Two-Agent Leader/Follower Formation Control Through Challenging Maneuvers and Limited Sensing Environments

James Dunn (jkdunn@bu.edu)
Boston University ME/SE 740 Project Final Report, May 2019

***Abstract* - The ability for a "following" robotic agent to autonomously follow another "lead" agent has many real-world applications. These include the ability of self-driving cars to maintain a safe following distance from the vehicle ahead of them and anti-missile seekers for missile defense. The scenario can be thought of as a two-agent formation maintenance problem where only the following agent is autonomously controlled.**

In any hardware implementation of a following agent, there are various challenges that present themselves and complicate what is otherwise an algorithmically straightforward problem. For real-world applications, a desire for low-SWAP (size, weight, and power) and low-cost hardware is common and presents challenges to the control software that a more capable system would not have to deal with. This project investigates and implements solutions to a handful of these challenges. A test hardware implementation with a basic COTS RC truck as the lead agent, and a custom-built low-cost, low-SWAP system running a Raspberry Pi Zero W and a single RGB camera on a hobbyist car chassis as the following agent is used. Challenges addressed include dynamic lighting conditions, limited processing power, low framerate, low resolution, and sporadic sensing outages.

Control code available for download at http://github.com/jimmykdunn/raspi_car

I. INTRODUCTION

In an ideal scenario, the fully-autonomous following agent in a leader-follower formation control problem follows a simple algorithm:

1. Detect the lead agent in each temporal frame of data coming from the sensor
2. Adjust steering and/or propulsion to keep the desired distance and angle behind the lead agent.

In any real-world implementation, the following agent is presented with various challenges that place restrictions on this simple but optimal 2-step algorithm. In particular, minimizing the size, weight, power consumption, and cost of the following agent robots is often a primary concern.

Most of the robotic formation control literature deals with optimizing control laws and implementing them either in simulation or on highly-capable robots. Multidirectional cameras, LIDAR, communication with the lead agent, et cetera are commonplace and make the problem easier to solve. Low-SWAP and low-cost systems, particularly those in which the lead agent is noncooperative, must operate under much more restrictive design limitations.

Low-SWAP and low-cost systems introduce a new set of problems, among them poor-dynamic range cameras, coarser resolution, limited processing power, slow framerates, small fields of view (FOV), and sporadic sensing outages. This projects seeks to overcome the design limitations brought on by using a single $15 RGB camera run by a $10 Raspberry Pi Zero W. All processing on the following agent is done onboard, and there is no communication with the lead agent, making the following agent truly autonomous.

By trading off complex image processing for higher framerate and by running range and angle measurements through a Kalman filter, we are able to make the following agent robust to the limitations brought on by the low-SWAP, low-cost design constraints.

II. RELATED WORK

The topic of formation control for teams of robots has received much attention in the last decade. Oh, Park, and Ahn (2015) [1] provide a good survey of the topic of formation control in the literature, focusing particularly on the methods by which different formations can be achieved. We deal with leader-follower formation control: a sub-branch of formation control in which a single agent is designated the "leader", and the other agent(s) are tasked to maintain a formation while following the leader's position. Specifically, we deal with the case of a single follower attempting to maintain a specified distance immediately behind the leader. Additionally, the follower is completely autonomous; it has no communication with the leader and no path has been agreed to beforehand. Of note, the leader in this scenario does not have to be cooperative with the follower.

Mariottini et al (2009) [2] implement a team of unicycle robots and come up with a novel way of constraining their motion to maintain observability. Their implementation uses only cameras, but notably, the cameras are omnidirectional, and the leader communicates directly with the follower(s). Consolini et al (2008) [3] develop a similar set of constraints, and they assume that measurements are made only by the following agents. Their implementation is only in simulation, so no strict limits are placed on the ability of the following agent's to sense or process data. Hassan, Aljuwaiser, and Badr (2018) [4] develop a control strategy in two components; one for tracking a desired trajectory, and one for limiting overshoots. However, like [3], their implementation is only in simulation.

Hogg et al (2001) [5] implement leader/follower formation control system for "backpack" sized robots that rely only on various combinations of GPS/INS, track odometry, LIDAR, magnetometers, inclinometers, and stereo-vision cameras. They implement a Kalman filter to smooth measurements, much like in this project. They investigate the utility of those various sensors, but do not limit themselves to just the camera sensor.

Cruz-Morales et al (2017) [6] have perhaps the most similar implementation to this project. They focus on the discrete-time aspect of their implementation, which uses only a Kinect stereo camera sensor onboard Pioneer 3-DX wheeled robots. Of note, they deal with the details of latency in feedback controls in an explicit manner. Their implementation however uses comparatively heavy laptops onboard each robot, enabling 30fps and 640x480 resolution: much higher quality than the Raspberry Pi Zero W in our implementation is capable of.

Vankadari, Das, and Kumar (2017) [7] present a leader-follower implementation using Parrot AR quadrotor drones. Notably, they use onboard Raspberry Pi's and visual cameras, along with an Aruco fiducial marker on the lead drone. The resulting package is lightweight and inexpensive. The primary difference between their approach and that of this project is of course that they use holonomic quadrotors, whereas we use non-holonomic wheeled vehicles.

III. EXPERIMENTAL SETUP

Our experimental setup consists of two small electric vehicles: one lead agent and one following agent. The hardware setup of each is described in detail below:

Lead Agent

The lead agent is an inexpensive ($10 USD) manually-controlled toy remote control truck (fig 1). It has been outfitted with one of two brightly colored spheres acting as a fiducial for the following agent to track. The fiducial spheres used for this implementation are a 7cm diameter dog toy mimicking a tennis ball, and a 6.3cm diameter green toy ball. These were chosen for their color, near-Lambertian reflectivity surfaces, and isotropic appearance. A small LED penlight is placed against the fiducial sphere so it can be seen in the following agent's camera in poor ambient lighting conditions (see Figure 2).

Removing the fiducial sphere would require implementing a much more detailed computer vision algorithm to detect the location of the lead agent, which is beyond the scope of this project. The use of

a separate fiducial sphere for tracking allows the following agent to follow anything that the sphere can be placed onto without any software changes or algorithm training. It was particularly helpful during development to remove the sphere from the lead agent and simply have it in-hand.

Figure 1: Lead agent (toy RC truck) with bright yellow fiducial sphere attached.

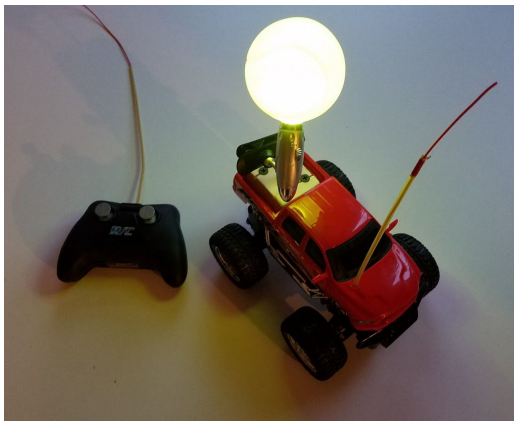


Figure 2: Lead agent with optional LED penlight illuminating the fiducial sphere.

This lead agent vehicle platform is suboptimal for this problem given its binary drive motor and binary steering servo-motor controls. This made driving the lead vehicle in a manner that the following agent was physically capable of mimicking cumbersome.

Following Agent

The autonomous follower agent was built from inexpensive COTS (commercial off-the-shelf) components and is a legitimately low-SWAP low-cost device. A low-SWAP and low-cost processor, the Raspberry Pi Zero W, is the most optimal processor on the market for this regime. A single Raspberry Pi-optimized 3-channel (RGB) visible band camera is mounted to the front of the car facing forwards. The cost of the parts for the following agent was approximately $90.

The entire vehicle measures 25 cm by 17 cm and weighs approximately 600 grams, with a large fraction of that weight being an unnecessarily large 5V rechargeable battery that was chosen for power capacity. A smaller and lighter battery could easily be used in its place.

The following agent's wheels rotate about fixed axles, so steering is accomplished by commanding differential rotation between the right and left wheels, and allowing slippage on the slower-rotating wheels. This makes the motion-to-wheelspin relationship highly dependent on the driving surface. Laminate flooring was used for all testing.

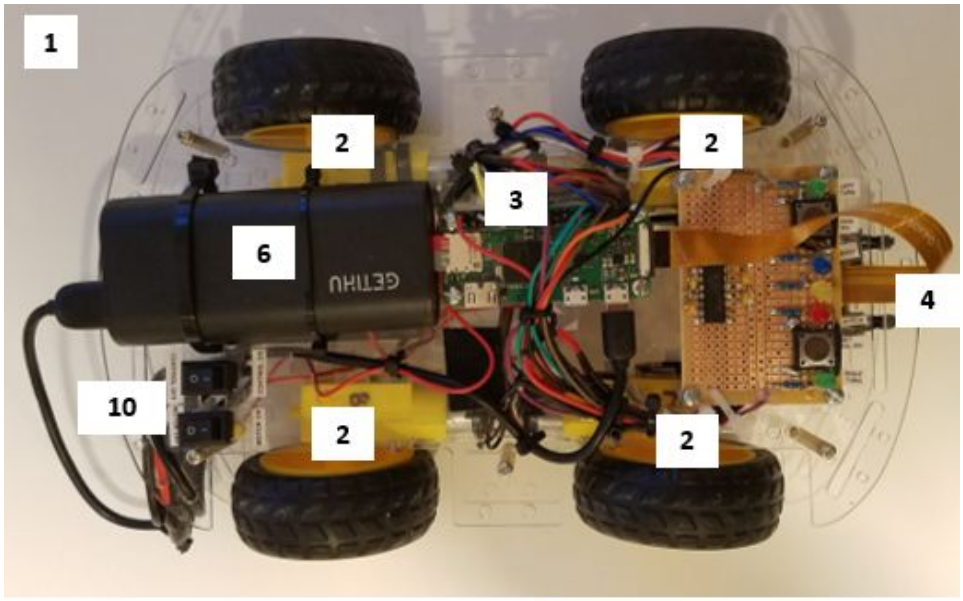
The following agent's motors are driven via a L293D motor driver, which use a pulse-width-modulated (PWM) drive voltage to vary their rotation speed. If we multiply the duty factor of the PWM signal, $d \in [0, 1]$, by the voltage applied to the L293D, we get the time-average voltage applied to the motors. The rotation speed of the motors is thus proportional to the PWM duty factor. Note that the L293D has some internal impedance, which caps its output voltage at approximately 6V when 9V is applied (determined by direct measurement with a potentiometer). Note also that below 3V, the motors cannot overcome their own internal friction, so applied voltages less than 3V will not rotate the wheels.

Measured characteristics of the following agent are shown below for reference. All motion measurements were made on a hard laminate floor; given the slip-steer configuration, these measurements are valid only for the floor surface used in testing. These are used later in Section V to calibrate feedback controls:

- The following agent's camera has a 50° field of view in the horizontal direction. This provides a challenge for the controller since the lead agent

can easily leave the field of view with only a modest-speed turn.

- The speed of the following agent when the applied PWM duty to all four motors is 1.0 is 0.52 m/s. This represents the maximum speed of the following agent
- The speed of the following agent when the applied PWM duty to all four motors is 0.6 is 0.31 m/s. This is strong evidence that PWM results in a linear duty-to-speed relationship when duty >= 0.6.
- Below about 0.6 PWM duty, the following agent generally does not move. At 0.6 PWM duty, the applied voltage to the motors is approximately 3.6V. This is roughly consistent with the motor manufacturer's stated 3.0V minimum voltage.
- When the right wheels have 1.0 PWM duty and the left wheels have 0 PWM duty, the following agent rotates counterclockwise more-or-less in place at a rate of 130°/second
- When the left wheels have 1.0 PWM duty and the right wheels have 0 PWM duty, the following agent rotates clockwise more-or-less in place at a rate of 134°/second.
- The motion of the following agent under intermediate left-right wheel duty ratios proved somewhat inconsistent and difficult to reliably measure with the limited instrumentation available for this project (tape measure and a stopwatch). This could be more accurately measured in a robotics lab, which would lead to a more reliable feedback control solution and thus improved formation maintenance.
- The lead agent was outfitted with a handful of different fiducial spheres in testing. The two most reliably detectable given the test environments were a 7cm diameter yellow ball and a 6.3cm diameter green ball. When placed 1 meter from the following agent's camera, these took up 0.59% and 0.48% of the following agent's camera's angular field of view, respectively.



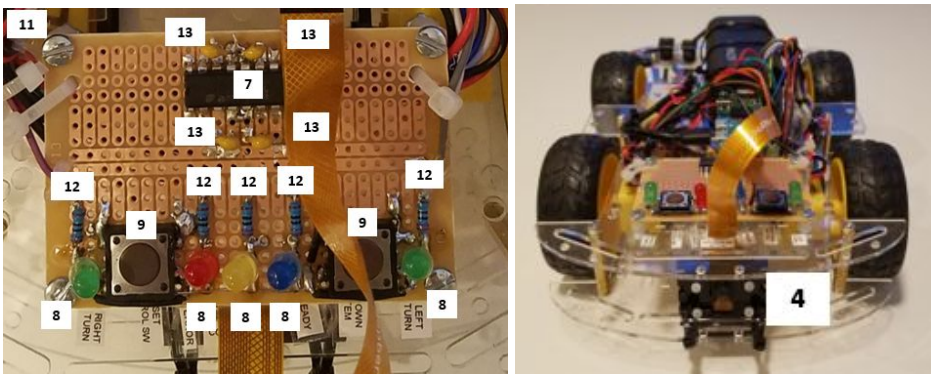


Figure 3: Photographs of the following agent. Top: top view. Bottom left: zoom-in of the circuit board. Bottom right: front view showing the location of the camera.

1. Plastic hobbyist car chassis (Perseids DIY Smart Car Chassis Kit)
2. 4x electric motors + wheels (included with the Perseids kit)
3. Raspberry Pi Zero W running Raspbian OS on a 64GB SD card
4. Raspberry Pi compatible visual-band RGB camera and mount
5. 9V alkaline battery (supplies motor drive power) (not visible underneath GETIHU battery)
6. 5V rechargeable battery (GETIHU) (supplies Raspberry Pi)
7. L293D motor driver (H bridge and MOSFET IC component)
8. 5x colored LEDs for indicating status to the user: left turn signal, right turn signal, error, software ready, controller active
9. 2x buttons (for starting/stopping the control software and gracefully shutting down the Raspberry Pi)
10. 2x rocker switches (controlling power to the Raspberry Pi and power to the motors)
11. Circuit board holding the buttons, motor driver, LEDs, series resistors, and capacitors
12. 5x 200Ω resistors in series with the LEDs
13. 4x 1μF capacitors across the motor driver outputs (for back EMF suppression)
14. Assorted wires, cables, heatshrink tube, cable ties, bolts, nuts, and standoff washers
15. Standard electrical solder and hot-melt glue

IV. FEEDBACK CONTROL LAW

The canonical state feedback control equation in control theory for discrete linear systems is:

$$x_{t+dt} = Ax_t + Bu_t \qquad [1]$$

For our experimental setup, we use a 4-dimensional state vector, x_t, with the following elements:

$$x_t = (r_t, \theta_t, \frac{dr_t}{dt}, \frac{d\theta_t}{dt})^T \qquad [2]$$

Where:

- r_t is the range from the following agent (specifically its camera) to the lead agent's fiducial sphere, measured in meters
- θ_t is the angle to the lead agent's fiducial sphere, in degrees clockwise from straight-ahead
- $\frac{dr_t}{dt}$ is the range rate, or the time derivative of r_t
- $\frac{d\theta_t}{dt}$ is the angle rate, or the time derivative of θ_t

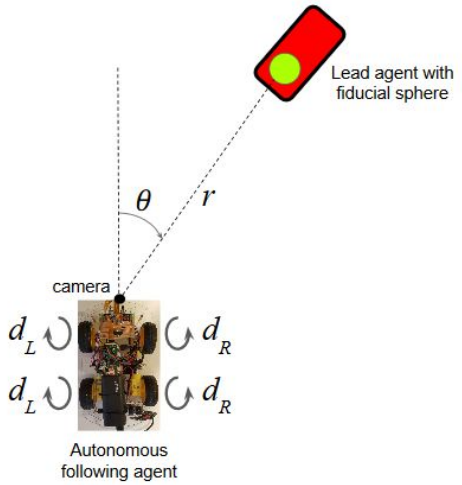


Figure 4: top-down diagram of the experimental setup

The "coasting" matrix in equation [1], A, is the standard first-orderization, where we can treat most of the off-diagonal terms as zero to first order since our timesteps dt are small relative to the motion of the agents:

$$A = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[3]

Our control vector, u, is a 2-element vector whose entries are the duty of the pulse-width-modulated (PWM) voltage applied across the left and right motors:

$$u_t = (d_R, d_L)^T \quad [4]$$

Where:

- d_R is the PWM duty applied to the right wheels (when $d_R > d_L$ the vehicle turns left)
- d_L is the PWM duty applied to the left wheels (when $d_L > d_R$ the vehicle turns right)

The time subscripts are dropped from d_R and d_L for readability, but both do depend upon time.

The DC motor and wheel assemblies on the autonomous following agent have significant friction and the speeds that we deal with are relatively slow. As a consequence, there is essentially zero "coasting" - if the voltage applied to the motors is zero, then the car stops in effect immediately. Accordingly, this makes the Bu_t product zero in the $\frac{dr_t}{dt}$ and $\frac{d\theta_t}{dt}$ entries.

We run into a conflict with equation [1], because the relationship between u_t and x_{t+dt} is decidedly nonlinear. Specifically, when the wheels on either side rotate at appreciably different rates, i.e. when $|d_R - d_L| > 0$, the vehicle's range rate decreases more or less exponentially. In simpler terms, instead of the PWM duty causing forward motion, it causes rotation. As such, the r_{t+dt} term of x_{t+dt} is best modeled as:

$$u_r = -\tfrac{1}{2} dt\, s_R (d_R + d_L) e^{-|d_R - d_L| s_d} \quad [5]$$

Where

- s_R is the speed of the vehicle when both $d_R = d_L = 1$. This is measured at 0.52 $\frac{m}{s}$
- s_d is the exponential decay in range rate as a result of differential duty between the left and right wheels. This is approximated as 5 (unitless) on the test floor surface.

Notice that the expression for u_r in equation [5] is negative, because positive PWM duty (forward motion) causes a *decrease* in the range from the following agent to the lead agent.

The remaining control term for the angular response from the applied PWM duty values takes a simple form:

$$u_\theta = dt\, s_\theta (d_R - d_L) \quad [6]$$

Where:

- s_θ is the rotation rate of the vehicle with the right wheels at full throttle, and the left wheels stopped. Measured at 132 $\frac{deg}{s}$ on the test floor surface.

Note from equation [6] that when the duty applied to the right wheels is greater than the duty applied to the right wheels, i.e. $d_R > d_L$, the angle term θ_{t+dt} *increases*. This is because $d_R > d_L$ causes the vehicle to turn left and thus the clockwise angle to the leader to *increase*.

Since equation [5] is nonlinear, we cannot represent our discrete state feedback control equation [1] in the linear form. Instead, we use:

$$x_{t+dt} = Ax_t + U_t \quad [7]$$

We assemble U_t from equations [5] and [6], along with the aforementioned mentioned lack of coasting in the applied controls giving zero in the velocity entries:

$$U_t = \begin{cases} (u_r, u_\theta, 0, 0)^T & \text{if } max(d_R, d_L) \geq 0.6 \\ (0, 0, 0, 0)^T & \text{if } max(d_R, d_L) < 0.6 \end{cases}$$

[8]

The DC motors on the following agent do not turn when less than 0.6 PWM duty is applied to them, thus the conditional in equation [8]. This is consistent with the manufacturer's specified minimum voltage.

To complete the feedback control law description, it remains to define how the PWM duty factors for the

motors on each side, d_R and d_L, are calculated based on the observations of the following agent's forward-facing RGB camera. This is done in the following section.

V. CONTROL FEEDBACK CALCULATION

The following agent's forward-facing camera captures between 5 and 15 frames per second, depending primarily on how much image processing is being performed on each frame. An onboard image processing algorithm performs the following steps on each frame to ultimately calculate d_R and d_L to send to the following agent's DC motors:

1. Map each pixel into a value $p_i \in [-1, 1]$, where $p_i \cong 1$ indicates a target-like pixel
2. Find the location of the fiducial sphere in image pixels, (x_p, y_p), using p_i
3. Calculate the area of the fiducial sphere in image pixels, A_L, using p_i
4. Use x_p to calculate the horizontal angle to the leader's fiducial sphere, θ_t
5. Use A_L to calculate the range to the leader's fiducial sphere, r_t
6. Optionally feed θ_t and r_t into a Kalman filter and update. Performance is evaluated both with and without the Kalman filtering of θ_t and r_t.
7. Use θ_t and r_t to calculate a the PWM duty factors for the wheels on each side of the vehicle, d_R and d_L.
8. Send d_R and d_L to the wheels and feed into equation [8]. These are the two control parameters.

We go through each of the above 8 steps individually.

Control Step 1

Each incoming frame is filtered to find all pixels belonging to the target by taking advantage of its unique color. Depending on the color of the local background, one of two different fiducial spheres are used: one bright yellow at approximately [255,255,0] in 8-bit [R,G,B] format, the other pure green at approximately [0,255,0] in 8-bit [R,G,B] format under strong ambient lighting. Consistent detection of the target sphere through varying lighting conditions proved challenging; the details of how we arrived at the following method are detailed in Section V.

Map the native RGB values for each pixel into a single value, p_i, in the range [-1,1] using one of the following equations depending on the color of the fiducial sphere being used:

for all pixels in the frame: $i = 0, 1, ..., N_x N_y$

$$p_i = \begin{cases} \frac{0.5\times(r_i+g_i) - b_i - |r_i - g_i|}{255} & \text{if fiducial sphere is yellow} \\ \frac{g_i}{1+r_i+g_i+b_i} - \frac{200-g_i}{255} & \text{if fiducial sphere is green} \end{cases}$$

$$p_i = max(min(p_i, 1), -1)$$

[9]

Where:

- N_x is the horizontal size of the frame in pixels
- N_y is the vertical size of the frame in pixels
- r_i, g_i, and b_i stand for the 8-bit [0:255] pixel values of the red, green, and blue channels, respectively.

Control Step 2

To make the fiducial sphere-finding process robust to similarly-colored background areas, we find the pixel with the largest p_i value, (x_p, y_p), and deem it to be the center of the fiducial sphere. This is not entirely optimal, since the peak pixel may be anywhere within the finite-area fiducial sphere. A more accurate method would be to find the centroid of all pixels deemed part of the target. In the interest of saving processing cycles over getting a perfectly accurate position, we just use the peak pixel:

$$p = p_i \text{ for } i = 0, 1, ..., N_x N_y$$

$$p_p = max(p)$$

$$(x_p, y_p) = argmax(p)$$

[10]

We must also allow for the possibility that the target's fiducial sphere has left the field of view of the camera or is otherwise not visible, so we impose the following condition:

if $p_p < p_{min}$:

Declare target not visible

Where p_{min} is an empirical tuning parameter that depends on the color of the fiducial sphere. 0.98 was found to be an optimal value for the green fiducial

sphere, and 0.2 was found to be an optimal value for the yellow fiducial sphere.

The action to take under the condition that the target is not visible is different depending on whether or not a Kalman filter is being used. See step 6 for details.

Control Step 3

Label all pixels within a radius, R_s, of (x_p, y_p), and above a tuned fraction, α_{rel}, of p_p as being part of the fiducial sphere. α_{rel} = 0.7 to 0.9 performed well in testing. This creates a mask, L, of pixels that belong to the fiducial sphere:

for all pixels in image $(x_{ij}, y_{ij}) \in ([1, N_x], [1, N_y])$

$$r_{p,ij}^2 = (x_p - x_{ij})^2 + (y_p - y_{ij})^2$$

if $r_{p,ij}^2 < R_s^2$ AND $p_{ij} > \alpha_{rel} p_p$ [11]

$$L_{ij} = 1$$

else

$$L_{ij} = 0$$

Calculate the total number of pixels labeled as part of the fiducial sphere as the area, A_L, of the sphere. Convert A_L into a percent of the total number of pixels in the image, $A_{\%}$:

$$A_L = \sum_i^{N_x} \sum_j^{N_y} L_{ij}$$

$$A_{\%} = \frac{100 A_L}{N_x N_y}$$

[12]

Control Step 4

Convert the horizontal pixel position of the peak of the leader's fiducial sphere, x_p, into an angle, θ_t:

$$\theta_t = \frac{100 x_p}{N_x} \alpha_x + \beta_0 \quad [13]$$

Where:

- α_x is an empirically measured parameter that converts from fractional image position (x100) into degrees. Measured as 0.495, i.e. 0.495° of physical azimuth corresponds to 1% of the image width. This means the total azimuth field of view is 0.495° x 100 = 49.5°
- β_0 is the azimuth angle of the far left edge of the image. Measured as -29.2°.

Comparing with β_0 with α_x, we notice that the image spans -29.2° through +20.3° azimuth, meaning that it is nearly 10° off-center. We suspect this is because the Raspberry Pi camera's firmware is cropping the imagery from a preset default down to our 80x60 pixel frame size, but did not have time to investigate further.

Control Step 5

The range to the lead agent in meters, r, is calculated from the measured area of the fiducial sphere as a fraction of the area of the image, $A_{\%}$:

$$r_t = \frac{\alpha_r}{\sqrt{A_{\%}}} \quad [14]$$

Where α_r is an empirically measured value, measured at 0.59 $\frac{m}{\sqrt{\%}}$ for the yellow sphere, i.e. if the fiducial sphere takes up 1% of the area of the image, then it is 0.59m away, or if it takes up 0.5% of the area of the image, then it is 0.83m away. Note that each image has nominally 80x60=4800 pixels, meaning one pixel is $\frac{1}{4800}$ = 0.02% of the image. Consequently, at 4.17m the fiducial sphere becomes sub-pixel.

For use in the Kalman filter updates, we also calculate an approximate error, σ_r, in the range measurement from equation [14]. This is approximated as 20% of the measured range:

$$\sigma_r = 0.2r \quad [15]$$

Control Step 6

θ_t and r_t are optionally fed into a Kalman filter to account for the uncertainty in their measurement. The Kalman filter also has the beneficial effects of smoothing out the commanded speed and direction of the following agent and coasting the assumed position of the fiducial sphere when the leader is no longer visible in the captured images.

The Kalman filter implementation consists of two steps: projecting the state, x_t (see equation [2]), forward to the time of the current frame, $t + dt$, then updating the state based on the measured range and angle to the leader at the current time. [8] was used as reference for these equations:

State projection:

$$x_{t+dt} = A x_t + U_t$$

$$P_{t+dt} = A P_t A^T + Q$$

[16]

State update:

$$K = P_{t+dt} H^T \left(H P_{t+dt} H^T + R_{t+dt} \right)^{-1}$$
$$x_{t+dt} = x_{t+dt} + K (z_{t+dt} - H x_{t+dt})$$
$$P_{t+dt} = P_{t+dt} - K H P_{t+dt}$$

[17]

Where

- A is the "coasting" matrix from equation [3]
- dt is the time, in seconds, since the previous frame
- x_t and x_{t+dt} are the states for the previous frame and the current frame, respectively. Initialized as $x_0 = (1, 0, 0, 0)^T$ (assumes leader starts stationary 1 meter directly ahead)
- U_t is the feedback control vector transformed to state space, per equation [8]
- P_t and P_{t+dt} are the 4x4 covariance matrices for the previous frame and the current frame, respectively. They are initialized with zeros.
- Q is the 4x4 process noise matrix. This accounts for unmeasured changes in the state by expanding the covariance ellipse. Four our purposes, this primarily accounts for any acceleration of the lead agent that *might* happen between measurements. We have found the following to give reasonable performance, given a framerate of 10 to 15Hz:

$$Q = \begin{pmatrix} 3e-4 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1e-3 & 0 \\ 0 & 0 & 0 & 0.05 \end{pmatrix}$$

[18]

Where the units of the 4 nonzero diagonal entries, from top-left to bottom-right, are meters, degrees, meters/second, and degrees/second.

- K is the Kalman gain matrix, which controls the relative contributions of the measurement versus the projected state. This is based on the relative sizes of the covariance ellipse, P_{t+dt}, and the measurement error, R_{t+dt}: whichever is smaller gets more weight in the update step.
- H is a 2x4 matrix that converts from state space to measurement (r, θ) space. For our purposes:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

[19]

- R_{t+dt} is a 2x2 measurement error matrix. This is approximated as:

$$R = \begin{pmatrix} \sigma_r & 0 \\ 0 & 0.5^\circ \end{pmatrix}$$

[20]

Where σ_r is calculated with equation [15].

- z_{t+dt} is the measurement of the range and angle to the leader, $(r_{t+dt}, \theta_{t+dt})^T$ from equations [13] and [14]

Note that when the fiducial sphere is deemed not visible per the condition in step 2, the state update part of the Kalman filter is skipped since there is no measurement, z_{t+dt}, available to update the state. The state projection is still applied, however, to coast the state to the current time.

Control Step 7

We use θ_t and r_t (previously θ_{t+dt} and r_{t+dt}) to calculate the PWM duty factors for the wheels on each side of the vehicle, d_R and d_L, as follows:

$$\varphi_s = \alpha_s \theta_t + \theta_0$$
$$\varphi_s = max(min(\varphi_s, 90), -90)$$
$$d_t = \alpha_r (r_t - r_d) + \alpha_\theta |\theta_t|$$
$$d_t = max(min(d_t, 1), 0)$$

$$d_R = \begin{cases} d_t & \text{if } \varphi_s <= 0 \\ s_s d_t cos(\varphi_s) & \text{if } \varphi_s > 0 \end{cases}$$

$$d_L = \begin{cases} s_s d_t cos(\varphi_s) & \text{if } \varphi_s <= 0 \\ d_t & \text{if } \varphi_s > 0 \end{cases}$$

[21]

Where:

- φ_s can be thought of as a "command angle"; it is merely an intermediate step in the calculation.
- α_s is an empirically chosen parameter that converts the angle to the lead agent, θ_t, to the command angle, φ_s. We found 12.0 to produce good results.
- θ_0 is an empirical correction to the command angle that accounts for mismatched speed of the left and right motors when given the same PWM duty. 15° proved adequate to keep the vehicle moving straight when $\theta_t = 0$
- d_t can be thought of as a "throttle duty"; it is

merely an intermediate step in the calculation. It is forced to be nonnegative to prevent backwards motion.

- α_r is an empirical parameter that controls the commanded speed of the vehicle in response to a difference between the desired range and the measured range to the lead agent. We found α_r =3.0 to lead to smooth operation.
- r_d is the desired range to the lead agent. This is 0.45m by default, but it is a freely adjustable parameter. Setting r_d to less than 0.3m is not advisable since the lead agent's fiducial sphere starts to go above the following agent's camera's FOV at close range. Setting r_d to more than 2m is not advisable since the limited resolution of the following agent's camera starts to cause the fiducial sphere to be poorly resolved at longer ranges, leading to sporadic detection.
- α_θ is an empirical parameter that boosts the "throttle duty", d_t, when the angle to the leader is large. This becomes important when $r_t - r_d$ is small, since we still want the following agent to rotate when the lead agent is near the desired range but away from zero degrees azimuth. 0.02 was found to work well.
- s_s is an empirical adjustment that scales the speed of the wheels on the inside of the turn. 1.0 (i.e. no adjustment) gave good performance.

Control Step 8

The following agent's motors are commanded by setting the PWM duty factors of the voltage applied to the wheels on the right and left sides to d_R and d_L, respectively.

V. CHALLENGES

Limited Field Of View

One of the primary challenges associated with a single forward-facing camera autonomous robot is a its limited field-of-view. The single camera on the following agent is mounted at the front of the vehicle facing forwards. Its horizontal field of view is approximately +/- 25° off of its boresight with the standard lens attached. The vertical field of view is slightly smaller, but for the 2D nature of this problem, that is not a concern.

The primary consequence of the limited field of view is the lead-agent turing and leaving the field of view of the following agent on either side of the camera frame. A naïve attempt at mitigating this problem is to simply have the following agent's controller maintain the previous command whenever it loses sight of the lead agent.

The more complete approach is to use a Kalman filter to account for and track the uncertainty in the range and angle measurements of the fiducial sphere, which results in a smoother control reaction. A Kalman filter also makes the controller more robust to camera noise, poor visibility, clutter/confuser objects, and dropped frames by giving the most reliable prediction of the position of the lead-agent possible.

We implement a Kalman filter in the following agent's control software per the description in Section V step 6 to mitigate these issues.

Limited Computing Capabilities

The Raspberry Pi Zero W, while a very inexpensive, small, power-efficient, and easy to work with microcontroller/mini-computer, suffers from limited processing power. It runs a single-core 1GHz processor with 512MB RAM, which pales in comparison to even modest contemporary smartphones.

The primary consequence of the limited computing capabilities of the Raspberry Pi Zero W is the need to heavily downsample the incoming video frames in order to maintain a reasonable framerate. For reference, using the baseline image processing algorithm, a resolution of just 80x60 pixels was necessary to maintain a framerate of 10 to 15Hz.

Most approaches in the literature get around this limitation by sending their agent's video feed to a much more capable remote computer via wifi or bluetooth. The remote computer then runs a suite of image processing algorithms on the video feed to extract a control solution, which is then sent back to the agent. This strategy comes with important limitations, chief among them the need to be in range of some manner of wireless network. This is infeasible for many real-world applications.

From a software perspective, the limited processing power of the Raspberry Pi Zero W - coupled with the desire for a fast framerate - means that any image processing must be quick and efficient. This is somewhat mitigated in our experimental setup by the addition of a brightly colored fiducial sphere on the lead agent, somewhat mitigating the need to run computationally-expensive computer vision algorithms.

Some computer vision algorithms that were not used because they consume too many processing cycles were dilation and erosion of the fiducial sphere mask and fitting a circle to the fiducial sphere mask. Nonetheless, a simple computer vision algorithm must still be run on the Pi to deal with dynamic lighting conditions and potential confusers.

Dynamic Lighting

Dynamic lighting is a very common scenario for the following agent to deal with. In testing, this was most prevalent when running indoors during the daytime with windows to the outside visible in the field of view. Windows viewed directly are more than an order of magnitude brighter than the ambient light level in the room.

The Raspberry Pi camera has automatic exposure time adjustment built in, meaning that in response to an increase in image brightness, it lowers its exposure time to avoid saturating the bright regions of the image. The consequence of this from a lead-agent detection perspective is that the overall brightness of pixels on the lead-agent rises and falls linearly with the change in exposure time. A simple RGB threshold is therefore not robust to dynamic lighting.

The ideal colorspace for detecting distinct colors in dynamic lighting is the hue-saturation-value (HSV) colorspace. In HSV, the hue and saturation of a bright object remain relatively consistent under dynamic lighting, while the value channel changes. This makes detecting a bright yellow object as simple as selecting a small window around the hue for yellow (55°) and a lower bound on saturation of about 0.7. Unfortunately, changing the colorspace from the camera's native RGB to HSV proved to lower the framerate by ~6Hz, rendering it infeasible.

The solution to this was to apply a fast custom algorithm to reliably detect the lead agent's fiducial sphere and determine its size, as is laid out in equations [9] through [12]. The algorithm's main features are a simplified mapping from an RGB colorspace to a "probability" that a pixel contains the fiducial sphere, and choosing the pixel with the peak "probability" to represent the location of the fiducial sphere. This algorithm enabled fast computation of a fiducial sphere mask and robustness to background clutter with color similar to the fiducial sphere.

Sporadic missed detections

Sporadic missed detections are a reality of any real-world camera system used for detecting a target. They can occur due to dynamic lighting as noted before, or from software errors, communication faults, or momentary obscuration of the camera by objects in the environment.

Sporadic sensor outages were simulated by simply putting a sheet of paper between the lead agent and the following agent for a short time - approximately one second - blocking the following agent's view of the lead agent's fiducial sphere. As evidenced in the results section (Section VI), the Kalman filter makes the following agent robust to these outages. Instead of simply stopping when it loses sight of the lead agent, the following agent continues to seek towards the coasted location of the lead agent until its visibility is restored.

VI. RESULTS

We find the following agent capable of successfully maintaining formation with the lead agent. To make this possible, the following agent and lead agent must have comparable speed and acceleration capabilities.

The Kalman filter is the primary aspect of the control algorithm that makes the following agent robust to the variety of challenges introduced by the limitations of the low-cost, low-SWAP system components used to build the following agent. We show the quality of the formation control both with and without the Kalman filter applied to demonstrate its importance in making the system robust to these challenges.

Videos showing both an overhead perspective and the feed from the following agent's onboard camera are included with the final report package. These demonstrate the performance of the following agent. The 30s circle run in particular demonstrates the robustness of the following agent to dynamic lighting: sunshine from the window in the background causes sporadic missed detections, but the following agent is able to maintain its formation through these missed detections. This is primarily thanks to the Kalman filter.

Figure [5] shows plots of the range and angle to the lead agent over time for the runs shown in the supplementary videos. The following agent was commanded to be 45cm behind the lead agent while keeping the lead agent immediately in front of it, i.e. $(r_t, \theta_t) = (0.45m, 0°)$. The range and angle in the plots are as measured by the following agent - use of external measurement verification systems were not feasible for the scope of this project.

Notice that the actual range that the following agent maintains behind the lead agent tends to be closer to 0.7m, rather than the commanded 0.45m. This is a consequence of the finite reaction time of the following agent. When the lead agent is measured at 0.45m, the following agent is, logically, commanded to stop. Consequently, when the lead agent moves forward, range increases, and in turn, the following agent attempts to catch up with the lead agent. It only reaches the commanded 0.45m when the lead agent is stationary long enough for the following agent to catch up.

The measured angle tends to oscillate somewhat due to the finite latency of the following agent's reactions. It also tends to be negative because the lead agent is consistently turning to the left. There is a tradeoff between this oscillation and the ability of the following agent to react quickly to fast maneuvers by the lead agent. Adjusting the α_θ parameter in equation [21] controls this tradeoff.

Two additional tests were executed to demonstrate the benefits of utilizing a Kalman filter. The videos of these tests are also included in the report package and their range and angle vs time plots are also shown in figure [5].

The "LEFT TURN" test had the lead agent start ahead of the following agent, then turn left quickly enough to leave the FOV of the following agent's camera. The videos and associated range and angle vs time plots show the ability of the Kalman filter to overcome this challenge, while running without the Kalman filter causes track to be lost indefinitely. Note the gap in the instantaneous measurements and the continued turn to reacquire the target from 4s to 6.5s.

The "SPORADIC OBSCURATION" test involved the following agent moving straight with sporadic sensor outages. The sensor outages were simulated by intermittently placing a black sheet of paper in front of the fiducial sphere, blocking the following agent's view of the fiducial sphere. The videos and the associated range and angle vs time plots show how the Kalman filter is able to keep the following agent closer to the desired range.

VII. FUTURE WORK

Potential improvements deal primarily with the hardware setup. The lead agent vehicle was suboptimal for this test due to its binary throttle and steering controls. The throttle had to be pulsed to force the lead agent to maintain a speed that the following agent was capable of mimicking. A higher quality lead vehicle would have made testing less cumbersome.

Applying a fisheye lens to the following agent's camera would enlarge its angular field of view, thereby making it harder for the lead agent to leave the following agent's field of view.

Using a more complex fiducial marker, ala the Aruco fiducial used in [7], would enable measurement of the relative orientations of the lead agent and following agent. The current isotropic fiducial sphere does not allow for measurement of relative orientation. Such a measurement could be used to inform the state projection of the likely motions of the lead agent and lead to smoother controls.

Running tests in an well-instrumented robotics lab would enable more precise measurement of the following agent's reaction to control commands, thereby improving the Kalman filter's estimates of the projected state.

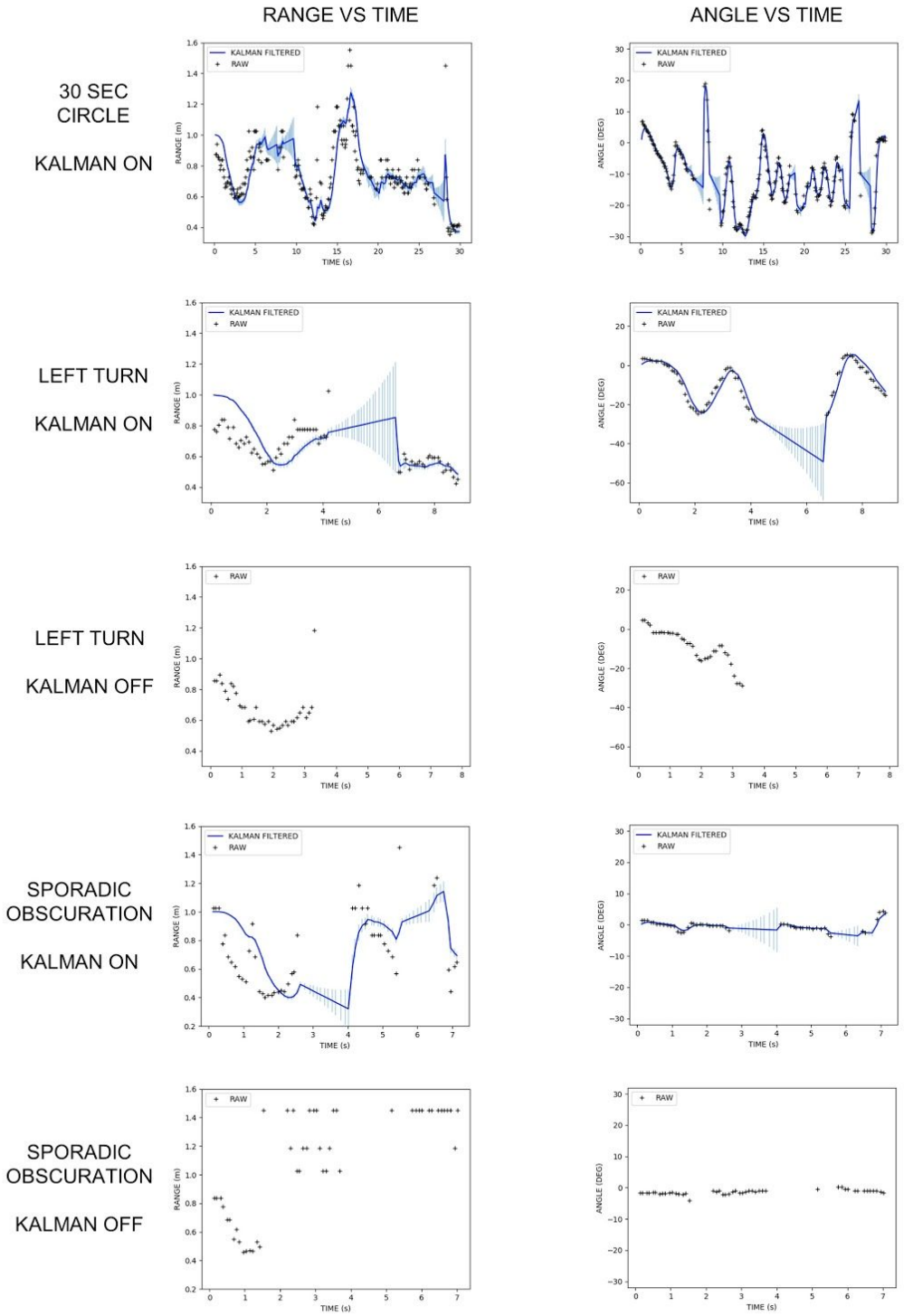


Figure 5: Range-time and angle-time plots for 5 tests, run with and without Kalman filtering. Kalman covariances (error bars) are shown at 2σ. These plots correspond to the videos that accompany this report.

Addition of a more complex lead agent detection algorithm would enable removal of the fiducial sphere as an aid in lead agent detection. Such an algorithm would likely involve a large amount of data collection and use of a convolutional neural network to determine the range and angle to the lead agent.

Further investigation into the firmware and software that run the Raspberry Pi camera may be able to decrease the latency of the captured imagery and further stabilize the framerate.

VIII. REFERENCES

1. Kwany-Kyo Oh, Myoung-Chul Park, Hyo-Sung Ahn, "A survey of multi-agent formation control", Automatica 53, pp 424-440, 2015
2. Gian Luca Mariottini et al, "Vision-Based Localization for Leader–Follower Formation Control", IEEE Transactions on Robotics, Volume 25, Issue 6, 2009
3. Luca Consolini, Fabio Morbidi, Domenico Prattichizzo, Mario Tosques, "Leader–follower formation control of nonholonomic mobile robots with input constraints", Automatica, Volume 44, Issue 5, 2008
4. M. Hassan, E. Aljuwaiser, R. Badr, "A new on-line observer-based controller for leader-follower formation of multiple nonholonomic mobile robots", Journal of the Franklin Institute, Vol 355, Issue 5, 2018
5. R. Hogg, A. Rankin, M. McHenry, D. Helmick, C. Bergh, S. Roumeliotis, and L. Matthies, " Sensors and Algorithms for Small Robot Leader/Follower Behavior", Proc. SPIE 4364, 2001
6. R Cruz-Morales, M, Velasco-Villa, R Castro-Linares, and E Palacios-Hernandez, "Leader-Follower Formation for Nonholonomic Mobile Robots: Discrete-Time Approach", International Journal of Advanced Robotic Systems, Vol 13, Issue 2, 2017
7. M. Vankadari, K. Das, and S. Kumar, "Autonomous Leader-Follower Architecture of A.R. Drones in GPS Constrained Environments", Proc. ACM Advances In Robotics, Article 51, 2017
8. www.bzarg.com, "How a Kalman Filter Works, in Pictures", https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/, accessed 4/12/19