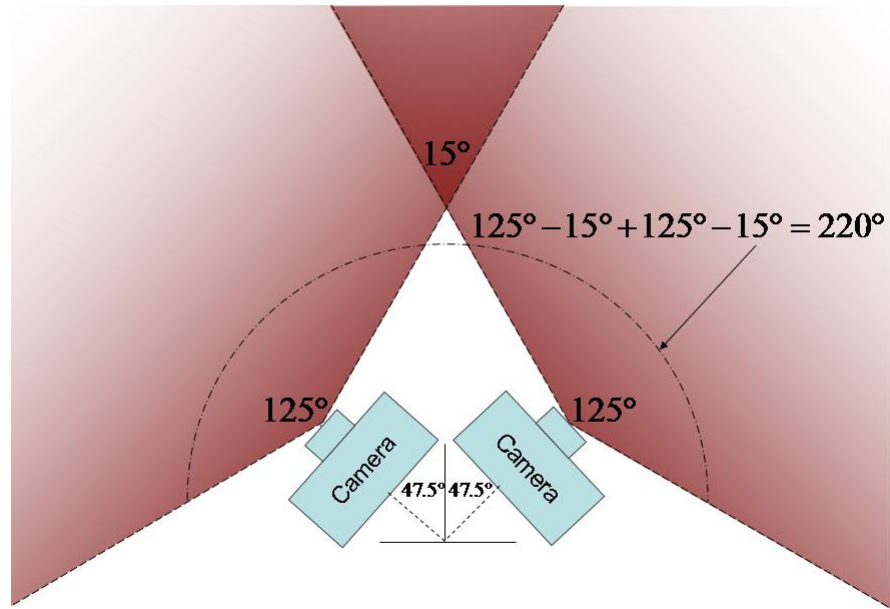
Figure 3.3:   Hardware Configuration for the Cameras

with an offset of ± 47.5 degrees. As shown in Figure 3.3 , this configuration results in an overlap of 15 degrees and allows a 220 degree vision overall. The bundled Videre Design DCAM software provides a means to view the dual images as they appear to the robot. This is shown in Figure 3.4.

3.2   Software

The DCAM software distributed with the MDCS cameras provides an application program interface (API) for capturing, opening and controlling images from the cameras. The DCAM software library is utilized for real-time capture of the camera images.

Player is an open-source program which provides a network interface to a variety of robot and sensor hardware, allowing robot control programs to be written in any programming language and run on any computer with a network connection to the robot. Player is used for controlling the Pioneer robots.

27

Figure 3.4:   View of the world as seen by the robots

3.3   Simulation

One of the preliminary steps in the project is to ensure that flocking is achievable with the hardware configuration of the Pioneer robots. As one can imagine, it might be difficult for some vehicles to achieve flocking. Any discrepancy between the courses of two supersonic jets with short range cameras, for example, would quickly take the jets out of visual range of each other.

To test the flocking capability of our P2-ATs, we use a flocking application written by Buckland [11]. For purposes of mapping our simulation to real distances, we choose to implement a ratio of 38.1 pixels in the simulation environment to 1 foot, since the first calculation results in a whole number (although any ratio works as long as it is consistent throughout). One of the primary concerns is speed and turning radius of the robots. The Pioneer specifications list a 0.8 meter per second top speed. A conversion to English units gives

$$\frac{0.8m}{s} * \frac{3.281ft}{m} = \frac{2.625ft}{s} \tag{3.1}$$

Using the pixel ratio above, this gives a top simulation speed of $2.625 * 38.1 \approx 100 \frac{pixels}{sec}$. The Pioneer platform can rotate in place by moving the wheels on either side of its body in opposite directions, and so the turning radius is actually zero. Since this is

28
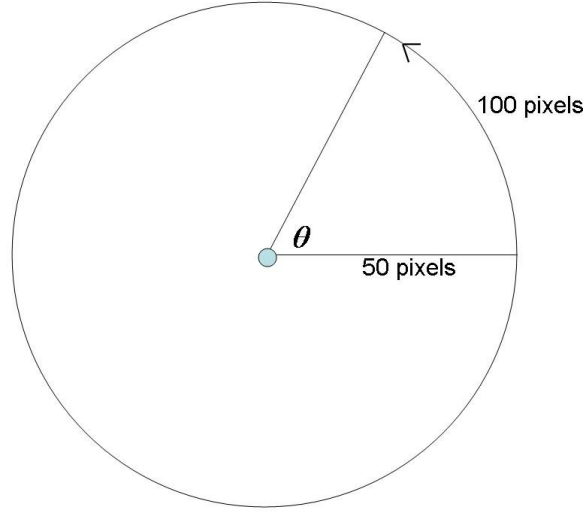
Figure 3.5:   Vehicle capability (units in pixels)

not realistic for most platforms, however, and since we do not desire our robots to stop during flocking, the turning radius is restricted to the motion gained by moving wheels on one side only, which is listed at 40 cm. Again, converting to feet gives

$$40cm * \frac{ft}{30.48cm} = 1.31ft \tag{3.2}$$

which is $1.31 * 38.1 \approx 50$ pixels.

These calculations indicate a robot capable of moving 100 pixels per second with a turn radius of 50 pixels as shown in Figure 3.5. Recall that the ratio of an arc length to the radius of a circle is the radian measure of the central angle which that arc subtends. Therefore, Θ is given by the simple equation $\frac{100}{50} = 2$ radians. Thus we are able to substitute a turn rate of two radians per second into Buckland's parameter set. Another modification required was visual range of the Videre Cameras. The effective range of the camera is approximately eight feet, or $8 * 38.1 \approx 305$ pixels.

The final modification to Buckland's application is updating the vision model which is slightly more problematic than the other adjustments. Buckland's original method of determining the neighbors of a given vehicle uses a square box centered on the vehicle with one side equal to the vision distance calculated above (see Figure

29

3.6(a)). Any vehicle within this range is considered a neighbor. Not only does this allow an actual range that is further than the given vision range (in the corners of the square), but it also allows rear vision, which our Pioneers do not have. As noted in Section 3.1, our robots actually have a 220 degree range of vision. Due to the setup of Buckland's application, it is easiest to relax our vision assumption somewhat and assume a 180 degree field of vision in the simulation. Therefore, the new function determines neighbors by first of all limiting the original box to only include the area in front of the vehicle (see the box in Figure 3.6(b)). Then all potential neighbors within this box are subjected to a second test which determines whether or not the distance to the neighbor is less than the the visual range of the vehicle. The area within this range is shown by the circle in Figure 3.6(b). The darker area is the intersection of the box and circle and gives the desired 180 degree arc of vision.

Given these calculations, the parameters are almost determined, however an empirical analysis of a two robot simulation shows that with Buckland's default urge weights, the simulated robots' ideal distance from each other is approximately 10 pixels. Rather than manually tweak the simulations cohesion and separation weights in order to arrive at an ideal distance of 38.1 pixels $\times$ 3 feet = 114 pixels, this research instead condenses the robot and a 104 pixel "safety bubble" into the point at which the robot in the simulation is located. Therefore, the 10 pixel simulation distance represents the $104 + 10 = 114$ pixel (three foot) safety bubble implemented in the flocking algorithm. This is illustrated in Figure 3.7. The visual range of the robot is 305 pixels, and subtracting the 104 pixel collapsed area gives a simulation vision range of 305-104 = 201 pixels.

Once the parameters are adjusted as described above, the simulation can run. Several runs were tested, including the basic run with 3 vehicles, no predators and no obstacles, as well as runs including many vehicles (up to 20), runs with obstacles, with predators and with combinations of these variations.

(a)

(b)
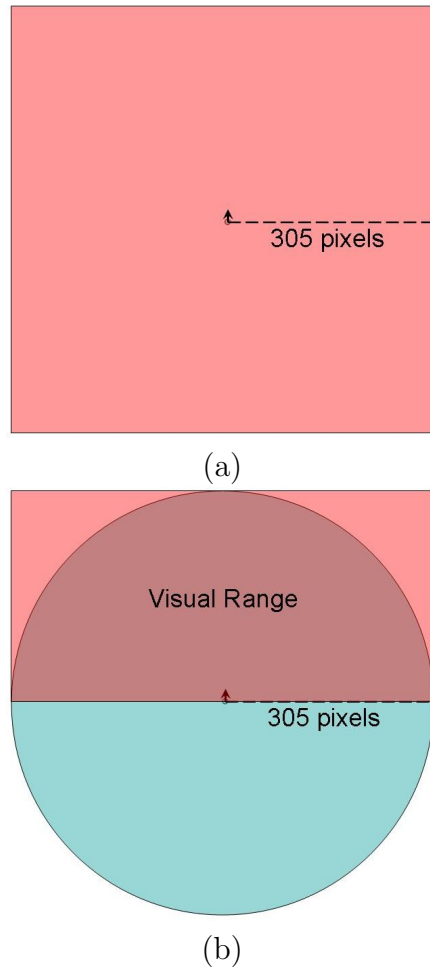
Figure 3.6:   (a) The original vision as implemented by Buckland (b) The modified vision, where the darker area (inside the forward facing hemisphere) gives the area of the neighborhood around the robot

There are varying degrees of flocking, and the definition of "successful" flocking is somewhat arbitrary. The decision of whether it is more desirable to have two separate highly cohesive flocks or one somewhat dispersed flock, for example, is largely based upon the application and/or personal preference. Since our version of flocking is ill-defined ("an approximate matching of velocity vectors"), there is plenty of room for interpretation.

In our runs, however, the simulation gave very definitive results. In all of the test cases using our hardware configuration (described above), the simulated robots

31
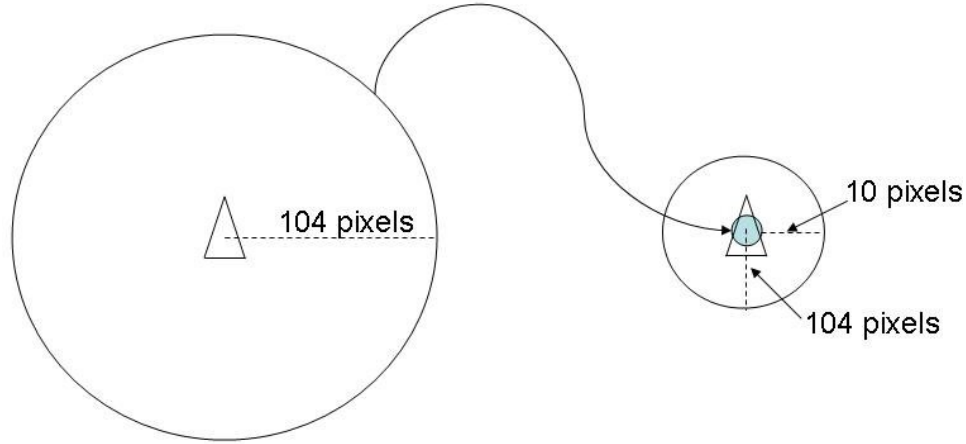
Figure 3.7:   Condensing the "safety bubble"

formed highly cohesive flocks quickly (within one minute), indicating potential success for the basic scenario as well as a great potential for future extensions to the scenario.

This research defines a "flock" within the simulation as a group of robots in which no robot is separated from the others by more than 48 pixels (i.e., within a foot of ideal distance). The results are then analyzed based upon the number of flocks at the end of one minute. Figure 3.8 shows the results of a typical test using three robots with a random initial configuration and the Pioneer hardware configuration described above. The large octagon is representative of a wall enclosing the area, and the flock after one minute is outlined with a box. The robot separated from the rest is a randomly wandering predator robot which the others avoid when it comes within visual range. This was introduced to the simulation to force variations in the paths of the other robots. Without a predator, the robots had a tendency to follow the wall indefinitely. In the case where all robots followed the wall in the same direction, flocking did not occur. Although this is a legitimate result (if the robots never see each other they can't be expected to flock), it does not provide useful information. The introduction of a predator also has the added bonus of testing the cohesiveness of the flock under more difficult conditions. Even with the predator, the flock tends to coalesce quickly and maintain its shape. After one minute in this environment,

32
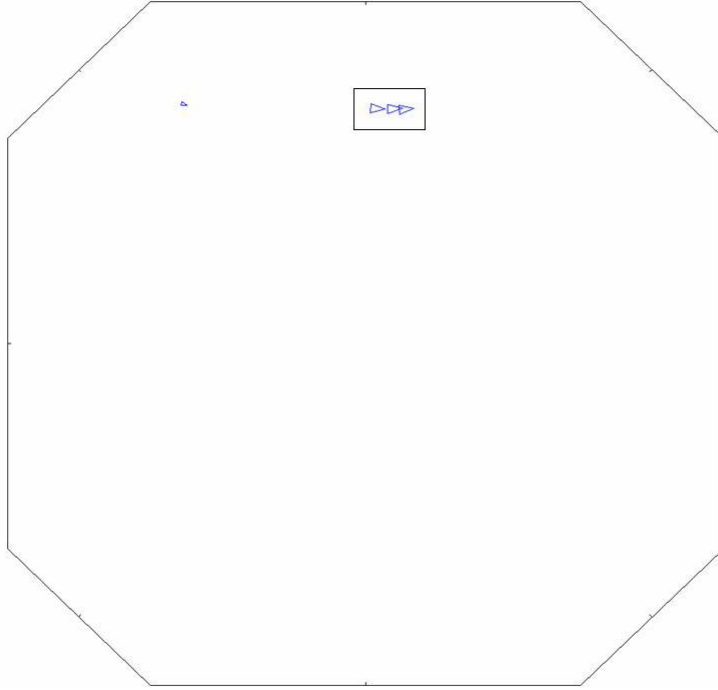
Figure 3.8:   Three Robot Simulation

the centroid of the flock was located at pixel coordinate (507, 118) and the standard deviation of the distance of each robot from the centroid location was about 14.7.

The simulation is also repeated five times with the same hardware configuration, but increasing the number of robots to twenty, with one predator. A typical result is shown in Figure 3.9. Again, the flocks (two in this case) are each outlined with a box. Their centroids are located at (223,640) for the smaller one and (375,664) for the larger, with standard deviations within each flock of approximately 15.1 and 45 pixels, respectively. Although the larger number of robots made the flock more susceptible to splitting apart from each other, it tended to rejoin quickly after the initial separation.

It is important to note that this simulation uses complete global knowledge to achieve flocking rather than our flocking algorithm as well as differing from our developed software in many other ways. The intent of the simulation is *not* to prove plausibility of our algorithm, rather to show plausibility of the hardware's flocking ca-

33
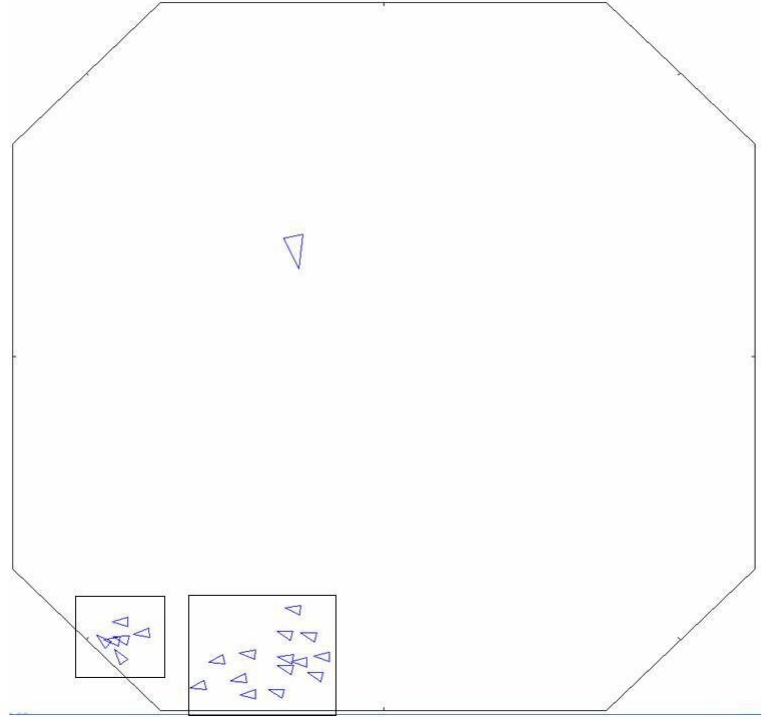
Figure 3.9:   Twenty Robot Simulation

pabilities; to guard against developing an algorithm for hardware physically incapable of flocking. As an example of poorly flocking hardware, a simulation with vehicles having the same speed and turn radius, but a vastly inferior visual capability was run five times each for both 3 and 20 robot setups. As seen in Figure 3.10 (3 robots) and Figure 3.11 (20 robots), reducing the camera range to 50 pixels (4 feet) caused a significantly worse performance from the robots. The results are less noticeable with fewer robots, though still worse than the previous configuration. The adverse effects of the reduced vision are greatly amplified, however, when considering the 20 robot setup. After one minute the robots are still in eight distinct groups. Each of these groups considered as a stand-alone entity has a standard deviation which is comparable to the first configuration (mean standard deviation of 9.2 pixels), but the discrepancy in the number of separate flocks highlights the poor performance. Furthermore, even when the simulation is allowed to run for 20 minutes, the number of distinct flocks fluctuates throughout that time, but never approaches the success seen in the earlier configuration. Any outside actor on a group of robots (another nearby

34
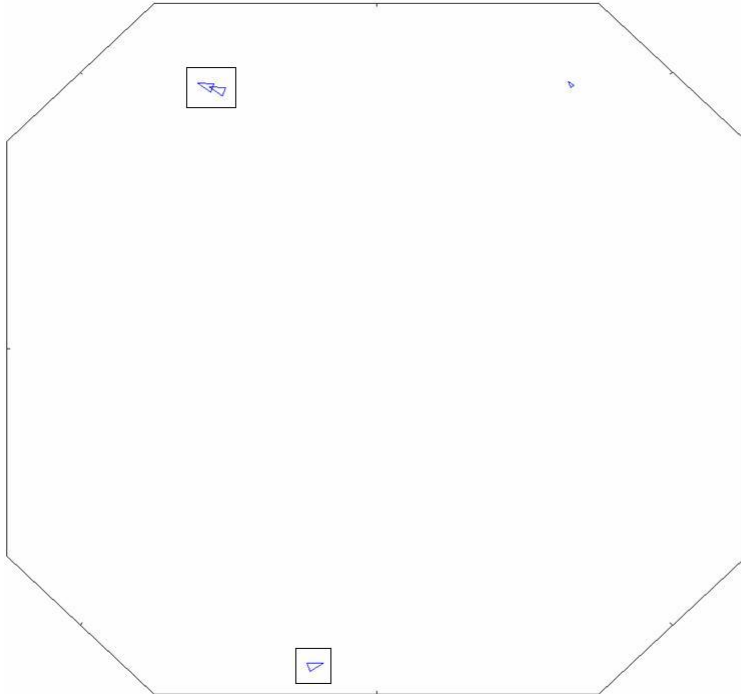
Figure 3.10:   Relative Positioning of the Observed
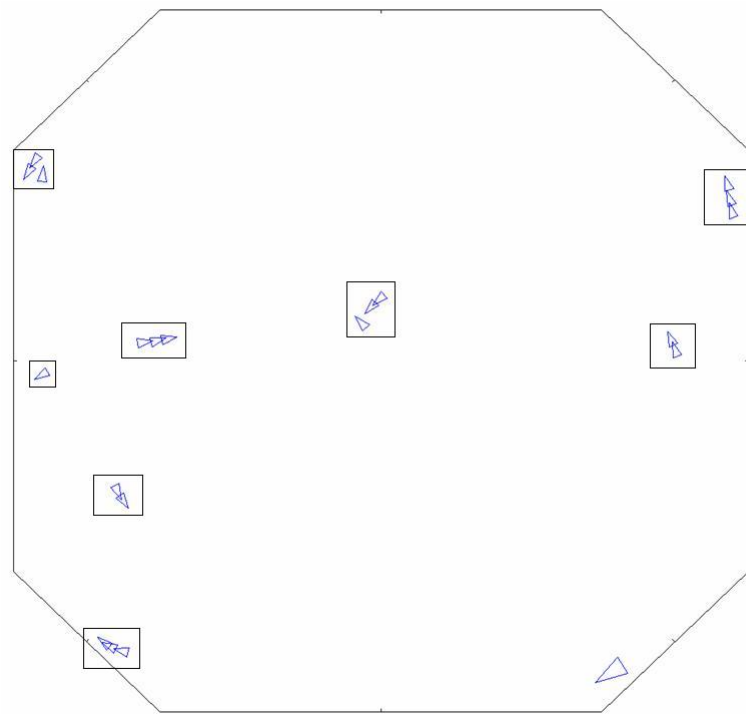
Figure 3.11:   Relative Positioning of the Observed

35

robot travelling in a different direction, a nearby wall, or the predator) quickly causes the stabilized robots to lose sight of each other and diverge.

3.4 Image Analysis

Some researchers have conducted experiments which suggest that many birds are able to estimate time to collision with a given object in the birds field of vision [32,52]. This time to collision estimation can be simulated in image processing either by using pixelwise optical flow techniques or by calculating the rate at which an object appears to change size and position. This research uses the latter approach in estimating the time to collision. Although not as accurate, it is much less computationally intensive and useful for a quick, efficient estimation.

For purposes of these calculations, two fundamental assumptions exist:

1. The birds are spherical in shape

2. The birds are fixed in size with radius r

These assumptions are made in lieu of more complicated image processing not directly associated with the research. The first assumption is convenient for math calculations since it ensures that the apparent size of a bird does not change based on its orientation, however more advanced image processing techniques would allow this assumption to be removed. The second assumption is more difficult to remove, however it is a reasonable one, especially when considering mass manufactured products such as robots or UAVs.

Using the notation from [42], let α represent the angle that bird A, the "observed," obstructs within bird B, the "observer's," vision field and β represent the angular offset of A relative to B (see Figure 3.12).

As seen in Figure 3.13, for a bird with radius r we know that

$$\alpha = 2 * \arctan(\frac{r}{|d| + r}) \tag{3.3}$$

36
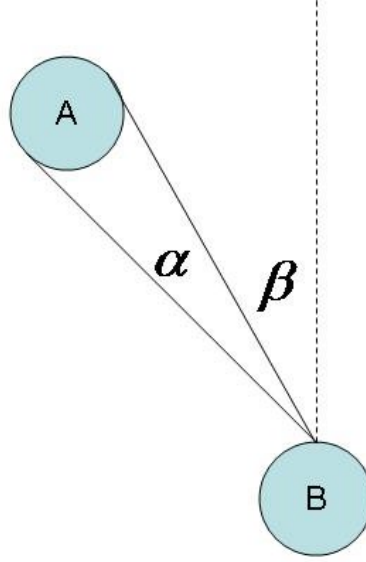
Figure 3.12:   Relative Positioning of the Observed

Or equivalently,

$$\tan(\frac{\alpha}{2}) = \frac{r}{|d| + r} \tag{3.4}$$

where d is the vector representing the position of bird A relative to B. Therefore the magnitude of d is

$$|d| = \frac{r}{\tan(\frac{\alpha}{2})} - r \tag{3.5}$$

To estimate the rate of change of the distance, the distance at the previous time-step, $|d|_{t-1}$ can be subtracted from the current distance, $|d|$ which is given by Equation 3.5.

$$\frac{\delta|d|}{\delta t} = |d| - |d|_{t-1} \tag{3.6}$$

The estimated time to impact, τ, then, is simply the current distance divided by the closure rate [32]:

$$\tau = -\frac{|d|}{\frac{\delta|d|}{\delta t}} \tag{3.7}$$

This calculated distance is prone to some error based upon both camera inaccuracy and the nature of the calculations. Even if the camera were 100% accurate, the distances still could not be measured precisely since, due to the resolution of the
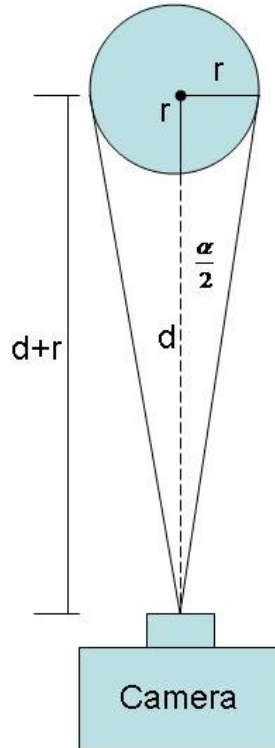
Figure 3.13:   Calculation of distance for a spherical bird of radius r

camera, a run length represents a (potentially large) range of distances. This type of inaccuracy is most noticeable at farther distances. For example, using our formula, a run of 15 pixels would result in an estimated distance of 76.21 inches. A run of 16 pixels would result in a distance of 71.32 inches. To gain a better understanding of the inaccuracy that this presents, a series of distance tests were run. An orange parking cone was placed at 6 inch intervals and distance measurements were taken using our software. The results were as expected – at closer distances, there is less constancy between the tests, but a lower standard deviation. At further distances the results are more constant, however if there is a deviation, it is more significant, due to the previously mentioned resolution issue. The complete results of the testing can be seen in Appendix A.

38

3.5 Locomotion

Once the necessary data items are calculated, namely β and τ, Moshtagh et. al. provide a method by which we can update the heading of the observer in order to achieve heading alignment and flocking [39]. They prove that providing angular acceleration, ω, as given in Equation 3.8 results in a stable flock.

$$\omega_i = \dot{\theta} = -k \sum_{j \in \aleph_i} \frac{\dot{\beta}_{ij}}{2} \sqrt{1 + \frac{1}{(\tau_{ij} \dot{\beta}_{ij})^2}} \tag{3.8}$$

where $\aleph_i$ is the "neighborhood" of the observer, i.e., those birds the observer sees, and k is a coupling constant.

This method of flocking using purely local information as opposed to global knowledge is a significantly different approach from much of the previous research regarding flocking. Moshtagh et. al. does not provide a hardware implementation of their method, but do provide a proof of correctness and simulations indicating that their method works practically as well as theoretically.

3.6 A Flocking Algorithm

The following section presents the details of our flocking algorithm. Although Equation 3.8 is used as a basis for our motor control, several important features of our hardware create a significantly different environment than was seen in [39].

3.6.1 Deviations from Moshtagh.   Although the formula derived by Moshtagh et. al. [39] provides the basis for our control algorithm, some modifications were made in order to conform to our specific needs. Specifically, this research modifies Moshtagh's assumptions on neighbors and robot speed.

3.6.1.1 Nearest Neighbor.   The first of the modifications is that while the control law given in [39] is based upon the information for all other robots within range of the robot (within its neighborhood), our software currently limits the scope

39

of the research by only considering the *nearest* neighbor. This can be extended later with more advance computer vision techniques.

3.6.1.2 Constant Speed. Another change from [39] is a departure from their assumption that all robots move at an equal, constant speed. Although a small change in our robot's control behavior would allow us to fix our robots at a constant speed as well, we allow each robot to accelerate or decelerate based upon the distance of its nearest neighbor as discussed in Section 3.6.8.

3.6.1.3 Neighborhood area. The final significant deviation from Moshtagh et. al. is that they assume a 360 degree field of vision with a fixed radius. The major implication of this is that the neighborhood relationship then becomes symmetric. So for every robot, j_n, in the neighborhood of robot i, i is also in the neighborhood of j_n which is inconsistent both with natural models (as seen in Figure 2.1) and our hardware (as seen in Figure 3.3). Instead of this approach, the robots are limited to responding to only those neighbors visible with our camera setup as depicted in Figure 3.3.

3.6.2 Algorithm Overview. In order to comply with the assumptions made in Section 3.4 and to focus research efforts on the flocking algorithm as opposed to the image recognition portion of the software, some simplifications are made. The most significant of these is that an orange parking cone is placed on top of each robot. The primary benefit of this is that it allows the "robots" (although in actuality we are observing only the cones) to appear the same from any angle. That is, the cone looks the same from the front as from the side, and, given its size (which is predetermined in accordance with the second assumption in Section 3.4), estimating the distance to the cone by measuring the cone at its widest point (the base) is easy.

The general approach taken to achieve flocking consists of the six sequential steps encapsulated within a loop seen in Algorithm 1.

Algorithm 1 High Level loop
while TRUE **do**
Capture image from each camera
Segment each image, looking for a specific color or multiple colors
Extract closest cone from image
Calculate change in velocity necessary to achieve flocking
Send information to robot controller
Update velocity
end while

3.6.3 Capture image from each camera.   The use of the DCAM software package makes the image capture portion of the program as simple as calling a single command. Although DCAM does not support synchronous image capture with non-stereo cameras, the capture time was found to be negligible, making consecutive calls (one for each camera) introduce minimal time disparity between the images.

3.6.4 Segment each image.   The segmentation portion of the software passed through several modifications throughout development, and thus has multiple capabilities. The segmentation algorithm is based on Zachary Gray's implementation [20] of the work of Bruce et. al. [29], and utilizes a masking algorithm which extracts one or more colors from an image.

3.6.4.1 RGB Masks.   The masks are created by capturing a color image with the DCAM software and analyzing it with the GNU Image Manipulation Program (GIMP) to determine the range of numeric values associated with each color of interest. This range of numeric values is then stored and is used in extracting the color from any image. The original algorithm analyzes the color using the red, green, blue (RGB) color scheme with each value ranging from 0 to 255. For example, the orange parking cone used in this work has color values ranging from 60-110 in red, 0-20 in green, and 0-20 in blue, resulting in the mask shown in Figure 3.14, where each digit in the mask represents a span of ten values. The segmentation algorithm, then, examines each pixel of a given image and extracts those which fall within the ranges specified by the mask. It then redraws the picture, coloring all pixels within

41

```
R = {0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
G = {1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
B = {1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
```

Figure 3.14:   An example mask for parking cone orange color

the specified range gray, and all others black. An example of the results of this segmentation process are shown in Figure 3.15.

3.6.4.2   Monochrome Subtraction.   Although the masking algorithm as described above does work, it turns out that it is extremely sensitive to lighting changes. Small changes to the camera aperture, variations in lighting arrangements, and even shadows from passing people affect the segmented picture. This happens in spite of built-in camera software which adjusts the camera aperture based on ambient light to mitigate the negative effects of varied lighting situations. This problem can be partially solved by simply expanding the range of values accepted by the mask, but at the cost of reducing the ability to discriminate between similar colors (e.g., red and orange).

One method of mitigating this problem consists of ignoring the darkness and lightness values associated with each pixel, concentrating only on the color. This is achieved by considering the RGB value of each pixel and subtracting the monochrome value from it. The monochrome value indicates the value that would be given a pixel if the picture was converted to black and white (monochrome), and is given by the equation

$$Pixel_{mono} = 0.2989 * R + 0.5870 * G + 0.1140 * B \qquad (3.9)$$

where R, G, and B are values ranging between 0 and 255. Note that under the RGB color representation scheme if all three numbers are equal, the resulting color is a shade of gray. The value (0,0,0) is black, (255,255,255) is white and all other values fall somewhere in between. The monochrome value, then, although usually referred to as one number ($Pixel_{mono}$), is really the shade of gray associated with the pixel value ($Pixel_{mono}$,$Pixel_{mono}$,$Pixel_{mono}$). The monochrome value is subtracted from

42
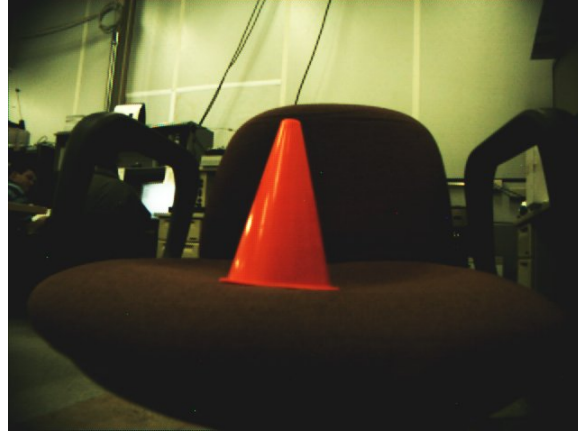
(a)
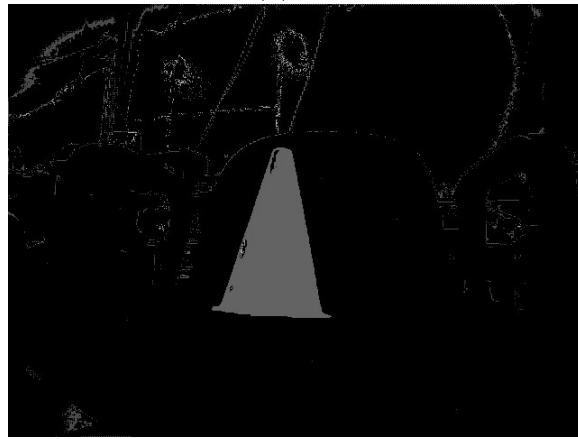
(b)

Figure 3.15:   Color Segmentation: The original image (a) is segmented, looking for the color of the cone, resulting in (b).

the pixel and the final pixel RGB value is given by the equation

$$Pixel = (R - Pixel_{mono}, G - Pixel_{mono}, B - Pixel_{mono}) \qquad (3.10)$$

Once again, although this modification does improve the cone extraction algorithm, the monochrome subtraction is not as effective as expected and the color extraction algorithm is still sensitive to minor changes in lighting. This necessitates nearly constant monitoring of the camera apertures, and a complete readjustment in any new environment. This is not an optimal situation for robust flocking behavior.

43
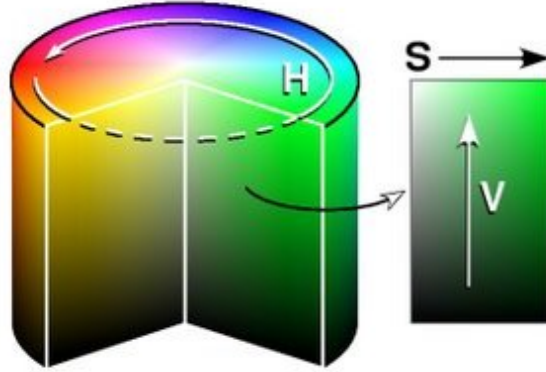
Figure 3.16:   The HSV Color Scheme

3.6.4.3 HSV.   An alternative to analyzing the RGB values of each pixel is to use the hue, saturation, value (HSV) color scheme. Although somewhat less common than RGB, the HSV scheme is often used in imaging applications because of its similarity to the way the human eye perceives color. Whereas the RGB color scheme is additive, describing each color as a composition of the primary colors, HSV encapsulates the same information with different values. The HSV scheme can be thought of as a cylindrical object as seen in Figure 3.16. Hue refers to the basic color (e.g., blue, red, orange) and ranges from 0-360 (around the color wheel on top of the cylinder). Saturation is essentially the "vibrancy" or "purity" of the color, and ranges from 0.0 to 1.0. A low saturation value indicates an impure color and is characterized by a grayish or faded appearance. Value simply refers to the brightness of the color and also ranges from 0.0 to 1.0.

The HSV values are obtained by a non linear transformation of the RGB values as follows. We first normalize the RGB representation so that given a color value (R,G,B), we let

$$R = \frac{R}{255} \qquad G = \frac{G}{255} \qquad B = \frac{B}{255} \tag{3.11}$$

and R, G, and B are between 0.0 and 1.0. We let MAX be the largest value of R, G, and B, and MIN be the smallest of R, G, and B. The H, S, V values are then given

44

by the following equations:

$$H = \begin{cases} 60 \times \frac{G-B}{MAX-MIN} & \text{if } MAX = R, \\ 60 \times \frac{B-R}{MAX-MIN} + 120 & \text{if } MAX = G, \\ 60 \times \frac{R-G}{MAX-MIN} + 240 & \text{if } MAX = B \end{cases} \qquad (3.12)$$

$$S = \frac{MAX - MIN}{MAX} \qquad (3.13)$$

$$V = MAX \qquad (3.14)$$

Note that several special cases need to handled as well. H can be less than 0 (for instance if $MAX = R$ and $B > G$). In this case, we add 360 to H so that the resulting value is between 0 and 360. Also, if $MAX = MIN$ (i.e., S=0.0) then H cannot be computed, and it becomes undefined. Finally if $MAX = MIN = 0$ (i.e., V=0.0) then both H and S are undefined. From the cylinder in Figure 3.16, we see that these exceptions make sense intuitively as well. If $MAX = MIN$ and thus S=0.0, then R, G, and B must all be equal, so the resultant color is a shade of gray. This explains the smaller gray cylinder within the cylinder modeling the HSV color space. Furthermore, if $MAX = MIN = 0$, the RGB color must be (0,0,0) which is black, as is the color in the HSV color scheme whenever V=0.0.

Using the HSV color scheme, our segmentation algorithm performs much better under varying lighting conditions. The likely reason for this is that by separating hue and value, varying intensities of a specific hue are extracted, whereas in the RGB scheme looking for varying intensities necessarily entails looking for varying hues. A sampling of pictures revealed that a standard orange pixel has a hue between 5 and 20, a saturation greater than 80%, and a value greater than 50%. Extremely dark spots or bright spots such as the glare on the cone in Figure 3.15(a) still do not fall within this range (as can be seen in the segmented image in Figure 3.15(b)), but this is compensated for during the cone extraction phase of the algorithm.

(a)
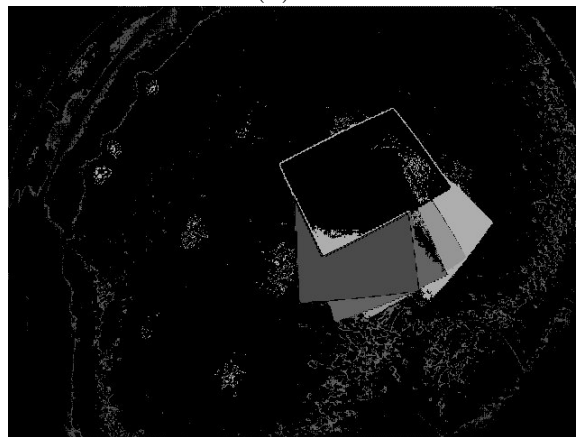
(b)

Figure 3.17:   (a) The original picture with (from top to bottom) orange, green, red, salmon and yellow papers (b) The resulting picture after the color segmentation algorithm was applied to extract all green, red, salmon and yellow colors (but not orange)

3.6.4.4   Multiple colors.   Our software also includes the ability to extract multiple colors from an image. Given a list of colors (in the form of HSV or RGB color ranges), it extracts them from the image. If the option to save the segmented image to a file is used, then each color is saved as a different shade of gray. It is important to note that if the ranges overlap, then order can matter. A pixel which falls into multiple color ranges is segmented as the matching color which is *last* in the list. Figure 3.17 shows an example of a multiple color extraction.

3.6.5 Extract closest cone from image. As each pixel is compared against the given mask, the algorithm also analyzes them, looking for contiguous, or nearly contiguous, runs of pixels of the desired color (in our case - orange). Determining the longest of these runs gives the width of the closest cone within the image. These runs of pixels are extracted by including a simple algorithm that looks for runs as the segmentor loops over every pixel in the image. Since this does not require a separate nested loop, this is a fairly inexpensive addition to calculation times. The basic logic is based upon whether or not the "run detector" is currently in the middle of a run of orange pixels. If it is, then it adds pixels to the run until the end of a row, or until the pixels are no longer orange If the run detector is not in the middle of a run of orange pixels, it simply looks for an orange pixel to start a new run. Also, to compensate for glare, etc. the run detector allows "holes" of up to 5 consecutive non-orange pixels within a run. Thus a series of 12 orange pixels, 4 non-orange pixels and 7 orange pixels would be evaluated as a run of 23 as opposed to two separate runs of 12 and 7 pixels. The complete algorithm is shown as Algorithm 2.

Algorithm 2 Calculate longest run of orange pixels

---
for row 1..numRows **do**
  for column 1..numColumns **do**
    segment pixel
    if pixelIsOrange **then**
      if !inRun **then**
        runBoolean = true
        runStart = pixel
      end if
    else if (inRun AND pixelsSkipped = 5) OR endOfRow **then**
      runBoolean = false
      pixelsSkipped = 0
      if longest run so far **then**
        store beginning and end of run
      end if
    else if inRun **then**
      pixelsSkipped = pixelsSkipped + 1
    end if
  end for
end for

3.6.6 Calculate change in velocity necessary to achieve flocking. As discussed briefly in Section 3.5, Moshtagh et. al. have described a method by which a collection of vehicles ultimately achieve flocking, using only the offset of nearby neighbors (β), and the time to impact of those neighbors (τ). This is accomplished by applying an angular acceleration as given by Equation 3.8.

At this point, all the necessary information is determined from the current image and the flocking algorithm proceeds with the calculations. The first additional item needed is alpha (α), the area of the view obstructed by the nearest cone (measured in degrees). Since the FOV of each camera is known, as well as the resolution, we determine the degrees represented by each pixel. For our specific camera, the FOV is $125°$ and the resolution is 640×480. Therefore, we know that every $\frac{640 pixels}{125°} = 5.12$ pixels is equivalent to one degree. The number of pixels obscured by the base (the widest part) of the cone is then used to determine the angle obscured by it. Thus

$$\alpha = Run_{max} \times 5.12 \qquad (3.15)$$

Where Run_{max} is the longest run of orange pixels.

This information allows us to compute the distance of the cone from the camera using Equation 3.5, using the calculated value of alpha and substituting the cone radius of 2 inches for r.

Using the information from the image segmentation, we can also calculate the angular offset (β) of the closest cone/robot. This is done in two steps. First, the angular offset of the center of the cone ($Cone_c = Run_{end} - Run_{start}$) from the center of the camera (Cam_c) is calculated using the following equation:

$$\beta = \frac{Cone_c - Cam_c}{640} * 125 \qquad (3.16)$$

The next step is that β is updated based on which camera observed the cone. The cameras are angled at 47.5 degrees off center, so if the left camera observed the

48

cone, 47.5 degrees is *subtracted* from the angular offset, otherwise 47.5 degrees is *added* to the offset.

Finally, the distance information from the previous image is used in estimating the closure rate of the closest cone and this value is used to calculate τ as described in Equation 3.7. This completes all the preliminary calculations and the equation proposed by Moshtagh et. al. (Equation 3.8) is used to calculate omega (ω).

3.6.7 Send information to robot controller. Once the calculations are complete, ω , the closure rate, $|d|$, and β are sent to the robot controller via User Datagram Protocol (UDP) packets. In this case, the robot controller is located on the same machine as the image processor, so the UDP packets are simply sent to another port, however this is not a requirement. The image processing, for example, could execute on a separate (faster) CPU and the information sent via wireless communication. Alternatively, the program supports Transfer Control Protocol (TCP), and this can be used instead of UDP. The only part of the UDP/TCP communication worth discussing specifically is that if no cone is found in either image (i.e., no orange pixels are extracted), then the predetermined distance of 111.1 (a distance value determined to be impossible for the robot to calculate) is sent to the robot controller to indicate that no cone is in view.

3.6.8 Update velocity. Using the information received from the image processing program, the robot control algorithm updates the robot velocity via the Player software. If no cone is in view (if the controller receives the "no cone" code of 111.1), the robot maintains its current velocity and travels straight ahead. Otherwise, the robot's actions are determined based upon the distance of the closest cone. The robot's ideal distance from each other is 36 inches. If the distance is greater, then the robot increases its speed by an amount proportional to the closure rate, given by the equation

$$|V|_{new} = |V|_{old} + \frac{closureRate}{150} \tag{3.17}$$

49

Note that the closure rate is positive when the robots are moving further apart, so $\frac{closureRate}{150}$ is always a positive number, resulting in an increase in speed.

If the robots are less than 36 inches apart (too close), the robot slows down by a preset amount until it either reaches a minimum speed or ends up at least 36 inches away. If it is less than 36 inches away, is at the minimum speed, and is still getting closer to the other robot, it turns away. This turn rate is calculated to be inversely proportional to the distance between the robots. That is, as the robots get closer, the turn rate increases so they are turning away more forcefully. Also, the turn rate increases as β approaches zero, causing a harder turn if the robot is straight ahead than if it is at the edge of the view. This turn rate is calculated with the equation

$$Turn_{new} = Turn_{old} + 15 - \frac{|\beta| + (d_{cone} - 10) * 4}{21.4} \qquad (3.18)$$

Since d_{cone} ranges between 10 and 36 inches and $|\beta|$ ranges between 0 and 110, this equation returns a turn rate between 5 and 15. Note that as the distance or angle increases, the turn rate decreases.

As a final failsafe, if the robots get close enough to each other that they are in danger of crashing (less than 10 inches) the speed is immediately set to zero. The turning logic remains the same, so the robots turn away until the other is out of view (or further than 10 inches if the other robot is moving). This portion of the logic does not aid in flocking, it is simply a safety mechanism to avoid collisions.

3.6.9 (Optionally) Record Odometry Data. For purposes of testing, the final robot controller software also has the ability to record odometry data. The odometry data is calculated by the robot's internal CPU and includes its current x-position, y-position, theta (heading), velocity, and angular velocity. With this option enabled, every time an image pair is processed, the robot's current odometry information is simply appended to a text file titled "odometry."

IV. Results and Analysis

```
...and the thousands of fishes moved as a huge beast, piercing the
water. They appear united, inexorably bound by common fate. How
comes this unity?
```
```
                                                ---Anonymous, 17th Century
```

This chapter presents the results of applying the software described in the previous chapter to a number of Pioneer robots. We discuss tests with both two and three robots, including tests focused on basic following capability, avoidance capability, and the ultimate goal of flocking capability. This chapter also includes an analysis of the simulation results, including a simulation which includes the possibility of robots failing.

4.1 Two Robot Tests

To initially test the capabilities of our algorithm, the first series of trials is performed with two robots, each having a specified role. One was designated as the follower and ran our software (including the option to record odometry data as discussed in Section 3.6.9) using the basic hardware configuration as discussed in Section 3.1. The other robot did not use any cameras. Rather, it was controlled by a simple keydrive program which controls the motors and records the odometry data to a file in the same manner as the following robot. Figures 4.1, 4.2, and 4.3 show a sequence of tests depicting the software's ability to extract the cone from an image and successfully update the robot's heading based on the cone's bearing change and time to impact. In each figure, the solid line depicts the path of the lead robot controlled by the keydrive program. Note that this robot generally starts six feet in front of the following robot, whose path is depicted by a dashed line.

Figure 4.1 shows a standard test, with the lead robot beginning 2 meters in front, at position (6.28,0), and then initiating a wide turn. As the lead robot begins the turn the following robot reacts immediately and initiates an inside turn, closing the distance. Not depicted by the graph is the fact that the following robot slows down as the distance closes and maintains a safe following distance.
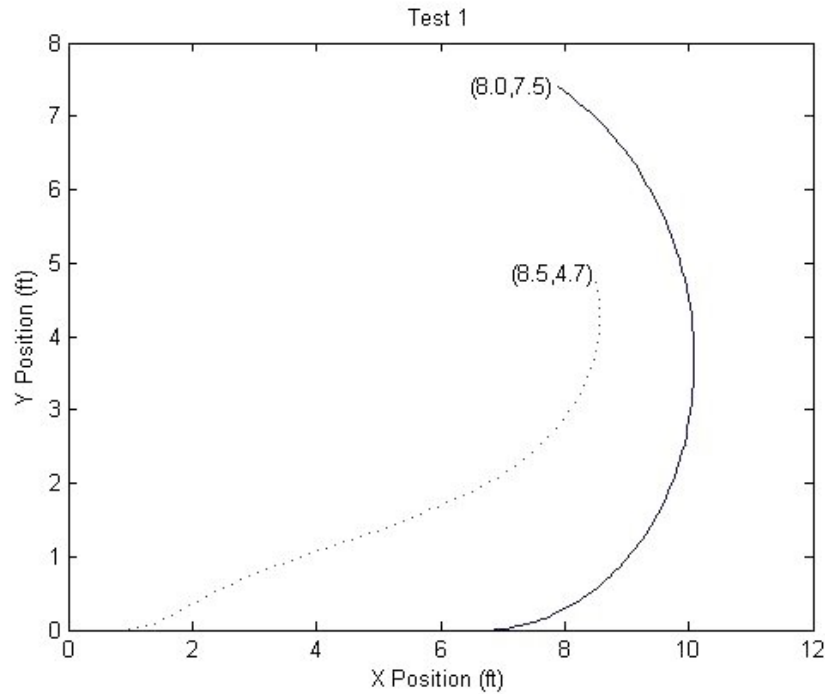
Figure 4.1:   Two Robots - Test 1

Figure 4.2 depicts a turn in the opposite direction which works, as expected, in the same manner as the previous test. Note that the lead robot follows a straight path for approximately 4.5 feet before beginning its turn. The follow robot travels approximately the same straight distance. Since the parking cone is placed on the back end of the lead robot, the cone does in fact swing to the following robot's left as the lead robot begins to turn to the right. This accounts for the slight left turn shown by the follow at approximately (3,0). This turn is quickly corrected as the cone (along with the lead robot) begins to move to the following robot's right.

Figure 4.3 shows a more complex course, this time with the lead robot following a meandering path. The following robot follows the general path, adjusting slightly but not reacting completely to every perturbation in the course. It maintains a position approximately behind the lead robot until the lead begins its final (larger) turn. The following robot then once again begins an inside turn and slows to maintain a safe distance.

52
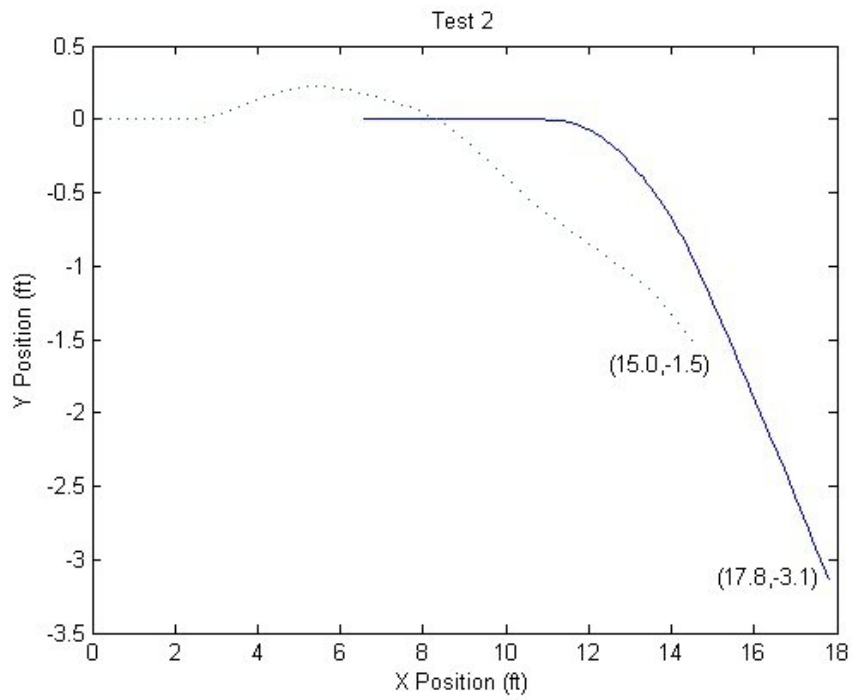
Figure 4.2:   Two Robots - Test 2
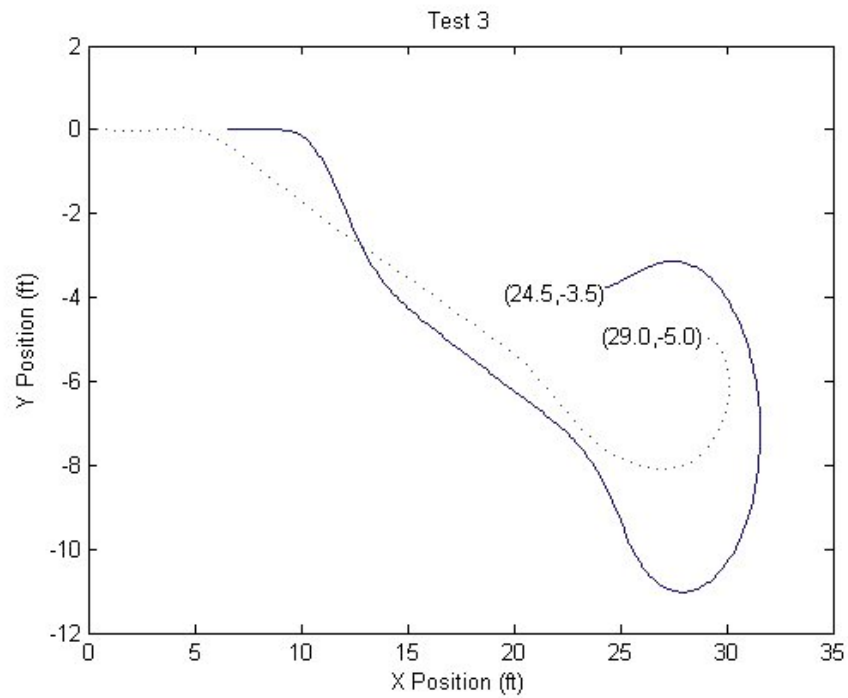
Figure 4.3:   Two Robots - Test 3
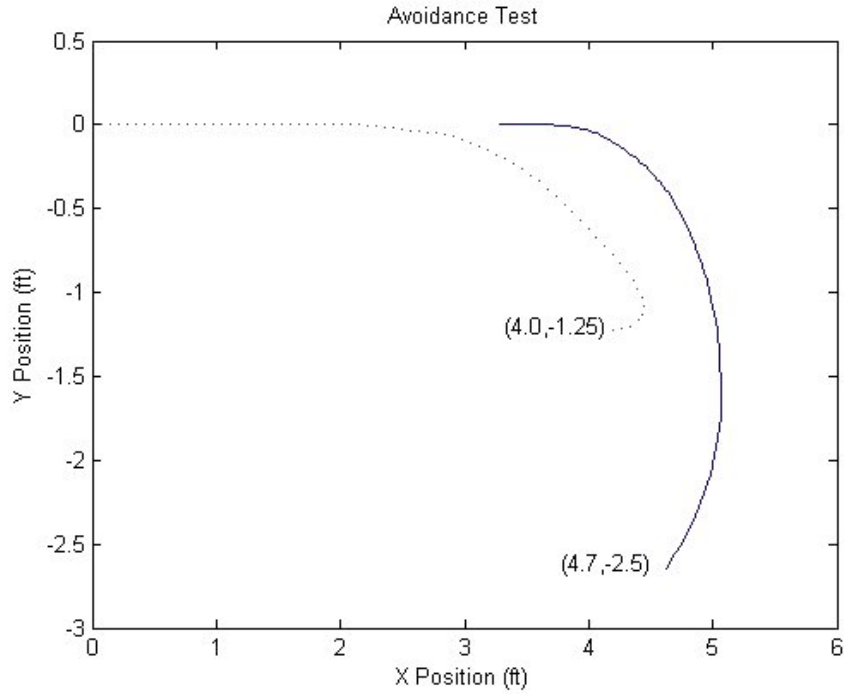
Figure 4.4:   Avoidance Test

To test the software's collision avoidance capability, the robots were placed in closer proximity to each other at the beginning of the tests. Note that in Figure 4.4 the lead robot begins only three feet in front of the follow robot. The lead robot then immediately begins a sharp turn, and the following robot immediately responds as usual. As the lead keeps turning, however, the distance continues to narrow and the following robot cannot slow down below its minimum speed. Finally, as seen at the end of the paths, the distance drops below the danger threshold (discussed in Section 3.6.8) and the following robot stops and turns away. Although only a couple tests were designed specifically to test this feature, in no case throughout the testing process did the robots ever collide with one another.

4.2   Three Robot Test

A run with all three robots is shown in Figure 4.5. The robot formation begins as a triangle with the rear robots placed parallel with each other, but separated by

54
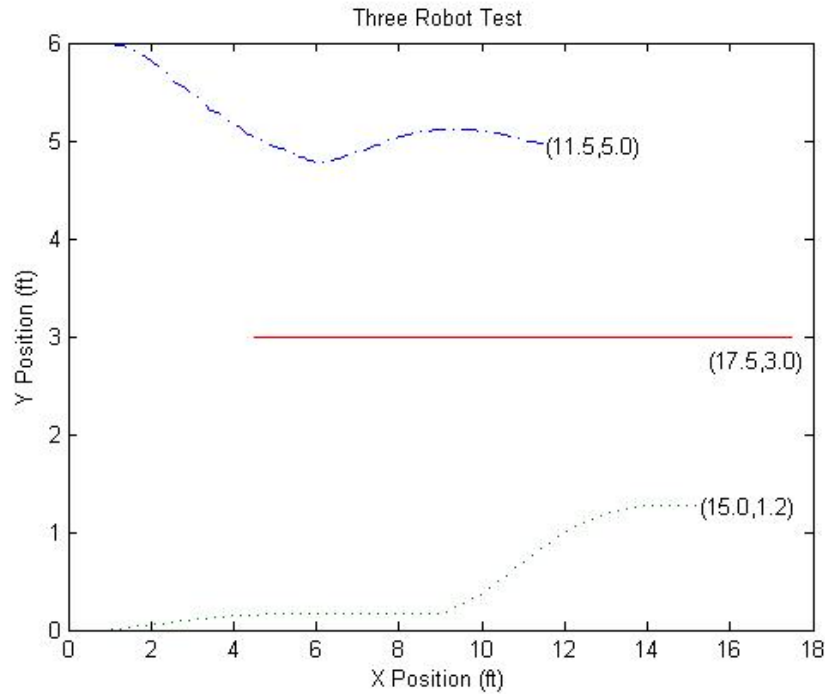
Figure 4.5:   Three Robot Test

six feet. The lead robot is placed in the middle of the two followers and offset forward
by 4.5 feet. Since the leading robot has no orange cones within its field of vision
it travels straight forward at a constant speed. The following robots initially begin
to converge on the path of the lead robot, but straighten out as they approach the
optimal distance from each other (approximately three feet).

4.3   Accuracy of Data

The odometry data is based upon the Pioneer robot's internal odometry estima-
tion, and as such, is limited to the accuracy of those guesses. Although this research
has not fully analyzed that accuracy, a preliminary testing of the distance and pose
estimations can be seen in Appendices B and C.

4.4   Comparison to Simulation

In general, the hardware tests are as effective as the simulation in terms of effective flocking. We use the final positions of the two robots and calculating the Euclidian distance between the two points with the formula

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (4.1)$$

Figure 4.1 shows a final distance of approximately $\sqrt{(8.0 - 8.5)^2 + (7.5 - 4.7)^2} \approx 2.9$ feet, within a foot of the ideal distance of three feet (as was the criteria for flocking in the simulation). The second test (Figure 4.2) is also as successful, with a final distance of approximately $\sqrt{(15.0 - 17.8)^2 + (-1.5 - (-3.1))^2} \approx 3.2$ feet. The third test (Figure 4.3) results in a configuration slightly outside of the ideal, with a final distance of $\sqrt{(24.5 - 29.0)^2 + (-3.5 - (-5.0))^2} \approx 4.7$ feet, however since it is just emerging from its turn at the end of the test run, it has not yet had a chance to speed up to compensate for this discrepancy. The avoidance test results in the robots being extremely close ($\sqrt{(4.0 - 4.7)^2 + (-1.25 - (-2.5))^2} \approx 1.4$ feet, but this is to be expected, given the nature of the test.

The three robot test shown in Figure 4.5 is also a little bit more spread out than desired, but not drastically so. The trailing robots are $\sqrt{(11.5 - 17.5)^2 + (5.0 - 3.0)^2} \approx 6.3$ feet and $\sqrt{(17.5 - 15.0)^2 + (3.0 - 1.2)^2} \approx 3.1$ feet from the robot leading the pack.

4.5   Scaling the Algorithm

Due to the limitation of having only three Pioneer robots, hardware scalability is unstested, however as shown in the simulations (Section 3.3), the scalability of this algorithm is promising. The simulation involving 20 robots performs as well as the simulation involving only three.

Moreover, the simulation is robust with respect to robots failing (i.e., stopping). To show this, the simulation to includes a small chance (0.01% for each robot at each timestep) of a robot failing. When this happens, the failed robot simply stops.

56

Figure 4.6:   Before robot failure

Although this causes a temporary disruption in the movement of its surrounding flockmates, they tend to recover quickly. An example of this is seen in Figures 4.6 and 4.7. In Figure 4.6, the robot that fails is shown in the middle of a flock of nine robots and is marked with a box around it. Another robot which failed earlier in the simulation is also marked with a black box. Figure 4.7 shows the same simulation several timesteps later, after the robot has failed. Note that the failure split the flock into two separate smaller flocks, but did not disperse them completely.

As seen in the tests discussed in this chapter, the implemented flocking algorithm shows promise and seems to be effective in an open environment. Clearly, the tests are not exhaustive, nor are the results infallible. Several possibilities for improvements and future extensions are discussed in more detail in Chapter V.
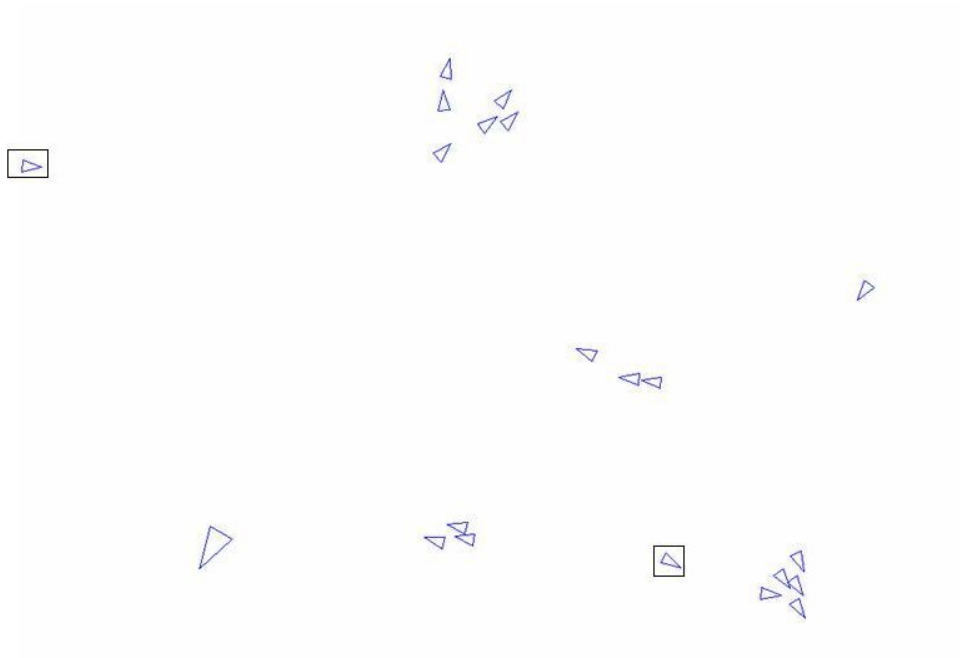
Figure 4.7:   After robot failure

V. Conclusions

The sciences do not try to explain, they hardly even try to
interpret, they mainly make models. By a model is meant a
mathematical construct which, with the addition of certain verbal
interpretations, describes observed phenomena. The justification
of such a mathematical construct is solely and precisely that it
is expected to work.

<div align="right">---John von Neumann</div>

This chapter presents a number of possible avenues for extension and future research
and discusses the conclusions of this research effort.

5.1   Future Extensions

Although our research was successful and achieved its intended goal, it is in
no way all-encompassing. The research opens the door for further exploration and
enhancement of our techniques and results.

One of the most obvious enhancements is to reduce the assumptions and achieve
the same results without the need of locating a parking cone. As discussed earlier,
the simplification of extracting the cone from the image instead of the actual robot is
made for the purpose of confining the scope of the research. Now that the algorithm is
developed and successfully completed its initial testing, some of these assumptions can
be lifted and future efforts can focus on applying our approach to the "real-world"
problem, i.e., achieving monocular vision based flocking in everyday environments.
This has been and continues to be a heavily researched field. For a survey of common
methods, the reader is referred to [7].

Likewise, another extension is to remove our second, somewhat unrealistic, as-
sumption of all robots being spheres of the same size. The area of vision based pose
recognition is another widely researched area and the reader can find a survey of these
methods in [27]. Given the dimensions and pose of a robot, estimating its distance as
done in our research becomes trivial.

Once the simplifying assumptions are removed, a natural extension is to test
scalability. As discussed in Chapter III, the simulation indicates that algorithm scala-

bility is possible, but unproven for our specific algorithm. To fully scale the problem, it would be beneficial to draw information from all neighbors instead of only from the nearest neighbor. This allows us to more easily ignore robots that have dropped out. Within our framework, the easiest implementation of this takes advantage of the multiple color segmentation ability discussed in Section 3.6.4.4. Using this feature, the robot can calculate vector updates for each robot within its field of vision and aggregate them to produce a final vector update.

Although flocking in an open environment is a first step and an indicator of a good control algorithm, a good algorithm is able to perform well in all environments. As such, another useful addition to this work is the ability to avoid obstacles (other than flockmates). This may be accomplished using sonar, but ideally vision based obstacle avoidance is more desirable from the standpoint of maintaining a passive presence in the environment. Successful flocking research in the past has successfully avoided obstacles including splitting the flock to flow around an obstacle and re-converging on the other side, and even successfully backing out of a dead end as well as self-organizing behaviors [6, 16, 31, 44].

Stationary obstacles are not the only hindrance to flocking, especially within a combat environment. Future extensions may add predator robots to the environment. As seen in the simulation, a predator introduces a new aspect to flocking, where the flock must avoid a moving obstacle as well as static ones.

Finally, another necessary improvement that became apparent during the final testing phase is robustness with regard to changing lighting. Although the improved HSV color mask and auto-adjusting camera aperture allow the segmentation algorithm to perform well in different lighting situations, changing lights still present a problem. The cause of this is the delay in the adjustment of the camera aperture, and some method is still needed to reduce this delay or make the algorithm more immune to its effects.

60

5.2   Conclusion

Overall, the developed flocking software behaved as expected (see Section 4.1 and 4.2 and provides a promising platform for future research efforts. The results of the initial follow capability testing were positive (Section 4.1, with the following robots responding quickly and accurately and closely matching the lead robot's path. In addition, the trial run of the complete, three autonomous robot testing demonstrates the success of the algorithm, with the robots forming a stable, coherent flock. Finally, the robustness of the flocking hardware with respect to failed robots was shown in simulation. In all, this research provides an excellent basis for extension and further research in the area of autonomous flocking. This research extends the current work in hardware flocking, applying monocular imaging constraints and using only local information.

Appendix A. Distance Tests

Actual Distance	Test1	Test2	Test3	Test4	Test5	Average
0.5	0.466667	0.434167	0.508333	0.493333	0.508333	0.482167
1	0.933333	0.910833	0.875833	1.01	0.935833	0.933167
1.5	1.419167	1.404167	1.456667	1.438333	1.404167	1.4245
2	2.216667	1.865833	2.001667	2.0375	1.909167	2.006167
2.5	2.558333	2.546667	2.546667	2.42	2.4725	2.508833
3	2.95	3.089167	2.984167	2.984167	2.989167	2.999333
3.5	3.65	3.591667	3.451667	3.451667	3.451667	3.519333
4	4.0825	4.0825	4.0825	4.0825	4.0825	4.0825
4.5	4.4875	4.4875	4.275833	4.4875	4.275833	4.402833
5	4.9775	4.9775	4.9775	4.72	4.9775	4.926
5.5	5.583333	5.583333	5.583333	5.583333	5.583333	5.583333
6	5.941667	5.941667	5.941667	5.941667	5.583333	5.87

All units in feet

Appendix B. Movement Tests

Vel	Robot X	Robot Y	Robot Theta	Actual X	Actual Y	Actual Theta	Delta X	Delta Y	Delta Theta
0.100	1.020	0.000	0.000	0.970	0.000	0.000	0.050	0.000	0.000
0.100	1.002	0.002	0.000	0.940	0.000	0.000	0.062	0.002	0.000
Average	1.011	0.001	0.000	0.955	0.000	0.000	0.056	0.001	0.000
Stdev	0.013	0.001	0.000	0.021	0.000	0.000	0.008	0.001	0.000
0.100	2.080	0.001	0.000	1.960	0.090	1.000	0.120	-0.089	-1.000
0.100	2.015	0.006	-1.000	1.960	0.045	4.000	0.055	-0.039	-5.000
Average	2.048	0.004	-0.500	1.960	0.068	2.500	0.088	-0.064	-3.000
Stdev	0.046	0.004	0.707	0.000	0.032	2.121	0.046	0.035	2.828
0.200	2.613	0.000	0.000	2.550	0.030	0.500	0.063	-0.030	-0.500
0.200	2.615	0.001	0.000	2.550	0.040	1.500	0.065	-0.039	-1.500
Average	2.614	0.001	0.000	2.550	0.035	1.000	0.064	-0.035	-1.000
Stdev	0.001	0.001	0.000	0.000	0.007	0.707	0.001	0.006	0.707

63

0.100	3.020	0.002	0.000	2.830	0.025	1.500	0.190	-0.023	-1.500
0.100	3.029	0.002	0.000	2.850	0.035	1.500	0.179	-0.033	-1.500
Average	3.025	0.002	0.000	2.840	0.030	1.500	0.185	-0.028	-1.500
Stdev	0.006	0.000	0.000	0.014	0.007	0.000	0.008	0.007	0.000
0.100	3.980	0.005	0.000	3.740	0.002	1.000	0.240	0.003	-1.000
0.100	3.990	0.004	0.000	3.750	0.100	3.500	0.240	-0.096	-3.500
Average	3.985	0.005	0.000	3.745	0.051	2.250	0.240	-0.047	-2.250
Stdev	0.007	0.001	0.000	0.007	0.069	1.768	0.000	0.070	1.768
0.100	5.020	0.005	0.000	4.710	0.055	2.000	0.310	-0.050	-2.000
0.100	4.970	0.007	0.000	4.670	0.140	3.000	0.300	-0.133	-3.000
0.200	5.052	0.004	0.000	4.935	0.155	3.500	0.117	-0.151	-3.500
0.200	5.102	0.003	0.000	4.985	0.130	4.500	0.117	-0.127	-4.500
Avg	5.036	0.005	0.000	4.825	0.120	3.250	0.211	-0.115	-3.250
Stdev	0.055	0.002	0.000	0.158	0.045	1.041	0.109	0.045	1.041

	Reported Theta (Deg)	Actual Theta (Deg)	X (cm)	Y (cm)
	46	41.5	1.2	-1.4
	46	42	1.4	-1.6
	46	42.5	1.8	-2
	47	43.5	1.8	-2
AVG	46.25	42.375	1.55	-1.75
STDEV	0.5	0.853912564	0.3	0.3

	Reported Theta (Deg)	Actual Theta (Deg)	X (cm)	Y (cm)
	90	85	7.8	-0.6
	90	85.5	6.2	-0.4
	90	85.75	6.4	-0.4
	90	86	7.2	-0.6
AVG	90	85.5625	6.9	-0.5
STDEV	0	0.426956282	0.7393691	0.11547005

65

	Reported Theta (Deg)	Actual Theta (Deg)	X (cm)	Y (cm)
	135	138.5	8.2	9.6
	135	138.5	7.4	9.2
	137	135.5	7.4	7.6
	138	140.5	7.8	9.8
AVG	136.25	138.25	7.7	9.05
STDEV	1.5	2.061552813	0.38297084	0.99833194

	Reported Theta (Deg)	Actual Theta (Deg)	X (cm)	Y (cm)
	180	197	-4.2	14.2
	180	199	-5.4	15.3
	180	199	-5	14.3
	181	203.5	-6.2	14.1
AVG	180.25	199.625	-5.2	14.475
STDEV	0.5	2.75	0.8326664	0.55602758

66

Bibliography

1. URL `http://www.mobilerobots.com/`.

2. URL `http://www.videredesign.com/`.

3. "Joint Unmanned Combat Air Systems". Press release. URL `http://www.darpa.mil/j-ucas/`.

4. A.B., Cao Y.U. Fukanaga A.S. Kahng and Meng F. "Cooperative mobile robotics: Antecedents and directions", 1995.

5. Badgerow, J.P. "An analysis of function in the formation flight of canada geese". *Auk*, 105:749–755, 1988.

6. Balch, T. and R. Arkin. "Behavior-based Formation Control for Multi-robot Teams". *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998. URL `citeseer.ist.psu.edu/balch99behaviorbased.html`.

7. Besl, Paul J. and Ramesh C. Jain. "Three-dimensional object recognition". *Annals of discrete mathematics*, 17(1):75–145, 1985.

8. Botea A., M., Muller and J. Schaeffer. "Near optimal hierarchical path-finding". *Journal of Game Development*, 1, 2004.

9. Braitenburg, Valentino. *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press, 1984.

10. Brooks, Rodney A. "A robust layered control system for a mobile robot". *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

11. Buckland, Mat. *Programming Game AI by Example*. Wordware Publishing, 2005. ISBN 1-55622-078-2.

12. Cliff, D. and G. F. Miller. "Co-evolution of pursuit and evasion II: Simulation methods and results". *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*.

13. Crowther, Bill. "Rules of Flocking". URL `http://www.eng.man.ac.uk/Aero/wjc/Research/Flocking/rules_of_flocking.htm`.

14. Curtis, Steven, 2005. URL `http://ants.gsfc.nasa.gov/ArchandAIwords.html`.

15. Dirk Helbing, Illés Farkas and Tamás Vicsek. "Simulating dynamical features of escape panic". *Nature*, 407:487–490, september 2000.

16. F. Michaud, et al. "Dynamic robot formations using directional visual perception". *Proceedings of the International Conference on Intelligent Robots and Systems*. 2002.

17. Fredslund, J. and M. J. Matarić. "Robot Formations Using Only Local Sensing and Control". *Proceedings of IEEE International Symposium on Computational Intelligence for Robotics and Automation.* 2001.

18. Glasius, R., A. Komoda, and S. Gielen. "Biologically Inspired Neural Networks for Trajectory Formation and Obstacle Avoidance". URL `citeseer.ist.psu.edu/glasius93biologically.html`.

19. Glasius, R., A. Komoda, and S. C. A. M. Gielen. "Neural Network Dynamics for Path Planning and Obstacle Avoidance". *Neural Networks*, 8(1):125–133, 1995. URL `citeseer.ist.psu.edu/glasius95neural.html`.

20. Gray, Zachary C. *A BEHAVIOR-BASED IMPLEMENTATION OF KADROVACH'S SWARM MODEL.* Master's thesis, Air Force Institute of Technology, 2004.

21. H. Tanner, A. Jadbabaie and G. J. Pappas. "Stable flocking of mobile agents, part I: Fixed topology,". *Proceedings of the 42nd IEEE Conference on Decision and Control.* 2003.

22. H. Tanner, A. Jadbabaie and G. J. Pappas. "Stable flocking of mobile agents, part II: Dynamic topology,". *Proceedings of the 42nd IEEE Conference on Decision and Control.* 2003.

23. Hayes, Dormiani-Tabatabaei P., A. "Self-organized flocking with agent failure: off-line optimization and demonstration with real robots", 2002.

24. Hebert, Adam. "New Horizons for Combat UAVs". *Air Force Magazine*, 86(12), 2003.

25. Hodos, W. and J.T. Erichsen. "Lower-field Myopia in birds: an adaptation that keeps the ground in focus". *Vision*, 30:653–657, 2004.

26. Horswill, I. "Visual Collision Avoidance by Segmentation". *ARPA94*, II:1135–1141. 1994. URL `citeseer.ist.psu.edu/horswill94visual.html`.

27. J. Park, B. Jiang and U. Neumann. "Vision-based pose computation: Robust and accurate augmented reality tracking". *Proceedings of the 2nd IEEE and A CM International Workshop on Augmented Reality.* 1999.

28. Jadbabaie, A., J. Lin, and A. S. Morse. "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules". *IEEE Transactions on Automatic Control*, 48:988–1001, June 2003.

29. James Bruce, Tucker Balch and Manuela Veloso. "Fast and Inexpensive Color Image Segmentation for Interactive Robots". *Proceedings of the International Conference on intelligent Robots and Systems.* 2000.

30. Julien Nembrini, Alan Winfield and Chris Melhuish. "Minimalist coherent swarming of wireless networked autonomous mobile robots". *In Proceedings of the 7th International Conference on Simulation of Adaptive Behaviour.* 2002.

31. Kadrovich, Tony. *A Communications Modeling System for Swarm-based Sensors*. Ph.D. thesis, Air Force Inst. of Tech., WPAFB, OH, March 2003.

32. Lee, D. N. and P.E. Reddish. "Plummeting Gannets - A Paradigm of Ecological Optics". *Nature*, 5830:293–294, 1981.

33. Lenser, S. and M. Veloso. "Visual sonar: Fast Obstacle Avoidance using Monocular Vision", 2003. URL `citeseer.ist.psu.edu/lenser03visual.html`.

34. Lorek, Helmut and Matthew White. "Parallel Bird Flocking Simulation", May 1993. URL `citeseer.ist.psu.edu/lorek93parallel.html`.

35. Lorigo, L., R. Brooks, and W. Grimson. "Visually Guided Obstacle Avoidance in Unstructured Environments", 1997. URL `citeseer.ist.psu.edu/lorigo97visuallyguided.html`.

36. Martinelli A, Tapus A., Tomatis N and Siegwart R. "Simultaneous Localization and Odometry Calibration". 2003.

37. Mataric, M. "Designing and understanding adaptive group behavior", 1995. URL `citeseer.ist.psu.edu/ mataric95designing.html`.

38. Miller, G. and D Cliff. "o-evolution of pursuit and evasion i: Biological and game-theoretic foundations". Technical Report CSRP311, 1994.

39. Moshtagh, Nima, Ali Jadbabaie, and Kostas Daniilidis. "Vision-Based, Distibuted Coordination of Multi-Agent Systems".

40. Naohiko Shimoyama, Tsuyoshi Mizuguchi Yoshinori Hayakawa, Ken Sugawara and Masaki Sano. "Collective Motion in a System of Motile Elements". *Physical Review E*, 76(20):3870–3873, 1996. URL `http://prola.aps.org/pdf/PRL/v76/i20/p3870_1`.

41. Olivier Alloyer, James Cremer Joseph Kearney Peter Willemsen, Esmail Bonakdarian. "Embedding Scenarios in Ambient Traffic". *Proceedings of DSC '97 (Driving Simulation Conference)*, 75–84. 1997.

42. Peterson, Garry D. "Animal Aggregation: Experimental Simulation Using Vison-Based Behavioural Rules". L. Nadel and D. L. Stein (editors), *1992 Lectures in Complex Systems*, 623–630. Addison-Wesley, Reading, MA, 1993.

43. Potts, Wayne K. "The Chorus-Line Hypothesis of Manoeuvre Coordination in Avian Flocks". *Nature*, 309:344–345, May 1984.

44. Price, Ian. *Evolving Self-Organized Behavior for Homogenous and Heterogeneous UAV or UCAV Swarms*. Ph.D. thesis.

45. Reynolds, C. W. "Competition, Coevolution and the Game of Tag", 1994.

46. Reynolds, Craig W. "Flocks, Herds, and Schools: A Distributed Behavioral Model". *Computer Graphics*, 25–34, July.

69

47. Schnapauff, Olaf and Josef Schule. "Simulating and Visualizing Natural Flocking Behavior". URL `citeseer.ist.psu.edu/article/schnapauff98simulating.html`.

48. Shen, Will P. Galstyan A. Chuong C. M., W. M. "Hormone-Inspired Self- Organisation and Distributed Control of Robotic Swarms", 2004.

49. T. Vicsek, E. Ben Jacob I. Cohen, A. Czirok and O. Schochet. "Novel type of phase transitions in a system of self-driven particles", 1995.

50. Tarlton, Malcolm. "Misjudgment in a Bird Flock".

51. Toner, John and Yuhai Tu. "Flocks, Herds, and Schools: A Quantitative Theory of Flocking". *Physical Review E*, 58(4):4828–4858, October 1998.

52. Wang, Y. and B. J. Frost. "Time to Collision is Signalled by Neurons in the Nucleus Rotundus of Pigeons". *Nature*, 6366:236–238, 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
23–03–2006	Master's Thesis	Sept 2004 — Mar 2006

4. TITLE AND SUBTITLE

A Monocular Vision Based
Approach to Flocking

5a. CONTRACT NUMBER

DACA99–99–C–9999

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Brian Kirchner, 2Lt, USAF

5d. PROJECT NUMBER

ENG 06-400

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management, (AFIT/EN)
2950 Hobson Way, Bldg 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/06-09

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Mikel Miller
Mikel.Miller@wpafb.af.mil
785-6127x4274
AFRL/SNRP (AFMC)
2241 Avionics Cir
Wright-Patterson Air Force Base, Ohio 45433

10. SPONSOR/MONITOR'S ACRONYM(S)

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Flocking is seen in nature as a means for self protection, more efficient foraging, and other search behaviors. Although much research has been done regarding the application of this principle to autonomous vehicles, the majority of the research has relied on GPS information, broadcast communication, an omniscient central controller, or some other form of "global" knowledge. This approach, while effective, has serious drawbacks, especially regarding stealth, reliability, and biological grounding. This research effort uses three Pioneer P2-AT8 robots to achieve flocking behavior *without* the use of global knowledge. The sensory inputs are limited to two cameras, offset such that the area of stereo vision is minimal, thus making stereo image analysis techniques effectively impossible, but allowing a much larger effective field of vision. The flocking algorithm analyzes these images and updates each robot's velocity vector according to the relative position, heading, and speed of its nearest neighbor. The result of this velocity update is an eventual stabilization of speed and heading, resulting in a coherent, stable flock, demonstrated in both software simulation and in hardware.

15. SUBJECT TERMS

flocking, swarming, robotics, control algorithm, monocular vision

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. Gilbert Peterson
U	U	U	UU	70	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636, ext 4281

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

www.manaraa.com