ABSTRACT

Title of Thesis:	Leader Based Cyclic Pursuit
Name of degree Candidate:	Kenneth L. Miltenberger
Degree and Year:	Master of Science, 2016
Thesis directed by:	P.S. Krishnaprasad Professor Department of Electrical and Computer Engineering

In this work a system of autonomous agents engaged in cyclic pursuit (under constant bearing (CB) strategy) is considered, for which one informed agent (the leader) also senses and responds to a stationary beacon. Building on the framework proposed in a previous work on beacon-referenced cyclic pursuit, necessary and sufficient conditions for the existence of circling equilibria in a system with one informed agent are derived, with discussion of stability and performance. In a physical test-bed, the leader (robot) is equipped with a sound sensing apparatus composed of a real time embedded system, estimating direction of arrival of sound by an Interaural Level and Phase Difference Algorithm, using empirically determined phase and level signatures, and breaking front-back ambiguity with appropriate sensor placement. Furthermore a simple framework for implementing and evaluating the performance of control laws with the Robot Operating System (ROS) is proposed, demonstrated, and discussed.

LEADER BASED CYCLIC PURSUIT

by

Kenneth L. Miltenberger

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2016

Advisory Committee:
Professor P.S. Krishnaprasad, Chair/Advisor
Dr. Kevin S. Galloway, Co-Advisor
Professor Andre L. Tits

Dedication

To my family, friends, and loved ones. I know you probably won't understand much of this paper, but I still love you all the same.

Acknowledgments

First I would like to thank Professor Krishnaprasad for all his encouragement and support. Without him I don't know if I would have even looked for opportunities to do research or do a Master's Thesis.

I would also like to thank Professor Kevin Galloway for all the great work he's done, so that I might expand upon it in a small way. Kevin is a great mentor, from helping me understand his work, to his detailed paper edits and revisions. Of course, my favorite times were trading sea stories in the lab.

Special thanks to Udit Halder for helping make the video, and being my TA for far too many classes. Thanks for putting up with me!

Finally, I'd like to thank my friends and family for their ongoing love and support! It's been a tough ride, but you've helped me through it!

This research was supported in part by the Air Force Office of Scientific Research under AFOSR Grant FA9550-10-1-0250 and FY2012 DURIP Grant from the AFOSR (FA2386-12-1-3002), and by the ARL/ARO MURI Program Grant No. W911NF-13-1-0390. Additionally, this research was supported by the U.S. Coast Guard Advanced Education program

Table of Contents

List of Figures

List of Abbreviations

α	interagent angle control parameter
α_B	agent's beacon angle parameter
β	spacial phase information
κ_i	the i^{th} agent's angle to the $i+1$ agent, relative to the agent's heading, measur
κ_{iB}	the i^{th} agent's angle to the beacon, measured the same as before
θ_i	the angle from agent i to agent $i-1$ relative to the agent's heading
ρ_i	distance from agent i to agent $i+1$
ρ_{iB}	distance from agent i to the beacon
λ	control parameter controlling the leader's attention to the beacon
μ	control parameter for equilibrium scaling
p	pressure, experienced by a microphone
ω	frequency

CB	Constant Bearing
CBB	Constant Bearing, Beacon focused
CBL	Constant Bearing, beacon focused, with a Leader
ARIA	software library for remotely manipulating robots
ROS	Robot Operating System
ILD	Interaural Level Difference
IPD	Interaural Phase Difference
FFT	Fast Fourier Transform
TI	Texas Instruments
DSP	Digital Signal Processing
ADC	Analog to Digital Converter
EPWM	Enhanced Pulse Width Modulation
SYSCLK	System Clock
ADCCLK	ADC Clock
ISR	Interrupt Service Routine
CLA	Control Law Accelerator
DSA	Dynamic Signal Analyzer
MDLE	Motion Description Language Extended
TVA	Topological Velocity Alignment

Chapter 1: Introduction

Missions such as search and rescue, persistent surveillance, and containment of a hazardous substance may be carried out more effectively by a team of diversely equipped agents, rather than a single highly-equipped agent. In this way a common goal can still be achieved, but with sensing redundancy. Specifically, organizations such as the US Coast Guard can benefit from using autonomous agents to carry out missions.

Drones and autonomous sea surface vehicles can significantly enhance Coast Guard rescue efforts by providing long range, persistent searching, with low personnel risk. Currently, vast expanses of open ocean such as the South Pacific prove difficult and costly to patrol and search. For example, in 2013 a sailing vessel was found adrift approximately 1300 nautical miles west of Oahu, HI. Multiple C-130 manned flights were conducted, while a Coast Guard Cutter with a 50 person crew was dispatched to board the vessel, and verify if there were any personnel aboard. The cutter took four days just to reach the vessel and required additional C130 flights to get a precise location of the adrift vessel. Due to fuel limitations, the C130 aircraft could only stay on station for a few hours at a time, requiring subsequent flights to relocate the adrift vessel. [2]

While the mission was accomplished and the vessel was verified to have no persons aboard, the entire effort was extraordinarily costly. According to U.S. Coast Guard Commandant Instruction 7310.1Q "Reimbursable Standard Rates", the inside government hourly rate was $7,140 and $14,975 for the cutter and C-130 respectively, including personnel cost. Assuming two 10 hour C130 flights and 8 days of cutter deployment, the cost of this search and rescue case exceeded 1.6 million dollars. By using Predator drones, with an estimated hourly cost of $3,600 an hour per drone, a team of drones could be deployed to significantly reduce mission costs. Thus we establish motivation for using a team of drones with diverse sensing and target tracking capabilities to perform search and rescue, as well as other Coast Guard missions.

In the interest of accomplishing missions such as search and rescue, cyclic pursuit schemes have been shown to be an effective method of controlling teams of n agents in a decentralized manner, generating desired group motions through dyadic pursuit interactions of agents interacting over a cycle graph (i.e. agent i pursues agent $i+1$, with n^{th} agent pursuing agent 1.) (See, for example, [3–5]). Recent work on beacon-referenced (or "target-centric") cyclic pursuit ([6–8]) has modified the framework to introduce a stationary beacon to which agents can sense and respond. However, in practice it is likely that not all agents will have the same sensor packages or ability to locate targets.

In the context of Coast Guard search and rescue, one drone might have an infrared sensor, while another might have a radio direction finder, surface radar, or no sensing equipment due to search and rescue assistance payloads. Constrained by

size and weight limitations, it might be advantageous from a payload standpoint to equip each drone with a different sensor package, with only a subset of the agents capable of sensing the beacon (or target).

In this thesis, performed in the Intelligent Servosystems Laboratory (ISL), we consider the setting of a collective of agents, with exactly one member dividing its attention between a beacon a neighboring agent in the collective: this agent is designated the "leader". The other agents of the collective are therefore designated as "followers", with each of them sensing a neighboring agent of the team in a cyclic way. This leader based, beacon focused, cyclic pursuit system is referred to as "CBL", in contrast to the works [6] and [9] where "CBB" is used to denote the setting where each agent has knowledge of the beacon and a neighboring agent of the collective. In Chapter 2, we demonstrate that under appropriate conditions on control parameters, circling equilibria still exist for the CBL system, with agent-beacon distance and shape determined by control parameters.

In the CBL context, the leader agent is equipped with a means to sense the beacon in a manner possibly different from how the agents sense their neighbors. In this thesis, we assume the beacon is like an ocean buoy, radially emitting sound, and we equip the leader to sense the direction of the beacon using only sound, whereas the agents sense neighbors through an indoor global positioning system (Vicon). Drawing on the earlier work of Handzel and Krishnaprasad in [10], and Andersson et. al. in [11], an apparatus for robotic phonotaxis is created and implemented using embedded systems. While the algorithm used in this thesis to recover the direction (angle) of a sound source is the same as in the earlier work, the sound

sensing apparatus, the method by which sound signatures are generated, and the way front/back symmetry is broken all constitute a departure from earlier work. Instead of using theoretical signatures for a perfectly spherical "head" apparatus, empirical signatures are generated in a calibration experiment for a somewhat realistic (nonspherical) Styrofoam manikin head, using "optimal" microphone placement. In Chapter 3 we demonstrate that this method of direction finding is good enough to be used in the CBL system.

And finally, given the broad range of controls experiments performed in this work, an overview of the lab's equipment setup is given in Chapter 4. Interactions between the Vicon^{TM} motion capture system, controls software, and robots are described and diagrammed in detail, with the hope that new users might become comfortable with this system quickly. Moreover in an effort to ease and standardize control software development, a software framework based on the Robot Operating System (ROS) and inspired by MDLE [12, 13] is proposed, which facilitates implementation of different control laws in a collective of mobile robots. Such a framework has the advantage of enabling repeatability of experiments and facilitating code maintenance, the results of which are demonstrated through all of the controls experiments performed in this work. Additionally, the power of the framework is demonstrated by creating hybrid, complex control laws based on individual control law libraries.

Chapter 2: Leader Based Cyclic Pursuit

2.1 Theory

2.1.1 System Modeling

The way we frame our model of interaction will be the same in [6], but with a small extension to account for the difference in agent attention. First we define the index set I,

$$I \triangleq \{1, 2, 3, ..., n\}, \tag{2.1}$$

where n is the total number of agents. We assign the agents to indices as follows: the leader is assigned index 1 (thus will be agent 1), the follower paying attention to the leader is assigned index 2 (agent 2), the follower paying attention to agent 2 will be assigned index 3 (agent 3), and so on until all n agents are assigned indices. Addition and subtraction within this set should be interpreted as modulo n such that,

$$n + 1 = 1. \tag{2.2}$$

Thus the leader (agent 1) pays attention to agent n. In addition the leader also pays attention to the beacon (see Figure 2.1). We will refer to this asymmetric scheme

as the CBL system, where L denotes "leader based".

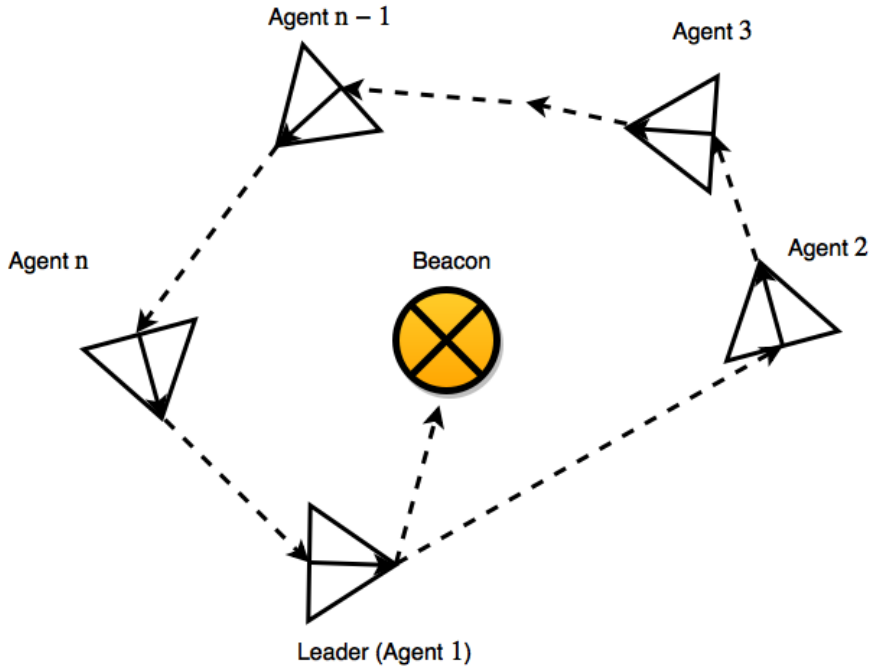

Figure 2.1: Attention graph for leader based cyclic pursuit with a beacon. Arrows denote the direction of an agent's attention.

Each agent is modeled as a unit-mass self-steering particle with a twice-differentiable motion path in $\mathbb{R}^2$. We can then use the natural Frenet frame equations [14] to describe the motion for a group of n agents. By letting $\mathbf{r}_i$ denote the position of the i^{th} agent, the underlying system dynamics can be expressed as,

$$\begin{aligned} \dot{\mathbf{r}}_i &= \nu_i \mathbf{x}_i \\ \dot{\mathbf{x}}_i &= \nu_i u_i \mathbf{y}_i \\ \dot{\mathbf{y}}_i &= -\nu_i u_i \mathbf{x}_i, \quad \forall i \in I, \end{aligned} \tag{2.3}$$

where $\mathbf{x}_i$ is the normalized velocity, $\mathbf{y}_i$ is $\mathbf{x}_i$ rotated $\frac{\pi}{2}$ in the counter clockwise direction, ν_i is the speed, and u_i is the natural curvature which will be the steering

control. The variables $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{r}_i$ are all in $\mathbb{R}^2$. We pack (2.3) into a matrix and define,

$$g_i = \begin{bmatrix} \mathbf{x}_i & \mathbf{y}_i & \mathbf{r}_i \\ 0 & 0 & 1 \end{bmatrix} \in SE(2), \quad \forall i \in I. \tag{2.4}$$

As in [6] we introduce a beacon, however it will only be referenced (i.e. paid attention to) by the leader agent. We denote its location as $\mathbf{r}_B \in \mathbb{R}^2$, along with a fixed frame $[\mathbf{x}_B\ \mathbf{y}_B]$ attached to it. Without loss of generality, the frame can be interpreted as the inertial reference frame. The corresponding element of $SE(2)$ is denoted by g_B. Differentiating g_i we write,

$$\begin{aligned} \dot{g}_i &= g_i \zeta_i \\ \text{where } \zeta_i &= \nu_i \begin{bmatrix} 0 & -u_i & 1 \\ u_i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{2.5}$$

Where from [6], two variables are introduced,

$$\begin{aligned} \tilde{g}_{i,i+1} &= g_{i+1}^{-1} g_i \\ \tilde{g}_{iB} &= g_B^{-1} g_i. \end{aligned} \tag{2.6}$$

By definition of $\tilde{g}_{i,i+1}$, we derive the cycle closure constraint,

$$\prod_{i=1}^{n} \tilde{g}_{i,i+1} = \mathbb{1}_3 \tag{2.7}$$

and beacon consistency conditions,

$$\tilde{g}_{i,i+1} = \tilde{g}_{i+1,B}^{-1} \tilde{g}_{iB}, \quad \forall i \in I \tag{2.8}$$

where $\mathbb{1}_3$ is the identity matrix in $\mathbb{R}^3$.

Following the same approach as in earlier works [6], the state of an agent relative to the beacon and neighboring agents is described as a reduction to scalar shape variables. We denote the counter-clockwise planar rotation by an angle $\phi \in S^1$ by,

$$R(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \in SO(2). \tag{2.9}$$

Then we define the set of scalar shape variables as,

$$\begin{aligned} &\rho_i = |\mathbf{r}_{i+1,i}| && \rho_{iB} = |\mathbf{r}_{B,i}| \\ &R(\kappa_i)\mathbf{x}_i = \frac{\mathbf{r}_{i+1,i}}{|\mathbf{r}_{i+1,i}|} && R(\kappa_{iB})\mathbf{x}_i = \frac{\mathbf{r}_{B,i}}{|\mathbf{r}_{B,i}|} \\ &R(\theta_i)\mathbf{x}_i = -\frac{\mathbf{r}_{i,i-1}}{|\mathbf{r}_{i,i-1}|}, \end{aligned} \tag{2.10}$$

for all $i \in I$, and $\mathbf{r}_{i,j} \triangleq \mathbf{r}_i - \mathbf{r}_j$, for any agent two agents i, j in the system. Figure 2.2 shows a diagram of the scalar shape variables.

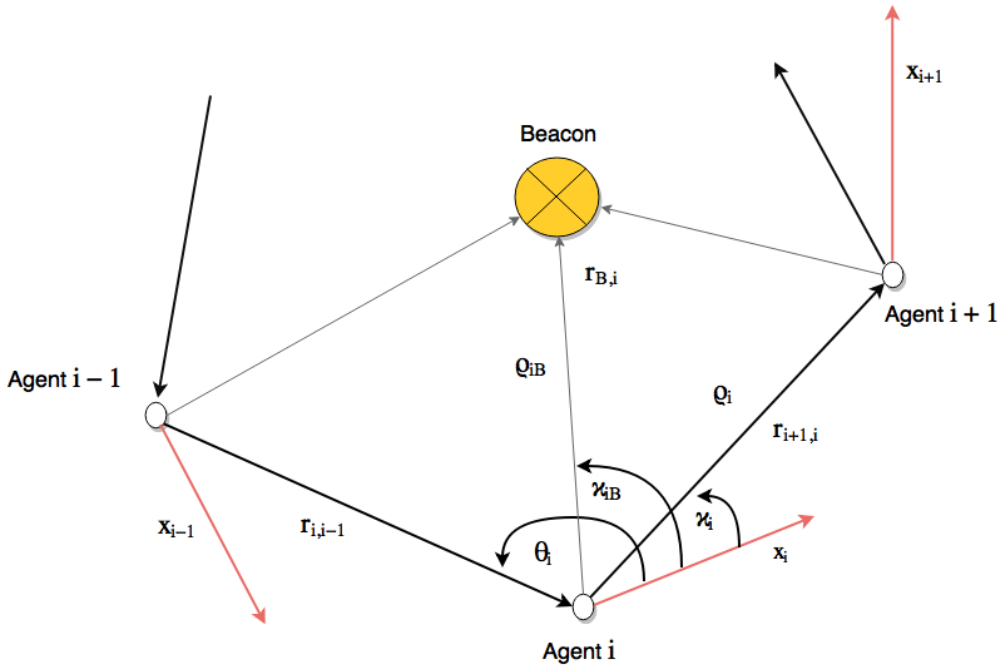

Figure 2.2: Diagram of scalar shape variables $\rho_i, \rho_{iB}, \kappa_i, \theta_i$

Using the scalar shape variables, the dynamics of each agent $i \in I$ can be expressed as follows,

$$\begin{aligned}
\dot{\rho}_i &= -\nu_i \cos(\kappa_i) - \nu_{i+1} \cos(\theta_{i+1}) \\
\dot{\kappa}_i &= -\nu_i u_i + \frac{1}{\rho_i}(\nu_i \sin(\kappa_i) + \nu_{i+1} \sin(\theta_{i+1})) \\
\dot{\theta}_i &= -\nu_i u_i + \frac{1}{\rho_{i-1}}(\nu_{i-1} \sin(\kappa_{i-1}) + \nu_i \sin(\theta_i)) \\
\dot{\rho}_{iB} &= -\nu_i \cos(\kappa_{iB}) \\
\dot{\kappa}_{iB} &= -\nu_i u_i + \frac{\nu_i}{\rho_{ib}} \sin(\kappa_{iB}),
\end{aligned} \tag{2.11}$$

subject to the cyclic closure constraint derived from (2.7),

$$R\Big(\sum_{i=1}^{n}(\pi + \kappa_i - \theta_{i+1})\Big) = \mathbb{I}_2, \tag{2.12}$$

and the beacon closure constraint derived from (2.8),

$$\rho_i \mathbb{I}_2 = \rho_{iB} R(\kappa_{iB} - \kappa_i) + \rho_{i+1,B} R(\kappa_{i+1,B} - \theta_{i+1}), \forall i \in I. \tag{2.13}$$

The value ρ_i denotes the distance between agent i and agent $i+1$, κ_i is the angle between the agent i's current heading and direction to agent $i+1$, and θ_i denotes the angle between the heading of agent i and direction to agent $i-1$. With the beacon introduced, ρ_{iB} is the distance between agent i and the beacon, and κ_{iB} is the angle between the heading of agent i and direction to the beacon.

It can be shown that if (2.12) and (2.13) are satisfied initially, they are satisfied for all further time under the dynamics on (2.11), regardless of the controls used.

2.1.2 CBL Control Law

In this section we recall the beacon-referenced, constant bearing (CBB) pursuit law from [6],

$$u_i = u^i_{CB+B} = (1-\lambda)u^i_{CB} + \lambda u^i_B, \quad \lambda \in [0,1] \tag{2.14}$$

and u^i_{CB} is the original CB pursuit law from [15]. The CB pursuit law references agent $i+1$, and u^i_B is based on the deviation from a desired bearing angle to the beacon, and the λ parameter is the weight of the beacon in the CBB control law. Now by the attention graph in [6], in terms of scalar shape variables, each agent i uses,

$$\begin{aligned} u^i_{CB} &= \mu_i \sin(\kappa_i - \alpha_i) + \frac{1}{\rho_i}(\sin(\kappa_i) + \sin(\theta_{i+1})) \\ u^i_B &= \mu_B \sin(\kappa_{iB} - \alpha_{iB}). \end{aligned} \tag{2.15}$$

Parameter $\mu_i > 0$ is simply a gain value, which as was shown in [6], can be used to adjust the radius of encirclement of the beacon. The α_i parameters describe the desired relative angle of the neighboring agent, and α_B is the agent's desired relative angle of the beacon.

For analysis, we simplify the controls and dynamics obeyed following simplifying assumptions:

1. The speeds of all agents are equal and constant, thus without loss of generality $\nu_i = 1 \quad \forall i \in I.$

2. Controller gains are equal, $\mu_i = \mu_{i+1} = \mu, \quad \forall i \in I.$

Then the dynamics simplify to,

$$\begin{aligned}
\dot{\rho}_i &= -\cos(\kappa_i) - \cos(\theta_{i+1}) \\
\dot{\kappa}_i &= -u_i + \frac{1}{\rho_i}(\sin(\kappa_i) + \sin(\theta_{i+1})) \\
\dot{\theta}_i &= -u_i + \frac{1}{\rho_{i-1}}(\sin(\kappa_{i-1}) + \sin(\theta_i)) \\
\dot{\rho}_{iB} &= -\cos(\kappa_{iB}) \\
\dot{\kappa}_{iB} &= -u_i + \frac{1}{\rho_{iB}}\sin(\kappa_{iB}),
\end{aligned} \tag{2.16}$$

for all $i \in I$. The point of departure for this thesis from [6] is that only agent 1, the leader, is aware of the beacon. Then the CBB control law is modified into CBL laws: the leader agent employs the CBB pursuit law, but the rest adopt the plain

CB law, leading to,

$$u_i = \begin{cases} \lambda\mu\sin(\kappa_{iB} - \alpha_{iB}) + (1-\lambda)\Big(\mu\sin(\kappa_i - \alpha_i) + \dfrac{1}{\rho_i}\big(\sin(\kappa_i) + \sin(\theta_{i+1})\big)\Big), \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } i = 1 \\ \mu\sin(\kappa_i - \alpha_i) + \dfrac{1}{\rho_i}(\sin(\kappa_i) + \sin(\theta_{i+1})), \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } i \neq 1, \end{cases} \tag{2.17}$$

with the dynamics subject to the cyclic closure constraint (2.12) and beacon closure constraint (2.13).

2.1.3 Relative Equilibria

Now we analyze the closed loop shape dynamics (2.16) to determine existence conditions for equilibria, and characterize system behavior at equilibria[1]. While in (2.14), λ ranges over the closed interval $[0, 1]$, we do not allow for λ to take the value 1 or 0 in order to keep the system performing leader based cyclic pursuit with a beacon. Our approach takes inspiration from [6]. From the form of $\dot{\rho}_i$ and $\dot{\rho}_{iB}$ in (2.16), we obtain the condition,

$$\theta_{i+1} = \pi \pm \kappa_i \text{ and } \kappa_{iB} = \pm\frac{\pi}{2}, \quad i \in I. \tag{2.18}$$

Now from setting $\dot{\kappa_i}$, $\dot{\theta}_i$ equal to zero, we see

$$\frac{1}{\rho_i}(\sin(\kappa_i) + \sin(\theta_{i+1})) = \frac{1}{\rho_{i-1}}(\sin(\kappa_{i-1}) + \sin(\theta_i)), \tag{2.19}$$

[1]Note that equilibria for the shape dynamics (2.16) are relative equilibria for the full dynamics (2.3).

for $i \in I$. Now define,

$$\delta_i \triangleq \frac{1}{\rho_i}(\sin(\kappa_i) + \sin(\theta_{i+1})), \tag{2.20}$$

where from (2.19) we see the relationship,

$$\delta_i = \delta_{i+1}, \quad i \in I. \tag{2.21}$$

Geometrically, δ_i is a quantity that relates the angle and distance between agents at equilibrium. Further we define,

$$\psi \triangleq \delta_1 = \delta_2 = ... = \delta_n. \tag{2.22}$$

Looking at the controls from $\dot{\kappa}_i$, (2.16), and (2.21) we see that at equilibrium,

$$u_i = u_{i+1}, \quad i \in I, \tag{2.23}$$

and more specifically,

$$u_1 = u_i, \quad i = 2, 3, ..., n. \tag{2.24}$$

This is to say that the controls for all agents are the same at equilibrium, despite the leader having a different control law from the followers. From the $\dot{\kappa}_i$ dynamics, we also see the relationship,

$$u_i = \delta_i, \quad i \in I. \tag{2.25}$$

Then from (2.22) and (2.25),

$$\begin{aligned}
\psi &= u_1 \\
\psi &= \lambda\mu\sin(\kappa_{1B} - \alpha_B) + (1-\lambda)\Big(\mu\sin(\kappa_1 - \alpha_1) + \frac{1}{\rho_1}\big(\sin(\kappa_1) + \sin(\theta_2)\big)\Big) \\
\psi &= \lambda\mu\sin(\kappa_{1B} - \alpha_B) + (1-\lambda)\big(\mu\sin(\kappa_1 - \alpha_1) + \psi\big) \\
\lambda\psi &= \lambda\mu\sin(\kappa_{1B} - \alpha_B) + (1-\lambda)\big(\mu\sin(\kappa_1 - \alpha_1)\big).
\end{aligned} \tag{2.26}$$

Now dividing through by lambda,

$$\psi = \mu \sin(\kappa_{1B} - \alpha_B) + (\frac{1}{\lambda} - 1)(\mu \sin(\kappa_1 - \alpha_1)). \tag{2.27}$$

Setting $\dot{\kappa}_{iB}$ equal to zero and using (2.24) ,

$$\frac{1}{\rho_{iB}} \sin(\kappa_{iB}) = \frac{1}{\rho_{i+1,B}} \sin(\kappa_{i+1,B}), \quad i \in I, \tag{2.28}$$

and from (2.26),

$$\psi = \frac{1}{\rho_{iB}} \sin(\kappa_{iB}), \quad i \in I. \tag{2.29}$$

Now suppose that in (2.18) we have $\kappa_i = \pi + \theta_{i+1}$. Substitution into (2.29) yields,

$$\frac{1}{\rho_{iB}} \sin(\kappa_{iB}) = \psi = \frac{1}{\rho_i}(\sin(\pi + \theta_{i+1}) + \sin(\theta_{i+1})) = 0, \tag{2.30}$$

for all $i \in I$. This results in a contradiction in (2.18) because $\rho_{iB} > 0$ and $\sin(\kappa_{iB}) \neq 0$. Therefore,

$$\kappa_i = \pi - \theta_{i+1}, \quad i \in I. \tag{2.31}$$

Then using (2.20), (2.27), and (2.31),

$$\psi = \frac{2}{\rho_i} \sin(\kappa_i), \quad i \in I. \tag{2.32}$$

Solving for ρ_i using (2.20),

$$\rho_i = \frac{2 \sin(\kappa_i)}{\psi}, \quad i \in I. \tag{2.33}$$

Solving for ρ_{iB} using (2.29),

$$\rho_{iB} = \frac{\sin(\kappa_{iB})}{\psi}, \quad i \in I. \tag{2.34}$$

Now looking at the $\dot{\kappa}_i$ dynamics for the followers, $i \in \{2, ..., n\}$, and using (2.17) (2.25),

$$\mu \sin(\kappa_i - \alpha_i) = 0, \quad i = 2, 3, ..., n \tag{2.35}$$

which implies for follower agents ($i \in I$ s.t. $i \geq 2$),

$$\kappa_i = \alpha_i \text{ or } \kappa_i = \alpha_i + m\pi, \quad m \in \mathbb{N} \tag{2.36}$$

As demonstrated in [5], under the CB pursuit law the κ_i dynamics are self-contained for $i \in \{2, ...n\}$, and the trajectories asymptotically approach $\kappa_i = \alpha_i$ for all initial conditions that do not start at $\kappa_i = \alpha_i + \pi$. Thus we will restrict our analysis to the invariant and attractive submanifold M_{CB} (see [5]) corresponding to all follower agents attaining the CB pursuit strategy (i.e. $\kappa_i = \alpha_i$, $i \in I$ s.t. $i \geq 2$).

Now evaluating solutions for κ_1 we use the cyclic closure constraint (2.12) with (2.31),

$$\begin{aligned}
\mathbb{I}_2 = & R\Big(\sum_{i=1}^{n}(\pi + \kappa_i - \theta_{i+1})\Big), \quad i \in I \\
= & R\Big((\pi + \kappa_1 - \theta_2) + (\pi + \kappa_2 - \theta_3) + (\pi + \kappa_3 - \theta_4) + ... \\
& + (\pi + \kappa_n - \theta_1)\Big) \\
= & R\Big((\pi + \kappa_1 - \pi + \kappa_1) + (\pi + \kappa_2 - \pi + \kappa_2) + (\pi + \kappa_3 - \pi + \kappa_3) + ... \\
& + (\pi + \kappa_n - \pi + \kappa_n)\Big) \\
= & R\Big(2(\kappa_1 + \sum_{i=2}^{n}\kappa_i)\Big).
\end{aligned} \tag{2.37}$$

Decomposing the R matrix and substituting with (2.36),

$$\begin{aligned} \cos(2(\kappa_1 + \sum_{i=2}^{n} \alpha_i)) &= 1 \\ \sin(2(\kappa_1 + \sum_{i=2}^{n} \alpha_i)) &= 0 \end{aligned} \tag{2.38}$$

From which we see that,

$$2(\kappa_1 + \sum_{i=2}^{n} \alpha_i) = 0,\ 2m\pi \quad \forall m \in \mathbb{Z}. \tag{2.39}$$

Then,

$$\kappa_1 = m\pi - \sum_{i=2}^{n} \alpha_i, \quad m \in \mathbb{Z}. \tag{2.40}$$

This reduces to two possible cases: either $m = 0$ or $m = 1$.

2.1.4 Existence Conditions of Relative Equilibria

Recall that we require ρ_i and ρ_{iB} to be strictly positive. From our equilibrium values ρ_i and ρ_{iB} given by (2.33) and (2.34), we see that for $i \geq 2$ if a single α_i is chosen such that $\sin(\alpha_i) > 0$, then by (2.33) a constraint is placed on ψ such that $\psi > 0$. Then by (2.34) this requires all κ_{iB}'s to be the same sign. And by a similar argument for the case of a single $\sin(\alpha_i) < 0$ for $i \geq 2$, we form the existence condition,

$$\sin(\kappa_{iB}) \sin(\kappa_i) > 0,\ \forall i \in I. \tag{2.41}$$

Claim 2.1.1. *All equilbria are circling equilibria.*

Proof. From the ρ_i and ρ_{iB} equilibrium values and positivity constraints, all $\sin(\alpha_i)$ must be the same sign for $i \geq 2$, else it would contradict the sign of ψ. This implies that all $\sin(\kappa_{iB})$ terms are the same sign, and by (2.18) all κ_{iB}'s must be the same value. Hence all agents are equidistant from the beacon such that $\rho_{iB} = \rho_{jB}$, $\forall i, j \in I$. All agents are therefore equidistant from the beacon and have unit speed; thus all equilibria are circling equilibria. □

We now state the main result for existence of circling equilibria, based on the above analysis.

Proposition 2.1.2. *Consider a CBL system evolving on the submanifold M_{CB}, consisting of $n-1$ follower agents and one leader agent, whose shape dynamics is governed by (2.16) and parameterized by μ, λ, and the CB parameters $\{\alpha_1, \alpha_2, ..., \alpha_n\}$ and beacon angle parameter α_B. The following statements are true:*

1. *All equilibria are circling equilibria.*

2. *A circling equilibrium exists if and only if either of the following mutually exclusive cases are satisfied,*

   (a)

$$\begin{aligned} &\sin(\alpha_i) > 0, \ \forall i \in I \ \textit{s.t.} \ i \geq 2, \\ &\lambda \cos(\alpha_B) + (1-\lambda)\sin(m\pi - \sum_{i=1}^{n} \alpha_i) > 0 \\ &\textit{where } m \in \mathbb{Z} \textit{ satisfies } \sin(m\pi - \sum_{i=2}^{n} \alpha_i) > 0 \end{aligned} \tag{2.42}$$

(b)

$$\begin{aligned} &\sin(\alpha_i) < 0,\ \forall i \in I \ \textit{s.t.}\ i \geq 2, \\ &\lambda \cos(\alpha_B) - (1-\lambda)\sin(m\pi - \sum_{i=1}^{n} \alpha_i) > 0 \\ &\textit{where } m \in \mathbb{Z} \textit{ satisfies } \sin(m\pi - \sum_{i=2}^{n} \alpha_i) < 0 \end{aligned} \tag{2.43}$$

3. *At equilibrium if* (2.42) *holds,*

$$\begin{aligned} \kappa_{iB} &= \frac{\pi}{2}, \quad \forall i \in I \\ \psi &= \mu \cos(\alpha_B) + (\frac{1}{\lambda} - 1)\big(\mu \sin(m\pi - \sum_{i=1}^{n} \alpha_i)\big). \end{aligned} \tag{2.44}$$

Alternatively if at equilibrium (2.43) *holds,*

$$\begin{aligned} \kappa_{iB} &= -\frac{\pi}{2}, \quad \forall i \in I \\ \psi &= -\mu \cos(\alpha_B) + (\frac{1}{\lambda} - 1)\big(\mu \sin(m\pi - \sum_{i=1}^{n} \alpha_i)\big). \end{aligned} \tag{2.45}$$

Then the equilibrium values satisfy,

$$\begin{aligned} \kappa_1 &= m\pi - \sum_{i=2}^{n} \alpha_i \\ \kappa_i &= \alpha_i, \quad i \geq 2 \\ \rho_1 &= \frac{2\sin(m\pi - \sum_{i=2}^{n} \alpha_i)}{\psi} \\ \rho_i &= \frac{2\sin(\alpha_i)}{\psi}, \quad i \geq 2 \\ \rho_{iB} &= \frac{1}{\psi}, \quad \forall i \in I \end{aligned} \tag{2.46}$$

Proof. The only part of the proposition that does not directly follow from the preceding discussion is the requirement that,

$$\lambda \cos(\alpha_B) + (1-\lambda)\sin(\kappa_B)\sin(m\pi - \sum_{i=1}^{n} \alpha_i) > 0 \tag{2.47}$$

in (2.42) and the analogous constraint in (2.43). We establish this constraint by starting from the equilibrium value for ρ_{iB} (2.34), such that for all $i \in I$,

$$\begin{aligned}
\rho_{iB} =& \frac{\sin(\kappa_{iB})}{\psi} \\
=& \frac{\sin(\kappa_{iB})}{\mu \sin(\kappa_{iB} - \alpha_B) + (\frac{1}{\lambda} - 1)(\mu \sin(\kappa_1 - \alpha_1))} \\
=& \frac{\lambda}{\mu(\lambda \cos(\alpha_B) + \frac{(1-\lambda)\sin(\kappa_1 - \alpha_1)}{\sin(\kappa_{iB})})} \\
=& \frac{\lambda}{\mu(\lambda \cos(\alpha_B) + (1-\lambda)\sin(\kappa_{iB})\sin(\kappa_1 - \alpha_1))}
\end{aligned}$$

The last line is equivalent because $\sin(\kappa_{iB})$ is required to be ± 1 at equilibrium. Since λ, μ, and ρ_{iB} are all required to be positive, the constraint in (2.47) follows. □

Remark 2.1.3. *If (2.42) is satisfied from Proposition 2.1.2, and if equilibrium is achieved, the agents will circle the beacon counter-clockwise. If (2.43) is satisfied and if equilibrium is achieved, the agents will circle the beacon clockwise.*

Remark 2.1.4. *It can be easily shown that these equilibrium values satisfy the cyclic closure constraint (2.12) and beacon closure constraint (2.13).*

2.1.5 Stability Analysis for Two Agents

Exploring stability for a problem like this is complicated, with results that are sometimes difficult to interpret. In this section we do our best to discuss the stability of the system. Lyapunov analysis was attempted, but is so far inconclusive. We therefore proceed by linearization.

The two agent analysis reveals some simplification in dynamics. To start, the measure of distance between agents, ρ_i, is the same for both agents, so notation simplifies to using ρ to denote inter agent distance. Due to the cyclic closure constraint, (2.12) and the relationship between κ_i and θ_{i+1}(2.18),

$$\begin{aligned} &\cos(\kappa_1) + \cos(\theta_2) = \cos(\kappa_2) + \cos(\theta_1) \\ &\sin(\kappa_1 - \theta_2 + \kappa_2 - \theta_1) = 0 \\ &\cos(\kappa_1 - \theta_2 + \kappa_2 - \theta_1) = 1 \end{aligned} \tag{2.48}$$

Suppose $\theta_2 = \kappa_1$ and $\theta_1 = \kappa_2$, then $\cos(\kappa_1) = \cos(\kappa_2)$. This is a contradiction because κ_1 need not equal κ_2. Therefore,

$$\theta_2 = \kappa_2 \text{ and } \theta_1 = \kappa_1. \tag{2.49}$$

Now define,

$$\delta \triangleq \frac{1}{\rho}(\sin(\kappa_1) + \sin(\kappa_2)). \tag{2.50}$$

And the two agent dynamics follow (not including beacon closure constraint),

$$\begin{aligned} \dot{\kappa}_1 &= -u_1 + \delta & \dot{\kappa}_{1B} &= -u_1 + \frac{1}{\rho_{1B}}\sin(\kappa_{1B}) \\ \dot{\kappa}_2 &= -u_2 + \delta & \dot{\kappa}_{2B} &= -u_2 + \frac{1}{\rho_{2B}}\sin(\kappa_{2B}) \\ \dot{\rho} &= -(\cos(\kappa_1) + \cos(\kappa_2)) & \dot{\rho}_{1B} &= -\cos(\kappa_{1B}) \\ & & \dot{\rho}_{2B} &= -\cos(\kappa_{2B}). \end{aligned} \tag{2.51}$$

With the controls,

$$u_1 = (1-\lambda)\big(\mu\sin(\kappa_1 - \alpha_1) + \delta\big) + \lambda\mu\sin(\kappa_{1B} - \alpha_B) \tag{2.52}$$

$$u_2 = \mu\sin(\kappa_2 - \alpha_2) + \delta \tag{2.53}$$

The two agent dynamics (2.51) take the form,

$$\dot{\zeta} = f(\zeta), \quad \zeta = \{\kappa_1, \kappa_2, \rho, \kappa_{1B}, \kappa_{2B}, \rho_{LB}, \rho_{2B}\} \tag{2.54}$$

Linearizing about the equilibrium values from 2.1.2 yields dynamics of the form,

$$\dot{\tilde{\zeta}} = A\tilde{\zeta}, \quad A \in \mathbb{R}^{7\text{x}7} \tag{2.55}$$

Now we show the beacon closure constraint (2.13) yields exactly one pair of imaginary eigenvalues, regardless of the control used. From (2.13) we see,

$$\begin{aligned} 0_{2\text{x}2} = \rho \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \rho_{1B} \begin{bmatrix} \cos(\kappa_{1B} - \kappa_1) & -\sin(\kappa_{1B} - \kappa_L) \\ \sin(\kappa_{1B} - \kappa_1) & \cos(\kappa_{1B} - \kappa_1) \end{bmatrix} \\ - \rho_{2B} \begin{bmatrix} \cos(\kappa_{2B} - \kappa_2) & -\sin(\kappa_{2B} - \kappa_2) \\ \sin(\kappa_{2B} - \kappa_2) & \cos(\kappa_{2B} - \kappa_2) \end{bmatrix} \end{aligned} \tag{2.56}$$

which yields the following constraints,

$$g_1(\zeta) \triangleq \rho - \rho_{1B}\cos(\kappa_{1B} - \kappa_1) - \rho_{2B}\cos(\kappa_{2B} - \kappa_2) \equiv 0 \tag{2.57}$$

$$g_2(\zeta) \triangleq \rho_{1B}\sin(\kappa_{1B} - \kappa_1) + \rho_{2B}\sin(\kappa_{2B} - \kappa_2) \equiv 0. \tag{2.58}$$

Now define the pursuit manifold, the space where the constraints (2.57) and (2.58) are satisfied,

$$M \triangleq \{\zeta \in \mathbb{R}^7 \text{ s.t. } g_1(\zeta) = 0 \text{ and } g_2(\zeta) = 0\}. \tag{2.59}$$

Claim 2.1.5. *The pursuit manifold M is an invariant manifold under the two agent dynamics of* (2.51).

Proof.

$$
\begin{aligned}
\dot{g}_1(\zeta) =& \frac{\partial g_1(\zeta)}{\partial \zeta} f(\zeta) \\
=& \cos(\kappa_{2B})\cos(\kappa_2 - \kappa_{2B}) - \cos(\kappa_1) - \cos(\kappa_2) + \cos(\kappa_{1B})\cos(\kappa_1 - \kappa_{1B}) \\
& - \rho_{1B}\sin(\kappa_1 - \kappa_{1B})\Big(\frac{\sin(\kappa_{1B})}{\rho_{1B}} + (\lambda - 1)(\delta - \mu\sin(\alpha_1 - \kappa_1)) + \lambda\mu\sin(\alpha_B - \kappa_{1B})\Big) \\
& - \rho_{2B}\sin(\kappa_2 - \kappa_{2B})\Big(\frac{\sin(\kappa_{2B})}{\rho_{2B}}) - \delta + \mu\sin(\alpha_2 - \kappa_2)\Big) \\
& + \rho_{1B}\sin(\kappa_1 - \kappa_{1B})\Big(\delta + (\lambda - 1)(\delta - \mu\sin(\alpha_1 - \kappa_1)) \\
& + \mu\rho_{2B}\sin(\kappa_2 - \kappa_{2B})\sin(\alpha_2 - \kappa_2)\Big) \\
=& \cos(\kappa_{2B})\cos(\kappa_2 - \kappa_{2B}) - \cos(\kappa_1) - \cos(\kappa_2) + \cos(\kappa_{1B})\cos(\kappa_1 - \kappa_{1B}) \\
& \rho_{2B}\sin(\kappa_2 - \kappa_{2B})(u_2 - \frac{\sin(\kappa_{2B})}{\rho_{2B}}) + \rho_{1B}\sin(\kappa_1 - \kappa_{1B})(u_1 - \frac{\sin(\kappa_{1B})}{\rho_{1B}}) \\
& \rho_{2B}\sin(\kappa_2 - \kappa_{2B})(u_2 - \delta) - \rho_{1B}\sin(\kappa_1 - \kappa_{1B})(u_1 - \delta) \\
=& \cos(\kappa_{2B})\cos(\kappa_2 - \kappa_{2B}) - \cos(\kappa_1) - \cos(\kappa_2) + \cos(\kappa_{1B})\cos(\kappa_1 - \kappa_{1B}) \\
& \rho_{2B}\sin(\kappa_2 - \kappa_{2B})(\delta - \frac{\sin(\kappa_{2B})}{\rho_{2B}}) + \rho_{1B}\sin(\kappa_1 - \kappa_{1B})(\delta - \frac{\sin(\kappa_{1B})}{\rho_{1B}}) \\
=& -\delta g_2(\zeta) - \sin(\kappa_1 - \kappa_{1B}) - \sin(\kappa_2 - \kappa_{2B})\sin(\kappa_{2B}) \\
& + \cos(\kappa_{1B})\cos(\kappa_1 - \kappa_{1B}) + \cos(\kappa_{2B})\cos(\kappa_2 - \kappa_{2B}) \\
& - \cos(\kappa_2) - \cos(\kappa_1) \\
=& -\delta g_2(\zeta)
\end{aligned}
\tag{2.60}
$$

Likewise it can be shown that,

$$\dot{g}_2(\zeta) = \delta g_1(\zeta) \tag{2.61}$$

Observe that M is invariant because, on M, $\dot{g}_1(\cdot) = \dot{g}_2(\cdot) \equiv 0$. Note from the above

calculations that u_1 and u_2 cancel out in both calculations. Thus M is an invariant manifold. $\square$

From [6], there exists a change of basis for the linearized dynamics such that the new dynamics take the form,

$$\dot{\tilde{\zeta}} = A\tilde{\zeta} = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0_{2x5} & A_{22} \end{array}\right] \tilde{\zeta}, \text{ where } A_{22} = \begin{bmatrix} 0 & -\delta_{eq} \\ \delta_{eq} & 0 \end{bmatrix}. \tag{2.62}$$

and using 2.1.2 to substitute for equilibrium values, taking advantage that at equilibrium $\rho_1 = \rho_2$, which means $\sin(\kappa_{1,eq}) = \sin(\alpha_2)$,

$$\delta_{eq} = \frac{\sin(\kappa_2) + \sin(\kappa_1)}{\rho} \mid_{\zeta_{eq}} \tag{2.63}$$

$$= \frac{2\sin(\alpha_2)}{\rho_{eq}} \tag{2.64}$$

$$= \frac{1}{\rho_{B,eq}} \tag{2.65}$$

By it's form, A_{22} represents a pair of imaginary eigen values on the jw axis s.t.

$$\lambda_{1,2} = \pm j\delta_{eq}. \tag{2.66}$$

It remains to obtain A_{11} and A_{12}. To this end, we make the simplification that

$\alpha_1 = \alpha_2 = \alpha$. Linearizing (2.51) about $\zeta = \zeta_{eq}$ yields,

$$A_{11} = \begin{bmatrix} \gamma_1 + \frac{(\lambda-1)\cos(\kappa_{1,eq})}{(2\rho_{B,eq}\sin(\alpha))} & -\frac{\delta_{eq}\lambda}{2\sin(\alpha)\rho_{eq}} & -\lambda\gamma_B & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ \sin(\kappa_{1,eq}) & \sin(\alpha) & 0 & 0 & 0 \\ (1-\lambda)\left(\gamma_1 + \frac{\cos(\kappa_{1,eq})}{2\rho_{B,eq}\sin(\alpha)}\right) & \frac{\eta(\lambda-1)}{2\sin(\alpha)} & \frac{(1-\lambda)\delta_{eq}}{2\sin(\alpha)\rho_{B,eq}} & -\lambda\gamma_B & \frac{-1}{\rho_{Beq}^2\sin(\kappa_{B,eq})} \\ 0 & 0 & 0 & \sin(\kappa_{B,eq}) & 0 \end{bmatrix}$$

$A_{12} = 0_{5x2}$

Where $\eta = \dfrac{\cos(\alpha)}{\rho_{eq}}$, $\gamma_1 = \cos(\alpha - \kappa_{1,eq})$, $\gamma_B = \cos(\alpha_B - \kappa_{B,eq})$.

(2.67)

It can be shown that the resultant characteristic polynomial from (2.62) is,

$$\begin{aligned} P(x) =& (\delta^2 + x^2)(x+1)(a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0) \\ s.t. \quad & a_4 = 1 \\ & a_3 = (1-\lambda)\cos(\alpha - \kappa_{1,eq}) - \frac{\lambda\cos(\kappa_{1,eq})}{2\rho_{B,eq}\sin(\alpha)} + \lambda\cos(\alpha_B - \kappa_{B,eq}) \\ & a_2 = \frac{1}{\rho_{B,eq}^2} + \frac{\lambda}{2\rho_{B,eq}^2} - \frac{\lambda\cos(\kappa_{1,eq})\cos(\alpha_B - \kappa_{B,eq})}{2\rho_{B,eq}\sin(\alpha)} \\ & a_1 = \frac{-\lambda\cos(\kappa_{1,eq})}{2\rho_{B,eq}^3\sin(\alpha)} + (1-\lambda)\frac{\cos(\alpha - \kappa_{1,eq})}{\rho_{B,eq}^2} + \frac{\lambda\cos(\alpha_B - \kappa_{1,eq})}{2\rho_{B,eq}^2} \\ & a_0 = \frac{\lambda}{2\rho_{B,eq}^4} \end{aligned} \tag{2.68}$$

Even though the eigenvalues corresponding with the manifold M are purely imaginary, because M is invariant, the Center Manifold Theorem can be used to determine local asymptotic stability of the linearized system. Therefore if the remaining eigenvalues have negative real part, we can claim local stability. By the

Routh Hurwitz stability criteria for quartic polynomials, the necessary and sufficient conditions for the quartic polynomial to have no positive roots are,

$$\begin{aligned} &a_1, a_2, a_3, a_4 > 0, \quad \text{(trivially satisfied for } a_4 \text{ and } a_0) \\ &a_3 a_2 > a_1 \\ &a_3 a_2 a_1 > a_4 + a_3^2 a_0 \end{aligned} \tag{2.69}$$

Unfortunately, even the quartic polynomial in its current form is challenging to analyze. However, a numerical analysis of stability has been performed for varying λ and $\alpha = \alpha_B$, as seen in Figure 2.3. Surprisingly, for parameter values where equilibrium exist (see Figure 2.4), high lambda values indicate instability in the linearized system. However in practice and simulations, high lambda values (of approximately 2/3) have not affected system stability. The reason this might be the case is due to the linearized stability result, whereas a nonlinear result might indicate these regions stable. Note that circling equilibrium does not exist for $\alpha = 0$, but is not visible due to the resolution of the figures.

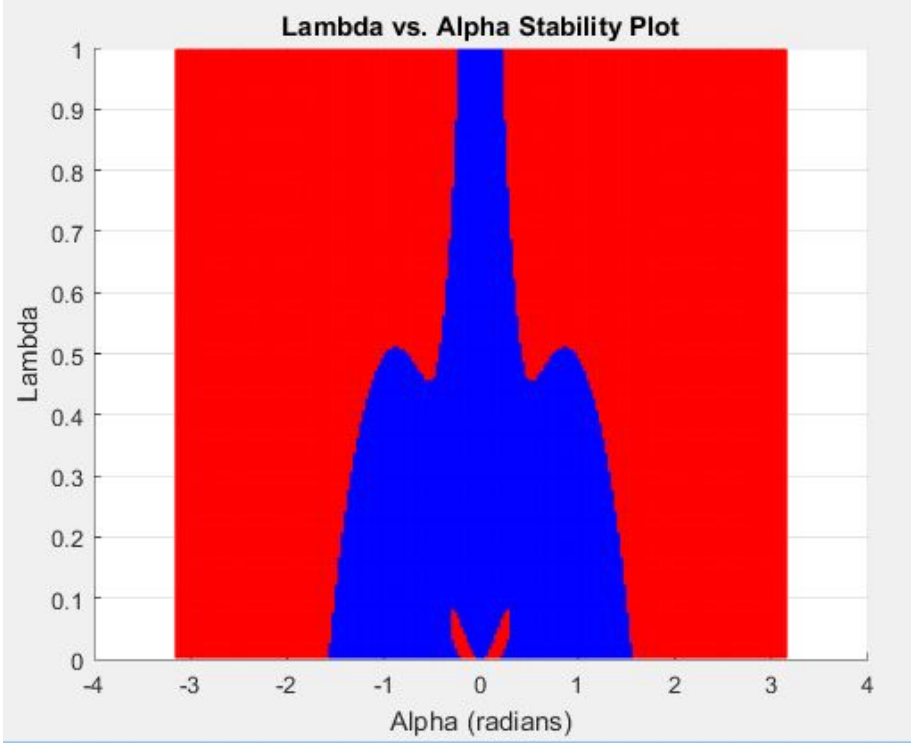

Figure 2.3: Numerical Stability Analysis of the Two Agent System, λ vs. $\alpha = \alpha_B$. Red indicates instability, blue indicates stability.

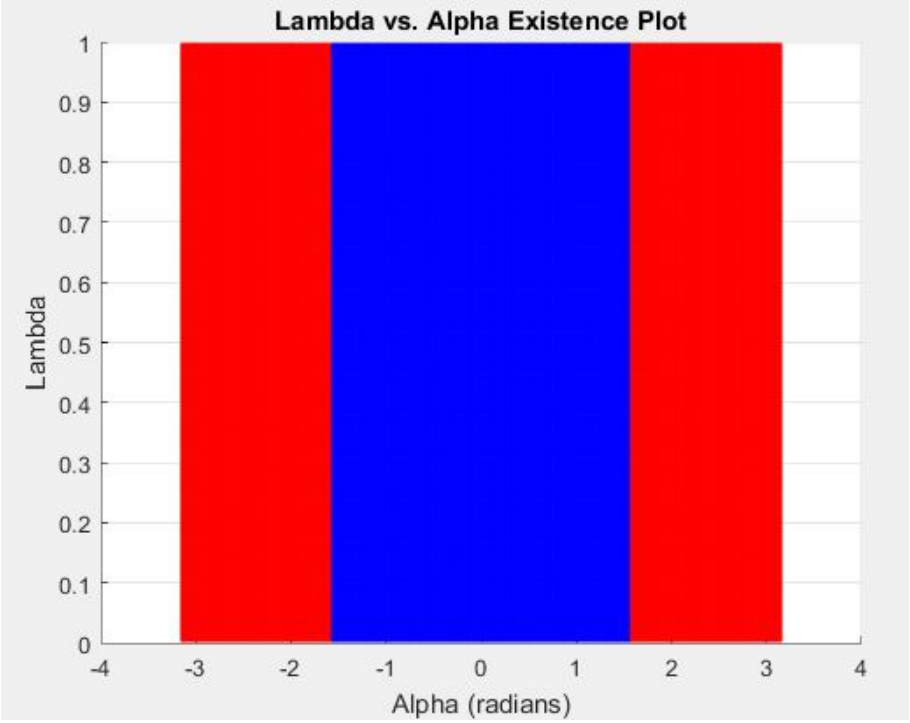

Figure 2.4: Numerical Existence Analysis of the Two Agent System, λ vs. $\alpha = \alpha_B$. Red indicates non-existence, blue indicates stability.

2.2 Experiments

Experiments were conducted using two agents, with dynamics from 2.1.2. The agents used were Pioneer 3 DX robots, a compact differential-drive robot with reversible DC motors, high-resolution motion encoders, as the experimental platform. Onboard computation is done via 32-bit Renesas SH2-7144 RISC microprocessors, including the P3-SH micro-controller with ARCOS. A software library ARIA was used that translates standardized remote motion commands into robot actions. On the back end, an Ubuntu server gathers position data from the lab's sub-millimeter accuracy Vicon positioning system, then uses open source Robot Operating System

(ROS) libraries to compute control laws, and use wireless communications (802.11g) to communicate to the agents. A more detailed discussion of implementation can be found in Chapter 4.

2.2.1 Two Agent CBL

The experiment was configured for two agents, circling counter clockwise, with the pursuit angle parameter $\alpha_1 = \alpha_2 = \alpha = \frac{\pi}{3}$ and $\alpha_B = \frac{\pi}{3}$, with a gain of $\mu = 1$, and a beacon weight $\lambda = \frac{1}{2}$. In this experiment, the beacon is displaced half way through by hand, to examine robustness of the equilibria. Based on the chosen parameters we expect the following equilibrium values,

Agent	κ	κ_B	ρ	ρ_B
Leader	2.094	1.572	1.267	0.732
Follower	1.047	1.572	1.267	0.732

Table 2.1: 2 Agent CBL, Theoretical Equilibrium Values

A plot of ρ_B is provided in Figure 2.5, and trajectories of the agents are shown in Figure 2.6. For the trajectory plot, squares denote initial positions, where circles denote final positions. In the experiment, we observe that all of our equilibrium values are reached, even after the beacon is moved half way through the experiment.

Figures 2.7, 2.8, and 2.9 are the κ, κ_B, and ρ plots respectively. In each plot we observe convergence towards the predicted equilibrium values, then once the

beacon is moved, the system again approaches equilibrium. Indeed near convergence to predicted equilibria is observed after the beacon is moved after 80 seconds, demonstrating the (empirical) stability of the system.

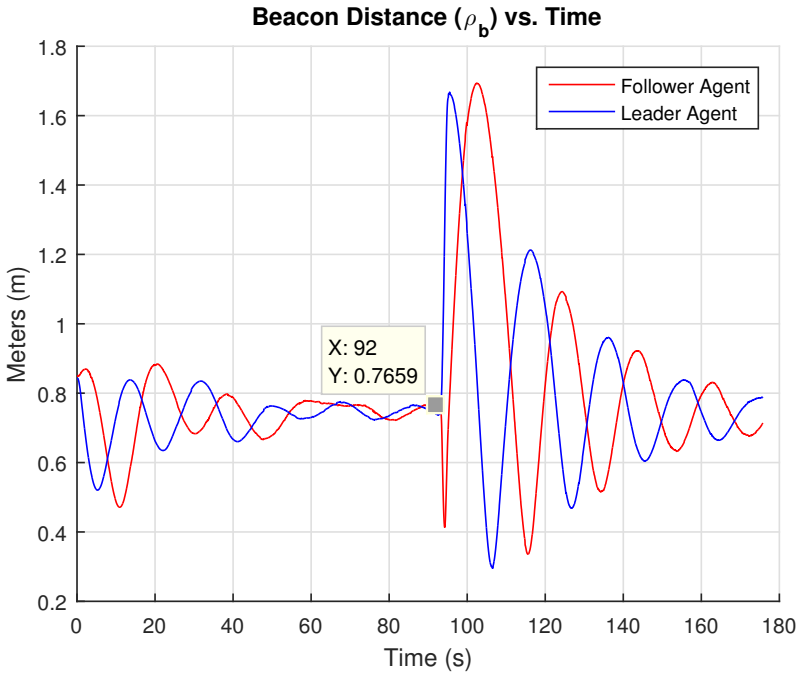

Figure 2.5: 2 Agent CBL, Plot of Agent Distance to Beacon (Beacon was moved by hand to a new location at approximately 92 seconds).

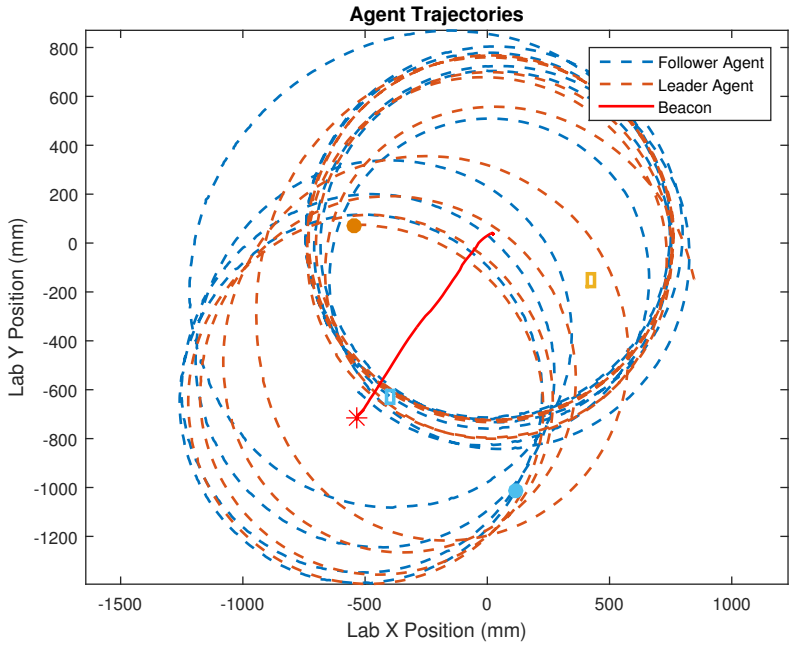

Figure 2.6: 2 Agent CBL, Plot of Agent and Beacon Trajectories; squares represent initial positions, circles represent final positions

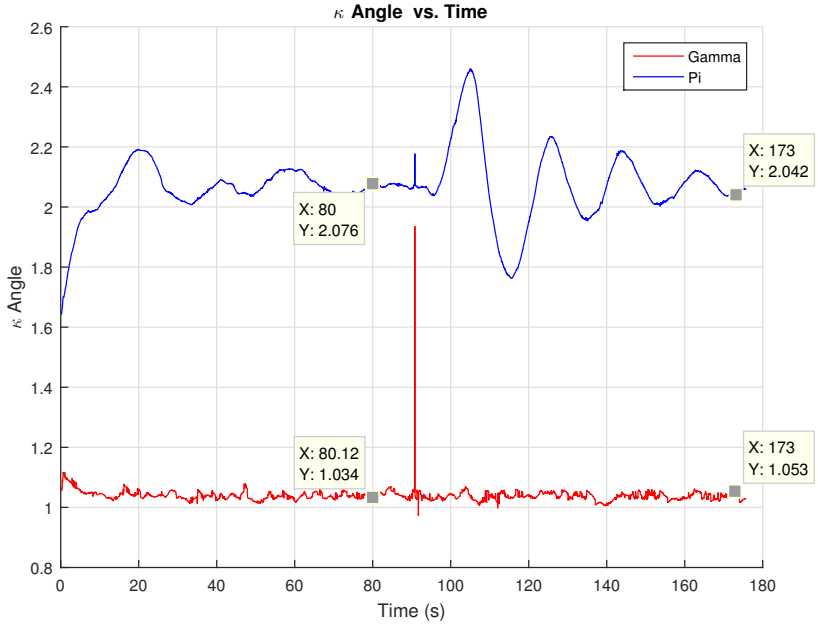

Figure 2.7: 2 Agent CBL, Plot of Kappa Angles

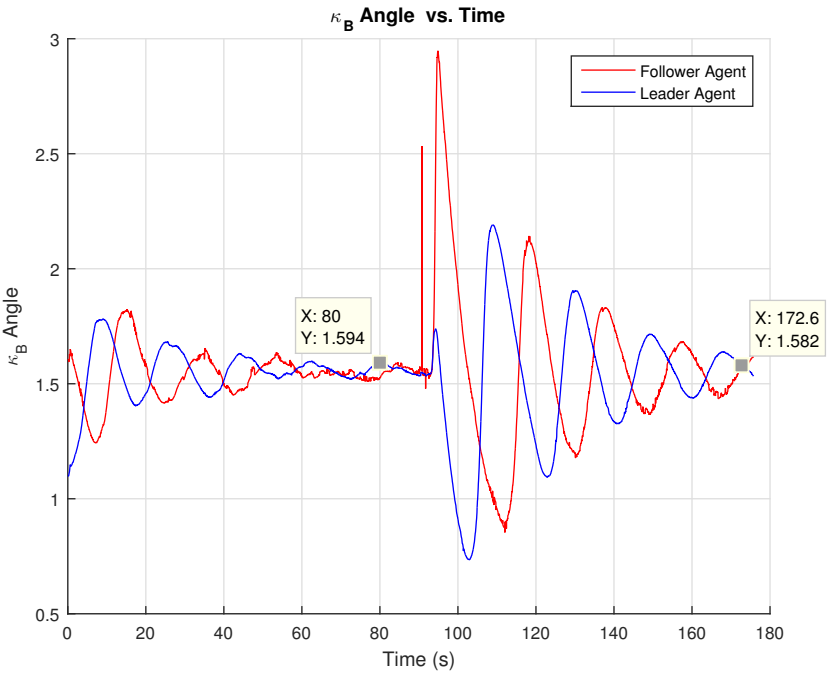

Figure 2.8: 2 Agent CBL, Plot of Kappa Beacon angles

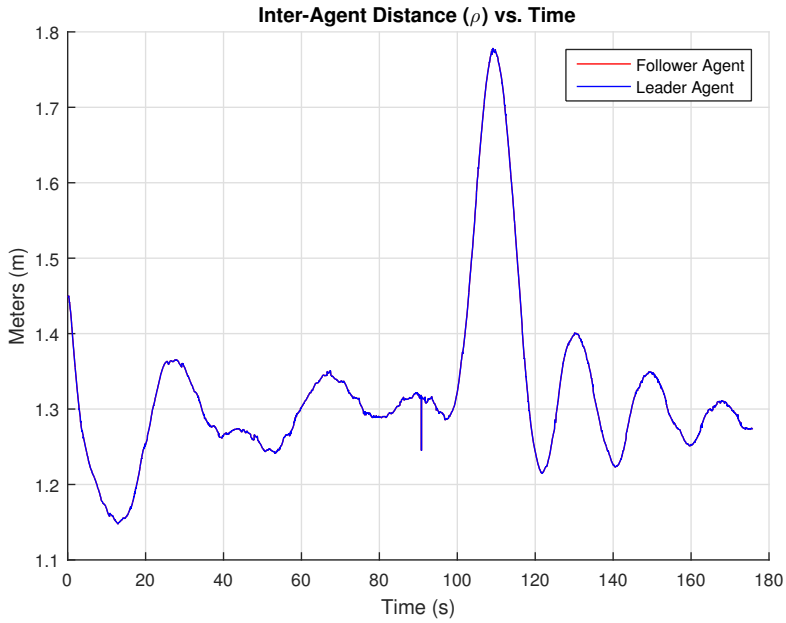

Figure 2.9: 2 Agent CBL, Plot of Rho (rho values are the same in the two agent case)

2.2.2 4 Agent CBL

Figure 2.10: 4 Agent CBL, lab setup. Agents are circling the cone, which serves as the beacon.

This experiment was configured for four agents, circling counter clockwise, with the pursuit angle parameter $\alpha_i = \alpha = \frac{\pi}{4}$ and $\alpha_B = \frac{\pi}{4}$, with a gain of $\mu = 1$, and a beacon weight $\lambda = \frac{2}{3}$. In this experiment, the beacon is moved approximately .75 meters at 53 seconds by hand, to examine robustness of the equilibria and speed of convergence. Epsilon is the leader, while Gamma, Upsilon, and Pi are followers respectively. Based on the chosen parameters, we expect equilibrium values as detailed in Table 2.2.

Note that due to parameters being chosen in a "symmetrical" way, we expect a symmetrical circling equilibrium. Figure 2.10 shows the agents circling the beacon

Agent	κ	κ_B	ρ	ρ_B
Epsilon (leader)	.7854	1.5708	2	1.4142
Gamma	.7854	1.5708	2	1.4142
Upsilon	.7854	1.5708	2	1.4142
Pi	.7854	1.5708	2	1.4142

Table 2.2: Expected Equilibrium Values for 4 Agent CBL

(cone) at the end of the experiment. In general, the agents converge to the expected equilibrium values after the beacon is moved. Figure 2.13 and Figure 2.15 depict slightly different equilibrium values than expected. The most likely explanation for this difference is that the experiment was cut short, prior to achieving the theoretical values. In both cases however, these differences are within less than 5% of the calculated equilibrium value.

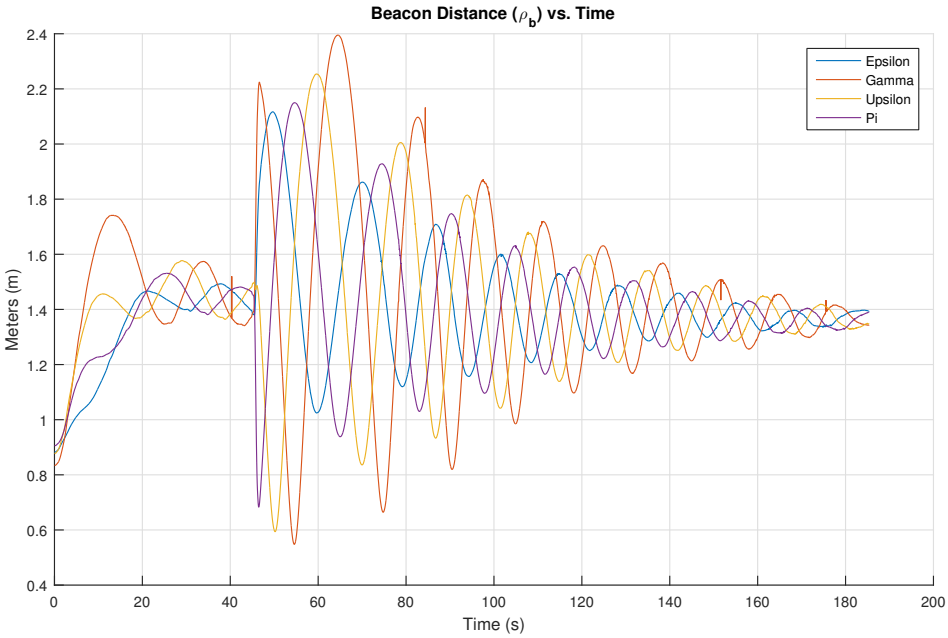

Figure 2.11: Plot of Agent Distance to Beacon. Beacon was moved by hand to a new location at approximately 53 seconds. We see that before the beacon was moved, ρ_B was approximately 1.4 for all agents, then once the beacon was moved, it once again settles to approximately 1.4.

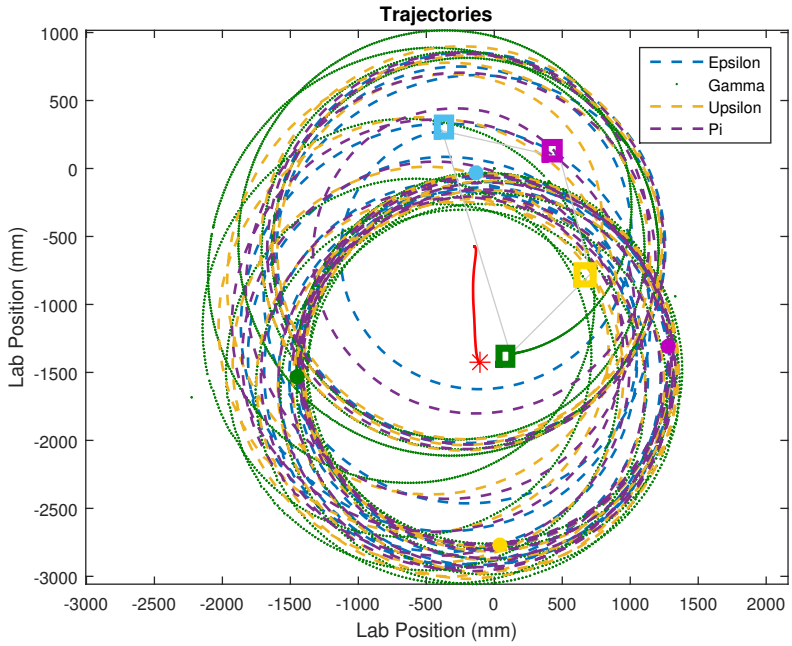

Figure 2.12: Plot of Agent and Beacon Trajectories. Beacon was moved by hand to a new location at approximately 53 seconds. Squares represent initial positions, squares represent final positions.

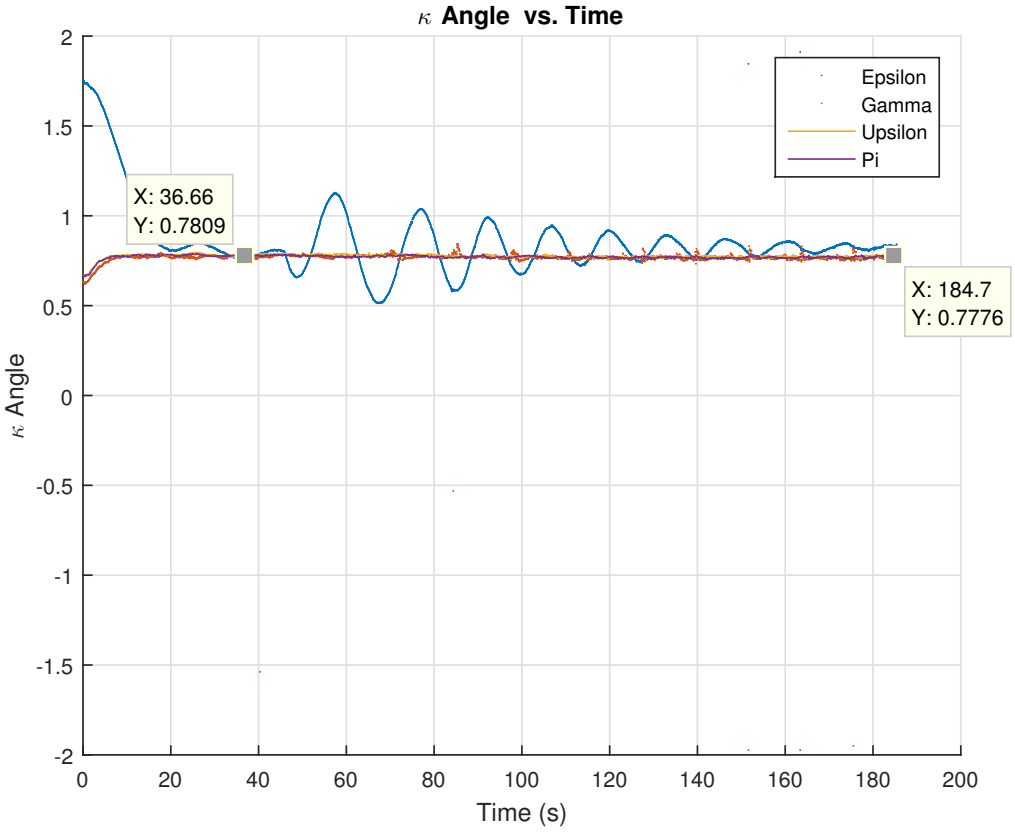

Figure 2.13: Plot of kappa angles. Prior to movement of the beacon, it appears as though all agents converge to a κ angle of approximately .78. After the beacon is moved, the followers maintain this angle, while the leader (Epsilon) converges to a $\kappa \approx .8$.

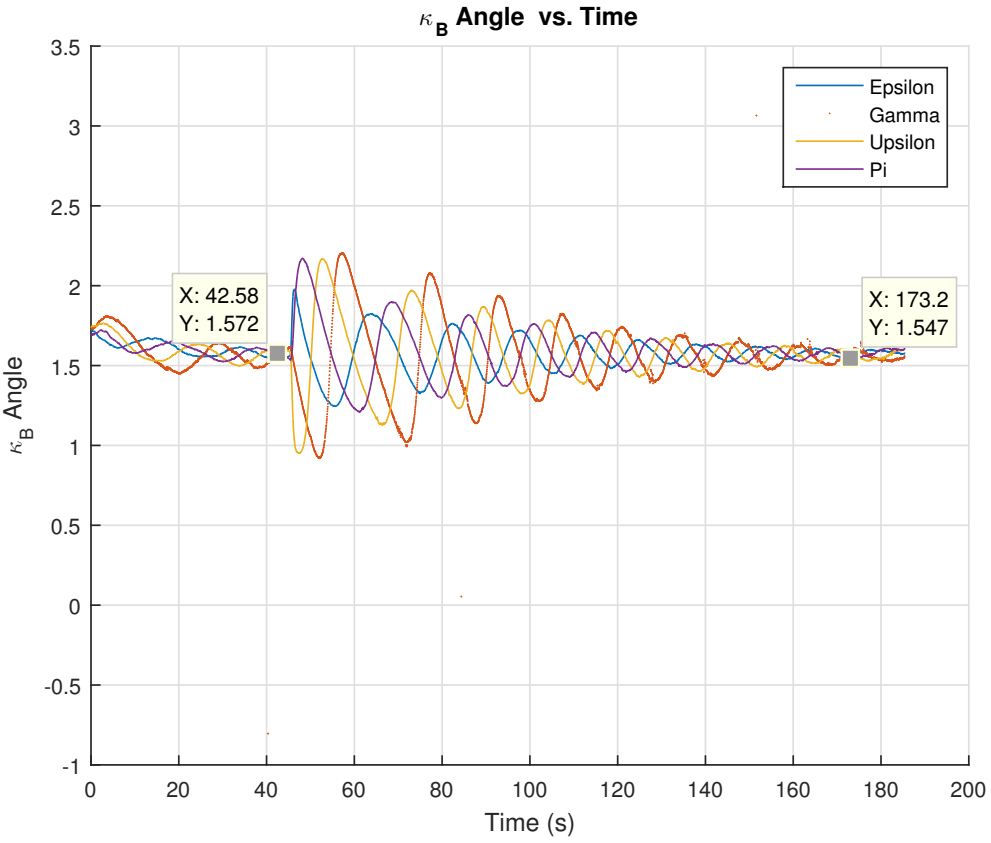

Figure 2.14: Plot of kappa beacon angles. Prior to movement of the beacon, all agents appear to converge to a κ_B of approximately 1.55. Once the beacon is moved, the agents once again settle to a κ_B of approximately 1.55.

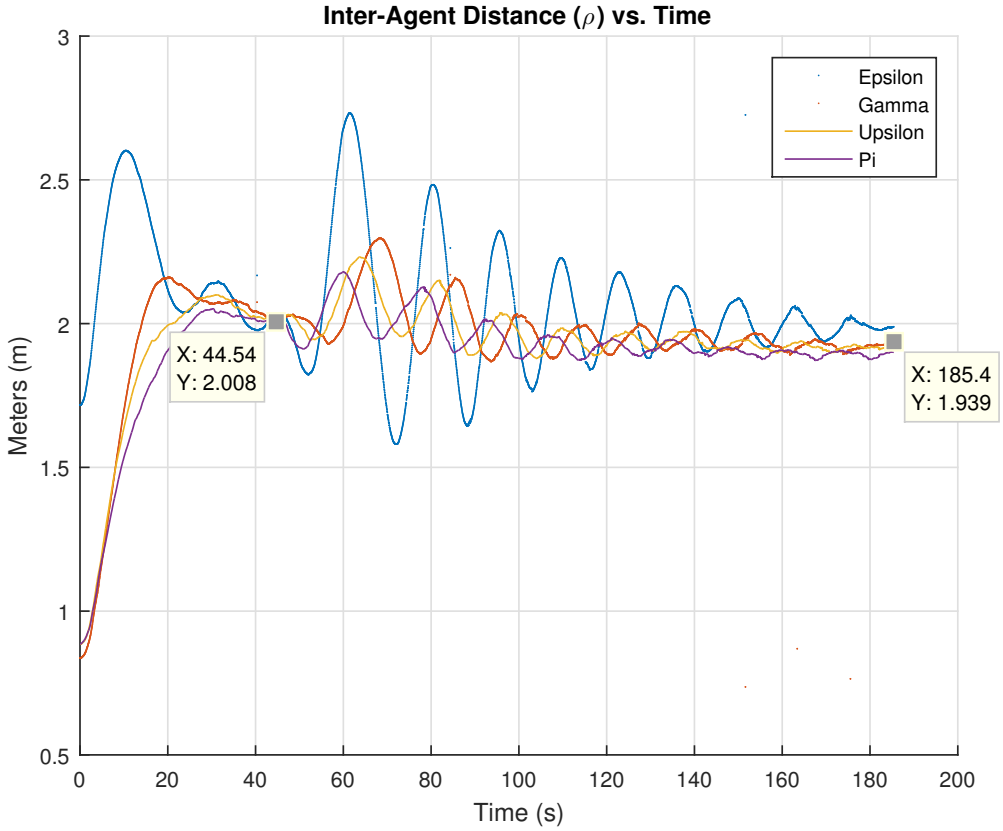

Figure 2.15: Plot of Rho, interagent distances. Prior to movement of the beacon, ρ values are approximately 2 meters. After the beacon is moved, the leader ρ converges to approximately 2 meters, while the follower ρ values converge to approximately 1.9.

2.3 Performance vs. CB Beacon

While discussing performance characteristics of the CBL system, it is prudent to compare its performance to the CB Beacon (CBB) system, where each agent has knowledge of the beacon. Due to a reduction of feedback information in the system, we would expect performance of the CBL system to be inferior in the majority of

tests. Because the followers in the CBL system are following the CB pursuit law, we expect the follower to quickly converge to the κ equilibrium value, while parameters like ρ_B and κ_B should take a longer time to settle than in the CBB system.

In this section we attempt to quantify CBL performance discussed in this paper vs. CB-Beacon, where every agent has knowledge of the beacon's location [6]. For comparison, we will use the following criteria:

- Time to Steady State: The time from the beginning of the simulation to when all agents are within 5% of their predicted equilibrium values.
- Maximum Actuation: The peak control (curvature) that was commanded during the experiment for each agent.
- Maximum Beacon Overshoot: The largest distance by which an agent exceeded the ρ_B equilibrium value.

These criteria were chosen to characterize and compare nonlinear performance because they reflect how well the system converges, at the expense of possibly reaching implementation constraints. Time to steady state is used to measure how long it takes for the system to reach predicted equilibrium values for a given parameter. Lower time to steady state is considered better. Maximum actuation reflects the threat of actuator saturation, and maximum actuation characteristics. In some applications, agent curvature could be limited by physical system characteristics. A lower maximum actuation is considered better. Finally, maximum beacon overshoot is inspired by classical PID step response performance. A large beacon overshoot in

an operational environment could result in the beacon moving outside of an agent's sensing range.

The following comparisons were carried out using MATLAB simulation, using a script to evaluate the above performance criteria. The first set of comparisons varies alpha for each simulation, while the second set varies lambda (beacon attention). A single simulation was performed for each data point. As such, the MATLAB parallel computing toolbox was use to vastly improve computing performance. Agent initial conditions were determined by a pseudo random position generator, whereby a seed number (Rand Seed) represents a consistent set of initial conditions.

Due to the peculiar nonlinear nature of these systems, simulations suggest that over a range of parameters for both α and λ, the CBL system has superior or similar performance under the above performance metrics.

2.3.1 Fixed Lambda and Initial Conditions, Varying Alpha

For this simulation, the beacon weight λ was chosen to be .5, α_B was chosen to be $\pi/2$, while α was varied from .3 to $(\pi/2 - .1)$ in .01 radian increments. This range was chosen so circling equilibria would exist for all alpha values. Results for all performance metrics can be see in Figure 2.16.

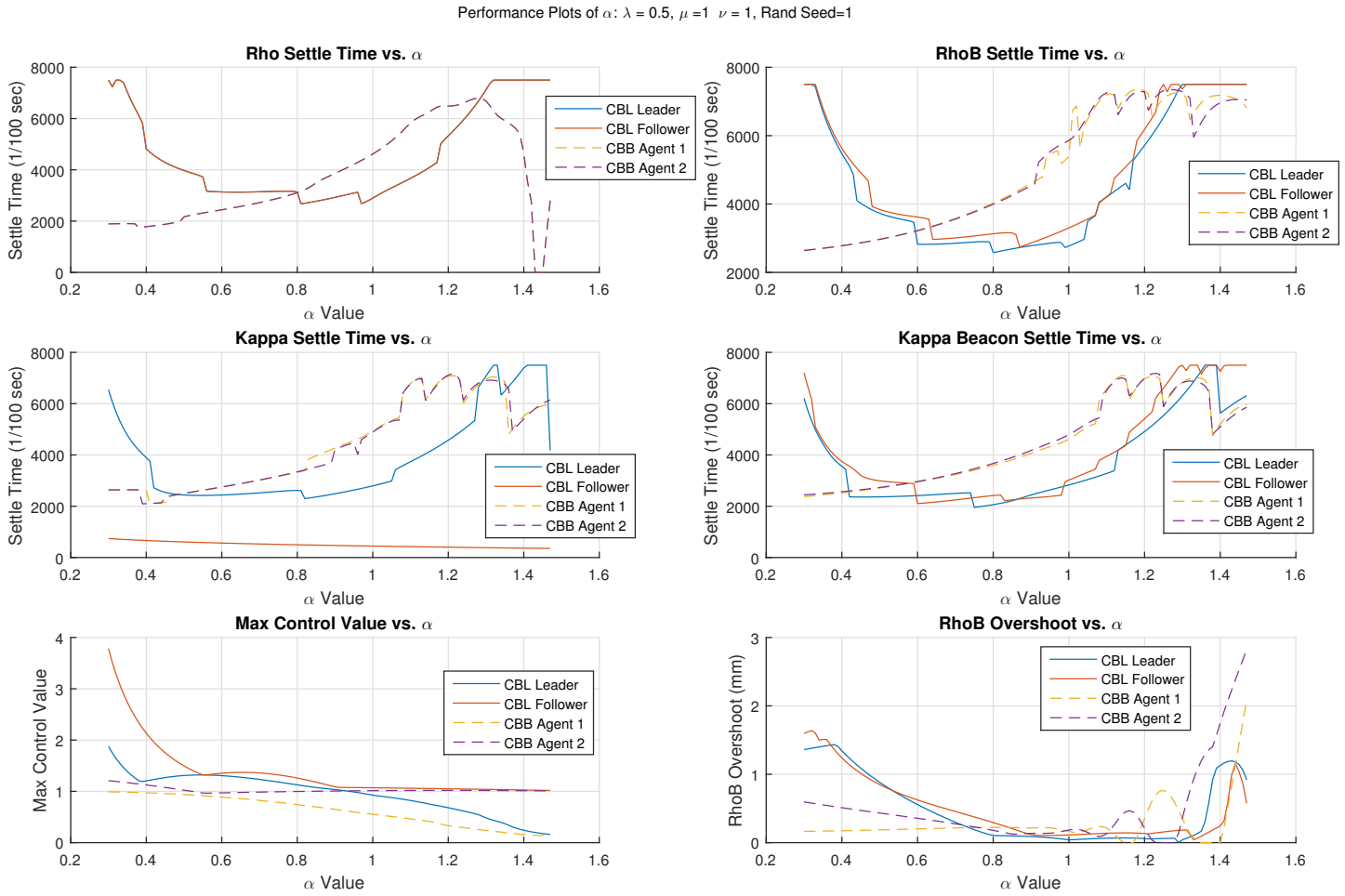

Figure 2.16: Performance Comparison of CBL and CBB, varied over λ, other parameters fixed. Initial condition parameter 1.

As expected, κ for the follower in the CBL system converges faster than all other agents regardless of the system. For α between .6 and 1.2, CBL performance appears better across most metrics. Settle times for ρ_B and κ_B are consistently better for CBL, sometimes by at most 30 seconds. For a series of initial conditions (not pictured), performance remains consistent with that of Figure 2.16. In all cases, there appears to be a point between .6 and .8 where CBL performance overtakes

CBB performance in ρ, ρ_B, κ, and κ_B settle times.

In terms of maximum actuation (max control) and beacon overshoot, the CBL system performs worse, though for α values greater than .8, performance is actually comparable between the two systems. Additionally the ρ_B overshoot is generally higher for the CBL system. This is most likely due to the follower having no knowledge of the beacon and in a sense, distracting the leader from converging to a circling equilibria about the beacon.

In general, we consider α between .6 and 1.2 to be the "operational range" of the CBB and CBL systems, where the most desirable circling geometries result from parameters in this interval. Generally speaking, the CBL system is as good if not better in terms of performance as compared to the CBB system. This is of course not necessarily true for all parameter combinations and initial conditions, though this series of simulations certainly gives insight into the nonlinear performance of both systems. The reason this might be the case is because the followers do not have divided attention, they are just following the next agent. In this way the followers might be more inclined to make more aggressive moves towards the beacon, because only the leader agent is essentially guiding the collective.

In practice, we noticed that while convergence to near equilibrium appeared to occur rather quickly, settling to the exact equilibrium values took longer than expected. This is to say that while the CBL system is faster in some cases to within 5% of theoretical equilibrium values, achieving the last 5% may take longer than CBB. This is because the leader is the only agent making adjustments specifically for the beacon's location, rather than all four agents at once.

2.3.2 Fixed Alpha and Initial Conditions, Varying Lambda

For the next set of simulations, λ is varied from .33 to .99, where α is constant, chosen to be $\pi/4$, and α_B is chosen to be $\pi/2$. The random seed was chosen to be the same as for Figure 2.16. Figure 2.17 shows the results from this set of simulations.

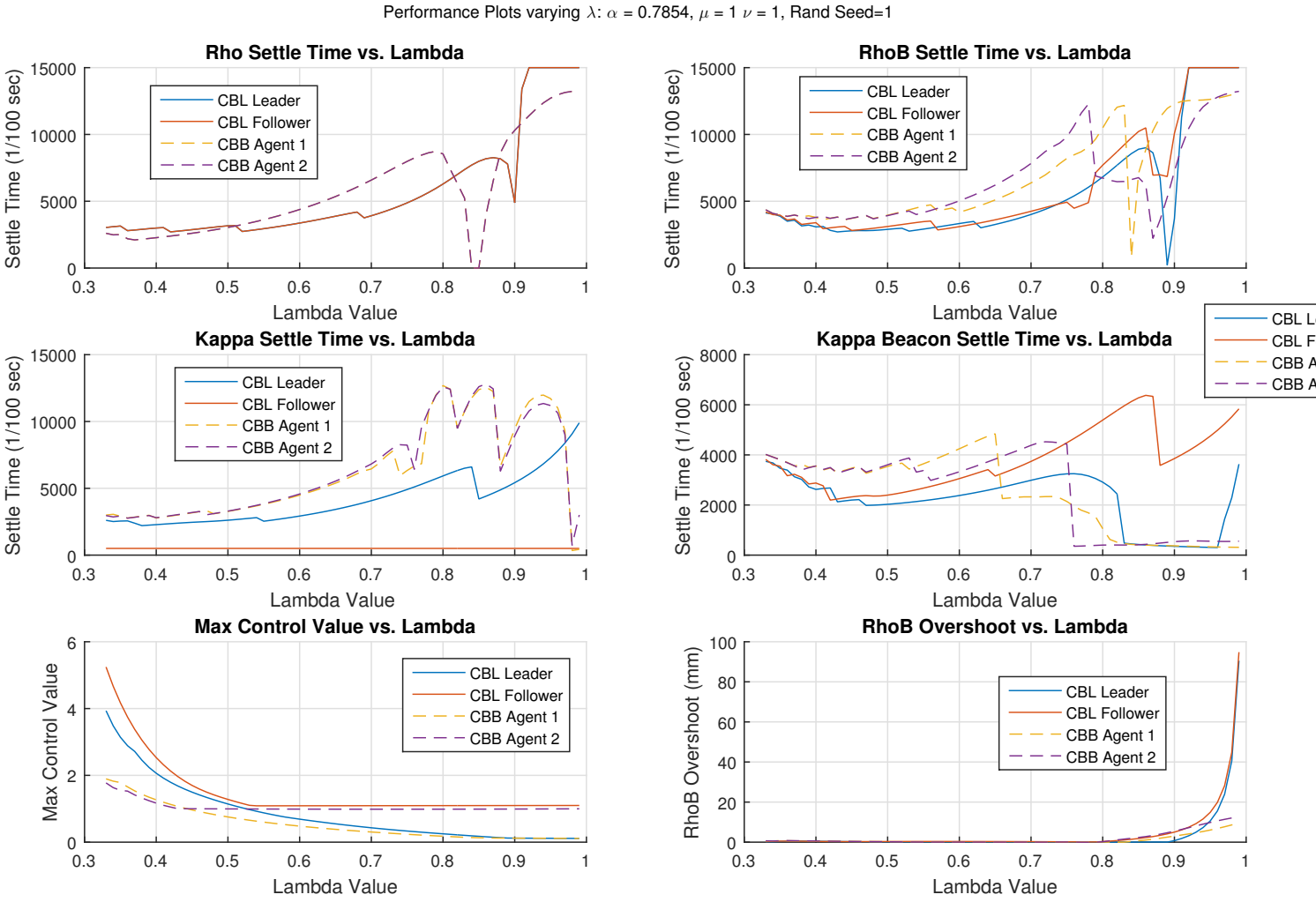

Figure 2.17: Performance Comparison of CBL and CBB, varied over λ, other parameters fixed.

Again for the CBL system, follower κ values converge significantly faster than

for the CBB system. Generally speaking, performance across various initial conditions (not pictured) is similar to that of Figure 2.17. Interestingly we see a similar trend to those of the previous section; there appears to be a λ value between .5 and .6 where CBL settle performance exceeds that of CBB in most metrics for an interval. Once the λ value exceeds .8 in the CBB system, ρ, ρ_B, and κ_B settle times become much faster than the CBL system, though it is at the expense of κ settle time, which is essentially the circling geometry. However for all sets of initial conditions, the CBL system has higher maximum actuation. Beacon overshoot appears consistent across all parameter values, until λ is greater than .9.

The most desirable "operational values" of lambda are between .4 and .7. These values provide generally the best performance across all metrics for both CBL and CBB systems, while providing a reasonable balance between beacon and agent attention.

Chapter 3: Sound Sourced, Leader Based Cyclic Pursuit

3.1 Introduction

In this section we demonstrate implementation of a phonotactic robot as the leader in the CBL system discussed in the previous chapter. The leader is given a "head", and uses sound to sense κ_B, the relative angle between the agent and the beacon. Sound sourced localization is performed by the Interaural Level Difference (ILD) and Interaural Phase Difference (IPD) algorithm, as introduced in [10], prototyped in [16], and used with a mobile robot to move to a sound source in [11]. These previous implementations have been carried out using MATLAB for signature acquisition and angle recovery. By modeling a head as a sphere, a database of theoretical signatures was utilized for angle recovery. Additionally, sensor placement on the head apparatus was traditional, with the microphones placed at $\pm$ 90 degrees on the head.

We build on these works in a several ways. First, signal acquisition, signature generation, and wireless communication is all done by fast controls-oriented embedded systems. Embedded systems have several advantages over Windows/Linux implementations; with an embedded system, real time deadlines can be met, with low cost, low profile, and energy efficient processing.

In terms of the sensing apparatus, instead of using a perfectly spherical head, we use a Styrofoam manikin head. The non-spherical nature of the manikin head means that theoretical signatures calculated based on a spherical head might not be valid for angle recovery. As such, we show that an empirical signature database generated by leveraging the accuracy of the Vicon positioning system can be used for angle recovery. To break front back symmetry, rather than use robot odometry and additional differential metrics as implemented in [11], we show in Section 3.2.2 that "optimal sensor placement" as discussed in [10] can be implemented to solve this problem in a more computationally efficient way.

3.2 Sound Sourced Localization

3.2.1 The ILD IPD Algorithm

In this work, we use the Interaural Level Difference (ILD) and Interaurel Phase Difference (IPD) algorithm, the rich theory and physics of which are introduced and discussed in [10]. In the works of Handzel and Krishnaprasad [10] the (acoustic) wave equation is solved, which is separate in time and space. By analytically finding a solution to this equation, the spacial information for a sound source at angle θ, with a head modeled as a sphere, can be computed. From this, theoretical signatures for source angles are generated. As we will not be using theoretical signatures, we will not delve into the physics involved in obtaining these equations. We only note that there are unique phase and magnitude signatures that exist based on source angles, where uniqueness is determined based on microphone placement. As in these

papers, we concern ourselves with recovery of the azimuthal angle of a sound source, rather than the inclination angle. A simple diagram of the head apparatus relative to the sound source can be seen in Figure 3.1.

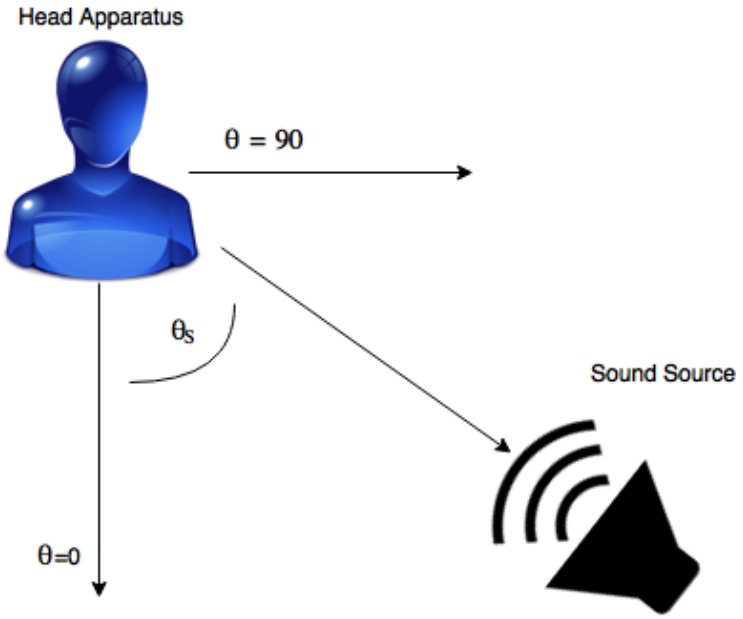

Figure 3.1: Sound Localization Diagram

The algorithm works as follows: microphones mounted on either side of a head are sampled simultaneously. For each set of samples, the microphone measures sound pressure, expressed as a complex response to excitation by a source,

$$p(\omega, \theta_S) = A(\omega, \theta_S) e^{j\beta(\omega, \theta_S)}, \quad j = \sqrt{-1}, \tag{3.1}$$

where A is the magnitude of the response, ω is the angular frequency of the sound source, β is the phase information, and θ_S is the source angle (more discussion in [10]). In practice for a set of samples, $A(\omega)$ is the magnitude response after a Fast Fourier Transform (FFT), and $\beta(\omega)$ is the phase response. The head has

both left and right microphones, so we define the Interaural Level Difference and Interaural Phase Difference as such,

$$ILD = \log A_L - \log A_R, \quad IPD = \beta_L - \beta_R. \tag{3.2}$$

Both ILD and IPD are smooth functions of ω when a broadband sound source is used. When a signal is received at the head, a specific ILD and IPD curve is generated based on the azimuthal angle of the source, which means that the IPD and ILD are also functions of the source angle θ_S. The angle recovery problem now becomes one of matching a sampled ILD and IPD function to the closest ILD/IPD signature associated with an azimuth angle θ, which we will call the "angle signature". In terms of notation, we purposefully refer to the actual sound source angle as θ_S and any arbitrary angle as θ. To accomplish angle recovery, we define the squared L^2 norm distance between the sampled $ILD(\omega)$ and $IPD(\omega)$ functions and the angle signature functions, $IPD_s(\theta, \omega)$ and $ILD_s(\theta, \omega)$. Then for each θ the IPD metric is,

$$D_2^{IPD}(\theta) = ||IPD_s(\theta, \omega) - IPD(\omega)||_2^2 = \sum_{\omega} \big(IPD_S(\theta, \omega) - IPD(\omega)\big)^2, \tag{3.3}$$

and likewise for ILD. Each metric is normalized over θ such that,

$$D(\theta) \to \frac{1}{M} D(\theta), \quad \text{where } M = \max_{\theta} D(\theta). \tag{3.4}$$

The IPD and ILD metrics are combined,

$$D_2^{\text{Comb}}(\theta) = D_2^{IPD}(\theta) + D_2^{ILD}(\theta) \tag{3.5}$$

and the recovered angle θ_R corresponds to the smallest combined metric value,

$$\theta_R = \arg\min_{\theta} D_2^{\text{Comb}}(\theta). \tag{3.6}$$

In practice this optimization is done by simply picking the smallest value of a real (discrete) vector. However, when microphones are mounted on either side of the head at ±90, angle recovery is only accurate to a front/back symmetry [10].

Ideally, θ_S will equal θ_R. Realistically, because signatures need to be generated for each θ, the recovered angle θ_R is only accurate up to a predetermined resolution. The resolution is determination is generally based on application necessity and limitations on processing and memory availability. For example, if a 512 point FFT is chosen and a 2 degree θ resolution is desired, there are 180 D_2^{IPD} and D_2^{ILD} metrics that need to be calculated every sample, each of which involves 256 subtractions, additions, and multiplications. In terms of memory requirements, assuming each floating point is 4 bytes, the signature database would occupy 458,752 bytes of memory. While this memory requirement is not significant for modern desktop computers, it is significant for an embedded system, in which a large amount of flash memory is on the order of 1 MB.

3.2.2 Breaking the Symmetry

Several methods exist to break the symmetry problem associated with microphone placement at ±90 degrees on the head. In [11], the symmetry problem is broken by taking the difference between subsequent ILD and IPD samples, and creating two additional metrics based of these differences, ILD' and IPD'. This method relies on a "flow" in the signature curves that is induced when the head is turned relative to a sound source. Using this method, angle signatures for these

two new functions are created, and by using odometry from a moving robot, the front/back ambiguity can be broken.

There are few disadvantages to this method, however. First, if there is no change in heading, there is a "divide by zero" scenario which occurs when calculating IPD' and ILD' quantities in practice. The second disadvantage is computational; two additional metrics ($D_2^{ILD'}$ and $D_2^{IPD'}$) are required to be calculated for each angle signature, effectively doubling the amount of computation and memory required to recover an angle.

Rather than using the difference (flow) method, we propose using "optimal sensor placement for localization", as discussed in [10]. Optimality in this case has been determined empirically, with respect to the ability to uniquely recover angles for the entire 360 degree spectrum. By mounting the microphones at ± 50 degrees, the combined metric (3.5) has a unique minima for each source direction angle θ.

3.3 Embedded Systems Design and Development

Embedded systems development can span the breadth of electrical engineering knowledge; it requires computer engineering knowledge for CPU management and peripheral configuration, digital signal processing for sampling data and manipulation, communications for data transmission, controls for feedback processing, and networking for TCP/IP and 802.11 communications. With such complex and capable systems, good software design principles should be followed throughout development. This section discusses the design and development of the embedded

systems used in implementing the ILD/IPD algorithm.

3.3.1 Hardware Selection

A Texas Instruments (TI) LAUNCHXL-F28377S was chosen as the primary computing platform. This microcontroller features several key features that are necessary to achieving proper implementation of the ILD/IPD algorithm for use in a control system. The F28377S is computationally fast, with a 200 MHz processor, a 200 MHz "control law accelerator" dedicated to computation, and a whole 1 MB of on-board flash memory. Several DSP libraries are supplied and maintained by TI, with support for floating point computation, and implemented in such a way that results can be directly compared with those from MATLAB. The microcontroller can run Real Time Operating Systems (RTOS), which can ease development complexity, while not sacrificing the real time capabilities of the system. Furthermore, this microcontroller has two analog to digital converters (ADC), many pins for general purpose input/output (GPIO), and multiple serial interfaces for external communication.

Unfortunately at the time of development, this microcontroller did not have direct support for 802.11 WIFI modules. As such, a TI MSP430 low power microcontroller with a CC3100 WIFI booster back was chosen as an intermediary between the F28377S and the lab's control server. The MSP430 is a slower microcontroller running at 20 MHz with significantly less RAM and flash memory.

3.3.2 System Design

The overall system design is as such - the MSP430 signals the F28377S to sample the microphones and calculate ILD/IPD values. The sampled signatures are sent back to the MSP430 via a serial connection, then transmitted over WIFI to the lab's control server, where MATLAB parses the transmission and performs an angle lookup based on the received data. Once κ_B is determined, the angle is published to the Robot Operating System (ROS) server. Then, the control program subscribes to MATLAB published κ_B angle, using that value in the control law calculation for the leader agent, and subsequently commanding the robots to execute the control laws. Figure 3.2 shows a detailed system design diagram. Each blue box is a module that was designed and tested independent of the rest of the other modules. In this way organized development could take place for this complex system.

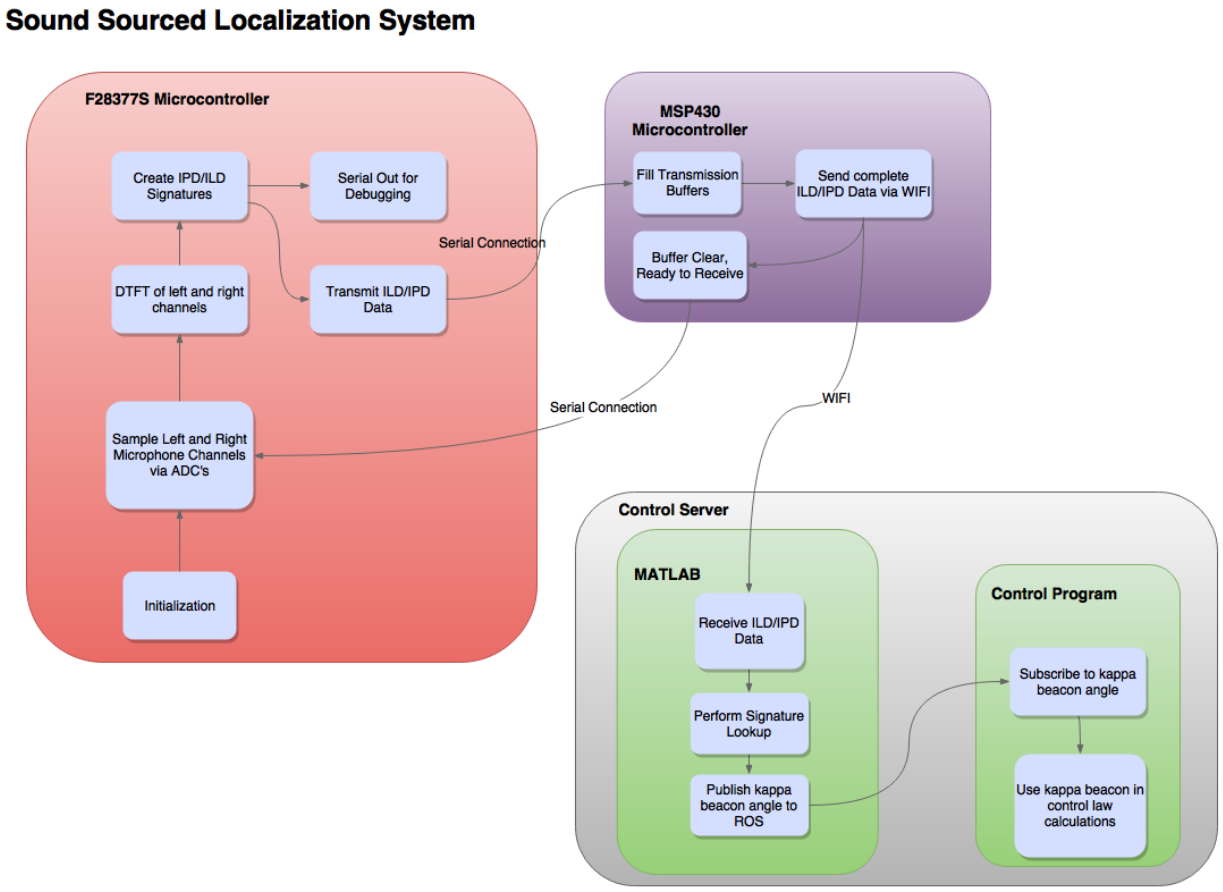

Figure 3.2: Embedded System Design Diagram

3.3.3 Development

Development started with ADC sampling on the F28377S microcontroller. Instead of using multiple pins of the same ADC to sample the left and right channel sequentially, two separate ADC's were using to sample the left and right channels in parallel. One advantage to this method is that no adjustment needs to be made for sampling delay between channels. In order to meet real time requirements,

precise timing and sampling is necessary. As such, a 20 kHz sampling rate for both ADC's was desired. 16 bit ADC sampling was chosen over 12 bit for enhanced sampling resolution, despite additional processing requirements. Figure 3.3 shows the information flow and timing for a single ADC. What is important to note, is that multiple on-board clocks control different aspects of ADC performance, including processing and sampling rates.

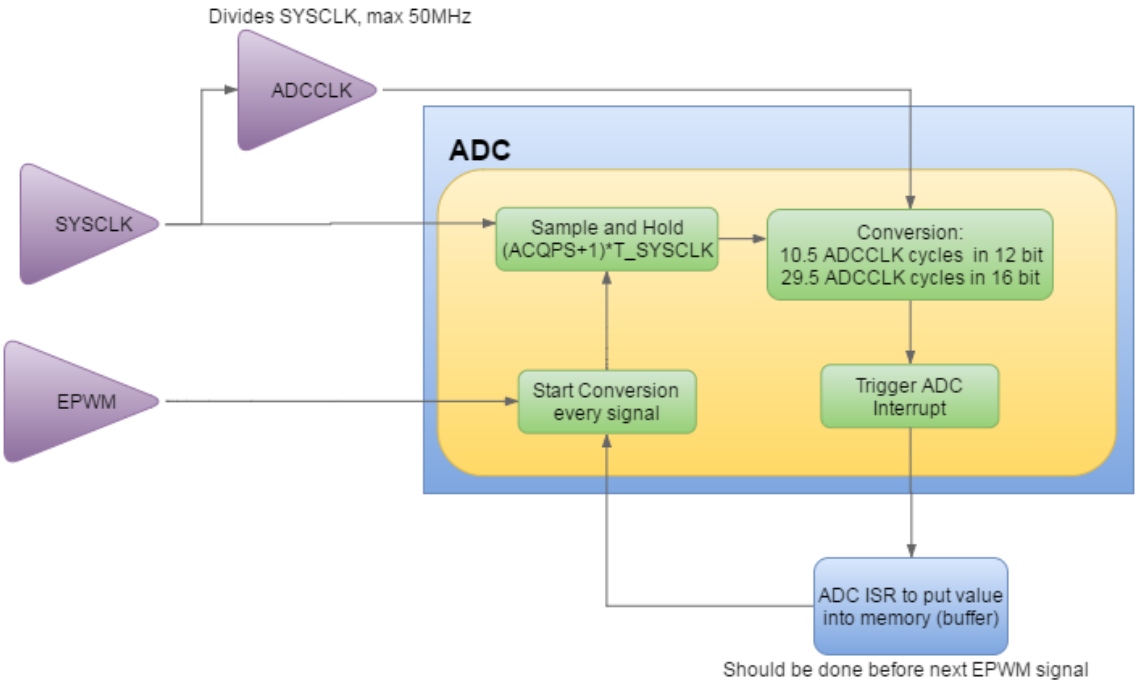

Figure 3.3: Processing diagram of the F28377S ADC. External clocks (triangles) feed into the ADC. Each ADC conversion cycle is triggered by the EPWM signal.

The maximum frequency of the broadband sound source was chosen to be 10 kHz, so by Nyquist, the sampling rate of each ADC was chosen to be be 20 kHz. The signal to begin analog to digital conversion essentially controls the sampling rate of the ADC, which is separate from the clock used to sample and convert the

data. Enhanced Pulse Width Modulation (EPWM) was configured on the F28377S to generate a 20 kHz square wave and supply both ADC's with the same trigger signal to start conversion. The EPWM signal was also output to a GPIO pin and checked with an oscilloscope to ensure proper frequency characteristics.

Once the 20kHz EWPM signal triggers the ADC to start conversion, there are approximately $50\mu s$ to sample and hold the voltage on the input pin, convert the voltage to an integer, trigger an ADC interrupt, and store the integer value in a memory buffer via an Interrupt Service Routine (ISR) before the next trigger signal occurs. While sampling, conversion, and interrupt triggering happen in parallel for each ADC, while servicing each ADC's interrupt is handled by the main processor, and therefore is executed sequentially.

The system clock (SYSCLK) was configured to run at its advertised maximum speed of 200 MHz, yielding a period of $5ns$ per cycle. As such, the ADC's Clock (ADCCLK) is based off the SYSCLOCK, and was configured at its maximum speed of SYSCLK/4. By design, the sample and hold duration must exceed both 1 ADCCLK period and the minimum sample and hold duration, 320 ns (for 16-bit conversion). The sample and hold time parameter ACQPS was chosen to be the minimal value 63, making the sample and hold duration,

$$\begin{aligned} \text{S\&H} &= (ACQPS + 1) * T\_SYSCLK \\ &= (63 + 1) * 5ns \\ &= 320ns. \end{aligned} \tag{3.7}$$

And the conversion time (CT) is approximately,

$$\begin{aligned} \text{CT} &= 29.5 \text{ ADCCLK cycles} \\ &= 29.5 \text{ ADCCLK cycles} * \frac{5\text{ns}}{1 \text{ SYSCLK cycle}} * \frac{4 \text{ SYSCLK cycles}}{1 \text{ ADC cycle}} \\ &= 590ns \end{aligned} \tag{3.8}$$

The total time for each ADC to sample and convert a pin voltage is,

$$\begin{aligned} ADCT &\triangleq \text{S\&H} + \text{CT} \\ &= 320ns + 590ns \\ &= 910ns. \end{aligned} \tag{3.9}$$

Then the remaining time to trigger an interrupt and service two ISR's is,

$$50\mu s - 0.910\mu s = 49.09\mu s \tag{3.10}$$

or 9818 SYSCLK cycles, which is more than enough to be ready for the next conversion cycle.

In order to implement the ILD/IPD algorithm, Fast Fourier Transforms (FFT) of each channel must be performed to retrieve phase and level information as a function of frequency (ω). A 512 point FFT for each channel was chosen as a balance between memory usage, computational requirements, and data transmission size. According to TI's DSP library data sheets, one 512 point FFT takes 13675 SYSCLK cycles, which means that two sets of FFT's cost 27350 cycles or 136.75μs. The immediate consequence of this processing time means ADC conversions cannot occur continuously, else ADC buffers will be overwritten during FFT computation. Therefore after 512 samples have been collected for each channel, EPWM triggering was configured to cease until the ILD/IPD information was sent to the MSP430.

Serial output was configured for both debugging (to a desktop computer) and data transmission (to the MSP430). The BAUD was configured for a rate of 460,800 bits per second, a fast rate which could be attained by both the F28377S and the less powerful MSP430, with low error.

Using the serial debugging feature, TI's FFT library was tested against MATLAB's to ensure proper results. After carefully aligning FFT buffers in memory on the microcontroller, the FFT results were nearly identical to MATLAB's.

Texas Instruments' Real Time Operating System (TIRTOS) was used on the F28377S to take advantage of a scheduler, to simplify development, and allow for extendibility of the code if necessary. Tasks (threads) can easily be added and managed, with precise control over priority of execution and interrupts. While the finished program on the F28337S only had one task, semaphores, hardware interrupts, and boot control modules were used considerably throughout development.

The MSP430 was configured to communicate with the F28377S via a serial connection at the same rate of 460,800 bits per second. The structure of the MSP430's program was as such,

1. Initialize network connection with static IP.
2. Send a signal to the F28377S to sample.
3. Fill transmission buffers until an "end of signal" character is received.
4. Transmit the buffer data over WIFI to the control server.
5. Repeat from step 2.

The MSP430 runs at a maximum clock speed of 20 MHz and has significantly less RAM and flash memory than the F28377S. As such, the MSP430 needs to control when it receives sampled data, because its transmission speeds act as a bottleneck for the whole system (this was determined empirically). Therefore the above algorithm ensures that the MSP430 is operating at maximal speeds, with the F28377S operating in sync.

The TCPIP toolbox in MATLAB was used to receive and parse the ILD/IPD signatures. After some testing, it was determined that the speed of the MATLAB angle recovery caused a bottleneck even slower than that of the MSP430. Given the MATLAB bottleneck, angle recovery was still achieved at a rate of approximately 3-5 Hz, which proved sufficient for implementation in the CBL system. However if faster speeds were necessary, the angle recovery procedure could be done on the F28377S. This would require angle signatures to be loaded into flash memory of the microcontroller, and calculations done in the C language. Moreover the microcontroller's Control Law Accelerator (CLA) could be used in parallel with the primary processor to increase computation speed.

The configuration used in this work requires the transmission of 512 floating points once by serial, then again by WIFI, then parsing by MATLAB. With on-board angle recovery, the MSP430 would only have to perform one 3 character transmission per sample cycle, and could potentially publish directly to ROS. This would shift the system's bottleneck to the speed of the F28377S, and would be significantly faster than the current setup. Due to the substantial size of the ILD/IPD signature database, flash memory would have to be used instead of RAM, which

would adversely affect computation speeds. Due to time constraints, this will be left to later work.

.

3.4 Head Calibration and Angle Recovery

3.4.1 Head Setup

Two Electret Microphone/Amplifier combination boards were mounted on a Styrofoam manikin head at ±50 degrees from a 0 degree heading. Each microphone was wired into 3.3V DC power, and an oscilloscope was used to calibrate the microphone gain to where voltage was roughly equivalent between the two channels. Figure 3.4 shows the leader agent setup with the two on-board microcontrollers.

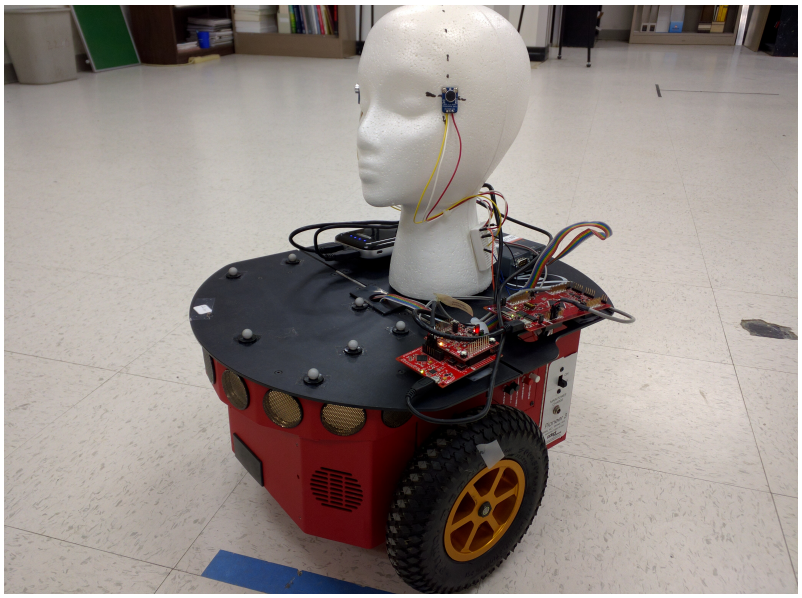

Figure 3.4: The Phonotactic Leader Agent with Optimal Microphone Placement

A dynamic signal analyzer (DSA) was used to verify microphone response

in the frequency domain, and compare with FFT results from the F28377S. Note there was no pre-ADC filtering performed with this setup, nor were the microphones embedded flush into the head. The microphones were simply mounted on the surface of the head.

3.4.2 ILD/IPD Signature Generation

In order for angles to be recovered and metrics computed, there must be a "database" of ILD and IPD signatures corresponding to a range of θ_S, the sound source direction. In the works of [10], [16], and [11], theoretical ILD and IPD signatures were generated for this database by solving the Helmholtz equation for a perfectly spherical head. While this method was very effective in angle recovery, it was done for a perfectly spherical head. Shape variations of the head and sound characteristics of a testing space may significantly impact the ability to use these theoretical signatures.

In this work, the lab's Vicon high precision indoor motion capture system was leveraged in conjunction with the Robot Operating System (ROS), and MATLAB ROS Toolkit to automate the creation of an empirical signature database (a detailed account of the the lab's setup and system intercommunication can be found in Chapter 4). The calibration algorithm works follows, assuming the speaker (source) location is the same as the beacon location,

1. Input the angle resolution, samples per angle, and start angle.

2. In MATLAB, publish the desired κ_B to ROS.

3. In the robot control program, subscribe to Matlab's published angle and change robot's κ_B to the desired angle.

4. Open a TCP server and await ILD/IPD data from the MSP430. Save the signature. Repeat for as many samples per angle as desired.

5. Repeat 2-4 until all samples have been taken.

6. In a post processing MATLAB script, smooth each sample with a moving average, then average all samples for each angle.

As a note, the phase samples are wrapped to 2π. This means when the IPD is calculated, the difference of phase between the left and right channel must again be wrapped to 2π, the rate of success that a κ_B angle will be recovered, will be extremely low. We refer to this success rate as the *angle recovery rate*, or just *recovery rate* for short.

Figure 3.5 shows the robot generating an ILD/IPD signature database. Note that the orange beacon (cone) is placed on top of the speaker, so as to provide the Vicon system with positioning data of the speaker, which is then used to position the robot appropriately.

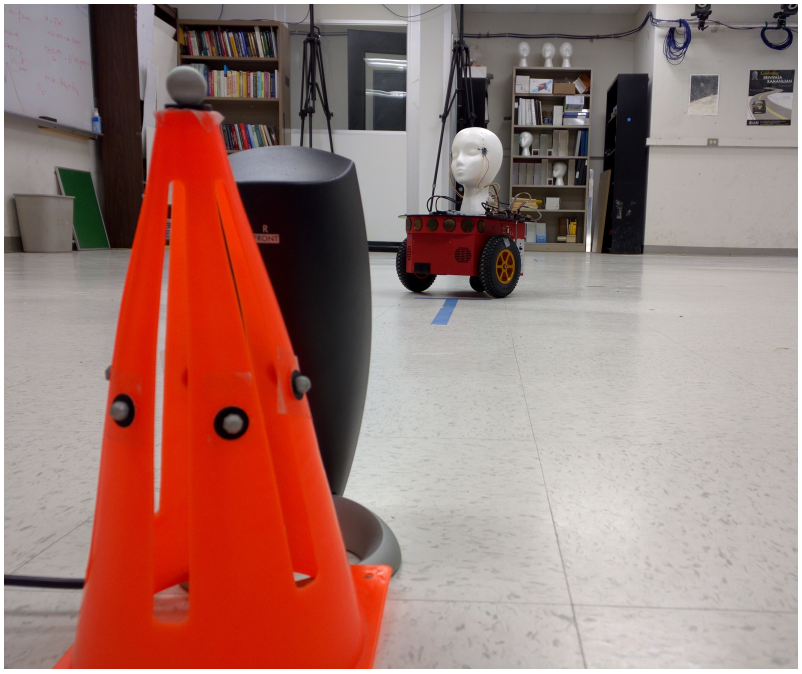

Figure 3.5: ILD/IPD signature database generation.

3.4.3 Stationary Beacon Tracking Law

In this section we discuss a stationary beacon tracking law, where an agent is stationary, and needs to track a target by changing its heading alone. The purpose of this is twofold, the first is to precisely change the heading of an agent equipped with a sound localization apparatus for ILD/IPD calibration. The second reason is to simulate a perfectly radial (omnidirectional) sound source; if a stationary agent with a speaker tracks (by heading alone) the position of the circling leader agent from the CBL system, then from the leader's perspective, the beacon (agent) is radially emitting sound.

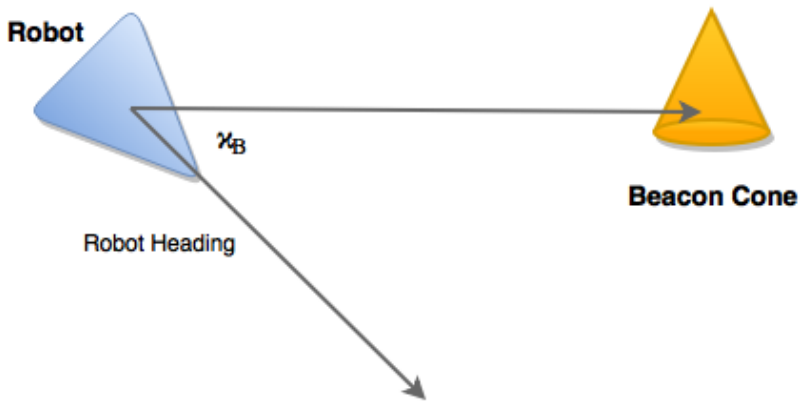

Figure 3.6: Stationary Tracking Diagram

Unfortunately, the dynamical model used for the CBL system (2.11) cannot be used to model the stationary agent, because that model assumes non-zero velocity. In this case there will be assumed zero translational velocity, so the system can be modeled as a simple unicycle [17],

$$\begin{aligned} \dot{x} &= u_T \sin(\kappa_B) \\ \dot{y} &= u_T \cos(\kappa_B) \\ \dot{\kappa}_B &= -u_\omega, \end{aligned} \tag{3.11}$$

where x and y are scalar coordinates in the lab's frame of reference. Here we assume without loss of generality that the coordinate frame is the vector that connects the leader at the beacon on the x axis. Figure 3.6 shows the simple diagram for the beacon tracking problem. Assuming the beacon and the robot are stationary

($u_T = 0$), the dynamics simplify,

$$\begin{aligned} \dot{x} &= 0 \\ \dot{y} &= 0 \\ \dot{\kappa}_B &= -u_\omega, \end{aligned} \tag{3.12}$$

Now drawing inspiration from the beacon tracking portion of the CBL control law (2.15) and using Vicon feedback, the beacon tracking control law and dynamics are,

$$\begin{aligned} \dot{\kappa}_B &= -u_\omega \\ u_\omega &= \mu \sin(\kappa_B - \alpha_B) \end{aligned} \tag{3.13}$$

where α_B is the desired κ_B angle. This is a separable ODE, and can be solved:

$$\begin{aligned} \frac{d\kappa_B}{\sin(\kappa_B - \alpha_B)} &= -\mu\, dt \\ \int_{\kappa_{B0}}^{\kappa_B} \frac{d\tilde{\kappa}_B}{\sin(\tilde{\kappa}_B - \alpha_B)} &= -\mu \int_0^t d\tilde{t} \\ \log\tan(\frac{\alpha_B - \tilde{\kappa}_B}{2})|_{\kappa_{B0}}^{\kappa_B} &= -\mu t \\ \tan(\frac{\alpha_B - \kappa_B}{2}) &= e^{-\mu t}\tan(\frac{\alpha_B - \kappa_{B0}}{2}) \\ \kappa_B &= \alpha_B - 2\tan^{-1}\left(e^{-\mu t}\tan(\frac{\alpha_B - \kappa_{B0}}{2})\right) \end{aligned} \tag{3.14}$$

Where κ_{B0} is the initial condition, and t is time. If $\alpha_B = \pi + \kappa_{B0}$, then $\kappa_B = \alpha_B - \pi$, $\forall t$. This is however an unstable equilibrium point, because for all other initial conditions such that $\tan(\frac{\alpha - \kappa_{B0}}{2}) = c \in \mathbb{R}$,

$$\lim_{t\to\infty} 2\tan^{-1}\left(ce^{-\mu t}\right) = 0 \tag{3.15}$$

And indeed κ_B asymptotically converges to α for initial conditions such that $\kappa_{B0} \neq \alpha_B - \pi$.

In practice, this control law works for tracking a circling agent if μ is chosen large enough, though we will not prove stability for this case. For experiments, μ was chosen to be 1000.

3.4.4 Broadband Sound

MATLAB was used to generate broadband sound for the experiments, to carry out phonotaxis. As discussed in previous sections, true broadband sound generates smooth ILD and IPD curves because they are functions of ω. In practice, perfectly broadband sound cannot be generated, but higher density will yield smoother signature curves. For the majority of experiments the broadband sound ranged from 200 to 10,000 Hz, at 43 Hz increments.

It was noticed over the course of head calibration and signature generation experiments that frequencies from 6.5kHz to 10 kHz were not being captured with a high enough magnitude response. As such, magnitudes of higher frequencies were increased relative to the magnitudes of lower frequencies. A linear frequency weighting was chosen to solve this issue, and the power spectral density estimate of the broadband source can be seen in Figure 3.7, where the total signal is given by,

$$BB(t) = \sum_{i=1}^{M} \frac{i}{M} \sin(\omega_i t), \tag{3.16}$$

where M is the total number of frequency components, and ω_i is the i^{th} individual frequency component.

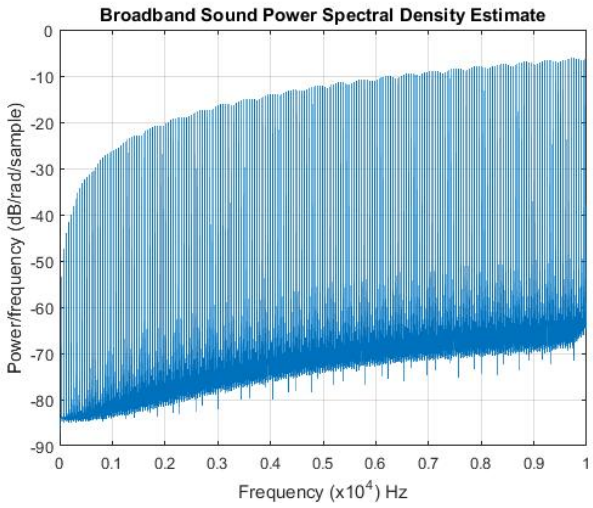

Figure 3.7: Linearly weighted broadband sound PSD

One distinct advantage to shifting the majority of the signature information to higher frequencies is lack of interference with human speech and other sounds. Humans normally speak at less than 300 Hz, and the majority of interfering environmental sounds are below 3 kHz (determined empirically in the lab).

For calibration experiments (signature generation), the sound was played from a computer speaker. For later implementation in the CBL system, due to the need for wireless sound, the broadband sound was played from a portable Bluetooth speaker. The sound itself was saved and compressed as a ".flac" file in MATLAB, and played over Bluetooth.

3.4.5 Building and Checking Signatures

Calibration experiments were carried out as follows; a signature database was generated from a set of data, a second set of signatures was collected using the same angles that generated the signature database from the first set of data. Then in a

MATLAB script, the signature database generated from the first set of samples was used to recover the angles for the second set of samples.

We define the ability to successfully match an ILD/IPD signature to the actual angle of incidence as "angle recovery". The rate of success that an angle will be recovered from a signature is called the "recovery rate".

A suite of experiments were run, most of which are left out of this section. Angle recovery was generally poor in initial experiments because IPD was not being wrapped to 2π.

3.4.5.1 Recovery Experiment 1: 5 Degree Resolution

This experiment was carried out with a 512 point FFT, from $\kappa_B = -90$ to 90 degrees, at 5 degree resolution, 25 samples per angle, and the broadband signal spanned from 200 Hz to 10 kHz, at 43 Hz steps, with no frequency weighting. This experiment was performed with the exact sound sensing apparatus from [11], not a styrofoam manikin head. This apparatus was used for a closer "ground truth", to test if the spherical head, with low pass filtering, and extremely high quality microphones would produce signatures capable of being recovered empirically. Due to the angles tested in this experiment, symmetry was not an issue.

Experiment results can be seen in Figure 3.1. "9 point DB averaging" denotes that each captured IPD/ILD signature was smoothed with a 9 point moving average, before every sample per angle was averaged to create the signature database. Recovery rates in this experiment are considered good - where the desired recovery

Experiment Conditions and Notes	Recovery Rate
No Smoothing	86.05%
9 point DB averaging	89.08%

Table 3.1: Recovery Experiment 1 Recovery Rates

rate would be greater than 80% with a low variance. This experiment suggests that smoothing the DB is advantageous to recovery.

3.4.5.2 Recovery Experiment 2: 2.5 Degree Resolution

This experiment was carried out with a 512 point FFT, from $\kappa_B = -90$ to 90 degrees, at 5 degree resolution, 50 samples per angle, and the broadband signal spanned from 200 Hz to 10 kHz, in 43 Hz steps, with linear frequency weighting. The second data set was captured at 5 samples per angle. This experiment was carried out using the same sound sensing apparatus as Recovery Experiment 1.

Figure 3.8 and 3.9 show plots for each of the 50 signatures for $\kappa_B = 90$ degrees and -90 degrees respectively; each color corresponds to a different sample. Clearly there are phase and level difference trends, which allow for unique angle identification. For the IPD in these plots, the majority of the unique phase information appears to be from 3 kHz to 8 kHz. Ideally the spacial information would be good throughout the entire frequency spectrum, but the 5 kHz interval proves sufficient for angle recovery. Table 3.2 shows the recovery results of the experiment.

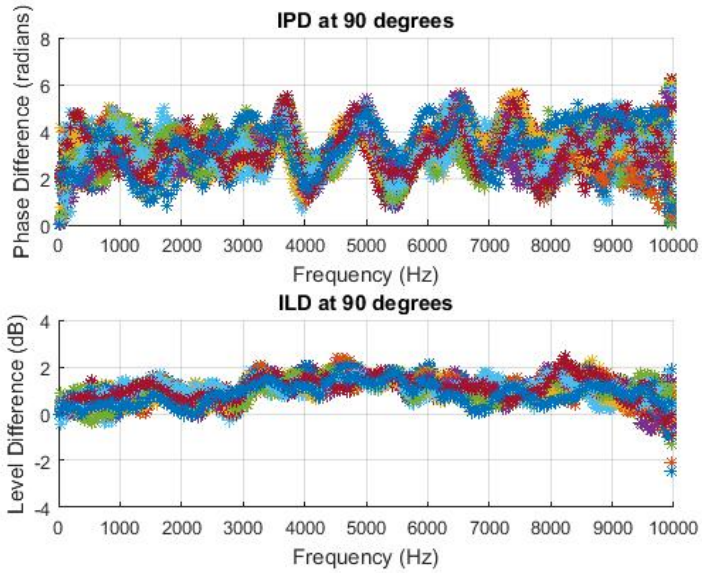

Figure 3.8: Captured Unsmoothed ILD and IPD signatures for $\kappa_B = 90$ degrees.

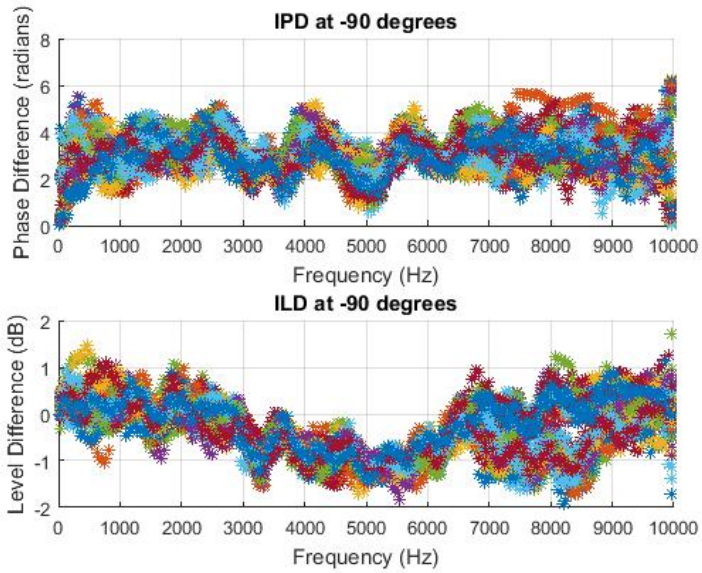

Figure 3.9: Captured Unsmoothed ILD and IPD signatures for $\kappa_B = -90$ degrees

Experiment Conditions and Notes	Recovery Rate
No DB Averaging	78.90%
9 point DB averaging	80.82%
9 point DB averaging, 5 point sample averaging	79.72%
9 point DB averaging, 9 point sample averaging	79.45%
9 point DB averaging, self check	89.17%

Table 3.2: Recovery Experiment 2 Recovery Rates

In the "Experiment Conditions and Notes" section of Table 3.2, "n point sample averaging" refers to smoothing samples prior to angle recovery. The "self check" refers to an experiment where samples used to create the database were checked against the database itself.

The recovery rate only decreased approximately 9% from the previous experiment, despite doubling the angle resolution. Figure 3.10 shows a plot of the recovered angles versus the actual angle. The blue line in the plot denotes the desired recovery curve. Figure 3.11 shows the variance of the error, per angle. Generally speaking the variance is low enough to be useful in controls experiments, with more error towards the poles ±90 degrees.

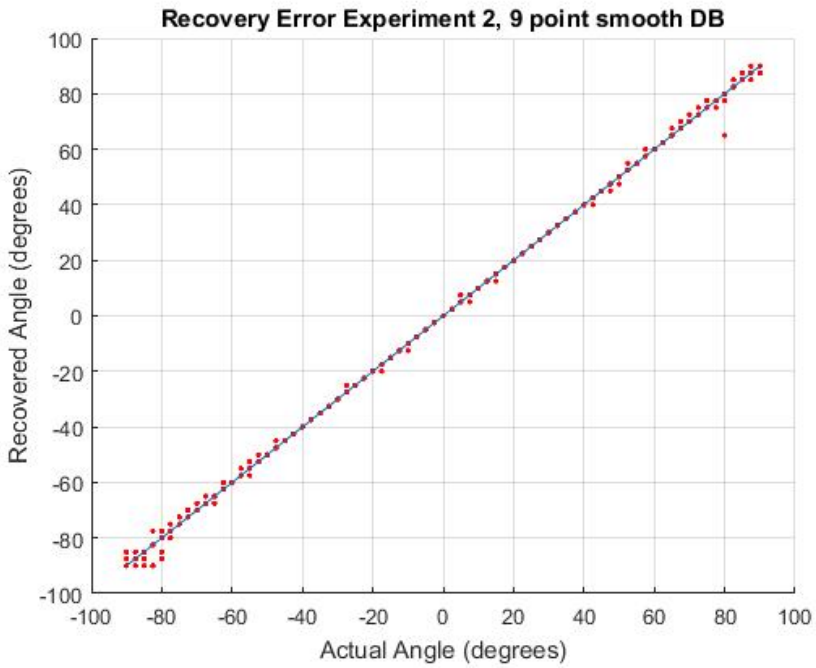

Figure 3.10: Recovery Plot from Experiment 2, 80.82% recovery rate

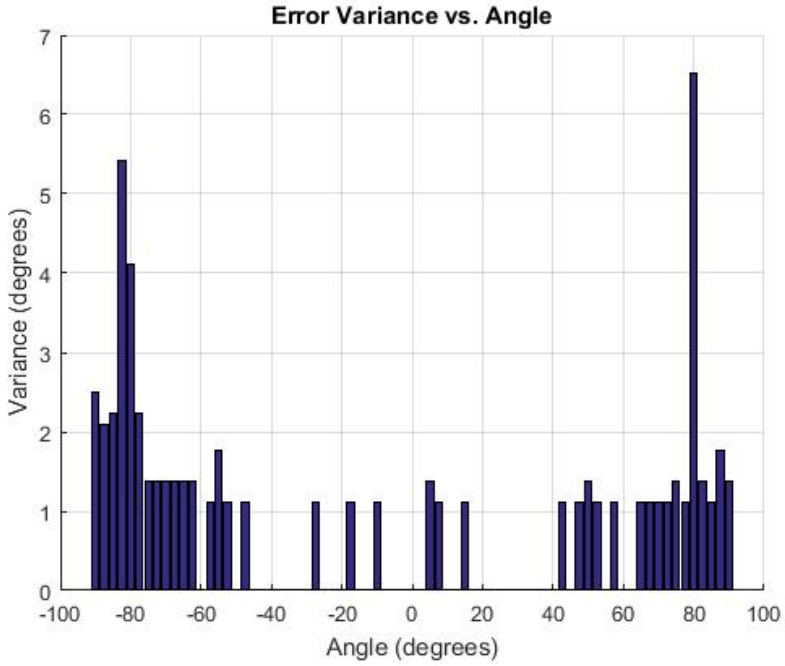

Figure 3.11: Recovery Error Variance Plot from Experiment 2, 80.82% recovery rate

Experiment Conditions and Notes	Recovery Rate
9 Point DB averaging	82.19%
9 Point DB averaging, 5 point sample averaging	84.11%
9 Point DB averaging, self check, 5 point sample averaging	92.35%

Table 3.3: Experiment 3, Manikin Head Recovery Rates

3.4.5.3 Recovery Experiment 3: The Manikin Head

In this experiment, the Manikin head was used with microphone placement at ±90 degrees. Similar to the previous experiment the following conditions and parameters were used: 512 point FFT, -90 degrees to 90 degrees, 2.5 degree increments, 50 samples per angle, broadband sound from 200-10000 kHz at 43 Hz increments, with linear frequency weighting. The second data set collected was the same, but 5 samples per angle. Table 3.3 shows the recovery results.

From the recovery rates, it appears that the non-low pass filtered, moderately priced microphones, and the styrofoam head perform just as well as the head apparatus from the previous experiments. While overall error remained low, Figure 3.12 and 3.13 show three distinct samples (out of 365) that were considerably inaccurate.

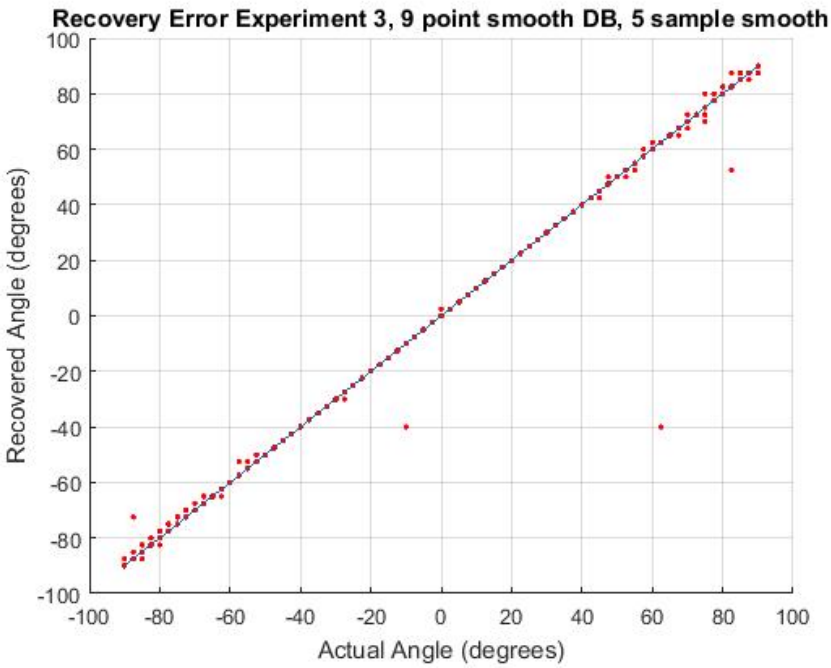

Figure 3.12: Recovery Plot from Experiment 3, 84.11% recovery

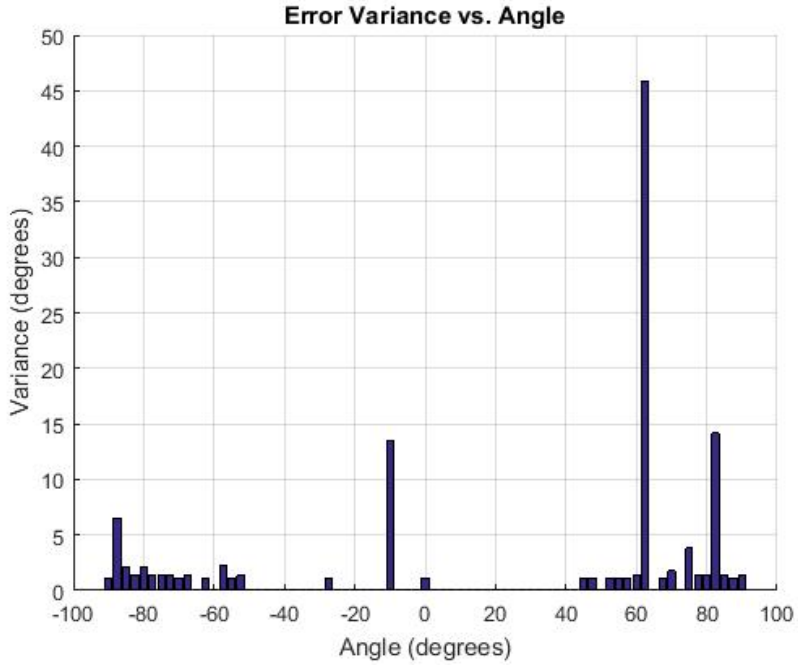

Figure 3.13: Recovery Error Variance Plot from Experiment 3, 84.11% recovery

To see if manikin head signatures were comparable with spherical head signatures from Experiment 2, another angle recovery test was performed. In this test, the samples that generated the signature DB (50 per angle) from the manikin head were tested against the signature DB from the spherical head in the previous experiment. The same test was also performed in reverse, where the signature DB from the manikin head experiment was used to recover angles from the spherical head experiment. Table 3.4 shows the results from this test. From this test it is abundantly clear that proper calibration is required based on the head shape and microphone type.

Experiment Conditions and Notes	Recovery Rate
Spherical Head DB vs Manikin Samples	.02739%
Manikin DB vs Spherical Head Samples	0%

Table 3.4: Experiment 3, Manikin and spherical recovery rates. For each recovery experiment, samples were taken with one agent and the database of the opposing agent was used for recovery.

3.4.5.4 Recovery Experiment 4: Full 360 Signature with Optimal Sensor Placement

The microphones were "optimally" placed at ±50 degrees on the manikin head. The parameters for the experiment were as follows: 512 point FFT, 0 degrees to

358 degrees, 2 degree increments, 50 samples per angle, broadband sound from 200-10000 kHz at 43 Hz increments, with linear frequency weighting. The second set of samples was taken at 5 samples per angle. Table 3.5 shows the recovery rates for this experiment.

Experiment Conditions and Notes	Recovery Rate
9 point DB averaging	84%
9 point DB averaging, 5 point sample averaging	85.55%

Table 3.5: Experiment 4 Recovery Rates

Given the increased angle resolution and increased number of total DB signatures from breaking front/back ambiguity, the results are very strong. Figure 3.14 shows no "spurious" angle recoveries, and Figure 3.15 shows a very low error variance. These results indicate that CBL implementation with the phonotactic robot is plausible. Moreover there were no recovery problems due to front/back ambiguity. These results also indicate that smoothing samples before checking against the database is advantageous, but not required.

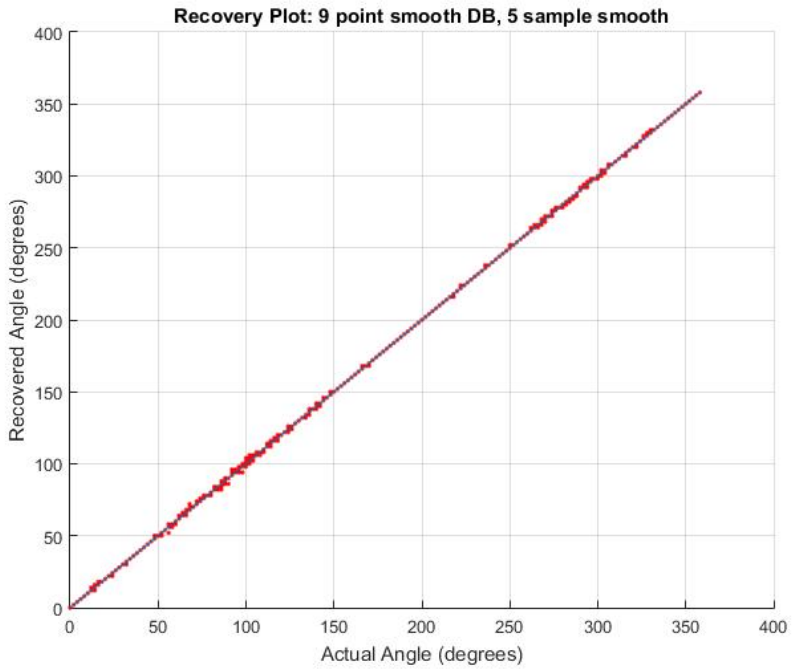

Figure 3.14: Recovery Plot from Experiment 4, 85.5% recovery rate

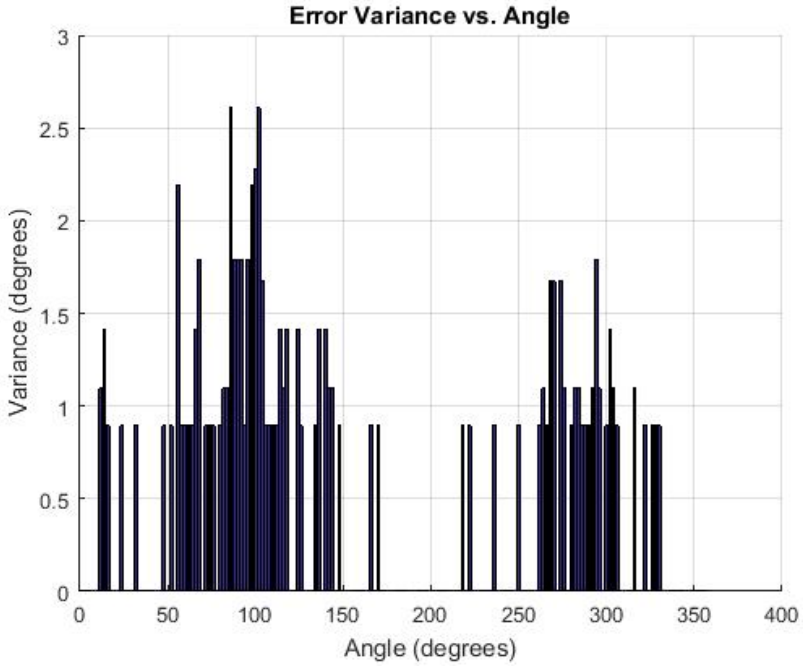

Figure 3.15: Recovery Error Variance Plot from Experiment 4, 85.5% recovery rate

3.5 CBL with Robot Phonotaxis

To show robustness of the phonotactic robot, we demonstrate the CBL system as discussed previously, but where the leader is guiding the collective through sound sourced localization of the beacon. This is intended to show that the ILD/IPD algorithm is indeed suitable for real-time control, and is reliable enough to be used in real controls systems.

The high level design of the CBL phonotactic leader system is shown in Figure 3.16. The MSP430 drives sampling, as it acts as a processing bottleneck for the rest of the system. No true design changes were necessary other than modifying the C++ control program to subscribe to MATLAB's published κ_B angles, so that the angle could be used in computations of the leader's control law for the CBL system.

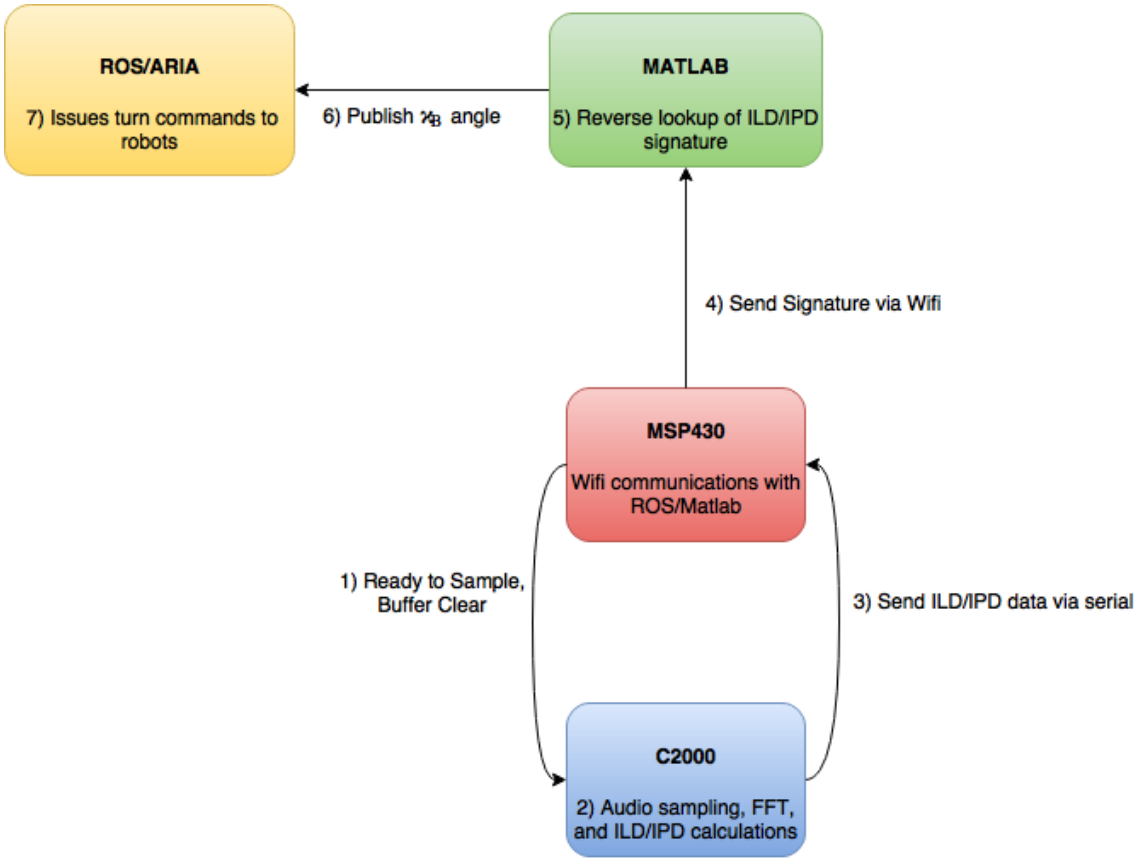

Figure 3.16: Information Flow of the CBL-Phonotaxis Embedded Implementation

3.5.0.1 Beacon Configuration

Due to the difficulties with fabricating a speaker or baffle configuration that radially emits sound adequately, a "simpler" more controls focused solution was carried out by creating a beacon agent. A speaker was fixed to the top of a robot, which was in turn assigned the Stationary Beacon Tracking Law from the head calibration section (3.13). The chosen α_B for the control law was 0, while the κ_B angle used for feedback was the κ angle between the beacon agent and the leader agent. The beacon agent has zero translational velocity, while its rotational velocity

is being changed by the stationary beacon tracking law. As per previous analysis, if the leader agent has reached circling equilibrium, the beacon agent's heading will asymptotically converge to tracking the leader agent if μ is chosen such that $\mu_{BA} > \frac{\nu_L}{\rho_{LB}}$ and $\kappa_{BA0} \neq \pi$.

A "DKnight MagicBox II" portable Bluetooth speaker was chosen as the beacon agent's sound source. The speaker had a 10W power output, and was connected to a mobile phone, playing a FLAC encoded audio file of the 10 kHz broadband signal as discussed above.

3.5.1 Experiments

Experiments were performed with the leader using an empirically determined signature DB for localization of the beacon, and an "asymmetrical" microphone configuration, whereby both microphones were offset from a 0 degree heading by 50 degrees. The beacon used was a "beacon agent" to simulate radially emitted sound. Recall that while only κ_B is being provided by sound localization, it is in fact the only piece of beacon information required by the control law,

$$u_L = \lambda\mu\sin(\kappa_{LB}-\alpha_{LB})+(1-\lambda)\left(\mu\sin(\kappa_L-\alpha_L)+\frac{1}{\rho_L}\big(\sin\left(\kappa_L+\sin(\theta_{f_1})\right)\right), \quad (3.17)$$

and the inter-agent information is being provided by Vicon. All experiments were carried out using two agents.

A video of 3 agent CBL performed in the lab can be viewed online at [18]. During the video, the sound source is stopped for several seconds, yet the collective

is able to keep the leader from straying too far from the equilibrium. Once the sound source is reenabled, the system reaches equilibrium.

3.5.1.1 Experiment 1: Anti-Clockwise Circling Equilibrium

Initial conditions for both agents were chosen such that the leader was facing away from the beacon, and neither agent was at equilibria. Parameters were chosen so as to achieve an anti-clockwise circling equilibrium. Both agents were assigned $\alpha = \alpha_B = \frac{\pi}{4}$. The leader's beacon attention, λ, was chosen to be $\frac{2}{3}$, and μ was chosen to be 0.8. Based on these parameters we expect equilibrium values as follows,

$$\begin{aligned} \kappa_1 &= 2.356 \quad \kappa_2 = .785 \\ \kappa_B &= 1.570 \\ \rho &= 1.462 \quad \rho_B = 1.034 \end{aligned} \tag{3.18}$$

A 5 point moving average was used to smooth recovered angles, adding robustness in the event of "spurious" angle changes. Figure 3.17 shows the recovered κ_B in degrees, comparing both leader and Vicon measurements. The angle recovery is reliable enough for the system to achieve equilibrium, and converge from the non-ideal initial conditions. Performance of the angle recovery is consistent, except at a few samples after 200, where there there appears to be a vertical line of leader samples. This was due to a small break in continuity of the broad band sound source. While this interrupt in the sound source occurred, it was not substantial enough to prevent the system from achieving equilibrium - or near equilibrium at 90 degrees. It is noteworthy to mention that the leader's recovered angles are slightly shifted

right in time compared to the Vicon angles (see samples 100-200). This shift is most likely due to MATLAB processing and communication delays associated with the embedded/MATLAB combined implementation, and the 5 point moving average for angle recovery.

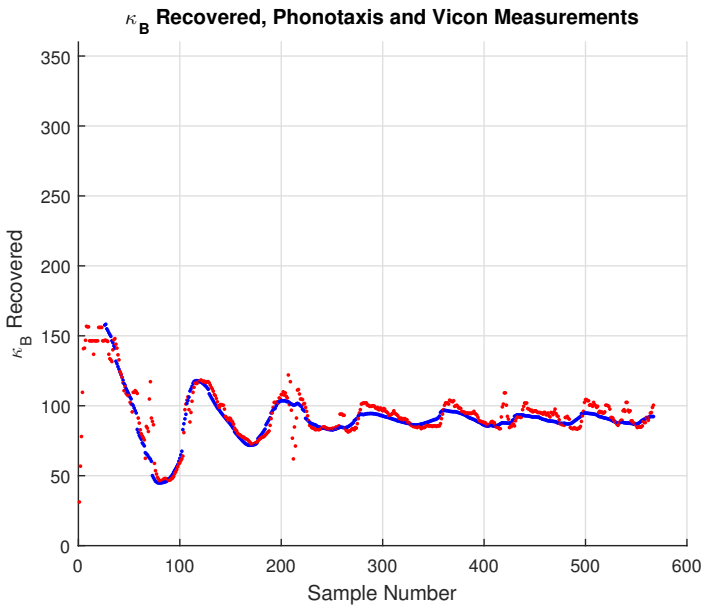

Figure 3.17: CBL-Phonotaxis Experiment 1: Plot of leader κ_B measurements (red) vs. ground-truth (Vicon based measurements, blue) in degrees.

Figure 3.18 shows the plots of $\rho, \rho_B, \kappa, \kappa_B$ for both the leader (Upsilon) and the follower (Epsilon). This figure shows convergence of all dynamics to their approximate theoretical values. There is a small amount of oscillation once the system has achieved equilibrium, due to the resolution of the leader's angle recovery and recovery error. However given the error, the equilibrium appears quite stable across all measurements.

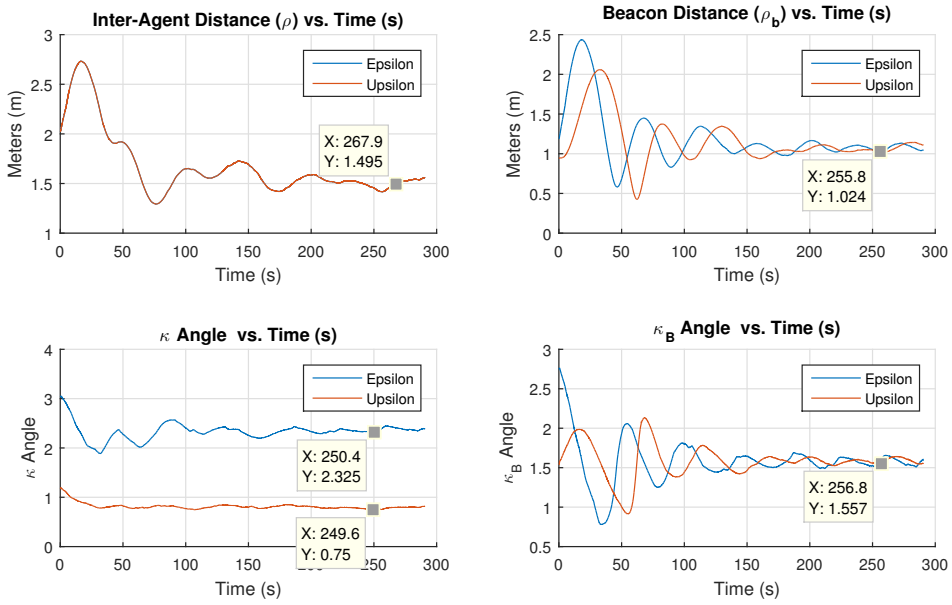

Figure 3.18: CBL-Phonotaxis Experiment 1: Plots of system dynamics vs. time for both the leader (Epsilon) and the follower (Upsilon).

Figure 3.19 shows the trajectories of the agents. Squares mark the initial positions of the agents, while circles mark the final positions. The agents clearly converge to a circling equilibria about the beacon.

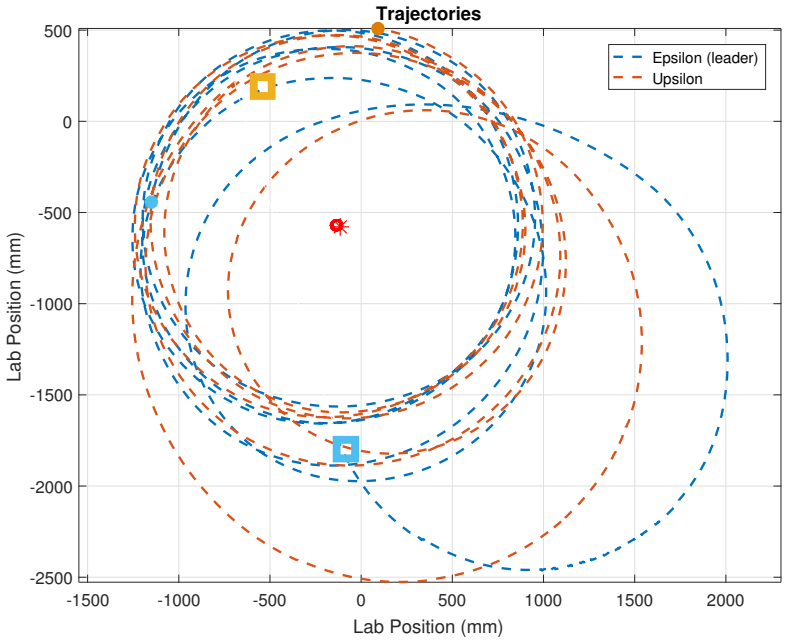

Figure 3.19: CBL-Phonotaxis Experiment 1: Plot of trajectories. Squares mark initial positions and circles mark final positions. Epsilon was designated the leader.

3.5.1.2 Experiment 2: Clockwise Circling Equilibrium

Initial conditions for both agents were again chosen in such a way that the leader was facing away from the beacon, and neither agent was at equilibria. Parameters were chosen so as to achieve a clockwise circling equilibrium. Both agents were assigned $\alpha = \alpha_B = \frac{-\pi}{4}$, the leader's beacon attention, λ was chosen to be $\frac{2}{3}$, and μ was chosen to be 0.8. Based on these parameters we expect equilibrium values

as follows,

$$\begin{aligned} \kappa_L &= 3.927 \text{ or } -2.356 \\ \kappa_B &= -1.570 \qquad\qquad \kappa_F = -0.785 \\ \rho &= 1.462 \qquad\qquad \rho_B = 1.034 \end{aligned} \tag{3.19}$$

Figure 3.17 shows the recovered κ_B in degrees, comparing both leader and Vicon measurements. The angle recovery is reliable enough to achieve equilibrium, and adjust to the non-ideal initial conditions. Initial error in this experiment was due to a break in the audio file, which then transitioned into reliable recovery. Again, there is a significant amount of recovery error around sample 350 due to a break in the audio file.

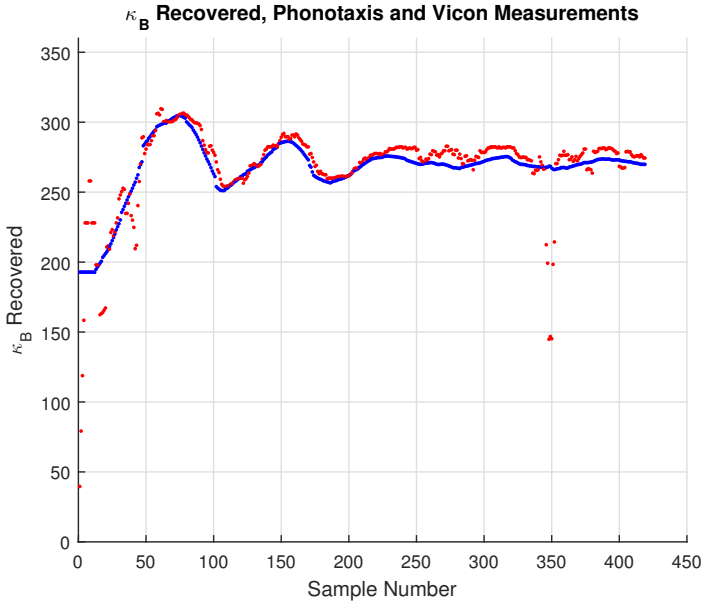

Figure 3.20: CBL-Phonotaxis Experiment 2: Plot of leader κ_B measurements (red) and Vicon measurements (blue) in degrees.

Figure 3.21 shows the plots of $\rho, \rho_B, \kappa, \kappa_B$ for both the leader (Epsilon) and the follower (Upsilon). This figure shows convergence of all dynamics to their approximate theoretical values.

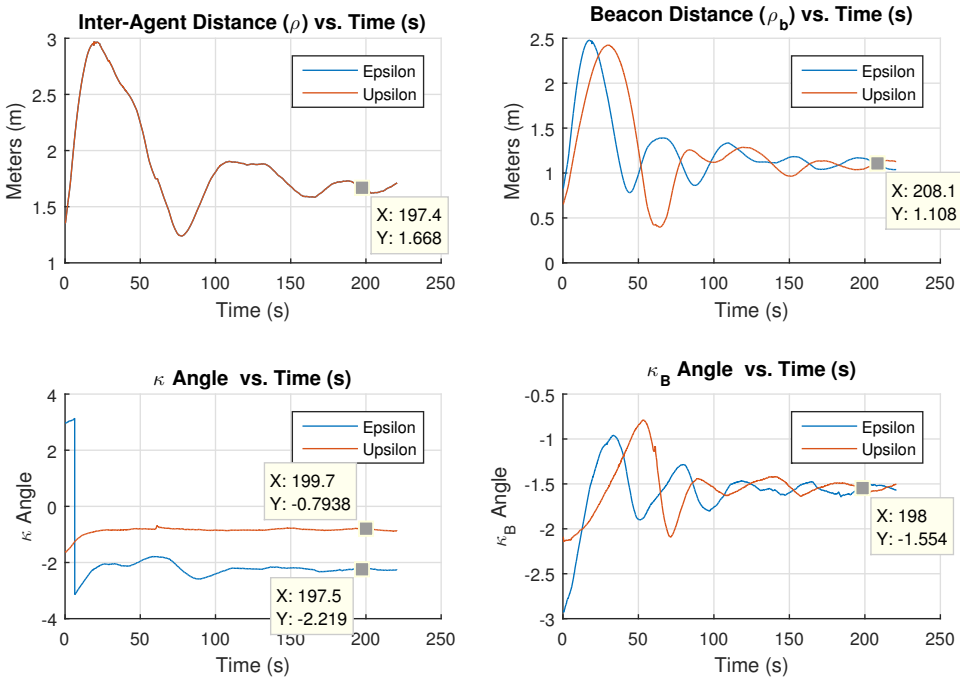

Figure 3.21: CBL-Phonotaxis Experiment 2: Plots of system dynamics vs. time for both the leader (Epsilon) and the follower (Upsilon).

Figure 3.22 shows the trajectories of the agents, with squares marking initial positions, and circles marking final positions. Again, a circling equilibrium is achieved and the system can be said to be at least (locally) stable.

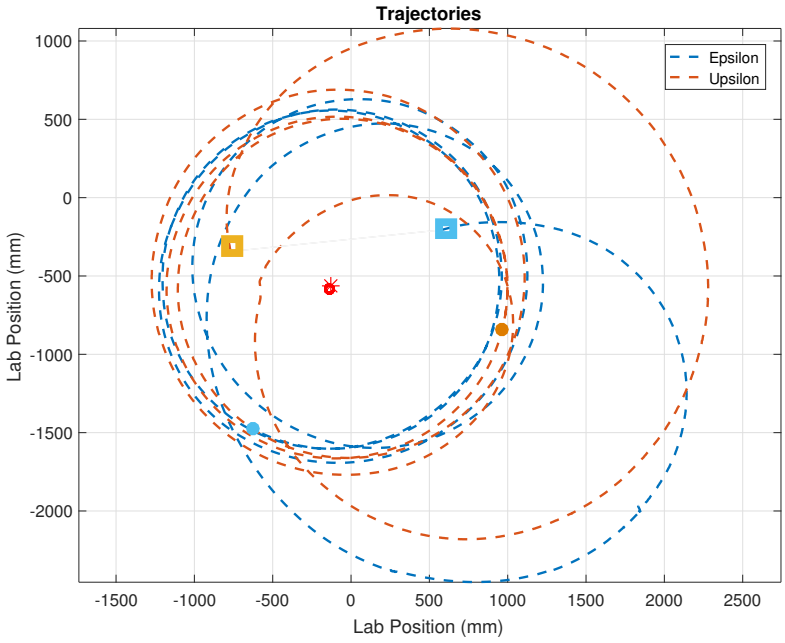

Figure 3.22: CBL-Phonotaxis Experiment 2: Plot of trajectories. Circles indicate agent final positions, squares indicate initial positions. Epsilon is the leader, Upsilon is the follower.

3.5.1.3 Experiment 3: Change of beacon location

Parameters were chosen to be the same as Experiment 1. As such, equilibrium values are the same as in (3.18). The experiment was started with the agents at near equilibrium positions. Once adequately settled at near equilibrium (at 190 seconds), the beacon was moved approximately 0.6 meters by hand. The purpose of this experiment was see if the system would still converge to equilibrium, given a change of beacon position.

Figure 3.23 shows the κ_B angle, red is the leader's recovered angle and blue is the actual Vicon angle. Again, angle recovery is good, and does not impede

convergence. Figure 3.24 shows the plots of the relevant dynamics. Again we see convergence across all parameters, though there appears to be a small oscillation in ρ by less than a tenth of a meter.

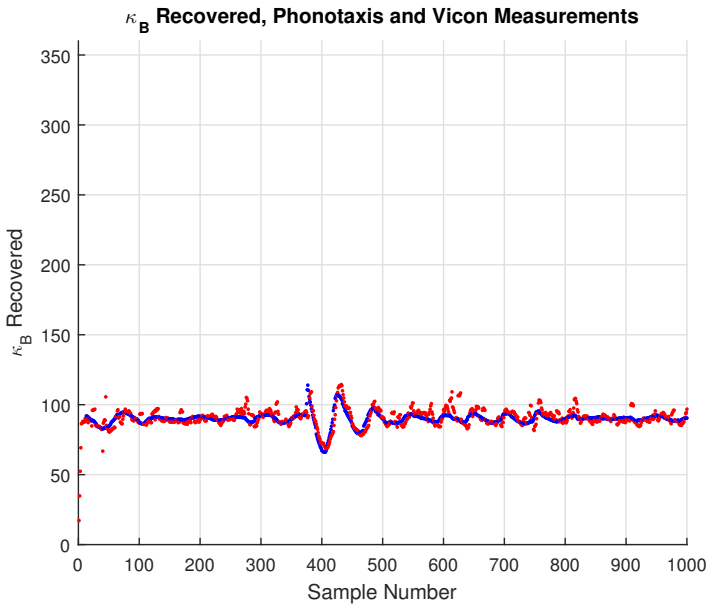

Figure 3.23: CBL-Phonotaxis Experiment 3: Plot of leader κ_B measurements (red) and Vicon measurements (blue) in degrees.

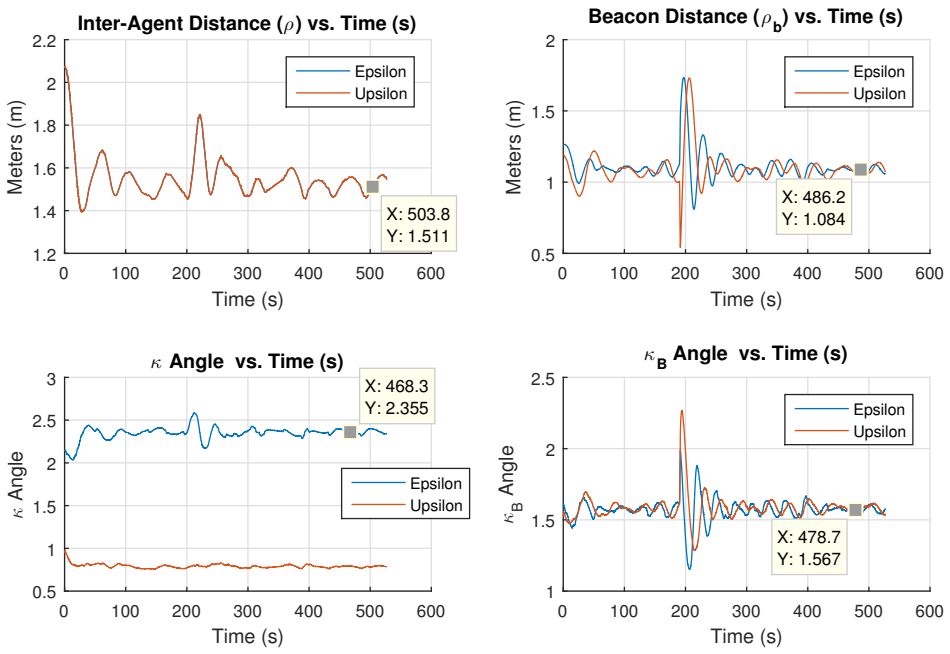

Figure 3.24: CBL-Phonotaxis Experiment 3: Plots of system dynamics vs. time for both the leader (Epsilon) and the follower (Upsilon).

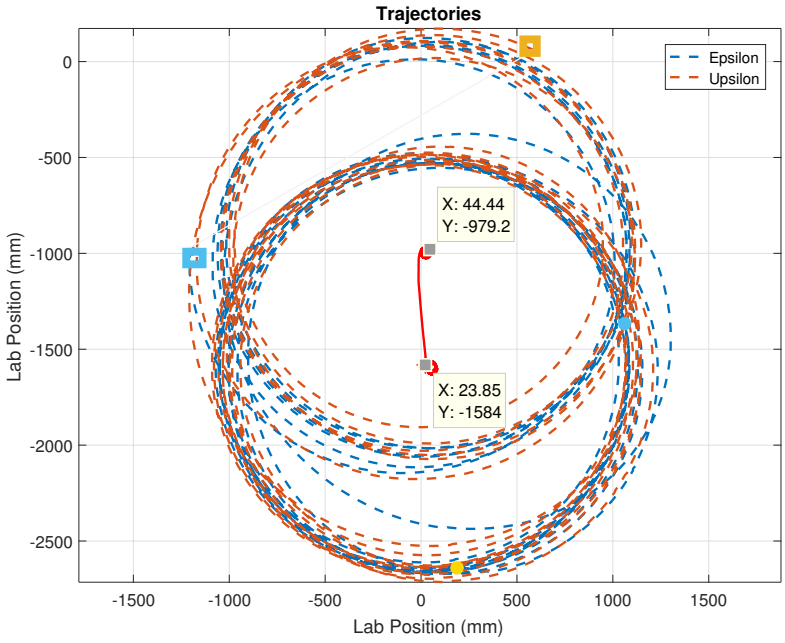

Figure 3.25: CBL-Phonotaxis Experiment 4: Trajectory plots of the leader (Epsilon) and the follower (Upsilon). Squares indicate initial positions while circles indicate final positions.

Chapter 4: A Controls Framework for ROS

4.1 Introduction

In this section we describe a simple object-oriented, extendable framework for implementing control theory experiments in ROS that is inspired by the Motion Description Language Extended (MDLe). This framework was utilized and realized in all experiments conducted in this paper. First, we will discuss some tools and concepts used in this chapter.

The Robot Operating System (ROS) is an open source set of tools for programming robots in C++ on a Linux based platform. It supports a number of different robots, positioning systems, and sensors. ROS itself is described as a toolbox rather than a framework. Because of how many options the developer has to implement control systems with ROS, structure issues can arise that directly impact software maintainability, extendibility, and portability.

The Motion Description Language Extended (MDLE) is an approach to translating control theory into software to interact effectively with the physical world. Generally speaking MDLE allows for complex, interrupt driven control, where an agent's actions are composed of a series of "atoms" consisting of a control law, sensor inputs, and timing information. In depth discussion and implementation details

of MDL and MDLe can be found in [12], [19], [13], and [20]. While specific development tools have utilized the MDLE structure in the past, none currently exist in conjunction with the ROS toolkit. Here we work toward an MDLE style framework for control using ROS.

In our framework, scalable actions, which can take the form of a control law or any movement, are compartmentalized and can be assigned in any combination to any number of agents. Each agent acts individually, determining what action to take via a "resolver" which is similar to a scheduler of an operating system. The resolver itself can be modified to consider timing information, scaling control inputs, sensor interrupts, and can even combine different actions into hybrid controls. All of these features can be utilized to provide a depth of motion planning similar to that of MDLe.

4.2 Lab Configuration

To best understand why certain decisions were made in the framework design, it is important to understand the lab configuration in which it's being used.

Robotic agents used in the lab were Pioneer 3 DX robots, a compact differential-drive robot with reversible DC motors, high-resolution motion encoders, as the experimental platform. Onboard computation is done via 32-bit Renesas SH2-7144 RISC microprocessors, including the P3-SH micro-controller with ARCOS. Upon each agent was placed a unique configuration of infrared sensitive "dots", detectable to the Vicon system.

Vicon was used as the motion capture system, providing sub-millimeter tracking accuracy of agents. The Vicon system consists of a series of infrared cameras, a digital signal processor, and a backend server. Each camera detects infrared objects within its field of view, sending the image data to the digital signal processor. The server works with the digital signal processor to compute three dimensional positions for each point in the lab's frame of reference. On the server itself, an application called Vicon Tracker or Vicon Nexus recognizes geometry of dots and combines them into objects. Each agent's unique dot geometry was used to create agent objects with associated positioning data. The positioning data is then sent over TCP IP to the controls server.

A "controls server" was used to handle all communications with robots, Vicon, and other peripherals. A C++ program was run on the controls server to listen for Vicon data on TCP IP, publishing agent positioning data to ROS. The ROS master server was also configured to run on the controls server, as well as all robot control programs. In this paper, the controls server was a Dell Precision T5600 running Ubuntu Linux. Figure 4.1 shows the overall information flow of communication between components in the lab. Within the C++ program, a software library called ARIA was used, which translates standardized remote motion commands into robot actions and manages the physical robots.

In addition to C++ programs interacting with ROS, the MATLAB ROS toolbox was used to process data, communicate to peripherals, and publish data to the ROS master server.

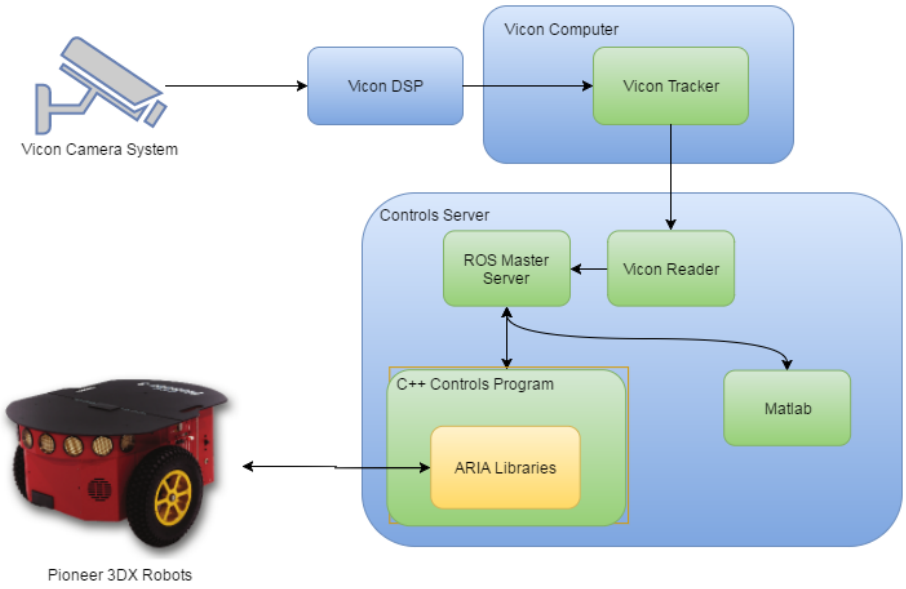

Figure 4.1: Information flow of lab components.

4.3 The Structure of a ROS Program

A control program for a collective of robots using ROS has some essential components regardless of its purpose. First the program needs to load run-time settings - specifying the number of agents, IP addresses of agents, control parameters, and any other relevant information. Then ARIA libraries are used to connect and initialize the robots over a wireless connection. Once the program has communication with the agents, they are configured to have a set of actions. The system itself must subscribe (using ROS functionality) to the published positioning data provided by an external system. After the control program is receiving data, the agents may then officially "start" and execute their control laws in a loop, with movement commands being sent via ARIA. Concurrent to the control law execution, the control program must continuously update the subscribed position data, placing it into appropriate

objects.

4.4 A Framework for ROS

The proposed framework is essentially a simple class structure that takes advantage of the native functionality of both ROS and ARIA libraries. A general information flow diagram of the framework is shown in Figure 4.2.

First the system must have a class for managing all the incoming positioning data and robot connections. We propose a "SystemData" class to do exactly this. Within the main loop of the program, the SystemData class should provide a means of managing all position data subscription, update agent position data structures within the class itself, and provide key handler functions for exiting and real time user input. Handles to the physical robots in the form of the ArRobot class should be managed within the SystemData class as well.

The ArRobot class is provided by ARIA and contains broad functionality. Each ArRobot represents one physical robot, and can manage the IP connection to the robot, issue motion commands, as well as a suite of other features. During configuration, "actions" can be added to each ArRobot individually in the form of an ArAction class provided by ARIA.

During program execution, each ArRobot object runs in its own thread. In a general sense the ArRobot and ArAction classes function as such: the ArRobot is configured and actions are added. For every control loop cycle, within each ArRobot, every assigned action "fires". Firing constitutes computing an action independent

of the other assigned actions, and passing the "desired action" to a "resolver" in the form of the ArResolver class (provided by ARIA). The resolver acts like a scheduler in an operating system, looking at each desired action, then deciding which action the robot should take for the cycle.

A key aspect to this framework is overloading the ArAction class in a standardized way to compute control laws. Each control law should be fully encompassed in an ArAction class and should act as a control-law library. Each ArAction should itself "subscribe" to only the position data structures it needs from the SystemData class. To avoid threading issues, each cycle the control ArAction should copy the relevant position data into local storage for computation.

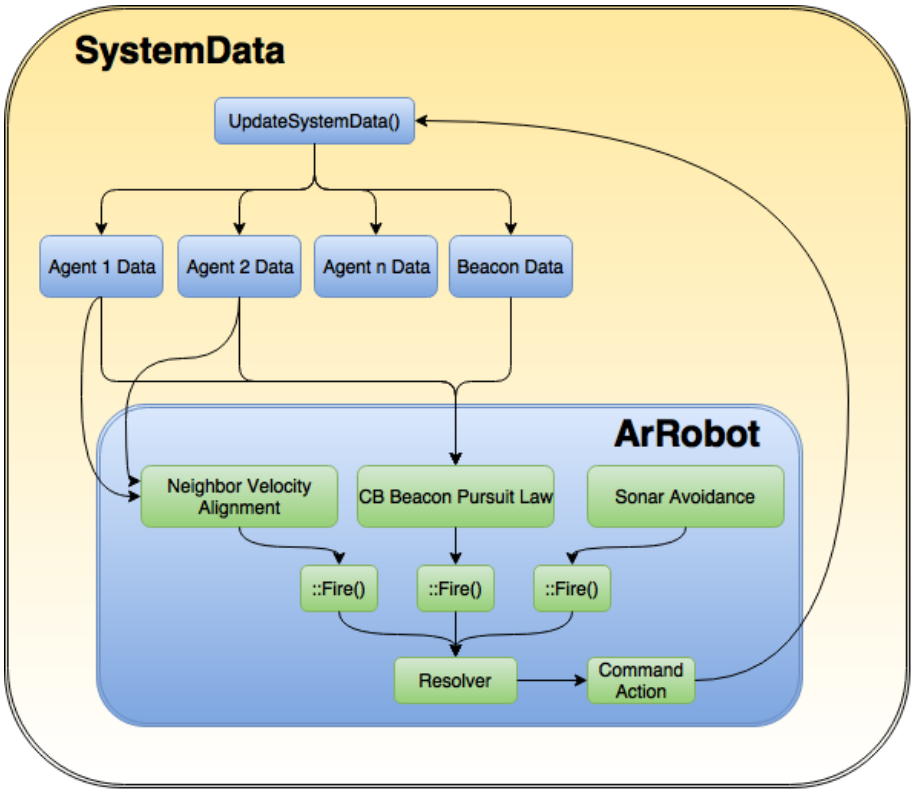

Figure 4.2: Example ROS Controls Framework Dataflow

By adding actions, we can give each robot a variety of control laws and general actions. Each action can be assigned a priority for the resolver to take into account. The resolver itself may be modified or overloaded for any or all robots to allow for interesting actions and motion planning. By doing so, this configuration easily supports sensor driven interrupt actions such as sonar based collision avoidance, time dependent behavior, and complex control laws. More interesting configurations of the resolver are also possible, allowing the resolver to return combinations of assigned actions.

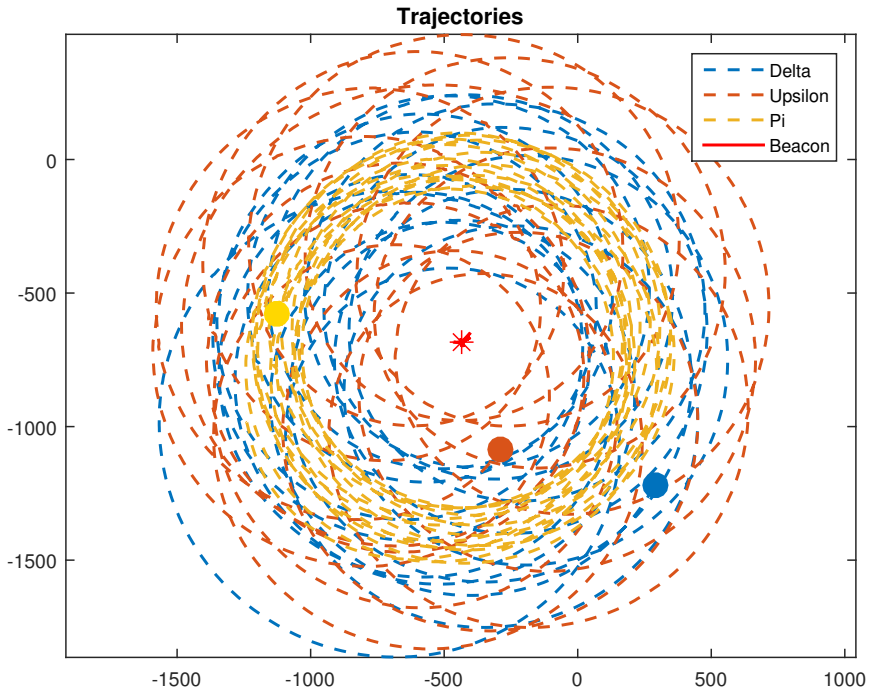

Figure 4.3: Trajectories of agents using hybrid control law within the proposed ROS framework, where two followers are performing topological velocity alignment from [1] on each other and cyclic pursuit on the leader. The leader is performing CB Beacon, paying attention to only one of the followers.

For example, a topological velocity alignment (TVA) control law from [1] can be combined with the CBL Beacon system discussed in this paper, using a hybrid resolver to decide on a robot's actions based on an assigned weight. Additionally we can have two followers both pursuing the leader, while attempting to align their velocities with each other. This modification can be done in several lines of code, while keeping the core control law actions compartmentalized and unmodified. The resulting agent trajectories for this hybrid CBL-TVA system are shown in Figure

4.3.

The proposed framework has a number of benefits, mostly relating to repeatability of experiments and maintainability of code. By compartmentalizing control laws into libraries in the form of ArActions, agents can be configured with different actions such that the actions can be easily reused in separate experiments without the need for re-implementing the control law. Thinking of control laws as individual libraries also reduces overall code complexity and greatly enhances the debugging process.

By creating a custom resolver, MDLe style behavior can be achieved. The resolver can be configured to prioritize certain actions as interrupts, as well as execute sequences of control laws or actions as if they were "atoms" in MDLe. Even more complex collective behavior can be achieved by leveraging ROS and ARIA's inherent flexibility by assigning different robots with different actions, and even different resolvers if desired. In a sense, the resolver becomes a configurable motion planner.

Chapter 5: Conclusion and Further Research

We have explored many aspects of the leader based cyclic pursuit system. First we were able to show conditions for existence of circling equilibria for an n agent CBL system. Based on desired circling geometry and circling radius, control parameters can be chosen and equilibrium values explicitly calculated from derived formulas. A linear stability analysis was performed, with formulas given to determine local stability for a set of parameters. From there, we implemented the CBL system in the two and four agent case, using the Vicon motion capture system for feedback. It was shown that the agents' behavior was as expected, and robust to different initial conditions and changes in the beacon location. Performance of the CBL system was analyzed in a series of simulations, varying both λ and α parameters, showing that the CBL system generally performed similarly, if not sometimes better than the CBB system. From here we changed gears to using sound to sense the target, rather than the Vicon$^{\text{TM}}$ motion capture system.

Taking advantage of the leader not needing to know its distance to the beacon, we explored sensing the beacon through sound. The inter-aural level difference and inter-aural phase difference algorithm was used to determine the beacon's location. We were able to show that this direction finding method could be implemented with

embedded systems, using an empirically determined signature database, a "realistic" styrofoam head, and optimal microphone placement to break front/back symmetry. In the processes of doing so, we showed that a beacon agent could track the circling leader with error asymptotically going to zero. It was then shown that the direction finding apparatus could be used in the CBL system, and was again robust to beacon position changes and different initial conditions.

Finally the implementation of the robotic experiments was discussed in detail. The lab setup was described, with information flow and dependencies addressed. A simple and standardized framework for implementing control laws was put forth, and used in all experiments in this paper. The flexibility of the framework was shown by easily combining two different controls libraries in a hybrid way. Thus a powerful, repeatable, and maintainable controls framework was demonstrated.

There are multiple ways one can proceed to expand upon this thesis for future work. First and foremost, a strong nonlinear stability result using Lyapunov analysis is desired. Given the strong empirical results and linear stability analysis, we would very much like to claim global asymptotic stability for sets of parameters. Unfortunately this result has escaped the CBB system as well, but is certainly worth the effort of exploration.

Further, it would be interesting to explore if a subset of agents had knowledge of the beacon, but perhaps with different (probabilistic) confidence levels. If multiple agents have beacon information, does their order in the circling equilibria matter? How does the amount of total beacon information in the system affect stability, robustness, and performance?

In regards to the phonotactic leader, it would be nice to implement the ILD/IPD algorithm fully on an embedded system, with angle recovery being performed on board the robot. An exploration of how little broadband sound is necessary to locate a target would be interesting to see. Can a sufficiently broadband piece of music be used to successfully locate the beacon? If the beacon only makes sound periodically, can estimation be used while not locating the beacon to successfully reach equilibrium?

For the ROS framework, it would be desirable to work towards a more complete MDLe implementation, while maintaining flexibility and simplicity of the current system. From a usability perspective, it would be advantageous to design a universal GUI for control law creation, as well as running experiments, and capturing data.

Bibliography

[1] U. Halder and B. Dey, "Biomimetic algorithms for coordinated motion: Theory and implementation," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5426–5432.

[2] USCGD14, "Vessel missing since october located adrift in pacific, unmanned," Nov 2013. [Online]. Available: http://www.uscgnews.com/go/doc/4007/1963974/vessel-missing-since-october-located-adrift-in-pacific-unmanned

[3] J. A. Marshall, M. E. Broucke, and B. A. Francis, "Formations of vehicles in cyclic pursuit," *IEEE Transactions on Automatic Control*, vol. 49, no. 11, pp. 246–251, 2004.

[4] M. Pavone and E. Frazzoli, "Decentralized policies for geometric pattern formation and path coverage," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 633–643, 2007.

[5] K. S. Galloway, E. W. Justh, and P. Krishnaprasad, "Symmetry and reduction in collectives: cyclic pursuit strategies," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, vol. 469, no. 2158, 2013.

[6] K. Galloway and B. Dey, "Station keeping through beacon-referenced cyclic pursuit," in *American Control Conference (ACC), 2015*, July 2015, pp. 4765–4770.

[7] S. Daingade, A. Sinha, A. V. Borkar, and H. Arya, "A variant of cyclic pursuit for target tracking applications: theory and implementation," *Autonomous Robots*, pp. 1–18, Online: 23 August, 2015.

[8] G. R. Mallik, S. Daingade, and A. Sinha, "Consensus based deviated cyclic pursuit for target tracking applications," in *Proceedings of the European Control Conference (ECC)*, Linz, Austria, 2015.

[9] B. D. Kevin S. Galloway, "Stability and pure shape equilibria for beacon-referenced cyclic pursuit," 2016, to appear in *Proceedings of the American Control Conference, 2016*.

[10] A. A. Handzel and P. S. Krishnaprasad, "Biomimetic sound-source localization," *IEEE Sensors Journal*, vol. 2, no. 6, pp. 607–616, Dec 2002.

[11] S. B. Andersson, A. A. Handzel, V. Shah, and P. S. Krishnaprasad, "Robot phonotaxis with dynamic sound-source localization," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, April 2004, pp. 4833–4838 Vol.5.

[12] R. W. Brockett, "Formal languages for motion description and map making," *Robotics*, pp. 181–193, 1990.

[13] V. Manikonda, P. Krishnaprasad, and J. Hendler, "Languages, behaviors, hybrid architectures, and motion control," in *Mathematical Control Theory*, J. Baillieul and J. Willems, Eds. Springer New York, 1999, pp. 199–226. [Online]. Available: http://dx.doi.org/10.1007/978-1-4612-1416-8_6

[14] R. L. Bishop, "There is more than one way to frame a curve," *The American Mathematical Monthly*, vol. 82, no. 3, pp. pp. 246–251, 1975. [Online]. Available: http://www.jstor.org/stable/2319846

[15] E. Wei, E. W. Justh, and P. Krishnaprasad, "Pursuit and an evolutionary game," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2009. [Online]. Available: http://rspa.royalsocietypublishing.org/content/early/2009/02/23/rspa.2008.0480

[16] A. A. Handzel, S. B. Andersson, M. Gebremichael, and P. S. Krishnaprasad, "A biomimetic apparatus for sound-source localization," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 6, Dec 2003, pp. 5879–5884 Vol.6.

[17] P. S. Krishnaprasad. (2015, jan) Lie brackets in control, enee661 spring 2015. [Online]. Available: http://www.ece.umd.edu/class/enee661.S2015/enee661_2011_lecture_2_b.pdf

[18] K. Miltenberger. (2016, apr) Leader-based beacon-focused cyclic pursuit using sound localization. [Online]. Available: https://www.youtube.com/watch?v=vJ-mZdeYoX0

[19] R. W. Brockett, H. Trentelman, and J. Willems, "Hybrid models for motion control systems," *Perspectives in Control*, pp. 181–193, 1993.

[20] Z. Kulis, V. Manikonda, B. Azimi-Sadjadi, and P. Ranjan, "The distributed control framework: A software infrastructure for agent-based distributed control and robotics," in *American Control Conference, 2008*, June 2008, pp. 1329–1336.