Mini-Robot Control and Distance Sensing Optimization

Susana C. Galicia, Franklin L. Nouketcha

Abstract — **Miniature, self-sufficient robotic swarms have a wide range of applications, including infrastructure monitoring and rescue mission assistance. The main goal of this project is to make a decentralized group of mini-robots perform formation following using distance sensing only. Currently, the robots cannot directly measure wheel velocity or perform position tracking due to size and power constraints. However, analyzing calibration data will enable better control and estimation of the robot's position over time using odometry. Furthermore, our control strategy requires a more precise distance sensor. We will refine sensor resolution through signal envelope interpolation. Solving these problems will result in an improved environmental awareness and optimal decision-making concerning motion control.**

Index Terms — **Distance Sensor, Goertzel Algorithm, Interpolation, Odometry, Time Difference of Arrival (TDOA)**

I. Introduction

In order for robots to be able to assemble in different formations successfully, it is essential for them to be aware of the distance that separates them from the other robots. One way through which this can be achieved is multilateration [1]. This process provides the time difference of arrival (TDOA) of two signals, which is directly proportional to the distance from transmitter to receiver. Once enough distance measurements are available, the robots will use this data to move towards a leader. This requires a robust control strategy. Thus, besides calibrating the motors correctly, we decided to incorporate an odometry system that estimates the robot's position and heading at a given moment, which will allow more control over the motion of the mini-robots. In addition, for a more accurate distance sensor, several interpolation techniques will be explored and implemented.

II. Background

A. Hardware

The current platform consists of a wireless sensor network board from Texas Instruments, the eZ430-RF2500, which includes a low-power microcontroller (the MSP430) that can be programmed using the IAR Embedded Workbench. The board also contains a radio chip, the CC2500, that enables the transmission of radio packets between boards.

The boards contain an electret omni-directional microphone (CMC-2742PBJ-A) and a piezoelectric buzzer (PS1240P02CT3) as well. The buzzer can emit a frequency in the 9-18 kHz range, but previous experiments show that a 12 kHz frequency is better picked up by the microphone.

The robots are powered by 3.7-volt rechargeable batteries. Part of this voltage is applied to each of the motors, giving the wheels certain turning rates, which in turn gives the robot an angle of rotation and a forward velocity. The mini-robots have a size of about 4 cm x 6 cm x 8 cm when mounted with all their components (Fig. 1).

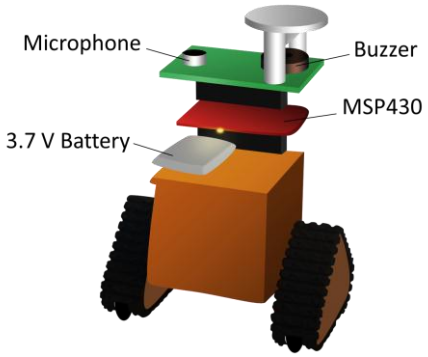

Figure 1: Current robotic platform

B. Distance Sensing

The microphone and buzzer in the boards allow transmission and reception of an audio pulse across platforms, which, along with the radio chip, are essential for calculating the time difference of arrival [2]. The process begins with the leader robot emitting a RF packet and a 12 kHz-sound pulse simultaneously. The radio packet travels at the speed of light and, upon reaching the follower bot, instructs the microphone to start listening. On the other hand, the pulse travels at the speed of sound, hence it reaches the follower several microseconds later, depending on the distance traveled from leader to follower.

From the moment the microphone is turned on ($t = 0$), the data is recorded in the form of a filtered signal, obtained through the Goertzel algorithm. This is a digital signal

processing technique that can detect specific frequencies using less CPU horsepower than the Fast Fourier Transform [3].

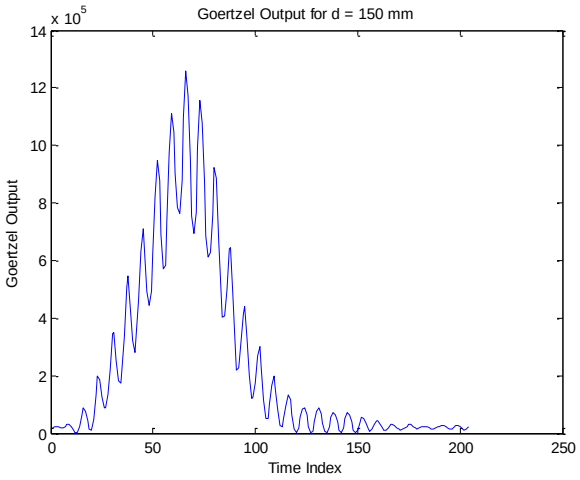

Figure 2: Goertzel algorithm output for a distance of 15 cm

The maximum peak of the waveform indicates the presence of the frequency of interest, in this case 12 kHz. For example, Fig. 2 shows that the sound pulse was detected around a time index equal to 65. Plotting several of these values against their respective actual distances helps us determine the relationship between time index and distance, fundamental for TDOA sensor calibration.

III. Signal Envelope Interpolation

Previous work on the project used the time index corresponding to the global peak of the Goertzel waveform for TDOA calculations [4]. However, there is a chance that the actual maximum of the signal envelope may occur between those peaks. For that reason, we decided to incorporate interpolation in the calculation of time index locations. Several interpolation methods [5] were tested in order to find which ones would yield the best resolutions.

A. Monomial Basis

The first attempt in calculating the interpolant employed monomial basis, which consists in constructing a Vandermonde matrix with the powers of the time indices of the three maximum peaks found in the filtered signal, as can be observed in (1). The product of the inverse of this matrix and a vector column containing the y-values of the peaks in the Goertzel output signal produces *a*, *b*, and *c*; the unknown coefficients of the second-degree polynomial that passes through all the points specified.

$$\begin{bmatrix} t_1 & t_1^2 & t_1^3 \\ t_2 & t_2^2 & t_2^3 \\ t_3 & t_3^2 & t_3^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \qquad (1)$$

B. Lagrange

Although monomial basis seemed like a promising method for interpolation, it requires matrix calculations, which consume a considerable amount of computational power on the microcontroller, as well as memory. For that reason, Lagrange interpolation was explored. Involving just simple arithmetic operations, this method intelligently picks the basis to decouple the system of linear equations, so that it is trivial to solve for the unknown coefficients.

C. Four-parabola Lagrange

As an attempt to maximize sensor resolution, a variation of the Lagrange method was tried by first interpolating between the three maximum peaks and their respective adjacent points at either side, forming three parabolas, as shown in Fig. 3. Afterwards, the maximum points of the three parabolas were calculated, and were used as the points to take into consideration to create a final interpolant. However, after testing, it turned out to have a poorer resolution than either one of the previous methods.

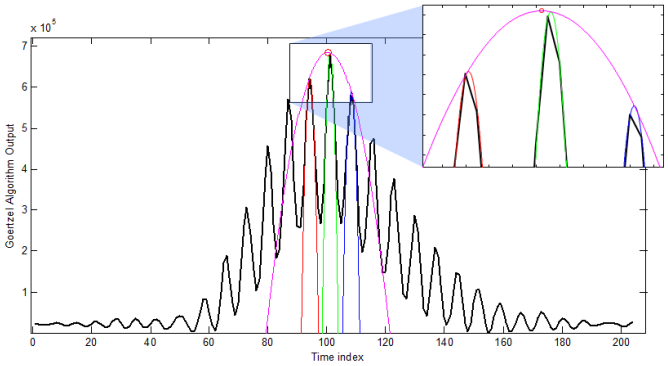

Figure 3: Four-parabola interpolation method

D. Results

Table I: Comparison between different interpolation methods

Interpolation Method	Mean Resolution
None (Raw Output)	1.06 cm
Monomial Basis	0.84 cm
Lagrange	0.86 cm
Four-parabola Lagrange	0.88 cm

Data for the output values of the Goertzel algorithm was provided for distances 5 cm, 15 cm, 30 cm, 45 cm, 60 cm and 75 cm. However, the microcontroller stopped sampling when recording the data around 75 cm, as it was out of range, making it senseless to apply interpolation. As a result, the data corresponding to 75 cm was discarded and thus not taken into account for distance sensor calibration.

After running the rest of the data through different methods of interpolation, we observed a series of trends: there was a

clear offset in the data, which may be caused by a delay in the reception of the signal; the resolution had improved by as much as 20%, and data for 5 cm slightly deviated from the rest of the measurements.

IV. Robot Kinematics

Unlike cars, robots do not have a steering that constantly controls their direction. It is only possible to regulate the amount of power that goes to each of the two motors using pulse width modulation (PWM). Depending on the average voltage applied to them, each motor will turn at a specific rate (Fig. 4). The average voltage applied to each motor is controlled by the microcontroller by varying the duty cycle (between 0% and 100%, where 0 means that no voltage is applied at all, while 100 means that the full voltage of the battery is applied to the motor).

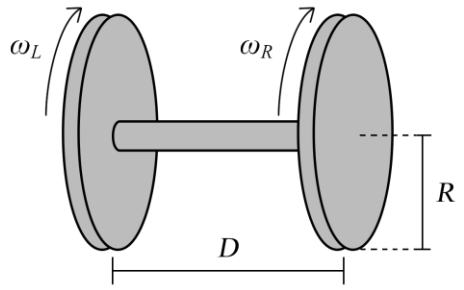

Figure 4: Wheel description

The velocities of the two wheels determine the robot's trajectory in Euclidean space (x, y, θ). An ideal robot will move either in a straight or circular trajectory. If the two pulses are equal, the ideal robot will move straight forward; however, if one pulse is greater, the corresponding wheel will move faster. A wheel turning faster will result in the robot making a turn, engaging the robot in a circular path (Fig. 5).

$$v = R\frac{(\omega_L + \omega_R)}{2} \quad (2)$$

$$\omega = R\frac{(\omega_L - \omega_R)}{D} \quad (3)$$

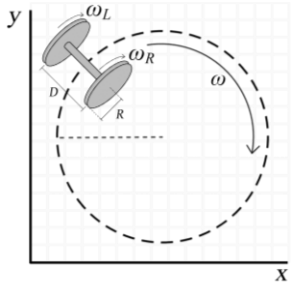

Figure 5: Robot trajectory

V. Motor Calibration

After sending a set of pulses to the micro-robot, one important piece of information to know is its corresponding velocities (υ, ω) or twist. The twist is a vector that includes both the forward and angular velocity of the robot. We can find the twist of the robot if we know the linear map that relates motor duty cycles to resulting robot twists. Calibrating a robot means finding the relationship between the power sent to each wheel and the resulting velocities.

$$PWM = \begin{bmatrix} P_L \\ P_R \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = A\begin{bmatrix} P_L \\ P_R \end{bmatrix} \quad (5) \qquad A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \quad (6)$$

Calibrating the robot is to find the 2 x 2 matrix (6) that will enable the prediction of the twist from the power modulation.

A. *Steps for calibrating the robot*

1. Download the calibration code to the micro-robot
2. Assign a power modulation to the robot and track its trajectory using the camera vision
3. Plot the trajectory of the robot
4. Find the angular and forward velocity of the robot in its trajectory
6. Find the forward velocity
7. Repeat steps 5 to 6 times using different power modulations
8. Rearrange the resulting velocities in a matrix and the pulse in another so that A can be solved

B. *Getting the data for the calibration*

To start the experiment, it is important to cover the robot with a piece of white paper with black tape on the top so that the robot can be seen as one entity; moreover, it will be used as reference for angle measurement. After the set-up, a specific command is assigned to the robot. For example, in the case of 'f 40,30,' the left motor is receiving a pulse or duty cycle of 40, meaning that 40% of the total voltage flows to the left wheel, while 30% flows to the right. From that trajectory, it is possible to find the twist by various methods.

C. *Camera distance vs. real distance*

The vision system does not give the x and y position in units of length. The time is tracked in nanoseconds, the angle is in radians, but the coordinates x and y are given by the camera in term of pixels. To convert the camera distance to real-world distance, one needs to place two objects on the experiment table so that they have the same x or y coordinate. If they have the same x coordinate, this means that the distance between them corresponds to the difference in y in term of pixels. The real distance can be measured directly from the table and the relationship between the pixels and a distance unit can be drawn.

Though this issue of distance seems to be solved, there is still a problem that affects data collection. When objects are

near the periphery of the camera's view, they appear smaller due to the camera's perspective. For the same distance, the camera will show different number of pixels.

D. Finding the twist from the trajectory

Circular regression – One way to find the twist from the trajectory of the robot is by doing a circular regression. In this case, some points are selected from the trajectory; those points are then used to find the equation of the circle that will best fit them. Since the camera also tracks the angle of the robot, the angular velocity can be estimated by taking a discrete time derivative. To find the forward velocity of the robot, the only step left is to multiply the angular velocity by the radius of the fitting circle.

Point-by-point method using odometry – In this case, the angular velocity is found the same way as described in the circular regression. As far as the forward velocity is concerned, we used (7) and (8), from robot kinematic models [8]:

$$\Delta D = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\sin c(\frac{\Delta\theta}{2})} \qquad (7)$$

$$v = \frac{\Delta D}{\Delta t} \qquad (8)$$

E. Finding the calibration constant A

Once the twist is computed for each time step, it is possible to use the initial pulse to find the constant of calibration *A*. Having *A* allows finding the twist from a given power modulation; from the twist it is possible to control the overall motion of the robot.

The constant calibration A is found by using (5). However, each experiment has a tendency to generate a different A. To solve this problem, all the twists resulting from the experiment are collected and rearranged in a single matrix in a way that they can all be used to find the matrix A. This is a way to optimize A taking into account several experiments. The matrices below show a way to reorganize the data for three experiments:

$$\begin{aligned} \upsilon_1 &= A_{1,1}*P_{L1} + A_{1,2}*P_{R1} \\ \upsilon_2 &= A_{1,1}*P_{L2} + A_{1,2}*P_{R2} \\ \upsilon_3 &= A_{1,1}*P_{L3} + A_{1,2}*P_{R3} \end{aligned} \qquad (9)$$

$$\begin{aligned} \omega_1 &= A_{2,1}*P_{L1} + A_{2,2}*P_{R1} \\ \omega_2 &= A_{2,1}*P_{L2} + A_{2,2}*P_{R2} \\ \omega_3 &= A_{2,1}*P_{L3} + A_{2,2}*P_{R3} \end{aligned} \qquad (10)$$

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \\ \upsilon_3 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} P_{L1} & P_{R1} & 0 & 0 \\ P_{L2} & P_{R2} & 0 & 0 \\ P_{L3} & P_{R3} & 0 & 0 \\ 0 & 0 & P_{L1} & P_{R1} \\ 0 & 0 & P_{L2} & P_{R2} \\ 0 & 0 & P_{L3} & P_{R3} \end{bmatrix} \begin{bmatrix} A_{1,1} \\ A_{1,2} \\ A_{2,1} \\ A_{2,2} \end{bmatrix} \qquad (11)$$

F. Results

The following table shows the normalized means for 4 data sets using the two methods of calibration:

Table II: Normalized means for 4 sets of experiments

Motor Comand(PL, PR)	Calibration Technique	CV (RMSE) (normalized wrt mean)	
		v	ω
(25,25)	SVD: Circular fit	0.133	-4.1686
	Odometry	0.136	17.49
(40,30)	SVD: Circular fit	0.104	0.209
	Odometry	0.102	1.085
(50,00)	SVD: Circular fit	0.353	0.382
	Odometry	0.35	0.504
(40,20)	SVD: Circular fit	0.104	0.2
	Odometry	0.102	1.08

VI. Odometry

A. Differential equation modeling the robot trajectory

It is assumed that the robot will only have a circular uniform motion; therefore, we will assume that its motion will be predicted by the following set of differential equations:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ \omega \end{bmatrix} \qquad (12)$$

Solving those equations will enable the prediction of the trajectory of the robot in Cartesian coordinates.

After applying a piecewise constant motor command for a specific time, it will be possible to predict the final position of the robot with respect to its initial position, therefore having more control over it. Moreover, it will allow the selection of the right pulse needed by the robot to reach a desirable final location.

Assuming that at t = 0, $\theta = \theta_0$, $x = x_0$ and $y = y_0$, the solution for the system of differential equations is:

$$\begin{aligned} \theta &= \omega t + \theta_0 \\ x &= \frac{v}{\omega}\sin(\omega t + \theta_0) + k_1 \\ y &= -\frac{v}{\omega}\cos(\omega t + \theta_0) + k_2 \end{aligned} \qquad (13)$$

Substituting the initial conditions:

$$\begin{aligned} \theta &= \omega t + \theta_0 \\ x &= \frac{v}{\omega}\left[\sin(\omega t + \theta_0) - \sin(\theta_0)\right] + x_0 \\ y &= -\frac{v}{\omega}\left[\cos(\theta_0) - \cos(\omega t + \theta_0)\right] + y_0 \end{aligned} \qquad (14)$$

When ω =0, practical issues arise as the equation presents a hole. However, there is no singularity, because the limit of those equations when ω approaches 0 does exist. When the robot is moving straight forward, there is no angular speed and the equation of motion can be simply modeled as:

$$\begin{aligned} \theta &= \theta_0 \\ x &= v\sin(\theta_0)t + x_0 \\ y &= y_0 - v\cos(\theta_0)t \end{aligned} \qquad (15)$$

B. *Predicting the trajectory of the robot*

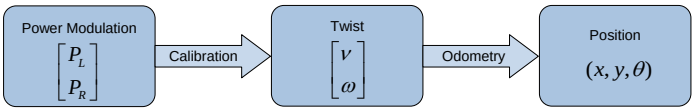

Figure 6: Flow chart for odometry and calibration

After calibrating the robot, we know the relationship between the power modulation and the resulting forward and rotational speed of the robot. A specific pulse will generate a unique twist; from a twist, it is possible to use odometry to predict the trajectory of the robot if initial conditions are known.

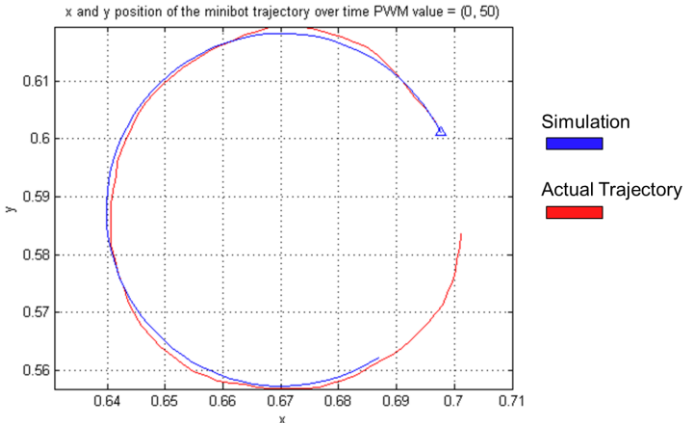

Figure 7: Predicted and experimental trajectory of the mini-robot

VII. Acknowledgment

We would like to thank Dr. Pamela Abshire, Michael Kuhlman, George Sineriz, Gary Sullivan and the members of the Antbots team for their mentoring and support. Special thanks to the MERIT BIEN program as well, and to the National Science Foundation (OCI Award #1063035) for funding this research project.

VIII. References

[1] SRA International, Inc. *Technologies: Multilateration*, http://www.multilateration.com/surveillance/multilateration.html

[2] Howard Fan, H. and Yan, Chunpeng. *Asynchronous Differential TDOA for Sensor Self-Localization*, ICASSP 2007.

[3] Banks, Kevin. *The Goertzel Algorithm*. EE Times, http://www.eetimes.com/design/embedded/4024443/The-Goertzel-Algorithm

[4] Perkins et al. *Distance Sensing for Mini-robots: RSSI vs. TDOA*.

[5] Heath, Michael T. 1997. *Scientific Computing: An Introductory Survey*. McGraw-Hill.

[6] Lei, Lydia L. and Perkins, Christopher L. *Control and Coordination of Micro-Robots*, MERIT BIEN 2010 Final Report, August 2010.

[7] Turner, John A. and Wilson, Ashia C. *Coordination of Realistic Decentralized Control Systems*, IEEE, MERIT BIEN 2009 Final Report, August 2009.

[8] Ming Wang, C. (1988), *Location Estimation and Uncertainty Analysis for Mobile Robots*, General Motors Research Laboratories, Warren, Michigan.

[9] Zhou, Yuchen. *Report on Calibrating Wallebots*, ENEE499, University of Maryland.