Localization, Tracking, and Odometry Calibration of a Multi-Agent Swarm System

by

Gregory Kealoha Fricke

Department of Mechanical Engineering and Materials Science
Duke University

Date: __________________________

Approved:

______________________________
Dr. Devendra P. Garg, Advisor

______________________________
Dr. Dejan Milutinović

______________________________
Dr. Jeffrey T. Scruggs

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Mechanical Engineering and Materials Science in the Graduate School of Duke University 2009
Abstract

Swarm robotics offers the promise of enhanced performance and robustness relative to that of individual robots, with decreased cost or time-to-completion for certain tasks. Having many degrees of freedom, the swarm related control and estimation problems are challenging specifically when the solutions involve a great amount of communication among the robots. Much of the research in the field of swarm robotics is conducted in simulation. While numerical simulations are an effective and valuable tool allowing extensive study across broad ranges of model and control parameters and situational conditions, verification of obtained results in a physical realization of the swarm is far from routine. With the intent of effecting a change in this practice, a sensor-integrated testbed has been developed for the validation of swarm algorithms, observation of swarm behavior, and measurement of swarm performance.

Implementation of swarm behavior in real hardware allows this validation and also serves to bring to light the practical issues that must be considered. These practical issues may include computer science issues relating to real-time computational efficiency, data processing methods to handle measurement dropouts or spurious measurement error outside of the usual Gaussian white noise approximation, communication difficulties, or any number of other uncertainties.

An arena for conducting swarm experiments has been established in the Robotics and Manufacturing Automation Laboratory (RAMA Lab) at Duke University. The main thrust of this arena development treats the detection and position measurement of multiple homogeneous members of a robotic swarm within the arena. This thesis covers the theory, simulations, and limited experimentation employed to produce this testbed.

The swarm agents in the RAMA Lab are Khepera-II differential drive robots
acquired from K-Team. Three sensors are utilized to measure the positions of the agents and provide feedback: a Cognex InSight 5400 industrial machine-vision system, and two SICK LMS400-1000 LADAR rangefinders. Multiple estimation tasks are required to integrate the position measurements into a usable set of data. To utilize the rangefinder data, the features of interest (in this case, the robots) must be identified relative to the background clutter. For the vision system to operate with a usable sample frequency, simple blob detection must be used rather than more computationally costly pattern recognition. This compromise to increase sample rate requires the sacrifice of robot orientation measurements. The orientation of the robots must then be estimated in order to properly provide feedback control, due to the non-holonomic limitations of two-wheeled, differential drive robots.

With simultaneous state estimation of multiple robots in the fields-of-regard of the sensors, the system must be capable of discerning how to properly apply the measurements. This requires the development of methods for tracking and data validation. The coupling of tracking and validation algorithm parameters with those of the estimation model adds another complication. For the individual robots to independently localize themselves without exteroceptive sensors or a priori knowledge of the arena, they must rely on odometry, which is notoriously susceptible to model error. The central supervisor thus provides absolute frame-of-reference position feedback at a low data rate, which the individual agents integrate into their on-board odometry calibration and localization models providing accurate localization results.

Real data sets are gathered, and these estimation methods are verified via simulation acting on the real data. Some limited real-time experiments are performed to provide further validation.
Contents

List of Tables viii
List of Figures ix
List of Abbreviations and Symbols xi
Acknowledgements xvii

1 Introduction 1
  1.1 Motivation .................................................. 1
  1.2 Thesis Outline ........................................... 3

2 Fundamental Swarm Characteristics 6
  2.1 Multiple-Robot Control .................................. 7
  2.2 Communication ........................................... 9
    2.2.1 Agent-Agent Communication ............................ 10
    2.2.2 Agent-Host Communication ............................. 10
  2.3 Localization ............................................ 11

3 Testbed 13
  3.1 Physical Layout ........................................ 13
  3.2 Agents: Khepera-II ....................................... 16
    3.2.1 Communication ........................................ 17
    3.2.2 Programming .......................................... 19
    3.2.3 Sensing Ability ....................................... 20
    3.2.4 Motion Control ....................................... 21
  3.3 SICK LADAR Sensor ....................................... 22
  3.4 Cognex InSight 5400 Vision System ....................... 23
  3.5 Coordinate Frame Definition ............................. 25

4 Estimation Techniques 27
  4.1 External Position Sensor Processing ...................... 28
    4.1.1 Proposed Method ...................................... 30
  4.2 Tracking ................................................ 33
4.2.1 Proposed Method ........................................ 35
4.3 External Localization ...................................... 36
4.3.1 Proposed Method ........................................ 37
4.4 Odometry .................................................... 40
4.4.1 UMBmark Method ......................................... 43
4.4.2 Least-Squares Method using X, Y, and Orientation .... 44
4.4.3 Least-Squares Method using Path and Orientation .... 47
4.4.4 Augmented, Extended Kalman Filter Method .......... 49
4.4.5 Augmented EKF for Non-Systematic Error Prediction .. 53
4.4.6 Proposed Method ........................................ 54
4.5 Identification of Agents by the Host System ............ 54
4.5.1 Proposed Method ........................................ 57

5 Simulations and Experimental Data Analysis ................. 59
5.1 Simulation Environment ..................................... 60
5.2 Experimental Data .......................................... 61
5.2.1 Individual Agents ........................................ 62
5.2.2 Multiple Agents ......................................... 63
5.3 External Sensor Data Processing ............................ 63
5.3.1 LADAR Arc-Detection .................................... 64
5.4 Controlled Trajectories ...................................... 66
5.4.1 Circular Motion .......................................... 67
5.4.2 Straight Line Motion ..................................... 68
5.5 External Localization ....................................... 69
5.5.1 Extended Kalman Filter .................................. 69
5.5.2 Tracking .................................................. 70
5.6 Odometry Calibration ........................................ 72

6 Results and Discussion ......................................... 81
6.1 External Localization and Tracking ......................... 81
6.1.1 Coupling of EKF, Data Validation, and Tracking .... 82
6.2 Odometry Calibration ....................................... 84
6.3 Sources of Error ............................................ 85
6.4 Best Use of Position Sensors ................................. 87

7 Conclusion ..................................................... 89
7.1 Future Work ................................................ 89
7.1.1 Variation of Process Noise ............................... 90
7.1.2 Coordinate Frame Estimation ............................. 90
7.1.3 Identification and Odometry Calibration ............... 90
7.1.4 Stochastic Control and Random Graph Theory ......... 91

Bibliography ...................................................... 92
A  Additional Images

B  Custom MATLAB® Code  104
   B.1  Arc-Detection  ......................................................... 104
   B.2  Arc-Prediction ......................................................... 105
List of Tables

4.1 Cognex pattern matching tests (left edge of arena) . . . . . . . . . . . . 57
4.2 Cognex pattern matching tests (center of arena) . . . . . . . . . . . . 57
5.1 Cognex sensor noise variance (from raw data) . . . . . . . . . . . . . 64
5.2 SICK LADAR combined measurement and estimation error . . . . . 65
5.3 Optimal parameters for $Q_{\theta}(v)$ . . . . . . . . . . . . . . . . . . . 69
5.4 Estimation errors for $Q_{\theta} \propto -v$ . . . . . . . . . . . . . . . . . . . 71
5.5 Estimation errors for $Q_{\theta} \propto -v^2$ . . . . . . . . . . . . . . . . . . . 71
5.6 Estimation errors for $Q_{\theta} \propto \frac{1}{|v|}$ . . . . . . . . . . . . . . . . . . . 71
5.7 Estimation errors for $Q_{\theta} \propto \frac{1}{v^2}$ . . . . . . . . . . . . . . . . . . . 71
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Composite photo of arena and sensors</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Overhead view of the Khepera-II arena</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Khepera-II robot as used in experiments.</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>IR proximity sensing function of front sensors on Khepera-II robots</td>
<td>20</td>
</tr>
<tr>
<td>3.5</td>
<td>LADAR coverage of the Khepera arena</td>
<td>22</td>
</tr>
<tr>
<td>3.6</td>
<td>LADAR occlusion of distant agents by a proximal agent</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Sample range data from SICK LADAR, 1st unit</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Sample range data from SICK LADAR, 2nd unit</td>
<td>30</td>
</tr>
<tr>
<td>4.3</td>
<td>Geometry of arc detection</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Unique IDs for Cognex identification via pattern-recognition</td>
<td>55</td>
</tr>
<tr>
<td>5.1</td>
<td>Histogram of range measurements (best case)</td>
<td>66</td>
</tr>
<tr>
<td>5.2</td>
<td>Histogram of range measurements (worst case)</td>
<td>67</td>
</tr>
<tr>
<td>5.3</td>
<td>$Q_\theta$ vs. $v$, all four candidate functions.</td>
<td>70</td>
</tr>
<tr>
<td>5.4</td>
<td>Raw data vs. EKF state estimates</td>
<td>73</td>
</tr>
<tr>
<td>5.5</td>
<td>Raw data vs. EKF state estimates</td>
<td>74</td>
</tr>
<tr>
<td>5.6</td>
<td>Semi-circular arc trajectory</td>
<td>75</td>
</tr>
<tr>
<td>5.7</td>
<td>Odometry calibration for semi-circular arc trajectory</td>
<td>75</td>
</tr>
<tr>
<td>5.8</td>
<td>Arbitrary trajectory</td>
<td>76</td>
</tr>
<tr>
<td>5.9</td>
<td>Odometry calibration for arbitrary trajectory</td>
<td>76</td>
</tr>
<tr>
<td>5.10</td>
<td>Circular trajectory</td>
<td>77</td>
</tr>
<tr>
<td>5.11</td>
<td>Odometry calibration for circular trajectory</td>
<td>77</td>
</tr>
<tr>
<td>5.12</td>
<td>Figure-eight trajectory</td>
<td>78</td>
</tr>
<tr>
<td>5.13</td>
<td>Odometry calibration for figure-eight trajectory</td>
<td>78</td>
</tr>
<tr>
<td>5.14</td>
<td>Square trajectory</td>
<td>79</td>
</tr>
<tr>
<td>5.15</td>
<td>Odometry calibration for square trajectory</td>
<td>79</td>
</tr>
<tr>
<td>5.16</td>
<td>Rounded-square trajectory</td>
<td>80</td>
</tr>
<tr>
<td>5.17</td>
<td>Odometry calibration for rounded-square trajectory</td>
<td>80</td>
</tr>
<tr>
<td>A.1</td>
<td>IR proximity sensing function of side sensors on Khepera-II robots</td>
<td>99</td>
</tr>
<tr>
<td>A.2</td>
<td>$Q_\theta$ vs. $v$, all four candidate functions (semilogx).</td>
<td>100</td>
</tr>
<tr>
<td>A.3</td>
<td>$Q_\theta$ vs. $v$, all four candidate functions (semilogy).</td>
<td>100</td>
</tr>
<tr>
<td>A.4</td>
<td>$Q_\theta$ vs. $v$, all four candidate functions (loglog).</td>
<td>101</td>
</tr>
</tbody>
</table>
# List of Abbreviations and Symbols

## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKF</td>
<td>augmented, extended Kalman Filter</td>
</tr>
<tr>
<td>COM</td>
<td>A component of the Khepera-II operating system, the i/o manager</td>
</tr>
<tr>
<td>DOF</td>
<td>Degree(s) of freedom</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>HSRT</td>
<td>High-Speed Radio Turret, an expansion turret for Khepera-II robots based on Bluetooth technology</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated development environment</td>
</tr>
<tr>
<td>IO</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman Filter</td>
</tr>
<tr>
<td>LADAR</td>
<td>Laser detection and ranging</td>
</tr>
<tr>
<td>LPF</td>
<td>Low-pass filter</td>
</tr>
<tr>
<td>MHT</td>
<td>Multiple Hypothesis Tracking</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean squared error</td>
</tr>
<tr>
<td>MSG</td>
<td>The local multi-microcontroller network manager component of the Khepera-II operating system</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed circuit board</td>
</tr>
<tr>
<td>RF</td>
<td>Radio frequency</td>
</tr>
<tr>
<td>RLS</td>
<td>Recursive least-squares</td>
</tr>
</tbody>
</table>
RT Radio Turret, an expansion turret for Khepera-II robots based on RF communication

SER A component of the Khepera-II operating system, the serial RS232 manager

SSE Sum of squared error

TCP/IP A standard network protocol

USB Universal Serial Bus, a standard hardware communication protocol

Symbols

\[ \alpha \] Empirically-defined parameter in the velocity-varying functions \( Q_\theta(v) \)

\[ \Delta \psi_{R/L}^k \] Incremental angular rotation of the right/left driveshaft over the interval \([k - 1, k]\)

\[ \psi_i \] Angle \( \angle OA_iC \) in Fig. 4.3.

\[ \psi_{R/L}(t) \] Accumulated encoder reading at time \( t \)

\[ \rho \] Range to the center of the reflected arc of a Khepera-II robot received by the SICK LADAR rangefinder

\[ \Delta \theta \] Spatial (angular) sample spacing of the SICK LADAR sensors

\[ \Delta \theta_k \] Incremental change in orientation over interval \([k - 1, k]\)

\[ \theta \] Orientation angle in the \( \hat{x} - \hat{y} \) plane; the fourth element of the state vector \( X \).

\[ \theta \] Half-angle extent of a Khepera-II robot at range \( \rho \) as seen by the SICK LADAR sensors

\[ \hat{\theta} \] Estimate of state \( \theta \)

\[ \theta_{i,j} \] Orientation at time-step \( i \) of calibration trajectory \( j \)

\[ \theta_k \] Orientation of robot at time-step \( k \)

\[ \Phi_\theta \] \([1 \times 2]\) vector of total angular change of the right and left driveshaft over interval \([0, N]\)

\[ \bar{\Phi}_\theta \] \([P \times 2]\) matrix of accumulated encoder readings
\( \Phi_{\theta,j} \) \([1 \times 2]\) vector of total angular change of the right and left drive-shaft over interval \([0, N]\) of calibration trajectory \(j\)

\( \Phi_{xy} \) \([2 \times 2]\) matrix of accumulated odometry states over \(N\) sample periods

\( \bar{\Phi}_{xy} \) \([2P \times 2]\) matrix of accumulated odometry states over \(N\) sample periods for each of \(P\) trajectories

\( \xi_{\theta} \) The random excitation of state \(\theta\)

\( \xi_{v} \) The random excitation of state \(v\)

\( \omega_{R/L} \) Angular velocity of right/left wheel

\( A_k \) The Jacobian of the state-transition model \(a\) at time-step \(k\)

\( b \) Separation of the wheels

\( \hat{b} \) Estimated separation of the wheels

\( C \) State-output matrix; the Jacobian of the output model \(c\)

\( \hat{C} \) State-transition matrix for Antonelli’s odometry

\( \hat{C} \) Least-squares estimate of the odometry state-transition matrix

\( c_{i,j} \) The \((i,j)\) element of the odometry state-transition matrix \(C\)

\( \hat{c}_{i,j} \) The \((i,j)\) element of the estimated odometry state-transition matrix \(\hat{C}\)

\( c_{R/L} \) Right/left encoder-counts-to-radians scale factor

\( \partial e_{R/L}^k \) Incremental encoder reading over interval \([k - 1, k]\)

\( d_{R/L}^k \) Incremental distance traveled by right/left wheel over interval \([k - 1, k]\)

\( E_r \) Ratio of right wheel radius to left wheel radius

\( E_b \) Ratio of actual wheelbase to estimated wheelbase

\( F1 - F4 \) Hessian matrices of each component of the state-transition function \(a\)

\( g \) Validation gate

\( K \) Kalman Gain matrix
\(k\) Time-step index

\(N\) Dimension (number of points) of the predicted arc \(S\)

\(N\) Total number of sample periods in the least-squares odometry estimation

\(P\) Estimated covariance of the estimated state \(\hat{X}\)

\(P\) Number of robots whose states are estimated by the central supervisor

\(P\) Number of calibration trajectories in the least-squares odometry estimation

\(\vec{P}_{\text{COG}}\) Displacement vector from the arena reference frame origin to the Cognex Insight vision system

\(\vec{P}_{\text{SICK}1}\) Displacement vector from the arena reference frame origin to the first SICK LADAR sensor

\(\vec{P}_{\text{SICK}2}\) Displacement vector from the arena reference frame origin to the first SICK LADAR sensor

\(Q\) Covariance matrix of process noise in the state-transition model

\(Q_\theta\) The \([4,4]\) element of the process covariance matrix \(Q\) corresponding to the fourth element, \(\theta\) of state vector \(X\)

\(q_\theta\) An empirically-defined parameter in the velocity-varying functions \(Q_\theta(v)\)

\(R\) The covariance matrix of measurement noise in the state-output model

\(R_K\) Radius of a Khepera-II robot

\(R_{\text{SICK}1}\) Rotation matrix to transform the first SICK LADAR sensor measurements into the arena reference frame

\(R_{\text{SICK}2}\) Rotation matrix to transform the first SICK LADAR sensor measurements into the arena reference frame

\(r\) Distance from the origin of the arena reference frame

\(r_{R/L}\) Radius of the right/left wheel

\(\hat{r}_{R/L}\) Estimated radius of the right/left wheel
$S$ Estimated arc representing a Khepera-II robot as seen by the SICK LADAR rangefinder

$S_i$ The $i$-th element of $S$, the predicted arc

$S_i$ Covariance estimate of the predicted measurement $\hat{Z}_k^i$

$S_k$ Curvilinear abscissa distance traveled up to time-step $k$

$S_{abs}$ Total magnitude of distance traveled up to time-step $k$

$S(t)$ Curvilinear abscissa distance traveled up to time $t$

$T$ Sample period

$T_{SICK1}$ Homogeneous representation of the transform for the first SICK LADAR sensor measurements into the arena reference frame

$T_{SICK2}$ Homogeneous representation of the transform for the first SICK LADAR sensor measurements into the arena reference frame

$\delta t$ Sampling interval

$V_{ij}$ The “cross-residual” matrix

$v$ Robot velocity, the third element of state $X$

$v_k$ Robot velocity at time-step $k$

$v_{max}$ The maximum velocity that the robot will nominally experience

$w_x$ Covariance of error in measurement $x_m$

$w_y$ Covariance of error in measurement $y_m$

$X$ The state vector

$\hat{X}_{k+1|k}$ Prediction of state $X$ at time-step $k+1$, including measurements up to time-step $k$

$X_0$ Initial state

$\vec{x}, \vec{y}, \vec{z}$ The basis vectors of the arena reference frame

$(x, y, z)$ Coordinates in the arena reference frame

$x_i$ Externally measured $x$-coordinate at time-step $i$

$x_{i,j}$ $x$-coordinate at time-step $i$ of calibration trajectory $j$
\( x_k \)  \( x \)-coordinate of robot at time-step \( k \)

\( x_m \)  Measurement of \( x \) coordinate of a robot

\( y_i \)  Externally measured \( y \)-coordinate at time-step \( i \)

\( y_{i,j} \)  \( y \)-coordinate at time-step \( i \) of calibration trajectory \( j \)

\( y_k \)  \( y \)-coordinate of robot at time-step \( k \)

\( y_m \)  Measurement of \( y \) coordinate of a robot

\( Z_k \)  The \( [2 \times P] \) measurement matrix at time-step \( k \)

\( \hat{Z}_i^k \)  Predicted \( [1 \times 2] \) measurement vector of the \( i \)-th robot at time-step \( k \).

\( Z_j^k \)  The \( j \)-th \( [1 \times 2] \) measurement vector reported by an external sensor at time-step \( k \)

\( Z \)  General expression for a measurement vector

\( z_\theta \)  Measured orientation change over interval \([0, N]\) in least-squares method

\( Z_\theta \)  \([1 \times P]\) vector of externally measured orientation changes

\( Z_{xy} \)  \([1 \times 2P]\) vector of externally measured changes in \( x \)- and \( y \)-coordinates

\( z_{xy} \)  External measurement vector of changes in \( x \)- and \( y \)-coordinates
Acknowledgements

To my parents, the Reverend Milton Fricke and Judy Fricke: for your encouragement, support, prayers, and love;
To my brothers and sisters, Dr. Matthew and Katharine Fricke and Scott and Melissa Fricke: for your belief in me and your words of wisdom, encouragement, and humor;
To my partner, Danielle Thomas: for your patience, understanding, support, and sacrifice;
To Linda Ogata, Edwin Ginoza, and Keith Imada: for inspiring me at a young age to be inquisitive and creative, and encouraging me to continuously challenge myself;
To Professor Devendra P. Garg: for your infinite patience, calm guidance, laughter, and for this opportunity;
To Professor Dejan Milutinović: for your lively debate, enthusiasm, wisdom, and eagerness to share knowledge;
To Professor Jeffrey T. Scruggs: for your rigorous instruction, entertaining conversation, and flexibility;

THANK YOU.

I could not and would not have done this without any of you.

This work was supported by Army Research Office grant number W911NF-08-1-0106 titled “Modeling, Analysis, and Control of Swarming Agents in a Probabilistic Framework”.

xvii
1.

Introduction

1.1 Motivation

Research in the area of swarm robotics continues to expand and move forward. Initially a problem of robotic navigation, the field has extended to other areas such as numerical problem solving and optimization [42]. Recent advances have been made in the application of modern control theory to swarms of independent robots with modest computational power linked via simple communication [59]. Various tasks may be accomplished with these simple architectures, including environmental mapping [29], border defense [20, 67], plume location [31], dynamic optimization and reconfiguration of sensor placement within a sensor network [24], trapping or escorting of environmental intruders [3, 21], strength-assistance [55], et cetera. The utility afforded by these swarm capabilities is broad reaching and merits exploration, ranging in application from macroscopic military or civilian applications [67] to microscopic understanding of motile networks of living cells [54].

The advantages and disadvantages of a swarm of simple robots relative to a single, complex robot have been investigated [20, 27, 30, 49, 55] and contrasted over such metrics as system cost, system reliability, energy expenditure, time to task
completion, probability of task success, and required complexity. While the last of these characteristics is difficult to quantify, the previous metrics are relatively easy to evaluate via both simulation and experiment. The frequent consensus is that swarm applications are best suited for mapping, area coverage (including searching or foraging tasks) or in hostile situations where damage to or loss of an agent would cause a single-robot mission to fail where a swarm would exhibit *graceful degradation* [67]; that is, performance may decrease, but overall mission success is still possible.

Over the last twenty years, there has been much discussion, experimentation, and postulation relating to the control of a swarm of independent robotic agents. Formation control via simple, distributed control [34, 35, 53, 69] largely arose after the seminal work of Reynolds [63] and Beni [13]. Research at the Duke University Robotics and Manufacturing Automation Laboratory (RAMA Lab) has recently investigated stochasticity as it applies to the construction and maintenance of connected swarm formations [44, 45]. Planned work in the RAMA Lab includes further development of these methods via real-world implementation and experimental validation.

In preparation of this future work, certain infrastructure must be developed. The work conducted and described in this thesis establishes a platform upon which swarm experiments can be performed and evaluated. Integral to these goals are:

- the implementation and evaluation of estimation methods performed by a centralized supervisor;
- development of accurate and timely position feedback via external position sensors;
- selection and appropriate use of external position sensors for metrology and/or feedback;
- validation of odometry calibration methods
• identification of swarm agents via cooperation with the central supervisor;

• localization of individuals via dead-reckoning, with absolute frame correction via external updates.

1.2 Thesis Outline

Chapter 2 discusses various characteristics of swarm robotics, including the points regarding what defines a swarm. Some early literature regarding swarm classification is introduced, and these classifiers are discussed. Different modalities of control are contrasted, and applications of the different types are given. The limitations or capabilities related to abstract communication modalities are laid out and contrasted. Finally, localization is introduced and described, and the different types of localization are contrasted for their benefits and costs.

The unifying element of this thesis is the swarm arena. Chapter 3 gives an extensive discussion of the capabilities and limitations of each of the major components of the arena. A physical description of the arena is provided and the reference frame is defined. The swarm members are then described. The features and components of the robots are discussed along with the real-world challenges in utilizing them. Next, the LADAR rangefinders are described, defining their relevant parameters and physical properties. The arrangement of the LADAR pair is explained and illustrated. Finally, the Cognex vision system is described, relating its position relative to the arena, its imaging limitations and features, and the trade-offs made to maximize its usefulness.

The heart of the thesis is Chapter 4, in which the estimation methods are discussed. Each section relates to a critical task that must be performed either by the central supervisor or by the individual robots. Within each section, competing methods for task completion are described and contrasted, and then a method is chosen
for implementation in the testbed. The first task discussed deals with processing the raw data received from the sensors. Section 4.1 is primarily related to the processing of LADAR data, and discusses methods of extracting swarm member locations from the raw range data. Section 4.2 deals with tracking the individuals, making sure that measurements are applied to the estimate of the appropriate agents in the central supervisor. This section additionally discusses the problem of dealing with bad data, i.e., data points that exhibit excessive error relative to the predicted position. The methods used to estimate the actual positions of the robots is discussed in Section 4.3. The chosen method here is believed to be a novel adaptation of the 2nd-Order Kalman Filter. Section 4.4 presents a lengthy discussion of different methods for odometry calibration and error-correction. Several models are discussed in detail, and their relative strengths and weaknesses are compared. Section 4.5 deals with agent identification, and provides a discussion of the importance of unique identification. Very little prior literature has been found on this topic, so after discussion of a prior method attempted at the RAMA Lab, a novel approach is proposed to match tracked and identified agents via the agents’ reported odometry estimates.

In Chapter 5, simulations and experiments relating to the methods described in Chapter 4 are described. The simulations primarily utilize real collected data from the sensor triad. The real-world data sets are described, as are the simulations which use the real data. Parameters for each of the methods are chosen via numerical optimization, and the relative performance of each method is reported. Performance of the external localization is reported as compared to measured orientation data. Tracking and data rejection performance is measured over several data sets with different trajectories. Finally, odometry calibration is simulated utilizing fully synthetic data.

The results of the simulations and experiments are discussed in Chapter 6, including observations regarding the successes and failures of each estimation goal.
Unexpected sources of error are identified and discussed. Some interesting observations of the behavior and interaction of the different estimation methods are made, and flaws in the methodology are postulated.

Conclusions relating to the success of this work are presented in Chapter 7, along with recommendations for future improvements and experiments relating to these tasks. Additional recommendations are made regarding ways to most successfully utilize the arena and its components in the future.
The works of Reynolds [63] and Beni [13] set off a firestorm of research in collective and cooperative robotics. According to Cao [20], over 200 related papers were published in the years 1987-1995. It has become apparent that swarm robotics systems have great promise for broad applicability in autonomous task completion. Equally apparent is the fact that the variety of tasks robot swarms may undertake is rivaled only by the variety of robot swarm system architectures.

Cao [20], Dudek [27], and most recently Mondada [55] have attempted to delineate different types of robot swarms, characterizing them based on task type, communication type, control method, control ownership (e.g., centralized, decentralized, distributed, or hybrid), degree of cooperation, degree of swarm homogeneity, and degree of situational awareness. Each of these categorizations is valid, yet none captures fully the segregation of different swarm types. Borrowing language from linear mathematics, the categories thus far presented comprise a non-orthogonal basis. Perhaps this is a contributing factor for the continued interest in swarm robotics among the research community.

Biologically inspired control methods continue to receive much attention, as natu-
nal systems are accepted to be optimized for certain specific tasks relating to survival. Examples include the (assumed) energy efficiency and (observed) robustness of the foraging behavior of ant swarms, the effective self-defense behavior of schooling fish, and the energy-efficient V-formations of migrating bird flocks. For this reason, there has been a significant amount of crossover work between behavioral biologists, computer scientists, mathematicians, and control theorists attempting to discover and implement the control laws that govern the behavior [18, 54].

An important discussion topic is the difference between collective behavior and cooperative behavior. Collective behavior can be defined as the actions of more than one robot in a common environment, where the actions of each robot may affect the environment or other robots, but any interaction is simply a by-product of overlapping domains. In contrast, cooperative behavior refers to actions where, given some task, there is an increase in the total utility of the system due to some underlying cooperative mechanism [20].

2.1 Multiple-Robot Control

There exists a broad range of control types for controlling systems of multiple-robot systems. The variation in control types ranges widely depending on communication, sensing, and processing ability, as well as the goal or desired utility of the system. Parker suggests four categories for grouping the competing control types: centralized, hierarchical, decentralized, and hybrid [59].

It may be argued that fully centralized control does not exist, and if it does exist it should not be considered a multi-agent system since there is only one decision process occurring. A system of this type could be considered to be a multi-link manipulator with flexible links, in the limit where link flexibility approaches infinity. For this reason, most discussion of centralized control refers to hierarchical control wherein a central processor makes behavioral decisions while the swarm agents continue to
perform sensing and localized motion control. Currently, only a few publications exist dealing with centralized control of a multiple-robot system, including [10, 54].

In cases of decentralized control, the complexity or sophistication of the individual agent must necessarily be greater. The individual agent’s processing power (and programmability) must be capable of making appropriate decisions. These decision processes may be complex, involving full state-machines, or may be as simple as Matarić’s Nerd Herd collision avoidance algorithm (as reproduced in [12]):

\[
\text{If another robot is on the right}
\]
\[
\text{turn left}
\]
\[
\text{otherwise turn right}
\]

or the algorithm for "following":

\[
\text{If an object is on the right only,}
\]
\[
\text{turn right.}
\]
\[
\text{If an object is on the left only}
\]
\[
\text{turn left.}
\]
\[
\text{If an object is on the left and right,}
\]
\[
\text{And has been there for some time,}
\]
\[
\text{stop.}
\]

In order to make decisions, agents must have the ability to sense conditions within the environment. Once again this may be as simple as a contact sensor or as advanced as a full-color camera, laser-rangefinder, synthetic olfactory sensor, acoustic sensor, \textit{et cetera}. While decentralized control is the most common approach for multiple robot teams [59], the system may be limited in its global coherency due to limitations in sensing or communication ability.

Hierarchical control refers to a type of control wherein certain robots are given control over other groups of robots. There may be multiple tiers of such grouping.
This type of control may allow certain scalability not possible with centralized control, but would experience significant performance degradation in the event of the singular failure of a high-level robot.

Hybrid control utilizes both localized (i.e., on-board) control processing and external (i.e., centralized) sensing and/or communication. Individual agents have sufficient processing and sensing ability to make local decisions, such as position- or velocity-control or obstacle avoidance. At the same time, the central supervisor oversees the system, making global decisions such as the identification of targets or goals.

2.2 Communication

Natural swarms have a major advantage in their built-in communication methods, advanced sensors, and advanced computing systems. The type, amount, and rate of communication among members of a swarm all depend on the type of tasks to be accomplished by the swarm. Natural swarms rely on vision, acoustic, and olfactory sensing for navigation, decision making, and problem solving.

Bee scouts communicate location information for fields of flowers by performing a precise dance within the hive. Other bees observe the scout’s dance and perceive information such as distance to the field and heading relative to the sun’s position [68].

Ant scouts leave a trail of pheromones as they explore, allowing other members of the colony to follow once a resource is detected [7, 10]. Communication between ants appears to be mostly chemical, including sharing of information regarding the type of resource detected. The collective behavior for retrieving large quantized items additionally seems intrinsic; all ants in the vicinity take hold of a piece, and once there is enough collective effort, the resource begins to be moved. This behavior does not appear to require shared information, just a shared understanding of the task at hand. Such behavior has been emulated using Khepera-II robots [52].
2.2.1 Agent-Agent Communication

Observation of these simple agent-agent interactive behaviors facilitating the transition from collective to cooperative behavior give hope that simple behavioral algorithms may be implemented to replicate this natural efficiency, robustness, and increased capability. Such cooperative tasking has been examined and implemented by Martinoli and Easton [52] with Khepera-II robots equipped with Gripper Turrets in cooperative stick-pulling experiments without using direct communication, relying instead on stigmergic communication. Stigmergy refers to communication via changes in or to the environment, such as moving objects in a manner that is detectable by other members of the multi-robot system.

Any method of communication requires some form of sensing. This may include any of the sensing modalities mentioned above, or their combination. In robotics, these communication forms are often replaced by some sort of wireless radio-frequency communication as it may be difficult to construct a system of pheromone deposition and detection. Other modes of communication includes multi-colored lights or LEDs such as used by “s-bots” [23, 55]. Recently Ampatzis, et al, published results of experiments on the evolution of auditory communication between robots [1].

2.2.2 Agent-Host Communication

Situations also may arise where a swarm should receive direction from or relate information to an individual. This could be analogous to the bee example above, where one agent must communicate in a broadcast modality to share one-way communication with a large number of other agents. This behavior may be reminiscent of the action of a platoon of soldiers in response to orders from a commander. In such a situation, there is likely to be communication both between a leader and the rest of the group for general orders as well as among members of the group to coordinate
actions in pursuit of the goal.

Scalability in a situation such as this should be carefully observed. It may be easy to scale a swarm which only requires host-to-agent communication as this is done in broadcast fashion. However, if individual swarm members must provide feedback to the host, a communication bottleneck may arise in the reception and processing of information received by the host.

2.3 Localization

For any autonomous mobile robot, whether acting independently or within a swarm, knowledge of its position within the environment is critical for successful navigation. This position may be relative to features in the environment, such as the starting position, goal, or other arbitrary point, or may be relative to other members of the swarm. If a supervisory positioning system is available, localization becomes a trivial task, although the speed and reliability of communication becomes the limiting factor. To mitigate the necessity of high-speed communication as well as to allow scalability of a swarm (where the communication and perceptive requirements of the central supervisory member quickly exceed capability), the individual mobile member should be tasked with additional localization responsibility. A thorough discussion of the reasons for localization is provided by Negenborn in his Master’s Thesis [58]. Published research in collective or collaborative localization includes that of Fenwick, Newmann, and Leonard, [28], Fierro, et al [29], Howard, Matarić, and Sukhatme [33], Rekleitis, Dudek, and Milios, [62], and Roumeliotis and Bekey [64, 65]. Odometry provides a well-known and simple method of maintaining an individual’s position or velocity estimate in the absence of position updates from external sources. A discrete time, geometry-based odometry model is given in Equations (4.18) and (4.19). The odometry must be used with caution, as errors in the model
cause errors in estimated position to grow in a quadratic fashion [40, 41]. For systems using odometry, correction of current position must be periodically performed based on some sort of external reference. Calibration of parameters in the odometry model can greatly reduce the errors, but since the error grows quadratically with distance traveled, close but imperfect odometry calibration will eventually result in error. Still, odometry calibration reduces the periodicity requirement on external updates, and much attention has been given to various methods for localization estimation and odometric calibration such as the methods of Antonelli [4, 5, 6], Bak [9], Borenstein and Feng [15, 16], Larsen [46], Leonard [47], and Martinelli [50, 51]. Several calibration methods are discussed in Chapter 4.
A main goal of the work described in this thesis is the creation of a testbed suitable for conducting experiments with multiple robots. This chapter details the physical aspects of the testbed. Section 3.1 illustrates the global coordinate frame of the testbed arena. Section 3.2 discusses the robots used in this work, including some usage notes. Sections 3.3 and sec:cognex describe the position sensors that are the sources of all measurement in this work. Section 3.5 gives the transformations between the sensor frames and the global reference frame.

All of the experiments were performed in the Khepera-II Arena of the Duke University Robotics and Manufacturing Automation Laboratory (Duke RAMA Lab), Hudson Hall 030D. A composite photo of the testbed, including six robots and all of the sensors, is shown in Fig. 3.1.

3.1 Physical Layout

The arena for this experiment is a wooden surface approximately 84cm by 168cm, constrained by walls 7cm tall. The traversable area is not a full rectangle, as the arena supports intrude 2cm by 2cm in each corner.
Figure 3.1: Composite photo of arena and sensors
The global coordinate system is defined as follows (See Figure 3.2):
x-axis is aligned with the short edge of the arena
y-axis is aligned with the long edge of the arena
z-axis completes the right-handed coordinate system
θ measures positive rotation about the z-axis, with θ = 0 aligned with the x-axis.
(x, y) = (0, 0) lies outside the traveled area due to the construction of the arena.

Figure 3.2: Overhead view of the Khepera-II arena

Three external sensors provide supervisory data for the arena: one Cognex InSight 5400 machine-vision camera (see Section 3.4) and two SICK LMS400-1000 LADAR sensors (See Section 3.3). The arena is painted black to provide good contrast for the black-and-white video system. To allow the IR proximity sensors on the Khepera-II robots to function more consistently, the vertical constraining walls are covered with
white paper. This is mandated by the tendency of the black surface to absorb the emitted IR pulses (see discussion in Section 3.2.3).

3.2 Agents: Khepera-II

Developed by K-Team [36], the Khepera-II robots provide an adaptable platform for performing real-world, real-agent experiments. The Khepera-II robot, shown in Fig. 3.3, is complete and straightforward to use and modify for simple experiments with mobile robotics. Their small size makes them easy to work with. The Khepera-II robots have a base of 69\,mm in diameter, and are 34\,mm tall (from bottom of wheels to top of base circuit board). The wheels are 15\,mm in diameter, and the wheel separation is 52.5\,mm from center to center. Simple communication pathways allow for reliable (albeit slow) two-way communication with typical PC hardware, requiring only a free RS-232 serial (or USB port with an inexpensive USB-to-RS232 adapter).

The robots utilize a processor, Flash memory for user-created programs, precise wheel encoders (1/12\,mm resolution), eight infrared proximity sensors, a rechargeable battery system providing roughly an hour of operation on full charge, and a modular expansion system allowing additional functionality.

The expansion system allows the capability of the robots to be extended via modules called turrets. The General IO turret allows users to add sensors and actuators to meet their needs without completely designing a custom PCB. Other available turrets include the Radio Turret (RT) and the High Speed Radio Turret (HSRT). Some researchers using these robots for swarm research and demonstration utilize the HSRT, as it provides more reliable and higher speed communication [2, 3, 5]. The HSRT does not, however, allow robot-to-robot (i.e., turret-to-turret) communication as does the RT. Although several turret options are offered, K-Team has defined a standard which allows users to design and build their own turrets if a particular need is not met.
3.2.1 Communication

The Khepera-II robots are capable of several speeds of serial communication, from 9600 baud to 115200 baud. The highest speed option is sometimes unreliable with older Khepera-II BIOS versions, and is thus not recommended for use in COM-critical tasks such as downloading a new user-defined program. All of these modes are for use with the provided serial cable.

The mode of operation for the robot is selected via a 16-position rotary switch on the robot base. There are 16 modes from which to choose, providing different permutations of 8 communication speed and boot-up mode. The operational mode, e.g., the mode that the robot enters upon reset or power-on, is set via this rotary switch.
While the operational mode can only be changed via this switch, the communication speed may be changed via external serial commands or internal (user-defined) executable programs.

Wireless communication with the Khepera-II is also possible using either the RT or HSRT expansion module. The RT allows communication between robots or with a Radio Base connected to a central computer. The RT system is limited to only 9600 baud, and is susceptible to general RF interference as well as network traffic. The HSRT utilizes the much faster and more reliable Bluetooth 2.4GHz communication standard. The utility of the HSRT is limited in a swarm implementation, however, as it only provides agent-host communication with no capability of agent-agent communication.

Upon power-up in terminal mode, the Radio Turrets provide a somewhat reliable method of communication. Implementation of communication in custom executables via the wired serial link are straightforward, and the behavior is consistent with typical serial port communication implemented in ANSI-C. Custom executable implementation of communication via the Radio Turrets, however, is atypical of serial port communications. Additionally the documentation of libraries provided by K-Team for control of the robots is spotty (more discussion of this is given in Section 3.2.2), with no information regarding the Radio Turrets. The only clues for communicating via the Radio Turrets are provided in a single example C file called radiotst.c, included in the KTProject development environment. It should be noted that even this sample program does not behave as expected without modification. After significant effort, attempts to utilize communication with the Khepera-II robots via their RTs were abandoned.
3.2.2 Programming

Implementing custom operational programs for the robots is straightforward when utilizing the KTProject library of functions (included with purchase of Khepera-II robots or via download from the K-Team website). The KTProject IDE includes several sample code files that can be quickly compiled and downloaded to the robots via the COM link. For these operations it is very beneficial to use the high-speed, wired serial link. For more complex controllers (resulting in larger binary files) the high-speed link provides a large time-saving advantage. Additionally, the robots provide a user-flash capability, allowing the user to burn an image into memory after downloading a new executable.

As noted previously, the documentation for programming the Khepera-II robots is spotty. The only programming manual available at the time of this writing is a document dated June 2002, with indicated version 0.2, and the watermark “Preliminary” on the cover. The document does provide critical information regarding control of the robots, including the functions necessary to control motor functions, communicate with sensors, establish and orchestrate multiple tasks, and monitor critical hardware resources. However, the sections regarding communication, which are critical to this experiment, are empty.

Additionally, there is a BIOS reference manual, which is essentially a listing of low-level functions built-in to the Khepera-II BIOS. This is a critical set of information to include, yet it provides only the syntax for calling these functions without discussing the meaning or usage of some parameters. Especially lacking is discussion of the MSG, SER, and COM subsystems, which may shed light on how best to utilize the Radio Turret, as well as other add-on extension turrets.
3.2.3 Sensing Ability

Each agent is equipped with eight infrared proximity sensors. The sensors are transceivers, sending an infrared pulse and measuring the reflected response. When detecting non-absorptive and non-dissipative obstacles, the sensors perform reliably with a usable range of roughly $1 - 5\text{cm}$. The minimum detection distance does not adversely affect the utility of the data, as the sensors are set back in the body of the robot roughly $1\text{cm}$. Depending on the incident surface and ambient light, the maximum detection distance may vary widely. In particular, flat-black surfaces prove nearly undetectable by the sensors. Figure 3.4 and Figure A.1 illustrate the negative-exponential function. The steep decay may induce large errors when using
the proximity sensors for distance measurement. As such, these sensors are most effectively used to detect the presence of an obstacle rather than to measure the distance to an obstacle.

The sensor data is easily accessed through the included software library. The difficulty lies in using the data when the objects in the environment vary in reflectivity.

3.2.4 Motion Control

The Khepera-II robots are Type (0,2) robots according to Canudas de Wit’s wheeled-robot definitions [19]. The library of functions provided with the KTProject IDE includes some closed-loop motor control functions. The inclusion of such functions makes for simple development and quick integration with user functions. These functions have a few shortcomings.

The motor velocity control is fairly robust and provides repeatable results. One limitation observed with the velocity control is an apparent lack of an integrator anti-windup mechanism. Thus, if a robot encounters an object that impedes its commanded motion and cannot pass the obstacle, the torque command to the wheels continues to increase until either the wheel-surface friction is overcome or the robot pushes past the obstacle. In the case that the robot pushes past, the torque command is not decreased quickly enough, and the robot accelerates to velocities much greater than the maximum values set in the controller. This is an issue that could be overcome with custom programming.

The position control is quite useful for simple motion, although it could be dangerous without careful implementation in a closed-loop control system. The position control uses a time-optimal velocity profile, commanding the maximum acceleration until maximum velocity is achieved, then maximum deceleration to stop at the commanded position. This time-optimal servocontrol has been shown to work well for single-degree-of freedom actuators; however, this 1-DOF control method fails for
motion of a body whose kinematics couple the actuators. Since the target position is expressed as some specific number of encoder counts for each wheel, it is easy to see that the final position and orientation of the robot could vary greatly from the target position and orientation if the rate of each wheel is not monitored closely. Small errors in the control loops cause variations in the body’s angular velocity.

For proper servocontrol, the wheels should be velocity- or torque-controlled based on wheel encoder feedback and the kinematic model of the agent. This model would likely be similar to one of the odometry models presented in Section 4.4.

3.3 SICK LADAR Sensor

![Figure 3.5: LADAR coverage of the Khepera arena](image)

Range measurements are made by two SICK LMS400-1000 industrial LADAR Measurement Systems. These units may communicate to the central host via a half-duplex serial link (RS-232) or 10MBit, half-duplex Ethernet (TCP/IP). For real-time use or display of range data, Ethernet communication must be utilized. The units
utilize diode lasers scanned over 70°; the phase of the reflected light is compared to the reference giving the distance the light has traveled. Clearly for a periodic signal the phase could incorrectly indicate the wrong distance, so the range of the LADAR unit is bounded. The range of the LMS400-1000 is \(70 \text{mm} - 3 \text{m}\).

The LMS400-1000 allows the user to configure several parameters of the scan. Scan frequency may be set from 360\(Hz\)-500\(Hz\), with angular resolution ranging from 0.1333°-1.0°. Measured distances (at each angular point) are reported from 500\(mm\)-3000\(mm\). Range resolution is 1\(mm\), although the reported standard error of systematic measurement error is 4\(mm\) for typical remission values of 40-100%, reaching as high as \(\pm 10\text{mm}\) for lower remission values. Remission values below 6.5% lead to invalid measurements, reported explicitly as 0\(mm\). To facilitate optimum data collection, range, edge, median, and mean filters may be set.

Two LADAR units are required for three reasons. The first two reasons are the minimum range and limited detection angle of the LADAR units. Utilizing two units at opposite ends of the arena ensures complete coverage of the entire arena by at least one sensor (see Figure 3.5). The third reason is the susceptibility to line-of-sight obstruction; that is, an agent may occlude one (or more) other agents lying within the wedge defined by the tangents from the first agent to the centerpoint of the LADAR unit (see Figure 3.6). The LMS400-1000 scan plane is approximately 4.5\(cm\) above the arena surface, which is about 1\(cm\) higher than the Khepera-II robots without any expansion turrets, rendering them undetectable. To work around this shortcoming, the robots are wrapped in a tube of typical white paper extending from the top of the IR proximity sensor ring to the bottom of the top-most expansion turret.

### 3.4 Cognex InSight 5400 Vision System

A real-time industrial manufacturing video system manufactured by Cognex provided overhead visual feedback for absolute positioning. The feedback is limited to two-
Figure 3.6: LADAR occlusion of distant agents by a proximal agent

dimensional black-and-white images, with a (best) spatial resolution of roughly 2.43 mm/pixel. This resolution is not constant throughout the field of view, however, as the image exhibits a pincushion effect, especially noticeable towards the edges of the arena.

This pincushion effect is clear in Figure 3.2 especially along the bottom wall. Furthermore it is easy to see the projection effects with increased lateral displacement from the nadir axis. Specifically, the vertical walls at the +y max wall are clearly visible, and the vertical components of the robots at the left extent of the image are easy to see.

The Cognex system utilizes two main hardware components. The camera itself
acts as a standalone computer, performing both image capture and image processing. The other is a desktop computer through which a user may control or configure the camera. The interface may be via serial (RS-232) or Ethernet ports. Through these interfaces, all of the imaging parameters may be set: integration time, sensor gain, field of interest, etc. For simple control and configuration, the serial connection provides sufficient bandwidth, but for real-time download and display of video the Ethernet interface must be utilized. Using Ethernet and with no image processing, live video at roughly 40 frames per second is possible with little or no apparent lag. Actual lag has not been quantitatively evaluated at this time.

The camera may be configured to perform certain image-processing tasks such as identifying user-defined patterns. The pattern matching routine is intended to be used in a manufacturing setting where parts (of known shape) may enter the work cell in an unknown location and orientation. This capability was exploited in a previous RAMA Lab experiment to perform identification of individual robots and allowing a more robust method of measuring robot orientation (see Section 4.5).

For this experiment, the Cognex camera provides measurements of \([x, y, \theta]\) for all identified features in the arena, but will not uniquely identify the robots. As will be discussed in Chapter 4, the reported measurement of orientation \(\theta\) is meaningless except in certain circumstances where the Cognex pattern-recognition capability is used.

### 3.5 Coordinate Frame Definition

The origin of the reference from for this experiment was chosen, for convenience, to be coincident with the \((0,0)\) point of the Cognex camera’s field of view. The location and orientation of each LADAR unit was found by direct measurements relative to this origin. For a brief discussion of future coordinate frame estimation, see Section 7.1.2.
The Cognex Insight 5400 machine-vision sensor is located at

\[
\vec{P}_{\text{COG}} = (584\, \text{mm}, 762\, \text{mm}, 2286\, \text{mm})^T
\]  

(3.1)

relative to the arena coordinate frame, with the nadir anti-parallel to \( \hat{z} \). The sensor frames of the SICK LMS400 LADAR units are given in Eqs. (3.2) through (3.7). Note that the rotation angles for the SICK sensor frames align the sensors’ 0° reference angle to the arena reference frame, although the sensor only reports angles from 55° to 125°.

\[
\vec{P}_{\text{SICK1}} = (787\, \text{mm}, 165\, \text{mm}, 45\, \text{mm})^T
\]  

(3.2)

\[
\vec{P}_{\text{SICK2}} = (168\, \text{mm}, 1536\, \text{mm}, 45\, \text{mm})^T
\]  

(3.3)

\[
R_{\text{SICK1}} = \begin{bmatrix}
\cos(22.5^\circ) & \sin(22.5^\circ) & 0 \\
-sin(22.5^\circ) & \cos(22.5^\circ) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]  

(3.4)

\[
R_{\text{SICK2}} = \begin{bmatrix}
\cos(205^\circ) & \sin(205^\circ) & 0 \\
-sin(205^\circ) & \cos(205^\circ) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]  

(3.5)

\[
T_{\text{SICK1}} = \begin{bmatrix}
\cos(22.5^\circ) & \sin(22.5^\circ) & 0 & 787\, \text{mm} \\
-sin(22.5^\circ) & \cos(22.5^\circ) & 0 & 165\, \text{mm} \\
0 & 0 & 1 & 45\, \text{mm} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(3.6)

\[
T_{\text{SICK2}} = \begin{bmatrix}
\cos(205^\circ) & \sin(205^\circ) & 0 & 168\, \text{mm} \\
-sin(205^\circ) & \cos(205^\circ) & 0 & 1536\, \text{mm} \\
0 & 0 & 1 & 45\, \text{mm} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(3.7)
In this chapter, a variety of estimation tasks required for robotic navigation and this experiment are discussed. The proposed architecture consists of two main components: a central supervisor and a homogeneous swarm of agents. The central supervisor has superior computational ability relative to the individual swarm agents, and receives measurements of the swarm agents in discrete-time via either a pair of laser rangefinders, a Cognex Insight 5400 industrial machine-vision sensor, or both. The members of the swarm are assumed to be visually indistinguishable and to utilize identical on-board estimation and control software. A limited communication ability via on-board radio turrets allows individual agents to communicate one-on-one with the central supervisor, and allows the central supervisor to communicate either directly with individuals or via broadcast messages.

In each section, current methods for each specific estimation task are described and compared. At the end of each section, the proposed method for performing that estimation task is described. The proposed methods have been chosen and designed to allow maximum performance under a scalable solution with respect to swarm size.
4.1 External Position Sensor Processing

External position measurements are available to the central supervisor from a Cognex Insight 5400 vision system and from a pair of SICK LMS-400 LADAR rangefinders. The data from an individual LADAR unit is in polar coordinates, and is reported over evenly spaced angular space. The vision system data consists of an arbitrarily ordered list of $x - y$ positions of detected swarm agents.

The vision system utilizes a simple background contrast comparison and an area (i.e., blob size) threshold. In order to provide ample contrast against the black background of the arena, identical discs of common white laser printer paper are attached to the uppermost turret on each robot. Since the discs are uniform, and the vision system uses only simple blob detection, orientation data is not available. The only measurement returned by the vision system is position data. The major impetus for this utilization of the vision system is to allow significantly faster image processing (see Section 3.4 for discussion of alternative use of the vision system). The Cognex video system in general provides robust and accurate positional data, but if multiple agents are present in the field of view the system is not able to uniquely distinguish them from one another. The issue of maintaining track of individual robots and properly associating the measurements with corresponding estimates is addressed in Section 4.2.

The LADAR units each return a set of data covering a wedge of $70^\circ$, discretized into $0.25^\circ$ samples. At each angular point, an estimated range is returned. This data is simply a sweep over the field of view of the sensor, so the positions of the agents within the field of view must be extracted from background features, e.g., the walls of the arena. Two sets of raw data taken simultaneously, with six Khepera-II agents in the arena, are shown in Fig. 4.1 and Fig. 4.2. For display purposes, the data has been converted into $x - y$ coordinates, but remain defined relative to each respective
The traditional method for extracting features from this sort of 1-D data is the Hough Transform. The Hough Transform was first introduced by Hough in a patent [32] and further developed by Duda and Hart [26]. Initially used for line or edge detection within an image, general forms of the transform have been developed to extract other features such as ellipsoids or arbitrary shapes. The algorithm is criticized, however, for its processing and memory inefficiency.

Wavelet transforms have recently gained traction in image processing applications due to their success at locating, within a set of data over an independent variable, features of interest, especially in 2-dimensional datasets such as digital images. For perceiving specific features, appropriate wavelet shapes must be used.
4.1.1 Proposed Method

For this work, the Circular Hough Transform and Wavelet Transform were rejected for use in arc-detection due to the perceived complexity and computational requirements associated with these methods, especially for one-dimensional processing. Two simple methods (from an implementation and processing standpoint) have been developed and compared.

The first arc-detection method proposed may be thought of as an edge-detection filter. A low-pass filter (LPF) to remove noise due to discrete spacing is convolved with a differentiating filter, and the result is applied to sample data, collected off-line with an agent at a known location. The resulting filtered data is truncated to the region containing the agent. This truncated, filtered data is now used as a template.
for the matching algorithm.

In the online setting, each new set of range data is filtered and compared point-by-point to the template in a sum-of-squared-error (SSE) sense. The points with the lowest SSE are identified as probable agents. The angular extent of the agents within the LADAR sensor is a function of range, so the template may not work over the full range of the LADAR.

The template described above is defined at a specific range, thus it is unlikely to work for ranges significantly different than that for which it was defined. Thus, multiple templates must be generated for different ranges. As it is based on empirical data, a comprehensive set of range data must be accumulated. Acknowledging that a more closed-form estimation method may be desirable, another arc-detection method was developed.

The second proposed method is similar to a correlation function. Exploiting knowledge of the geometry of the Khepera-II robots, it is possible to predict the signal returned by the LADAR sensors if a robot is in the field of view. Simple trigonometry defines the angular extent of the robot at a given range, as illustrated in Fig. 4.3. The radius of the Khepera-II robot is $R_K$, and the angular extent of the robot at range $\rho$ is $2\theta$. The predicted arc is stored to a vector $S$ of length $N$. The incremental angle used in the algorithm is $\theta_i$, and the estimated range to the robot at angle $\theta_i$ is $S_i$. Relating Fig. 4.3 to Eq.(4.1),

$$\rho = OF, \psi_i = \angle OA_i C, \quad R_K = CA_i = CF, \quad S_i = OA_i.$$  

The formulae for calculating the predicted arc (i.e., the elements $S_i$ of $S$) are given in Eq.(4.1), and the MATLAB® software implementation is shown in Section B.2.
\[
\theta = \arcsin\left(\frac{R_K}{\rho + R_K}\right)
\]
\[
N = \text{floor}\left(\frac{2\theta}{\Delta\theta}\right)
\]
\[
S_0 = \rho
\]
\[
\text{if } N \text{ even, } N = N - 1
\]
\[
\text{for } i = -\frac{N - 1}{2}, \ldots, -1, 1, \ldots, \frac{N - 1}{2}
\]
\[
\theta_i = |i\Delta\theta|
\]
\[
\psi_i = \arcsin\left(\frac{\rho + R_k}{R_k} \sin \theta_i\right)
\]
\[
\text{if } \psi_i < \pi/2
\]
\[
\psi_i = \pi - \psi_i
\]
\[
\phi_i = \pi - \psi_i - \theta_i
\]
\[
S_i = R_K \frac{\sin \phi_i}{\sin \theta_i}
\]

At short ranges just larger than sensor minimum range, the agents extend over approximately 5.46°, which corresponds to 21 sample points at 0.25° angular resolu-
tion. At the maximum extent of the arena, the agents exhibit an angular extent of approximately 1.8°, or only 6 to 7 sample points.

Using these equations, a predicted arc $S$ is created for every point in the raw data. The raw data surrounding the current point is truncated to the same number of points $N$ as the predicted arc. These two vectors are then normalized, and their inner-product is evaluated. The resulting scalar is multiplied by a confidence factor; this final product is the score for that range point. Offline simulations of this method utilizing real, previously gathered data have verified that $N$ is a very good confidence factor, as it varies appropriately over the range.

Upon completion of the point-by-point scoring described above, the top scores are selected. The top scores must be separated by some number of points, otherwise the arc of a single robot may contain multiple top scores. When used with a smoothing algorithm to alleviate the multiple peaks observed for a single robot, this method has proved to be not perfect yet quite robust at identifying agents within its field of view with few missed identifications or false positives. The MATLAB implementation is shown in Section B.1.

This algorithm does require that the number of agents is known, so in cases where there are fewer agents in the field of view (as may happen due to range limits or occlusion as discussed above) false matches are guaranteed. The handling of these situations are discussed in Section 4.2.

4.2 Tracking

Track adherence is critical to maintaining good estimation by ensuring that measurements are applied properly to the estimates of the correct agents. Due to the presence of multiple agents in the field-of-regard for the LADAR and vision systems, the order of measurements within a sample may be rearranged from sample to sample. Additionally, there may be occasional excessive error in the measurements due
to a sensor glitch or communication error, or false identifications returned either by
the arc-detection described above or by the vision system.

The lack of discriminating information due to physical homogeneity prevents the
vision system from maintaining consistent tracking or ordering of the $P$ individuals
in the arena. At each measurement step, the $[2 \times P]$ array of measurements will be
reported, but the order of the measurements varies depending on factors outside of
the normal model. The order of identified agents within the dataset from the vision
system shows variation even when all agents are stationary. Additionally, the order of
measurements from the LADAR system is based on the best score, and thus varies
from sample to sample due to sensor noise. The individual LADAR sensors lack
complete arena coverage (necessitating the redundancy as described in Section 3.3),
so there must be a way to acknowledge that the reported measurements may not be
agents, but instead background clutter.

The supervisor must estimate the future positions of the agents and compare
the reported results with the estimates for matches. Matching measurements to
predictions could be done by comparing each ordering possibility and choosing the
measurement order which minimizes the total error. However, this requires $P!$ iter-
ations, imposing a computational limit on the scalability of this method. This also
does not take into account the possibility of a false identification of a background
feature or the occlusion of an agent. In a situation where a background feature in
the LADAR data receives a higher score than one of the agents, the measurement
for that agent should be marked invalid. Using a total \textit{sum of squared error} (SSE)
criterion without verifying individual data point validity, a single mismatch could
incorrectly force wrong choices for multiple other agents’ measurements.

Data validation in tracking has been studied for several years, in many cases
alongside Kalman Filter methods and especially in related fields such as radar track-
ing. Reid’s method of tracking multiple targets \cite{60} provides clear applicability to this
case, although this method does not account for drop track, the case where a target is no longer in the field of view. Bar-Shalom [11] covers many methods for single- or multiple- target tracking, including Reid’s Multiple Hypothesis Tracking algorithm, generally centered on the Kalman Filter, Extended Kalman Filter, or Second-Order Kalman Filter. Cox and Hingorani [25] proposed a computationally efficient implementation of Reid’s MHT algorithm, claiming optimal assignment results in $O(P^2)$ time.

4.2.1 Proposed Method

A matching and validation algorithm has been developed to validate and reorder the data to the extent possible. The method draws inspiration from all of the above methods, but with an eye towards simplicity of implementation.

The target matching or data association portion of the algorithm is based on the Hungarian Algorithm, first identified by Kuhn [43] and further developed by Munkres [56]. The algorithm solves the minimum-cost assignment problem in polynomial time rather than factorial time. The error between a measurement and an estimate is the cost, and the algorithm finds the association that minimizes the error.

The implementation of the algorithm utilizes a software package by Markus Buehren, downloaded from the Mathworks MATLAB®Central website\(^1\). The inputs to the algorithm are the latest prediction from the EKF, $\hat{X}_{k+1}$, and the latest set of measurements from the position sensors $Z_k$. The output is a list of indices to be used to reorder $Z_k$.

If one or more of the measurements exhibits error that is too large due to background clutter or other error, those measurements would still be applied erroneously in the estimation algorithm. The method for data validation is adapted from Leonard’s geometric beacon tracking method of localization [47], which is based

\(^1\) http://www.mathworks.com/matlabcentral/fileexchange/6543, version updated Jan 30, 2009
on the prior works of Reid [60] and Bar-Shalom [11].

At each prediction step $k$ of the central supervisor’s EKF, a validation gate $g$ is established around the estimated output of the $i$-th agent, $\hat{Z}_k^i = C\hat{X}_k$. The range of the validation gate is dependent on the estimated covariance of error in the prediction, $S$. The predicted state for the $i$-th agent $\hat{Z}_k^i$ is then compared to the $j$-th measurement, $Z_j^k$.

$$V_{ij} = Z_j^k - \hat{Z}_k^i \quad (4.2)$$

$$g^2 \geq V_{ij}S^{-1}V_{ij}^T \quad (4.3)$$

The parameter $g$ defines the region of acceptance, and in much of the literature is called the “number of standard deviations” of error allowed. Bar-Shalom [11] points out that the probability that a measurement lies in the validation region is truly $\chi^2$-distributed and depends on the dimension of the measurement vector. For a one-sided, 90% confidence region for a 2-dimensional measurement vector, $g = \sqrt{4.61} \approx 2.15$. If the measurement vector includes $[x_m, y_m]$ from all three sensors and is thus 6-dimensional, the same confidence region is $g = \sqrt{10.6} \approx 3.26$.

For the matching to be successful, the Kalman Filter described above clearly must maintain estimates within a certain error bound. Experiments and simulations continue at the RAMA Lab in search of the appropriate relationships between acceptable error within the KF estimate and the appropriate value for $g$.

4.3 External Localization

Once the set of individuals is defined, the central supervisor uses the measurements as updates to the estimated state. The estimated state maintained by the central supervisor uses an Extended Kalman Filter for observation of four states for each agent. Individual agents in this swarm are characterized by the state vector $X = [x, y, v, \theta]^T$. 

36
Direct measurement of the first two states are possible given the sensors available in this experiment. The sensors are imperfect and the measurements include noise, so an appropriate measurement-blending filter should improve results. For maintenance of the states of individual swarm members, tracking is required, and, as discussed in the previous section, the state estimates must be accurately maintained for validation and tracking to be successful.

The third state, velocity $v$, cannot be directly measured with this set of position sensors but could be discretely estimated given a sufficiently short sample time. The final state, $\theta$, cannot in any way be directly measured, requiring a state-blending filter to exploit the model (in particular, the non-holonomic dynamics) allowing $\theta$ to be estimated.

In order to preserve scalability of the swarm by limiting required agent-to-host communication, the control input to the individual robots is unknown to the external observer. Bar-Shalom [11] offers multiple methods for dealing with such maneuvering targets: modeling the maneuver input as a random white noise process or autocorrelated Markov noise; estimating the input to correct the state estimate; or augmenting the state with the maneuver input to be estimated.

4.3.1 Proposed Method

To propagate the model, changes in states $v$ and $\theta$ are attributed to process noise, assumed to be zero-mean Gaussian-distributed white-noise with covariance $\xi_v, \xi_\theta$ respectively. The process noise parameters $\xi_v, \xi_\theta$ thus account for the unmodeled and unmeasured control signal. By including these contributions as noise, it can be seen that the magnitude of the parameters are larger than required, but with the benefit of a method that requires no communication with the robot, and thus does not limit scalability. The state transition and measurement models are defined in Eq. (4.4) and Eq. (4.5)
\[
\dot{X}(t) = \begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{v}(t) \\
\dot{\theta}(t)
\end{bmatrix} = \begin{bmatrix}
v(t) \cos \theta(t) \\
v(t) \sin \theta(t) \\
0 \\
0
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
\xi_v(t) \\
\xi_\theta(t)
\end{bmatrix}
\] (4.4)

\[
X(t) = [x_1 \ x_2 \ x_3 \ x_4]^T, \quad F(X(t)) = [f_1 \ f_2 \ f_3 \ f_4]^T
\]

The measurements provided by the vision system or LADAR pair are absolute position measurements, assumed to include zero-mean Gaussian-distributed white noise with covariance represented by \( w_x, w_y \). Measurements are denoted by \( x_m, y_m \).

\[
Z = \begin{bmatrix} x_m(t) \\ y_m(t) \end{bmatrix}
\]

\[
= \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \end{bmatrix}
\] (4.5)

\[
= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} w_x \\ w_y \end{bmatrix}
\]

These estimates are then propagated and updated utilizing a 2nd-order Kalman Filter \([8, 38]\). The Jacobian \( A \) of the state transition model must be found for propagation of the estimated model covariance, and is shown in Eq. (4.6). Additionally, the Hessian matrices, which will be denoted \( F_1, F_2, F_3, F_4 \) respectively, of each state’s transition function must be found for the 2nd-order prediction model. These are shown in Eqs. (4.7) and (4.8). Note that, as the third and fourth rows (\( f_3 \) and \( f_4 \)) of the state-transition function depend only on the noise parameters and not any of the states, \( F_3 = F_4 = 0 \).

\[
A_k = \begin{bmatrix}
1 & 0 & \Delta t \cos(\theta_k) & -v_k \Delta t \sin(\theta_k) \\
0 & 1 & \Delta t \sin(\theta_k) & v_k \Delta t \cos(\theta_k) \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (4.6)
The prediction step is given in Eqs. (4.9) and (4.10), which is used at each iteration to initially update states before applying the correction. The subscript \( k \) indicates the iteration step. These estimates are then propagated and updated in typical form. The propagation and update equations are presented without derivation. Additional specific details may be found in references [8] and [38].

\[
\bar{X}_{k+1} = \hat{X}_k + F(\hat{X}(t))\Delta t \\
+ \frac{1}{2} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T tr(F_1 P_k) \\
+ \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T tr(F_2 P_k) \\
\bar{P}_{k+1} = A_k P_k A_k^T + Q_k \\
Q_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Xi_v \Xi_\theta^t(\nu) \\
0 & 0 & 0 & 0 & 0 & \Xi_v \\
0 & 0 & 0 & \Xi_\theta^t(\nu) \\
0 & 0 & 0 & \Xi_\theta^t(\nu) \end{bmatrix}
\]

Without measurements, the system propagates via the state transition nonlinear function. In update steps where measurements are available and valid, the innovation is performed as shown in Eq. (4.12).

\[
S_{k+1} = C \bar{P}_{k+1} C^T + R \\
K_{k+1} = \bar{P}_{k+1} C^T S_{k+1}^{-1} \\
\hat{X}_{k+1} = \bar{X}_{k+1} + K_{k+1}(Z_{k+1} - C \bar{X}_{k+1}) \\
P_{k+1} = (I - K_{k+1} C) \bar{P}_k
\]
\[ R = \begin{bmatrix} W_x & 0 \\ 0 & W_y \end{bmatrix} \quad (4.13) \]

In Eq. (4.10) and Eq. (4.11) above, the process noise covariance matrix (more specifically, \( \Xi_k^\theta \)) is indicated as a function of time-step \( k \). This is due to the observation that as \( v(t) \to 0 \), the uncertainty in \( \hat{\theta} \) rises due to the dynamics of the robot. For this reason, \( Q_{1.4} = \Xi_k^\theta \) is defined as a function of \( v \), such that it achieves its maximum value for \( v = 0 \) and its minimum value when \( v = v_{\text{max}} \). Candidate functions are given in Eqs. (4.14) through (4.17) and compared in Chapter 5.

\[ Q_\theta(v) = q_\theta(1 - \frac{v}{v_{\text{max}}}) + \alpha \quad (4.14) \]

\[ Q_\theta(v) = q_\theta(1 - \left(\frac{v}{v_{\text{max}}}\right)^2) + \alpha \quad (4.15) \]

\[ Q_\theta(v) = \frac{q_\theta}{\alpha + |v|} \quad (4.16) \]

\[ Q_\theta(v) = \frac{q_\theta}{\alpha + v^2} \quad (4.17) \]

### 4.4 Odometry

Individual self-localization requires the robot to have the ability to make appropriate measurements of its environment. In the scenario where \textit{a priori} information about the environment is known and where the robot has the ability to make measurements with sufficiently high resolution, the self-localization problem has been extensively covered \cite{39, 47, 51, 65, 66, 64, 70}. The Khepera-II robots do not have such a measurement capability, and thus must rely on external position updates. To maintain the on-board position estimate, odometry is utilized. The odometry model should
be calibrated in order to minimize the required rate of the external updates. This section covers several methods of odometry calibration.

For a differential-drive, two-wheeled robot with a passive third-wheel, slider, or caster, the approximate odometry equations may be written in discrete time as shown in Eq. (4.18) and Eq. (4.19). The position and orientation (which together define the pose of the robot) are given by the triplet \((x, y, \theta)\) in environmental coordinates. For simplicity of frame mapping, the position of the robot is taken at the midpoint of the collinear wheel axes, which is the natural center of rotation of the robot. This frame is right-handed, with \(\theta\) measured positive about the (unobserved) \(z\)-axis. For incremental parameters, the subscript \(k\) denotes the change in value over the interval \([k-1, k]\), while for cumulative parameters, the subscript \(k\) denotes the value at the end of the interval. The incremental angular rotation of each driveshaft over the interval is \(\Delta \psi^{R/L}_k\). The wheel radii are represented as \(r^{R/L}_k\). The encoder scale factor mapping encoder counts to radians of wheel shaft rotation is \(c^{R/L}_k\), assumed distinct for each shaft. Incremental encoder measurements for each wheel are given by \(\delta e^{R/L}_k\). The incremental distance traveled by each wheel is given by \(d^{R/L}_k\). The separation of the points of contact of the wheels is given by \(b\). The incremental change in orientation of the robot is given by \(\Delta \theta_k\), and the incremental, scalar change in position is denoted \(\Delta S_k\).

\[
\begin{align*}
\Delta \psi_{k}^{R/L} &= \delta e_{k}^{R/L}c_{R/L}
\[4.18]\n\Delta \theta_{k} &= \frac{d_{k}^{R} - d_{k}^{L}}{b} \\
\Delta S_{k} &= \frac{d_{k}^{R} + d_{k}^{L}}{2}
\end{align*}
\]
\[
x_{k+1} = x_k + \Delta S_k \text{sinc}\left(\frac{\Delta \theta_k}{2}\right) \cos(\theta_k + \frac{\Delta \theta_k}{2})
\]

\[
y_{k+1} = y_k + \Delta S_k \text{sinc}\left(\frac{\Delta \theta_k}{2}\right) \sin(\theta_k + \frac{\Delta \theta_k}{2})
\]

\[
\theta_{k+1} = \theta_k + \Delta \theta_k
\]

where,

\[
\text{sinc}(x) = \frac{\sin x}{x}
\]

by convention, \(\text{sinc}(0) = 1\)

Additional navigation values of interest are the curvilinear abscissa \(S\), the total direction-ignorant accumulated distance traveled \(S^{\text{abs}}\) (\(S^{\text{abs}} \geq 0\)), and the total orientation change from initial position \(\theta_0\). These are accumulated as shown in Eq. (4.20).

The recording of these additional values allows the use of methods such as Antonelli’s Least-Squares method [4, 5, 6] discussed in Sections 4.4.2 and 4.4.3, where the initial and final encoder values may be compared to initial and final orientation measurements. These parameters are used in individual Identification (Section 4.5), and may also be useful for comparing errors for various estimation methods.

\[
S^{\text{abs}}_{k+1} = S^{\text{abs}}_k + |\Delta S_k|
\]

\[
S_{k+1} = S_k + \Delta S_k
\]

\[
\theta_{k+1} = \theta_k + \Delta \theta_k
\]

Odometric error increases quadratically with distance [40, 41], hence it is quite susceptible to model errors. Many methods for calibrating odometry or mitigating the associated errors have been proposed [4, 5, 6, 15, 16, 22, 46, 50, 51, 61, 70]. Most of these authors agree that the sources of odometric error can be reduced to two major subgroups: systematic error and non-systematic error. For any of these methods to work properly there must be a known position reference to allow
correction. Frequently this reference is an on-board, exteroceptive sensor such as a laser rangefinder or vision system. Alternately, this reference may be provided by an off-board supervisory sensor that communicates absolute position updates to the robot, or may be implicit via sensing of and communication with other members of a swarm, providing collective (but possibly relative) localization and calibration.

The systematic error can be easily calibrated by appropriate parameterization of the odometry model, followed by application of an appropriate estimation method. Some possibilities include geometric estimation, Least-Squares Filters, or variants of the Kalman Filter.

4.4.1 UMBmark Method

One such parameterization of the model is Borenstein and Feng’s UMBmark technique [15, 16], which assumes two error parameters, $E_r$ and $E_b$, to estimate wheel radius uncertainty and wheelbase uncertainty, as defined in Eq. (4.21).

$$E_r = r_R/r_L$$
$$E_b = b_{actual}/b_{nominal}$$

The UMBmark technique is not ideal for multiple reasons. The first is that the true average value of the wheel radii is assumed to be known and thus is not estimated. The effect of this is that orientation error will be reduced, but total distance traveled may still have large (and growing at least linearly with distance) error. This method also requires that the robot follows a specific square-shaped closed trajectory during calibration, with the robot ideally stopping with final pose the same as the initial pose. Further, the trajectory must be repeated several times in clockwise and counter-clockwise directions. At the completion of each trajectory traversal, accurate measurements of the final pose relative to the starting pose must
be made. This type of calibration is unsuitable for online calibration, as the robot cannot perform meaningful navigation tasks while constrained to such a path.

4.4.2 Least-Squares Method using X, Y, and Orientation

Antonelli’s method of odometry calibration [4, 6] starts with the physical mapping of measured driveshaft angular velocities $[\omega_R, \omega_L]^T$ to the state parameters $[v, \dot{\theta}]^T$ (linear and angular velocity) via the state-transition matrix $C$ as shown in Eq. (4.22).

$$
\begin{bmatrix}
v \\
\dot{\theta}
\end{bmatrix}
= \begin{bmatrix}
r_R/2 & r_L/2 \\
r_R/b & -r_L/b
\end{bmatrix}
\begin{bmatrix}
\omega_R \\
\omega_L
\end{bmatrix}
= C \begin{bmatrix}
\omega_R \\
\omega_L
\end{bmatrix}
$$

(4.22)

The method requires the construction of two regressor matrices whose $P$ rows are trajectory data sets, one for orientation and the other for position. This data is compared to start and end values of accumulated wheel encoder values in the over-defined system shown in Eqs. (4.23) through (4.26). The notation used here replaces Antonelli’s $T \ast \omega_{R/L}$ with $\Delta \psi^R_{k}$, the incremental change in the driveshaft angle. Note that the model in Eq. (4.25) is the same as Eqs. (4.18) and (4.19) but with the $\text{sinc}(\Delta \theta/2)$ term dropped under the small-angle approximation.

$$
\theta_N - \theta_0 = c_{2,1} \sum_{k=0}^{N-1} \Delta \psi^R_k + c_{2,2} \sum_{k=0}^{N-1} \Delta \psi^L_k
$$

(4.23)

$$
z_{\theta} = \Phi_{\theta} \begin{bmatrix}
c_{2,1} \\
c_{2,2}
\end{bmatrix}
$$

Assembling data from multiple trajectories (i.e., trajectories 1 – $P$) defines the re-
\[
\begin{bmatrix}
\theta_{N,1} - \theta_{0,1} \\
\vdots \\
\theta_{N,P} - \theta_{0,P}
\end{bmatrix}
= \begin{bmatrix}
\Phi_{\theta,1} \\
\vdots \\
\Phi_{\theta,P}
\end{bmatrix}
\begin{bmatrix}
c_{2,1} \\
c_{2,2}
\end{bmatrix}
\tag{4.24}
\]

\[Z_\theta = \bar{\Phi}_{\theta} \begin{bmatrix} c_{2,1} \\ c_{2,2} \end{bmatrix}\]

Likewise for the position data
\[
\begin{bmatrix}
x_N - x_0 \\
y_N - y_0
\end{bmatrix} = \begin{bmatrix}
\sum_{k=0}^{N-1} \Delta\psi_k \cos(\theta_k + \frac{\Delta\theta_k}{2}) \\
\sum_{k=0}^{N-1} \Delta\psi_k \sin(\theta_k + \frac{\Delta\theta_k}{2})
\end{bmatrix}
\begin{bmatrix}
c_{1,1} \\
c_{1,2}
\end{bmatrix}
\tag{4.25}
\]

\[z_{xy} = \Phi_{xy} \begin{bmatrix} c_{1,1} \\ c_{1,2} \end{bmatrix}\]

The position regressor is then constructed from the \(P\) trajectories in the same way as the orientation regressor. Finally, the Least-Squares estimator is used to compute the elements of \(\hat{C}\).

\[
\begin{bmatrix}
\hat{c}_{1,1} \\
\hat{c}_{1,2}
\end{bmatrix} = (\bar{\Phi}_{xy}^T \bar{\Phi}_{xy})^{-1} \bar{\Phi}_{xy}^T \begin{bmatrix}
x_{N,1} - x_{0,1} \\
y_{N,1} - y_{0,1} \\
\vdots \\
x_{N,P} - x_{0,P} \\
y_{N,P} - y_{0,P}
\end{bmatrix}
\tag{4.26}
\]

\[= (\bar{\Phi}_{xy}^T \bar{\Phi}_{xy})^{-1} \bar{\Phi}_{xy}^T Z_{xy}\]

\[
\begin{bmatrix}
\hat{c}_{2,1} \\
\hat{c}_{2,2}
\end{bmatrix} = (\bar{\Phi}_{\theta}^T \bar{\Phi}_{\theta})^{-1} \bar{\Phi}_{\theta}^T \begin{bmatrix}
\theta_{N,1} - \theta_{0,1} \\
\vdots \\
\theta_{N,P} - \theta_{0,P}
\end{bmatrix}
\tag{4.27}
\]

\[= (\bar{\Phi}_{\theta}^T \bar{\Phi}_{\theta})^{-1} \bar{\Phi}_{\theta}^T Z_{\theta}\]

At this point in the estimation, Antonelli ignores the physical basis for the model, and relies on the Least-Squares estimate of \(C\). This method results in a model which
reduces odometric error (as verified experimentally in [4]) by directly substituting
the numerical results of $\hat{C}$ into Eq. (4.22), but the resulting values of $\hat{C}$ do not have
an intuitive physical meaning. Specifically, one cannot uniquely back-derive values
for the physical parameters $r_R, r_L$, and $b$. As shown in their published experimental
data, disparate estimates of these parameters are found, i.e., using $\hat{c}_{1,1}$ and $\hat{c}_{2,1}$ to
eliminate $r_R$ to find $\hat{b}$ will give a significantly different result than using $\hat{c}_{1,2}$ and
$\hat{c}_{2,2}$ to eliminate $r_L$. The authors acknowledge this and propose the addition of a
constraint linking $[\hat{c}_{1,1}, \hat{c}_{1,2}]^T$ with $[\hat{c}_{2,1}, \hat{c}_{2,2}]^T$, but also acknowledge that this would
likely lead to larger reconstruction error due to the loss of an estimated parameter.

For Antonelli’s method to be successful, the $P$ traversed trajectories need not (and
perhaps should not) be identical. Rather, for most-accurate calibration, this method
requires sufficiently rich stimuli, not unlike requiring a broad-spectrum excitation
used in frequency domain system identification. Using the UMBmark trajectories,
Antonelli’s method results in greater error than with both direct measurement of
the odometry model parameters and the UMBmark technique. However, the Least-
Squares method clearly performs better with trajectories containing a greater diver-
sity of trajectory shape elements. The richness of the trajectory can be quantified
by inspecting the right-hand-side of Eq. (4.26). Specifically, the regressor matrices
$\bar{\Phi}_{xy}$ and $\bar{\Phi}_\theta$ must be well-balanced in a singular value sense, and the observation vec-
tors $Z_{xy}$ and $Z_\theta$ must have large norms. For additional discussion of relative scaling
between the orientation and position estimators to ensure numerically well-behaved
results, see reference [4].

Antonelli’s method may be adapted to perform in an online setting, where the
individual robot follows some arbitrary trajectory consistent with its task (mapping,
foraging, etc.) but repeatedly stops after some interval to allow a measurement.
The regressor matrices could then be augmented with the new information, and a
new \( \hat{C} \) found. Implementation of such an algorithm on an on-board processor would be limited by memory restrictions (as the regressor matrices and observation vectors must be augmented at each time step) and processor capability (foremost, the ability to efficiently perform matrix inversion; at each update this inversion would involve a larger matrix).

4.4.3 Least-Squares Method using Path and Orientation

Antonelli solved the issue of the lost physical meaning of \( \hat{C} \) in reference [5] by exploiting the cumulative abscissa of the trajectory \( S \). This method is introduced as a continuous-time model in Eq. (4.28) and discretized in Eq. (4.34).

\[
\dot{S}(t) = v(t) = \frac{r_R}{2} \omega_R(t) + \frac{r_L}{2} \omega_L(t) \tag{4.28}
\]

\[
S(t) = \int_0^t \left[ \frac{r_R}{2} \omega_R(\tau) + \frac{r_L}{2} \omega_L(\tau) \right] d\tau \tag{4.29}
\]

\[
S(t) = \frac{r_R}{2} \psi_R(t) + \frac{r_L}{2} \psi_L(t) \tag{4.30}
\]

The model is thus freed of the non-linear, orientation-dependent constraint between \( x, y, \) and \( \theta \), allowing \( S \) to be integrated independently of \( \theta \). Now the odometry parameters may be constrained by algebraically eliminating \( b \) as in Eq. (4.31).

\[
r_R = -\frac{c_{2.1}}{c_{2.2}} r_L \tag{4.31}
\]

Substitution of Eq. (4.31) into Eq. (4.28) and making a discrete sum approximation defines the regressor for \( S \).

\[
S(t) = r_L \left( -\frac{c_{2.1}}{2c_{2.2}} \psi_R(t) + \frac{1}{2} \psi_L(t) \right) \tag{4.32}
\]

47
\[ S_k \approx r_L(-\frac{c_{2,1}}{2c_{2,2}} \sum_{i=0}^{k} \Delta \psi_i^R + \frac{1}{2} \sum_{i=0}^{k} \Delta \psi_i^L) \]

\[ = \Phi_S r_L \]

\[ \hat{r}_L = (\bar{\Phi}_S^T \bar{\Phi}_S)^{-1} \bar{\Phi}_S^T \begin{bmatrix} S_{k,1} \\ \vdots \\ S_{k,P} \end{bmatrix} \] (4.34)

Observing \( S \) thus allows the estimation of the three odometry model parameters by estimating \( \hat{c}_{2,1} \) and \( \hat{c}_{2,2} \) using Eq. (4.26), \( \hat{r}_L \) from Eq. (4.34), and \( \hat{b} \) and \( \hat{r}_R \) by applying Eq. (4.31) as shown in Eq. (4.35).

\[ \hat{r}_R = -\frac{\hat{c}_{2,1}}{\hat{c}_{2,2}} \hat{r}_L \]

\[ \hat{b} = \hat{r}_R / \hat{c}_{2,1} \]

\[ = -\hat{r}_L / \hat{c}_{2,2} \] (4.35)

This method is far superior to the earlier result, as the calculations of \( S \) and \( \theta \) are decoupled, and the odometry model parameters can be found independently in a physically meaningful way. Some of the other limitations remain, however, including the necessity of calculating matrix inverses in finding the Least-Squares estimator and the potentially unbounded size of the regressor matrices when used in online calibration. This limitation (growth of matrices) may be mitigated by implementing it as a Recursive Least Squares (RLS) method [48] or by limiting the number of calibration iterations. A system using this method is further constrained by the requirement of a fast external position-tracking system to accurately observe \( S \), hence the velocity of the robots must be bounded in relation to the position-tracking system’s sampling rate. The measurement system for this type of calibration must be capable of accurately measuring orientation. This method is likely not scalable in a swarm setting due to the requirements placed on the vision system and supervisor.
4.4.4 Augmented, Extended Kalman Filter Method

Murata and Hirose [57] proposed the use of an Extended Kalman Filter (EKF) for odometry correction based on observations of known waypoints. Larsen et al [46] extended this to an augmented, Extended Kalman Filter (AKF) method for online calibration of odometry parameters, allowing odometry model calibration. Starting with the nonlinear odometry model given in Eq. (4.19) with state \( X_k = [x_k, y_k, \theta_k]^T \) and assuming a simple measurement model represented by Eq. (4.36), the system may be expressed as in Eq. (4.37).

\[
Z = \begin{bmatrix}
  x_m(t) \\
  y_m(t)
\end{bmatrix} \\
= \begin{bmatrix}
  x(t) \\
  y(t)
\end{bmatrix} + \begin{bmatrix}
  w_x^n \\
  w_y^n
\end{bmatrix}_{W_k}
\]

\[
= \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0
\end{bmatrix} X(t) + \begin{bmatrix}
  w_x \\
  w_y
\end{bmatrix}
\] (4.36)

\[
X_{k+1} = a(X_k, U_k, W_k, k)
\]

\[
Z_k = c(X_k, W_k, k)
\] (4.37)

The input vector \( U_k \) contains \( \Delta S_k \) and \( \Delta \theta_k \) based on encoder readings at iteration \( k \), as shown in Eq. (4.18). The measurement noise inputs, \([w_x^n, w_y^n]\), are assumed to be Gaussian-distributed white noise with mean zero and respective covariances \( W_x \) and \( W_y \). The process noise inputs are likewise assumed to be zero-mean Gaussian-distributed white noise with covariances \( V_x, V_y, \) and \( V_\theta \) respectively.

The Extended Kalman Filter algorithm may then be constructed.
PREDICTION:

\[ \hat{X}_{k+1} = a(\hat{X}_k, U_k, 0, k) \]

\[ \bar{P}_{k+1} = A_k \bar{P}_k A_k^T + Q \]  

\[ S_{k+1} = C \bar{P}_{k+1} C^T + R \]

\[ K_{k+1} = \bar{P}_{k+1} C^T S_k^{-1} \]  

\[ \hat{X}_{k+1} = \hat{X}_{k+1} + K_{k+1} (Z_{k+1} - C\hat{X}_{k+1}) \]

\[ P_{k+1} = (I - K_{k+1} C) \bar{P}_k \]

where

\[ A_k = \left( \frac{\partial a}{\partial \hat{X}_k} \right) \bigg|_{U_k, W_k=0} \]

\[ = \begin{bmatrix} 1 & 0 & -\Delta S_k \sin(\theta_k + \frac{\Delta \theta_k}{2}) \\ 0 & 1 & \Delta S_k \cos(\theta_k + \frac{\Delta \theta_k}{2}) \\ 0 & 0 & 1 \end{bmatrix} \]  

\[ Q = \begin{bmatrix} V_x & 0 & 0 \\ 0 & V_y & 0 \\ 0 & 0 & V_{\theta} \end{bmatrix} \]

\[ R = \begin{bmatrix} W_x & 0 \\ 0 & W_y \end{bmatrix} \]

\[ C_k = \frac{\partial c}{\partial \hat{X}_k} \bigg|_{V_k=0} \]

\[ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \]  

At this point the state is augmented with the odometry model parameters \( \delta b \), \( \delta r \), and \( \delta l \), defined as in Eq. (4.43). These are correction factors that correct the
odometry model during operation. These new definitions are substituted into the original odometry model, Eqs. (4.18) and (4.19), resulting in the AKF state model. The Jacobians $A$ and $G$ for this augmented model are derived in Eq. (4.44) through Eq. (4.54).

\[
\delta_R = \frac{r_{R,act} \cdot c_{R,act}}{r_{R,est} \cdot c_{R,est}}
\]

\[
\delta_b = \frac{b_{act}}{b_{est}}
\]

\[
d_k^R = r_R \delta_R \Delta \psi_k^R + \nu_R
\]

\[
d_k^L = r_L \delta_L \Delta \psi_k^L + \nu_L
\]

\[
A_{aug} = \begin{bmatrix} A & F \\ 0 & I \end{bmatrix}
\]

\[
F = \frac{\partial a}{\partial \delta_i} \bigg|_{\nu_{k,w}=0,i=\{R,L,b\}} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}
\]

\[
G_{aug} = \frac{\partial a}{\partial w} = \begin{bmatrix} \cos \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) & -\frac{1}{2} \Delta S_k \sin \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) \\ \sin \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) & \frac{1}{2} \Delta S_k \cos \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) \\ 0 & 0 \end{bmatrix}
\]

\[
F_{11} = \frac{r_R \Delta \psi_k^R}{2} \sin \left( \frac{\Delta \theta_k}{2} \right) \cos \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) + \Delta S_k \frac{2r_R \Delta \psi_k^R}{\delta_b \Delta \theta_k^2} \left[ \frac{\Delta \theta_k}{2} \cos \left( \frac{\Delta \theta_k}{2} \right) - \sin \left( \frac{\Delta \theta_k}{2} \right) \right] \cos \left( \theta_k + \frac{\Delta \theta_k}{2} \right)
\]

\[
- \Delta S_k \sin \left( \frac{\Delta \theta_k}{2} \right) - \frac{r_R \Delta \psi_k^R}{2 \delta_b} \sin \left( \theta_k + \frac{\Delta \theta_k}{2} \right)
\]
\[ F_{12} = \frac{r_L \Delta \psi_k^L}{2} \text{sinc} \frac{\Delta \theta_k}{2} \cos \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) \]

\[ - \Delta S_k \frac{2r_L \Delta \psi_k^L}{\delta b \Delta \theta_k^2} \left[ \frac{\Delta \theta_k}{2} \cos \left( \frac{\Delta \theta_k}{2} \right) - \sin \left( \frac{\Delta \theta_k}{2} \right) \right] \cos \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]

\[ + \Delta S_k \text{sinc} \left( \frac{\Delta \theta_k}{2} \right) - \frac{r_L \Delta \psi_k^L}{2 \delta b} \sin \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]  

(4.47)

\[ F_{13} = -\Delta S_k \frac{2}{\Delta \theta b} \left[ \frac{\Delta \theta_k}{2} \cos \left( \frac{\Delta \theta_k}{2} \right) - \sin \left( \frac{\Delta \theta_k}{2} \right) \right] \cos \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]

\[ + \Delta S_k \text{sinc} \left( \frac{\Delta \theta_k}{2} \right) \frac{\Delta \theta_k}{2 \delta b} \sin \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]  

(4.48)

\[ F_{21} = \frac{r_R \Delta \psi_k^R}{2} \text{sinc} \frac{\Delta \theta_k}{2} \sin \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) \]

\[ + \Delta S_k \frac{2r_R \Delta \psi_k^R}{\delta b \Delta \theta_k^2} \left[ \frac{\Delta \theta_k}{2} \cos \left( \frac{\Delta \theta_k}{2} \right) - \sin \left( \frac{\Delta \theta_k}{2} \right) \right] \sin \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]

\[ + \Delta S_k \text{sinc} \left( \frac{\Delta \theta_k}{2} \right) \frac{r_R \Delta \psi_k^R}{2 \delta b} \cos \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]  

(4.49)

\[ F_{22} = \frac{r_L \Delta \psi_k^L}{2} \text{sinc} \frac{\Delta \theta_k}{2} \sin \left( \theta_k + \frac{\Delta \Theta_k}{2} \right) \]

\[ - \Delta S_k \frac{2r_L \Delta \psi_k^L}{\delta b \Delta \theta_k^2} \left[ \frac{\Delta \theta_k}{2} \cos \left( \frac{\Delta \theta_k}{2} \right) - \sin \left( \frac{\Delta \theta_k}{2} \right) \right] \sin \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]

\[ - \Delta S_k \text{sinc} \left( \frac{\Delta \theta_k}{2} \right) - \frac{r_L \Delta \psi_k^L}{2 \delta b} \cos \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]  

(4.50)

\[ F_{23} = -\Delta S_k \frac{2}{\Delta \theta b} \left[ \frac{\Delta \theta_k}{2} \cos \left( \frac{\Delta \theta_k}{2} \right) - \sin \left( \frac{\Delta \theta_k}{2} \right) \right] \sin \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]

\[ - \Delta S_k \text{sinc} \left( \frac{\Delta \theta_k}{2} \right) \frac{\Delta \theta_k}{2 \delta b} \cos \left( \theta_k + \frac{\Delta \theta_k}{2} \right) \]  

(4.51)
\[ F_{31} = \frac{r_R \Delta \psi_k^R}{\delta_b b} \]  
\[ (4.52) \]

\[ F_{32} = -\frac{r_L \Delta \psi_k^L}{\delta_b b} \]  
\[ (4.53) \]

\[ F_{33} = -\frac{\Delta \theta_k}{\delta_b} \]  
\[ (4.54) \]

### 4.4.5 Augmented EKF for Non-Systematic Error Prediction

Non-systematic errors are difficult to model and thus predict, as they arise due to irregularities between the mobile robot and the surface being traversed, and may include sources such as wheel slip, wheel skid, or loss of contact between wheel and surface. Therefore, error resulting from non-systematic sources is unpredictable, although the intensity of this error may be bounded in a probabilistic sense. The predicted error intensity places an upper bound on the required frequency of external position update.

One common weakness of the above methods is that they acknowledge non-systematic error but do not account for them. It is perhaps for this reason that the parameter estimates in Antonelli’s earlier works, [4, 6] do not converge, and why Borenstein and Feng’s UMBmark method [15, 16] requires its specific trajectory.

Martinelli *et al* expanded Larsen’s AKF method [50] and provided experimental results [51] for estimating the same systematic odometry parameters as well as for the non-systematic error. Estimating the non-systematic error is discussed by Martinelli in references [50] and [51].
4.4.6 Proposed Method

Each agent uses an augmented Extended Kalman Filter to estimate its current state. The Khepera-II code library allows access to cumulative encoder readings, which are polled at a regular sample period $\Delta t$. The readings are used in the Prediction phase of the AKF. The sample period $\Delta t$ is chosen such that it is large enough to allow the computations to complete, yet small enough that the geometric approximations in the odometry model remain valid. The upper bound of $\Delta t$ is related to the nominal velocity of the agents.

Observations allowing updates in the model come from the central supervisor which cannot occur before identification is accomplished. The first observation received from the central supervisor defines both the initial pose of the agent $X_0$ and the current pose. Future observations only provide the current pose.

4.5 Identification of Agents by the Host System

For a central supervisor to provide feedback and unique information to specific members of a swarm, the system must be able to identify the robots. In a prior RAMA Lab experiment conducted by the author, the vision system (see Section 3.4) was used to uniquely identify individual robots, as well as to provide position and orientation feedback ($x, y, \theta$) relative to the arena environment. The individual swarm members were identified by exploiting the pattern recognition capability of the Cognex system.

In order to generate position and orientation data along with unique identification using this camera system, it was necessary to design patterns that the pattern-recognition system would be able to use reliably. With a bit of experimentation it was determined that the smallest square feature the image system could identify is approximately 0.25in. Note that this is not the pixelation scale factor of the camera. With this in mind, and knowing that the circular footprint of the Khepera-
II robots is approximately 2 in, asymmetric patterns were generated on a circle with 2 in diameter, inscribed with a 3x3 grid of 0.5 in squares. Grid squares with half the spatial frequency were used in the hope that this much larger feature size would be more than sufficient to guarantee robustness in pattern matching. This problem turned out to be much more difficult to solve than suspected. In order to be uniquely identifiable, and provide orientation data, the patterns must have no more than 1 axis of symmetry. This in itself is simple to execute, and ten of these ID discs (see Figure 4.4) were generated. Initial tests with the vision system gave disappointing results with several cases of misidentification, and the complexity of the problem became apparent.

**Figure 4.4**: Unique IDs for Cognex identification via pattern-recognition
The issues that cause difficulty are blur due to focus, and off-orthogonal pixelation that occurs when the robot is askew to the cardinal axes of the camera system. In these cases, the sharply drawn grid boundary lines are no longer well-defined, and the squares are no longer well-defined. Couple these issues with the non-linear pixel scaling towards the edges of the arena, and the pattern-recognition software is easily spoofed. For example, the patterns of ID-7 and ID-d (see Fig. 4.4) are different when the sharp shapes are visible. However, if ID-7 is rotated by 45 degrees, and the blurring operations of the camera are considered, these patterns end up being quite similar and close enough to spoof the pattern recognition software. Some tests were performed with the ten initial patterns in order to select a few patterns that would work best together. In these tests, all of the pattern discs were randomly placed in the arena, closely spaced. The pattern-recognition software was then enabled for one pattern at a time, observing the real-time results. This series of tests was performed with the patterns grouped in two regions of the arena: towards the left-most edge (min-y) of the arena, and clustered around the middle of the arena. Results of these tests are shown in Tables 4.1 and 4.2.

One of the interesting results of these tests is the widely varying times required for the Cognex software to achieve a match. Based heavily on the timing results along with the accuracy results, patterns 4, 6, 7, and 8 were chosen for the remainder of that experiment. Unfortunately, the timeline of the experiment prevented expenditure of effort optimizing this process. As a result, the patterns are still not always identified uniquely from one another. Furthermore, the time required to match those patterns is quite costly. For a swarm of only four members, the data sample rate was reduced to approximately 1 frame/sec, with nearly 2 seconds of lag.

The use of this type of video recognition clearly places a severe limitation on the scalability of the swarm when pattern recognition is required for identification and orientation. Additionally, this type of implementation makes the otherwise homoge-
neous robots heterogeneous, arguably violating the spirit of swarm robotics.

<table>
<thead>
<tr>
<th>ID</th>
<th>time (ms)</th>
<th>Pos (+/− pixels)</th>
<th>Angle (+/−°)</th>
<th>Score</th>
<th>Misses</th>
<th>Crossovers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>255</td>
<td>0.01</td>
<td>0.5</td>
<td>88+/-2</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>always</td>
<td>a,b,c,d,7</td>
</tr>
<tr>
<td>4</td>
<td>190</td>
<td>–</td>
<td>(0,15)</td>
<td>–</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>6</td>
<td>210</td>
<td>0.03</td>
<td>0.2</td>
<td>–</td>
<td>some</td>
<td>none</td>
</tr>
<tr>
<td>7</td>
<td>170</td>
<td>0.1</td>
<td>0.4</td>
<td>90</td>
<td>rare</td>
<td>none</td>
</tr>
<tr>
<td>8</td>
<td>125</td>
<td>0.04</td>
<td>0.2</td>
<td>84+/-2</td>
<td>some</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.1: Cognex pattern matching tests (left edge of arena)

<table>
<thead>
<tr>
<th>ID</th>
<th>time (ms)</th>
<th>Pos (+/− pixels)</th>
<th>Angle (+/−°)</th>
<th>Score</th>
<th>Misses</th>
<th>Crossovers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>220</td>
<td>0.01</td>
<td>0.1</td>
<td>98+/-3</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>-1 (-13,0,17)</td>
<td>90-94</td>
<td>some</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>170</td>
<td>0.02</td>
<td>0.3</td>
<td>95+/-3</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>6</td>
<td>205</td>
<td>0.01</td>
<td>0.2</td>
<td>98+/-3</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>7</td>
<td>160</td>
<td>0.02</td>
<td>0.1</td>
<td>93+/-1</td>
<td>rare</td>
<td>none</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>0.01</td>
<td>0.1</td>
<td>94+/-1</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 4.2: Cognex pattern matching tests (center of arena)

### 4.5.1 Proposed Method

As discussed above, the central supervisor will locate probable agents in the arena based on position measurements from the pair of SICK LADAR sensors and from the Cognex Insight 5400 vision system. Identification, defined here as matching detected robots to radio communication identities, is necessary for the system to uniquely communicate individual update information.

As discussed in Section 4.2, the individual agents are indistinguishable to both the LADAR pair and to the vision system. The task of identification is then a more difficult estimation problem. The identification of individuals will be accomplished via comparison of the central supervisor’s state estimates with those of the individual agents. The agents have a limited amount of memory, so the shared data should be limited as much as possible, i.e., it should not contain a full time history of estimates.
Under this constraint, it is desired to find a set of parameters that, under non-identical motion of agents, should provide distinguishing information. The proposed set of data to be used for identification is \([\hat{X}_k, \hat{S}(k)]\), the current state estimate and the current estimated magnitude of the estimated abscissa. As the agents do not know the true values of \(X_0\) in the absolute reference frame, this is actually the agent’s estimate of the changes in state and abscissa estimate since sample \(k = 0\).

The methods discussed in previous sections, such as Antonelli’s Least Squares method and the validation gating used by Reid, Bar-Shalom, and Leonard, may be adapted and combined to form probability-based identification hypotheses for matching external estimates to on-board estimates of these proposed parameters.
Simulations are widely used in collective and cooperative robotics research to verify algorithms and performance. While a major focus of this thesis is the construction of a testbed for real-world experiments, simulations are still valuable and in some cases necessary to meet timelines and to overcome hardware limitations. To the extent possible, the simulations conducted under this work utilize real data collected in the testbed with real sensors observing real robots.

For the current investigations, MATLAB® was used extensively both for conducting simulations and performing real-time data collection and processing. The MATLAB® environment allows access to a wide variety of mathematical and signal processing tools, as well as easy interfacing with Java programs allowing access to any communication ports available on the central computer.

Several simulation packages are available for robotics simulation, including the commercial products Webots [71] from Cyberbotics and Robotics Developers Studio from Microsoft, as well as open-source GNU Public Licensed packages such as Gazebo/Player/Stage. Each of these packages offers specific advantages.

Webots is the simulation software recommended by K-Team with promises of
cross-compiling ability, allowing control algorithms to be run in simulation for testing and validation before being downloaded directly to the Khepera-II robots for use. While this does appear to be possible, significant effort in writing support libraries is required before this functionality can be utilized.

Microsoft recently introduced a product called Robotics Developers Studio. According to online documentation, this software development package appears to also allow offline simulation as well as assistance in writing hardware control code. The package contains some libraries for common hardware and popular robotic systems such as LEGO MindStorms NXT, KHR-1, MobileRobots PioneerP3DX, and iRobot Create and Roomba. The package additionally contains generic code for common components such as differential-wheels, motors, sensors, and communications. An “Express” version of the package is available for free download, and academic discounts do apply to the full version. The commercial cost is more than Cyberbotics’ Webots, but the cost to academic institutions is comparable.

There are several zero cost options available under the GNU Public License or similar. One in particular is the Player/Stage/Gazebo. This package makes available many common software components related to robotics, with similar coverage to the Microsoft Developers Studio. A review of recent robotics literature reveals that many researchers have chosen this package and have been able to use it successfully. Discussions with fellow researchers at Duke University as well as at other universities have confirmed these reports. The package is currently released only for Linux, UNIX, and BSD-based Mac OSX operating systems; a port to Windows XP and Vista is planned but not yet available as of this writing.

5.1 Simulation Environment

Several algorithms are designed and tested under this work. The mathematical bases for these algorithms are described in Chapter 4. These algorithms have been,
in general, developed and tested individually before being brought together for the central supervisor. After the communication interfaces to the SICK LADAR LMS-400 sensors and the Cognex InSight 5400 vision system were developed, several sets of raw data were gathered for development and test. Due to the failure to establish communication with the Khepera-II Radio Turrets (see Section 3.2.1), the algorithms for Identification (Section 4.5.1) and Odometry Calibration (Section 4.4.6) have not been implemented on the physical hardware. The odometry calibration method has been validated via simulation. Simulations of the Identification method have been dropped altogether, and will be investigated by the RAMA Lab in the future.

The simulations have generally been developed as m-file functions with the intent of providing modularity in their use. In some situations, the functions were called via Simulink models for ease of constructing input-output relationships. This is especially true in the case of the Java communication interface to the Cognex and SICK sensors. These methods can be called from functions as well, but the graphical depiction helps to clarify and, in some cases, enforce the causality relationships.

5.2 Experimental Data

Experimental data sets with the Khepera-II robots, SICK LMS-400 LADAR sensors, and the Cognex InSight 5400 vision system have been collected to provide real inputs to the algorithms. The simulations thus include real-world data corruption such as dropped or misread measurement packets, non-real values, noise due to environmental conditions, and sensor noise.

The simulations utilizing these experimental data sets serve to partially replace the originally planned experiments that have been rendered untenable due to the lack of success in establishing user-defined communication between the Radio Turrets and the Radio Base.
5.2.1 Individual Agents

Initial data was gathered from the Cognex InSight 5400 vision system with a single robot present within the arena, either stationary or moving under the Braitenberg obstacle avoidance algorithm built in to the Khepera-II standard library. The implementation is a variant of Braitenberg’s Vehicle 3b, also called Like [17]. The resulting trajectories are non-deterministic and depend strongly on the initial orientation relative to the arena geometry, as well as on the variation among the on-board IR proximity sensors.

These initial data sets were gathered using the distinct identifier discs to allow comparison of direct orientation measurements with orientation estimates. These sets were used to validate methods without the concerns of processing time for the vision system. These data sets contain only data from the Cognex vision system. Four data sets were taken with a stationary robot in different locations to quantify the sensor noise, including evaluation of whether the sensor error is a function of location within the arena.

Since orientation data was measured via pattern-matching, the maximum sample rate is $f \approx 2Hz$. It is desirable, clearly, to maintain comparable sampling rates among data sets such that the dynamics of the system will be equivalently measured, e.g., velocities, turning speeds, etc.

Five data sets were collected with the robot moving. These sets were used to validate the orientation estimation performance of the central supervisor extended Kalman Filter, and to optimize the parameters $g$, $\sigma_g^2$, and $\alpha$. Additionally, an artificial data set was constructed combining these single-robot data sets, and used in some tracking tests.

The sample time of these data sets was crudely controlled to be roughly 2Hz. This is primarily important in the consideration of the estimated values of $\hat{v}$ and
the related parameter $\sigma_v$, as they both scale with $\Delta t$. Since this data is used in artificial-time simulations, these relationships must be carefully observed.

### 5.2.2 Multiple Agents

Further observations were performed without the identifier discs, which therefore do not allow comparison of orientation with truth data. However, some conclusions can still be made regarding the error in the later tests, as some of the later data-collection events were conducted with precisely specified control commands to the Khepera-II agents.

These data sets were collected with six robots in the fields-of-view of all three sensors. The number of robots moving in each set varies from none to two. These data sets were collected with the Cognex vision system and both SICK LADAR units.

For comparison, data without orientation information (using the uniform white discs) was collected with a sample time of $\Delta t \approx 0.5s$; this data included measurements for six agents, rather than just one. However in simulation, $\Delta t$ may be varied to assess its effect on estimation accuracy, as long as the estimated covariance of velocity error is appropriately scaled to account for the different time-scale. Additionally, the update rate relative to the prediction rate may be varied to ascertain the requirement for data convergence and error-robustness.

### 5.3 External Sensor Data Processing

The proper conditioning and use of the data from the external sensors is clearly the basis for the majority of the estimation tasks proposed here. Evaluation of the stationary data sets yields the error intensities shown in Table 5.1. This table shows standard deviation of the measured orientation $\theta$, measured position $x$ and $y$, and distance from the origin $s$. The column headings indicate the nominal location and
orientation of the robot when the measurements were made. These values established the covariance estimates used in the EKF model.

<table>
<thead>
<tr>
<th>(x,y,θ)</th>
<th>(611,818,92°)</th>
<th>(249,793,89°)</th>
<th>(1020,757,89°)</th>
<th>(627,51,89°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>σ_θ (degrees)</td>
<td>0.089</td>
<td>0.0646</td>
<td>0.099</td>
<td>0.1966</td>
</tr>
<tr>
<td>σ_x (mm)</td>
<td>0.0135</td>
<td>0.0181</td>
<td>0.0404</td>
<td>0.0596</td>
</tr>
<tr>
<td>σ_y (mm)</td>
<td>0.0129</td>
<td>0.0118</td>
<td>0.0202</td>
<td>0.0415</td>
</tr>
<tr>
<td>σ_s (mm)</td>
<td>0.0139</td>
<td>0.0135</td>
<td>0.0330</td>
<td>0.0597</td>
</tr>
</tbody>
</table>

Table 5.1: Cognex sensor noise variance (from raw data).

5.3.1 LADAR Arc-Detection

One of the data sets described above included six stationary robots in the field of view of one of the LADAR units. Five of the six robots were visible to the other LADAR unit. This data set was used to quantify the error in LADAR measurements due to sensor variance and estimation, i.e., arc detection, error.

The data was processed via the algorithm for arc detection described in Section 4.1.1. The data from the LADAR sensor with only five visible robots contained at least one false-positive measurement at every time step, as the algorithm is configured to identify a set number of arcs. Due to this systemic flaw in the algorithm, the simplest of tracking methods described in this thesis, the Hungarian Algorithm, fails to properly segregate the measurements.

The data for the LADAR with six robots in view was successfully processed, and the measurements were appropriately grouped by the Hungarian Algorithm. During this data processing, an inherent weakness in the Hungarian Algorithm was identified. Specifically, the Hungarian Algorithm may return different optimal order of measurements if, instead of the cost matrix, the transpose of the cost matrix is passed. This is a result of the nature of the algorithm itself, wherein the minimum costs are assigned first in row order, and, if a mapping is indeterminate, the column minima is used to break the tie.
To overcome this limitation, the Hungarian Algorithm was called twice during each time step: first with the cost matrix, i.e., the error matrix, and then with the transpose of the cost matrix. If the optimal order for both calls is the same, it is used to map the new measurement to the previous estimate. If the orders differ, the normalized error for each order is found. That is, the previous estimate is differenced from the new measurements with each of the two optimal orders, and the resulting error with the lower 2-norm is chosen as the optimal order. In this way the measurements were successfully segregated for the sensor with all 6 robots in the field of view.

Following the successful matching, statistics of the measurement quality were evaluated. The results are shown in Table 5.2. The columns of the table indicate the identified (mean) center of the robots in polar coordinates ($\bar{r}, \bar{\theta}$ in $mm$ and degrees) relative to the sensor frame. The middle of the LADAR range is defined as 90°. The errors listed are variances with units $mm^2$. Note that this error includes sensor noise as well as uncertainty introduced by the arc detection method.

<table>
<thead>
<tr>
<th></th>
<th>866.1</th>
<th>1243.3</th>
<th>773.0</th>
<th>739.7</th>
<th>774.1</th>
<th>835.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{r}$</td>
<td>61.20</td>
<td>66.54</td>
<td>73.70</td>
<td>88.67</td>
<td>104.35</td>
<td>115.8</td>
</tr>
<tr>
<td>$\bar{\theta}$</td>
<td>61.20</td>
<td>66.54</td>
<td>73.70</td>
<td>88.67</td>
<td>104.35</td>
<td>115.8</td>
</tr>
<tr>
<td>$\sigma_r^2$</td>
<td>33.6380</td>
<td>6.4910</td>
<td>23.3824</td>
<td>20.1348</td>
<td>7.6611</td>
<td>6.7543</td>
</tr>
<tr>
<td>$\sigma_{\theta}^2$</td>
<td>11.1027</td>
<td>3.6504</td>
<td>3.5451</td>
<td>2.7610</td>
<td>3.0882</td>
<td>3.9852</td>
</tr>
<tr>
<td>$\sigma_\theta^2$</td>
<td>24.8988</td>
<td>7.0397</td>
<td>21.6320</td>
<td>21.0867</td>
<td>4.1114</td>
<td>6.1245</td>
</tr>
</tbody>
</table>

Table 5.2: SICK LADAR combined measurement and estimation error.

This set of data provides good spatial variation across the 70° scan, but does not offer much variation in range. There is large variation in the error intensity for each of these robot locations, yet there is not a clear correlation between the error intensity and either the measured range or measured angle.

The most likely source of this discrepancy is the quantization of the measurements. The arc-detection algorithm does not estimate the center of the robot; rather it returns the angular index closest to the point on the arc that lies along the line be-
Figure 5.1: Histogram of range measurements (best case)

tween the sensor origin and the robot center. Additionally, the range measurements are integer values, discretizing the measurement to the nearest mm. Two histograms of range measurements are shown in Figure 5.1 and Figure 5.2, illustrating the ranges of measurements corresponding to the best and worst covariances (columns one and two of Table 5.2).

5.4 Controlled Trajectories

Due to the limited communication ability between the Khepera-II agents and the central supervisor, the identification, odometry updates, and localization portions of this experiment have been limited to simulations with real data. Some real-time experiments are still possible, however, making certain assumptions about the
trajectories of the robots.

Certain trajectories may be specified by sending velocity commands to the built-in motion control. Real-time odometry reporting is not possible with this system. However, by making certain assumptions about the robot control model, odometry errors estimation is possible. Additionally, orientation information may be assumed allowing use of these trajectories in validating the external observer estimation.

5.4.1 Circular Motion

It is quite simple to command Khepera-II robots to move in circles. Velocity is controlled by the built-in motion control; velocity is specified by directly commanding the velocities of each wheel. Commanding different velocities to each wheel will
result in circular motion, with the radius of the circular path \( R(\omega_L, \omega_R) \) defined in Eq. (5.1). It is clear (and expected) that \( \lim_{\omega_L \to \omega_R} R = \infty \), indicating straight-line motion. Likewise, when \( \omega_L = -\omega_R \to R = 0 \) indicating pure rotation.

\[
R(\omega_L, \omega_R) = b \times \frac{\omega_R}{\omega_L} \frac{\omega_R}{\omega_L} - 1 \tag{5.1}
\]

These velocity commands may be sent to the robot through the normal Khepera-II SerCom protocol. Once received, the robot will operate in velocity-control mode with these commanded velocities until powered off or sent a new command. Two data sets were collected with a single robot moving under directly commanded wheel velocities \((12, 10)\), corresponding to \([\omega_L, \omega_R] = [12.8 \, \text{rad/s}, 10.6 \, \text{rad/s}]\). This effectively commands a counter-clockwise circular path of radius 267.5mm.

5.4.2 Straight Line Motion

Commanding the robots to travel in a straight line is straightforward using the above method. However, if it is desired to command the robots along a square or rectangular path, the commanding becomes much more complex. The simplest way to establish trajectories such as these is to write a sequential control program to download to the robots. The cost of this is that the scripted trajectory must be explicitly written and commanded. The velocity must be hard-coded into the program, and the timers within the trajectory script are functions of the desired distance and desired velocity.

A data set was collected in which a single robot was commanded in simple straight lines. The commands were issued as position target commands, leading to trajectory errors as discussed in Section 3.2.4 regarding the control limitations due to coupled dynamics of the implemented time-optimal servocontrol. Ideally, these straight-line trajectories would be dictated via velocity commands as for the circular trajectories described above.
5.5 External Localization

Simulations to verify the external localization methods were conducted using several of the data sets described above. All of these simulations were run as synchronous extended Kalman Filters, i.e., the prediction step and innovation step were always conducted together. The only cases in which a prediction occurred without a corresponding innovation were those in which the measurement for that step was marked invalid.

5.5.1 Extended Kalman Filter

Simulations conducted with the above described data sets as inputs allowed the tuning of model parameters, as the estimated states could be directly compared to the measured states. The relationships between the noise intensity \( Q_\theta \) and estimated velocity proposed in Eqs. (4.14) through (4.17) have been optimized and compared. The parameters for each of these velocity-dependent noise profiles were all optimized under the same set from the single-robot data. The particular data set was chosen somewhat arbitrarily, and the optimal parameters were then applied to the other data sets for performance comparison. The four candidate functions are plotted in Figure 5.3 using the optimized parameters. For additional clarity, the candidate functions are also shown plotted on logarithmic scales in Figure A.2, Figure A.3, and Figure A.4

<table>
<thead>
<tr>
<th>profile</th>
<th>( q_\theta )</th>
<th>( \alpha )</th>
<th>( Q_\theta(0) )</th>
<th>( Q_\theta(v_{max}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \propto -v )</td>
<td>0.91567</td>
<td>1.1411</td>
<td>2.0568</td>
<td>1.1411</td>
</tr>
<tr>
<td>( \propto -v^2 )</td>
<td>0.99853</td>
<td>0.99141</td>
<td>1.9899</td>
<td>0.99141</td>
</tr>
<tr>
<td>( \propto 1/v )</td>
<td>5.2658</td>
<td>3.0141</td>
<td>3.0141</td>
<td>7.0047e-3</td>
</tr>
<tr>
<td>( \propto 1/v^2 )</td>
<td>2.9929</td>
<td>10.587</td>
<td>10.5872</td>
<td>5.3208e-6</td>
</tr>
</tbody>
</table>

Table 5.3: Optimal parameters for \( Q_\theta(v) \)

The MATLAB® Optimization Toolbox was used extensively in generating ap-
appropriate parameters for the central supervisor’s EKF. Optimal parameters for $\alpha$ and $q_{\theta}$ were found for each profile under the condition of minimizing the mean squared error between $\cos Z_{\theta}$ and $\cos \hat{\theta}$. This metric was chosen over a direct comparison of $(Z_{\theta} - \hat{\theta})^2$ to account for angle-wrapping; i.e., to account for the fact that $\hat{\theta} = \hat{\theta} + 2\pi$.

The optimization routine utilized was “fminsearch” over two variables, $q_{\theta}$ and $\alpha$. The optimal parameters for each profile found via the optimization are shown in Table 5.3.

5.5.2 Tracking

The algorithm for tracking multiple robots by the central supervisor is discussed in Section 4.2. Several simulations have been performed with raw data in order to validate these methods and understand the requirements levied on the central su-
<table>
<thead>
<tr>
<th>profile</th>
<th>$g = 50$</th>
<th>min</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>0.1394</td>
<td>0.1394</td>
<td>43</td>
</tr>
<tr>
<td>Set 2</td>
<td>0.2121</td>
<td>0.2121</td>
<td>50</td>
</tr>
<tr>
<td>Set 3</td>
<td>0.1739</td>
<td>0.1677</td>
<td>40</td>
</tr>
<tr>
<td>Set 4</td>
<td>0.2831</td>
<td>0.2758</td>
<td>40</td>
</tr>
<tr>
<td>Set 5</td>
<td>0.1663</td>
<td>0.1512</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 5.4: Estimation errors for $Q_\theta \propto -v$

<table>
<thead>
<tr>
<th>profile</th>
<th>$g = 50$</th>
<th>min</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>0.3074</td>
<td>0.3020</td>
<td>15</td>
</tr>
<tr>
<td>Set 2</td>
<td>0.5524</td>
<td>0.4354</td>
<td>30</td>
</tr>
<tr>
<td>Set 3</td>
<td>lost track</td>
<td>0.2889</td>
<td>10</td>
</tr>
<tr>
<td>Set 4</td>
<td>0.4841</td>
<td>0.3192</td>
<td>15</td>
</tr>
<tr>
<td>Set 5</td>
<td>0.4562</td>
<td>0.2923</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 5.5: Estimation errors for $Q_\theta \propto -v^2$

<table>
<thead>
<tr>
<th>profile</th>
<th>$g = 50$</th>
<th>min</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>0.1796</td>
<td>0.1796</td>
<td>50</td>
</tr>
<tr>
<td>Set 2</td>
<td>0.4301</td>
<td>0.3918</td>
<td>13</td>
</tr>
<tr>
<td>Set 3</td>
<td>0.1883</td>
<td>0.1883</td>
<td>50</td>
</tr>
<tr>
<td>Set 4</td>
<td>0.2731</td>
<td>0.2696</td>
<td>43</td>
</tr>
<tr>
<td>Set 5</td>
<td>0.2274</td>
<td>0.1899</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 5.6: Estimation errors for $Q_\theta \propto \frac{1}{|v|}$

<table>
<thead>
<tr>
<th>profile</th>
<th>$g = 50$</th>
<th>min</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set 1</td>
<td>0.2028</td>
<td>0.2028</td>
<td>50</td>
</tr>
<tr>
<td>Set 2</td>
<td>0.3548</td>
<td>0.3103</td>
<td>40</td>
</tr>
<tr>
<td>Set 3</td>
<td>0.2318</td>
<td>0.1859</td>
<td>40</td>
</tr>
<tr>
<td>Set 4</td>
<td>0.3354</td>
<td>0.3354</td>
<td>50</td>
</tr>
<tr>
<td>Set 5</td>
<td>0.2241</td>
<td>0.1997</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 5.7: Estimation errors for $Q_\theta \propto \frac{1}{v^2}$
pervisor. These simulated experiments confirmed the proper operation of the measurement matching and validation tasks. Simulation was critical to the tuning of the necessary parameters.

After performing the optimization shown above, the optimal values for the orientation noise variation were implemented. The tracking and estimation was then performed on different data sets collected without orientation information but with multiple moving robots. In these simulations, exceptional tracking performance was achieved with $g = 3$ to $g = 5$. For smaller values of $g$, the estimates still tracked successfully but the estimates exhibited additional noise due to occasional “good” measurements being marked invalid.

Figure 5.4 (for $g = 4$) and Figure 5.5 (for $g = 4$) illustrate both the successful track adherence and bad data rejection for two different data sets. Both of these examples utilize the “1 minus V” noise function given in Eq. (4.14) and with the optimal parameters listed in Table 5.3.

5.6 Odometry Calibration

Due to the limitations encountered in establishing radio communication between the swarm agents and the central supervisor, the method described in Chapter 4 for odometry calibration is primarily examined under simulated conditions.

Simulations were attempted utilizing data from the controlled-trajectory data sets. Due to the uncertainty in matching velocity (due possibly to timeline jitter in the collected data), these simulations were inconclusive.

A model of the Khepera-II robot was implemented with errors of magnitude corresponding to values reported in previous odometry calibration papers [4, 5, 14, 15, 16]. The proposed AKF was implemented as a MATLAB® function. The simulated sharing of information between the central supervisor EKF and the agent’s AKF was then simulated at different rates to determine the appropriate rates of information
sharing. For these simulations, the Identification of agents by the central supervisor was assumed to be perfect.

Simulations were performed utilizing six different trajectories:

- counter-clockwise circular path (two revolutions);
- arbitrary path of five segments in time: straight line, rotation about the robot center, straight line, rotation about one wheel, and a circular arc;
- figure-eight (two circles with an inflection point at the tangent);
- alternating semi-circles;
- square path with perfect rotation at the corners; and
To define the trajectories, a fictitious set of incremental encoder readings was generated according to the geometry of the desired path. These artificial readings were then passed to the AKF as control input. The measurements provided to the innovation step of the model were the truth position of the simulated robot with Gaussian noise of intensity corresponding to the Cognex position variances.
Figure 5.6: Semi-circular arc trajectory

Figure 5.7: Odometry calibration for semi-circular arc trajectory
Figure 5.8: Arbitrary trajectory

Figure 5.9: Odometry calibration for arbitrary trajectory
Figure 5.10: Circular trajectory

Figure 5.11: Odometry calibration for circular trajectory
Figure 5.12: Figure-eight trajectory

Figure 5.13: Odometry calibration for figure-eight trajectory
Figure 5.14: Square trajectory

Figure 5.15: Odometry calibration for square trajectory
Figure 5.16: Rounded-square trajectory

Figure 5.17: Odometry calibration for rounded-square trajectory
The simulations described in the previous chapter provide a mixed-bag of results. In general, the methods investigated were successful at making somewhat reliable predictions. The magnitude of error in some of these methods was larger than expected, while other tasks yielded impressive results.

6.1 External Localization and Tracking

The method of varying the estimated process noise covariance of the orientation state as a function of velocity proved to work admirably. All four candidate functions attempted outperformed the normal method of a fixed value for the covariance of $\hat{\theta}$. It should be noted, however, that it is important to properly set the parameters for each of these functions.

The best individual score (minimal error) was realized under the optimized function $Q_\theta(v) \propto -v$ defined in Eq. (4.14). The errors experienced under that function were, for four of the five test trajectories, the lowest among all of the candidate functions. Interestingly, for these particular trajectories, decreasing the validation gate value $g$ raised the error estimates, although tracking was successfully maintained.
even for values as low as $g \approx 10$. For the other noise functions, decreasing $g$ from the value of 50 used in the optimization generally reduces the error to some extent (note the values of $g$ for the minima in Table 5.5.1 through Table 5.5.1). In only one case a minima lower than the $Q_{\theta}(v) \propto -v$ profile score is achieved, that case being Set 4 under $Q_{\theta}(v) \propto \frac{1}{v}$ defined in Eq. (4.16). In that particular case, the result is still very close to the optimal result for the $Q_{\theta}(v) \propto -v$ profile.

It is interesting to note that, in conducting the optimization for the profile function parameters, both the $\propto \frac{1}{v}$ and $\propto \frac{1}{v^2}$ functions exhibited a clear minima in $q_{\theta}$, yet in the $\alpha$ direction there was not a strong gradient. In contrast, the parameter spaces for the $\propto -v$ and $\propto -v^2$ functions were very flat and in general resulted in poor scores except very near to the region of the optimal parameters. Surface plots of the parameter spaces are provided in Figure A.5 through Figure A.8.

6.1.1 Coupling of EKF, Data Validation, and Tracking

Simulation of the external localization algorithm alongside the tracking and validation algorithms revealed an important constraint: the interaction between the validation gate $g$ and the estimated error in the model prediction. It is desired to set the size of the validation gate $g$ as low as possible to maximally reject measurements exhibiting excessive error. If the confidence in the model is too large (i.e., the estimated model covariance, $P$, is too small) good measurements will be rejected.

The validation method used here establishes a probabilistic range such that, even though $g$ is a fixed parameter, the gate effectively shrinks as confidence in the model (and prediction) grows. It is thus very important to use proper covariance estimates (i.e., measurement noise and process noise) which maintain the minimum values of the model covariance $P$ and estimated output $S$. In this case, however, the process noise covariance $Q$ is intentionally varied, thus for larger velocities $Q_{\theta}$ may be too small, leading to an overconfident $S$ estimate.
Setting the validation parameter $g$ aggressively, e.g., $g \leq 10$, resulted in strong rejection of bad measurements. Bad measurements are those in which the vision system reported non-real values due to failed detection, as well as those in which a reported measurement did not correspond to any agent in the field. However, more aggressive values ($g < 3$) in most cases led to rejection of valid measurements. In some cases, this resulted in a non-recoverable loss of track. This is due to the lack of a control input term in the supervisor’s EKF, as it is implicitly included as zero-mean process noise. Thus, the supervisor fails to properly apply new measurements as the prediction propagates away from the last-valid measurement.

Before continuing the discussion, it should be noted that this loss of tracking may also indicate problems in the model as implemented, including improper values of covariance estimates or incorrect execution of either the prediction or innovation steps in the model. For many iterations of these simulations (before the optimal parameters were employed and while errors existed in the simulation code) values of $g$ were required to be very high; in all cases, $g > 1000$ and in some cases $g > 5000$ in order to maintain tracking.

Due to this coupling, the issue of track reacquisition came to light. There are several *ad hoc* methods of dealing with reacquisition, including Bar-Shalom’s Multiple Model Tracking [11] or other similar hierarchical methods. In these simulations, a method was implemented wherein, upon marking a measurement invalid, the model covariance $P$ for that robot’s estimated state is reset to large values along the diagonal (the same values used at model initialization) in order to ensure that upcoming good measurements will be validated.

This discovery prompted the idea that the tracking algorithm must have a method of “reacquisition” available when measurements are marked invalid. As this model does not have access to control input information, the model estimate may quickly propagate away from the truth. A quick reacquisition capability was introduced by
simply resetting the covariance matrix of the state estimate, $P$, to a large value when measurements are marked invalid.

This quick fix provided tolerable results, but additional work is warranted to implement this more judiciously. Specifically, if a measurement is truly a bad measurement, the model should not be “punished”. However, if measurements have been persistently marked invalid for some period of time, there is additional likelihood that the model has propagated away and thus the values of $P$ should be reset.

This is not an ideal solution to the problem. Better methods could employ persistence counters which wait to reset $P$ until several consecutive measurements have been marked invalid, or some other method of resetting the state estimate. Possible implementations of this include hierarchical discrete steps where $P$ grows as multiple levels of persistence are reached (corollary to the multiple-model approach proposed by Bar-Shalom in reference [11]), or continuous growth for at each time step.

6.2 Odometry Calibration

The magnitude of the fictitious noise is important for convergence of the odometry model parameters. Setting this noise too large results in gross deviation from truth and very large model errors, while setting the noise too small prevents the model from converging to truth values.

The requirement for sufficiently rich stimuli for odometry calibration noted by Antonelli [4] is strongly observed here. Intuitively, straight-line trajectories will provide strong feedback to correct error in $\partial L$ and $\partial R$; these trajectories explicitly cannot affect the estimate of $\partial B$, however.

For curved sections, the relationship is less clear, but from the results it is evident that changes in the radius of curvature most strongly stimulate the model for $\partial B$. For trajectories with constant radius-of-curvature, it appears that the model correction
is indeterminate, i.e., the augmented states converge to a set of values that result in correct odometry output for that trajectory yet are not the truth parameters for the odometry model.

The importance of a rich trajectory extends to the magnitude of the fictitious noise in the AKF. With lower richness, the fictitious noise must be larger in order for the model to converge. The larger magnitude, however, introduces excessive noise into the estimation, which can increase the overall odometry error. By ensuring a rich trajectory, the fictitious noise can be kept small (Martinelli [51] uses $0.05^2$; the same value is used in these simulations), allowing convergence with minimal overshoot provided the rich stimuli is sustained.

The richness of these simulated trajectory has not been evaluated relative to Antonelli’s conditioning criterion, but it is quite likely that the “rich” trajectories in this set (arbitrary path, alternating semi-circular path, and rounded-square path) would score favorably.

In contrast to Antonelli’s results, both the square path and the path of alternating semi-circular arcs in this simulation converge well. This is due to the continuous nature of this method, whereas Antonelli’s Least Squares method breaks the trajectory into large segments.

6.3 Sources of Error

A few systematic sources of error have emerged during this experiment. Some errors were predicted but exhibited larger than expected magnitude, while others were not expected at all.

From the outset, the necessity for tracking was acknowledged for both the vision and rangefinder sensors. The conscious decision to neglect orientation and identification data from the vision system forced the use of some sort of tracking and identification estimation. Furthermore by nature of the type of information returned
by the rangefinder, it was clear that under any motion with multiple agents, one or more agents would likely leave and enter the rangefinders' fields-of-view with some regularity.

Previous experience and data collection with the Cognex vision system indicated very stable reporting of position data. Data gathered in experiments conducted prior to this work exhibited very low random error, with no observations of the large jumps encountered in this set of experiments. The working understanding of the vision system was that the only errors that would need to be handled are those due to switched measurement order (in which case, the Hungarian Algorithm would solve the problem) or failed identification (in which case the data is marked explicitly bad).

As such, the relatively frequent observation of such jumps in these experiments was quite unexpected. These jumps may be due to the change in image processing relative to prior experiments, namely the switch to somewhat generic blob detection as opposed to higher fidelity (but much slower) pattern recognition. An additional possibility is the increased sensor sampling rate. These large jumps do seem to occur in a correlated fashion; specifically, the bad measurements often exhibit the correct measurement in one dimension while exhibiting a large deviation in the other dimension, as shown in Figure 5.4a and Figure 5.5a. Future use of the vision system must consider this finding and consider the methods presented here for tracking and data validation.

Another source of difficulty and error was the noise in the laser rangefinder. The observed standard deviation in range measurements is consistent with the specifications provided by the manufacturer as long as the field does not contain sharp variation. In detection regions where corners or edges are detected (e.g., where a robot is in the field with a wall some distance behind it) the error over repeated samples is very large. Review of the data suggests that there is error in the absolute angle of the measurements.
According to the documentation, the normal to the detection window is defined as 90°, and the range data is reported from across a 70° from 55° to 125°. Based on qualitative inspection, the nominal angle varies up to 2∆θ. With this in mind, future experiments employing these sensors should utilize an active calibration to stabilize the LADAR frame.

6.4 Best Use of Position Sensors

The Cognex vision system had proven to be much easier to deal with, and in general provides much more consistent data. There are still bad data points, but the methods laid out in this work have been shown to reject the majority of those bad data points.

The SICK LADAR sensors exhibit an RMS noise level that is nearly two orders of magnitude larger than the Cognex and require significant additional processing before the data is usable. The main advantage of the SICK system is that theoretically the sample rate is much higher. Under the current implementation, which is acknowledged to be a non-ideal kludge, the sample rates of the LADAR and Cognex systems are roughly equivalent.

The only other advantage of the SICK LADAR system is that it is on the same plane as the robots themselves. While this is also a disadvantage considering the added susceptibility to occlusion, the implication is that these or similar sensors could be mounted on robots (assuming sufficient carrying capacity) and become members of the swarm themselves. This may be possible for an overhead vision system with a full 3-dimensional swarm as well, although in such a situation with one or more airborne members, the vision system employed would likely be a simpler camera system requiring off-sensor image processing. As the image processing capability of the Cognex is among the greatest advantages, the clear choice of the vision system over the rangefinder system may not be as clear in such a scenario.

Given the choices of these sensors in this arena, the Cognex vision system is
clearly the better sensor in every category except sample frequency, where it is tied with the SICK system. If the sensors are to be used for metrology, the best choice would be to use all three. In a feedback application where reliable data is paramount, the Cognex is the clear choice unless the SICK system can be made to operate at its full sample rate.
A testbed has been established to facilitate swarm robotics experiments. Multiple estimation methods from somewhat disjoint fields have been brought together in an integrated measurement system. The system has been developed to maximize accuracy without sacrificing sampling frequency. As such, this measurement system can be used for experimental metrology as well as for real-time feedback.

The sensor processing proposed, developed, and utilized in this thesis will facilitate future experiments in the RAMA Lab at Duke University. The methods developed in this thesis have been optimized for the available equipment, but are general enough to be used with many types of external sensors.

The discovery of the less-than-expected communication capability of the Khepera-II robots has clearly limited the work possible in this facility. It is fortunate that this has been discovered as alternative methods or equipment can now be sought.

7.1 Future Work

While many questions have been answered in the research work conducted towards the completion of this thesis, others have now been posed. Several additional in-
vestigative paths of research or development work in the RAMA Lab have been identified either as interesting or critical.

7.1.1 Variation of Process Noise

This work put forth the development of an effective extended Kalman Filter utilizing a process noise term that depends on one of the states of the filter. While several functions were proposed and tested with pleasing results, it is the author’s belief that there is an analytic relationship between the optimal variation function and either the dynamic model of the robot or the control input to the robot (which, in this experiment, was the Braitenberg algorithm). Further investigation on this front will continue, including additional experiments generating different trajectories to continue to improve the understanding of the optimal parameters.

7.1.2 Coordinate Frame Estimation

Utilizing the methods already discussed in this thesis and with the equipment currently available, it should be straightforward to implement a frame-of-reference calibration method for the unified system. By fixing the reference frame to some feature of the mobile system (e.g., a particular robot, one of the sensors, the centroid of the robots, etc.) the measurement system becomes a part of the swarm. The sensors could be made mobile while still allowing quasi-external measurement.

This will additionally allow collective localization of the swarm, facilitating cooperative behavior. Work is planned in the RAMA Lab to supplement the current swarm with additional members including aerial vehicles equipped with vision sensors, taking the place of the Cognex vision system.

7.1.3 Identification and Odometry Calibration

Due to the ineffectiveness of the radio system, the odometry calibration was not successfully implemented. The simulated results of the proposed method, however,
yielded very promising results. Given an alternative method of communication, the identification methods proposed can be tested and implemented. If effective, online calibration should be relatively easily achieved.

7.1.4 Stochastic Control and Random Graph Theory

Research is currently underway at the RAMA Lab to develop a sound mathematical foundation for modeling, analyzing, and controlling a robotic swarm operating in an unknown and possibly dynamic environment. This study focuses heavily on graph theory, specifically the application of random graph theory to a system of autonomous agents in order to control the creation (or destruction) of networks, and the control and stability of these formations in the presence of system uncertainties. The future work will build on recent RAMA Lab studies including [44, 45].

Experimental validation of these control and estimation models is paramount to this study. The characterizations of the sensors currently available for these experiments has provided a solid empirical basis for sensor uncertainty models to be employed. Additionally, the communication difficulties encountered in this work highlight the importance of the planned work regarding uncertainties in communication connectivity.

The estimation methods developed in this method will additionally facilitate the planned experiments. As the planned experiments focus on autonomous agents, the central supervisor will not be used to directly control the agents. The role of the central supervisor (and indeed, its measurement and estimation ability) are critical for the metrology in these experiments. As the agents are independent of the central supervisor, accurate and timely measurements of the systems will be critical for assessing the actual performance of the formation control. The central supervisor capabilities developed here (including both the localization and the sensor processing) will be critical to assessing the efficacy of the newly developed control model.
Bibliography


Appendix A

Additional Images

Figure A.1: IR proximity sensing function of side sensors on Khepera-II robots [37]
**Figure A.2**: $Q_\theta$ vs. $v$, all four candidate functions (semilogx).

**Figure A.3**: $Q_\theta$ vs. $v$, all four candidate functions (semilogy).
Figure A.4: $Q_\theta$ vs. $v$, all four candidate functions (loglog).

Figure A.5: $q_\theta$ vs. $\alpha$ surface plot for “1 minus V” profile
Figure A.6: $q_\theta$ vs. $\alpha$ surface plot for “1 minus $V^2$” profile

Figure A.7: $q_\theta$ vs. $\alpha$ surface plot for “1 over $V$” profile
Figure A.8: $q_\theta$ vs. $\alpha$ surface plot for “1 over $V^2$” profile
Appendix B

Custom MATLAB® Code

B.1 Arc-Detection

A snippet of the code used to detect arcs in the returned LADAR range data is given below. This snippet is included in a function that includes some basic data-validation before performing the loop shown below. The data-validation simply consists of handling data-point dropouts, i.e., those points where 0 is explicitly returned are averaged with their neighbors.

```matlab
% Scan point-by-point for circles
% Don’t scan the endpoints of the angular range, though
for j=plusminus:280-plusminus
    r = range(i,j);
    if r<minrange
        continue
    end
    % Assume that r is on the line between the rangefinder origin
    % and the center of the circle Estimate the "arc" that would
```
% be there if this is the case
[u,Npts] = sickcircles2(r,R);

% Normalize the returned arc:
u=detrend(u);
u=u/norm(u);

% Extract the correct number of points before and after the
% current range point
signal = range(i,j-(Npts-1)/2:j+(Npts-1)/2);
% Normalize the signal
signal = detrend(signal);
signal = signal/norm(signal);

% Take the inner product of the expected arc and truncated
% signal, and scale by Npts, akin to a "confidence" factor.
score(j) = (signal*u)*Npts;

end

B.2 Arc-Prediction

The MATLAB function for generating the predicted arcs for the arc-detection algorithm is given below:

function [u,Npts] = sickcircles2(r,R)
% This function returns a vector u of length Npts representing the
% concave tangent arc expected if a circular object of radius R is
% normal to the LADAR at range r.
%
% The discretization of the LADAR is hardcoded (below) as
% dtheta=0.25 degrees (SICK LADAR default angular resolution).
%
% The arc returned is the "perfect" estimate of the arc at that range
% with no shaping or scaling.

if ~exist('R','var') || isempty(R)
    R = 31;
end
if ~exist('r','var') || isempty(r)
    error('r not defined')
end
dtheta=0.25*pi/180;
Npts = floor(2*asin(R/(r+R))/dtheta);
if mod(Npts,2)==0
    % Npts is even, needs to be odd
    Npts = Npts-1;
end
u=zeros(Npts,1);
% Construct the expected arc:
for k=[-(Npts-1)/2:-1 1:(Npts-1)/2]
    theta=abs(k*dtheta);
    psi=asin(((r+R)/R)*sin(theta));
    if psi<pi/2
psi = pi - psi;
end

phi = pi - psi - theta;
u(k + (Npts - 1)/2 + 1) = R * sin(phi) / sin(theta);
end

u((Npts - 1)/2 + 1) = r;