THE DESIGN AND IMPLEMENTATION OF AN EFFECTIVE VISION-BASED LEADER-FOLLOWER TRACKING ALGORITHM USING PI CAMERA

Songwei Li

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2016

APPROVED:

Yan Wan, Major Professor
Xinrong Li, Committee Member
Shengli Fu, Committee Member and Chair of the Department of Electrical Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Victor Prybutok, Vice Provost of the Toulouse Graduate School
Li, Songwei. *The Design and Implementation of an Effective Vision-Based Leader-Follower Tracking Algorithm Using Pi Camera*. Master of Science (Electrical Engineering), August 2016, 44 pp., 28 figures, 14 numbered references.

The thesis implements a vision-based leader-follower tracking algorithm on a ground robot system. One camera is the only sensor installed in the leader-follower system and is mounted on the follower. One sphere is the only feature installed on the leader. The camera identifies the sphere in the openCV Library and calculates the relative position between the follower and leader using the area and position of the sphere in the camera frame. A P controller for the follower and a P controller for the camera heading are built. The vision-based leader-follower tracking algorithm is verified according to the simulation and implementation.
ACKNOWLEDGEMENTS

First, I would like to offer my sincerest gratitude to my major advisor Prof. Yan Wan, who has supported me throughout my M.S. study with her great patience, encouragement and mentoring. My sincere thanks also go to the rest of my thesis committee: Prof. Shengli Fu and Prof. Xinrong Li for their insightful comments and encouragement. I would also like to thank all of my friends for their company, support, and discussion.

I would also like to thank the Toulouse Graduate School at UNT, and Dr. Wan's grants from the National Science Foundation under numbers of GF 1453722, GF 1522458 and GF 154483 for the financial supports.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapters</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS ..................................................................................</td>
<td>iii</td>
</tr>
<tr>
<td>1. INTRODUCTION ........................................................................................</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation .........................................................................................</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Overview of the Thesis .....................................................................</td>
<td>1</td>
</tr>
<tr>
<td>2. ANALYSIS OF APPROACHES FOR OBJECTS’ POSITION USING CAMERA VISION ....</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Introduction .....................................................................................</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Camera Projection Model ...................................................................</td>
<td>4</td>
</tr>
<tr>
<td>2.3 The Location-Feature-Area Method .................................................</td>
<td>6</td>
</tr>
<tr>
<td>2.4 The Location-Feature-Position Method ...........................................</td>
<td>11</td>
</tr>
<tr>
<td>2.5 Conclusion .......................................................................................</td>
<td>15</td>
</tr>
<tr>
<td>3. LEADER-FOLLOWER TRACKING ALGORITHM BASED ON A SINGLE SPHERE FEATURE</td>
<td>16</td>
</tr>
<tr>
<td>3.1 Introduction .....................................................................................</td>
<td>16</td>
</tr>
<tr>
<td>3.2 Estimation of Sphere’s Center in the Camera Coordinate System .......</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Estimation the Bearing Angle and the Relative Position of the Sphere</td>
<td>20</td>
</tr>
<tr>
<td>3.4 Controller Design ...........................................................................</td>
<td>22</td>
</tr>
<tr>
<td>3.5 Conclusion .......................................................................................</td>
<td>25</td>
</tr>
<tr>
<td>4. SIMULATION AND EXPERIMENT ..................................................................</td>
<td>27</td>
</tr>
<tr>
<td>4.1 Simulation .........................................................................................</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Hardware and software .......................................................................</td>
<td>29</td>
</tr>
<tr>
<td>4.3 Detecting the Target Using Raspberry Pi 2 and OpenCV .....................</td>
<td>33</td>
</tr>
<tr>
<td>4.4 Experiment Result ............................................................................</td>
<td>37</td>
</tr>
</tbody>
</table>
5. CONCLUSION AND FUTURE WORK.................................................................40
   5.1 Conclusion.........................................................................................40
   5.2 Future Work....................................................................................40

APPENDIX : DIAGRAM OF LEADER-FOLLOWER TRACKING.......................41

REFERENCES ..............................................................................................................43
CHAPTER 1
INTRODUCTION

1.1 Motivation

UAVs are envisioned to play a big role in disaster rescue, with their capabilities such as information transmission at a long distance, searching in a disaster area, and surveillance of the target in the air or on the ground. In the article [1], a drone-to-drone communication system was built to transfer information at a long distance for a catastrophe scenario. Through the drone-to-drone communication system, real-time video of the disaster zone can be transmitted to the rescue center. Drones sometimes need to work with robots on the ground to scan the disaster area so to get close to a target to obtain the detailed information, in scenarios such as earthquake or the catastrophe of a chemical factory. In these cases, the drone can bring a robot with a camera to the disaster area. Then the drone drops the robot on the ground and follows it using the camera equipped on the drone. Meanwhile, the drone transfers the real-time video from both cameras to the rescue center.

Motivated by the above scenarios, our goal is to make the follower drone track the leader robot as much as possible, as the air-to-ground communication requires the robot to be within a distance to the drone. In this research, we address the leader-follower tracking problem using camera vision. As the preliminary effort, we implemented the vision-based tracking algorithm on a ground robot system, and leave the implementation on the drone platform to the future work.

1.2 Overview of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews two approaches to leader-follower tracking problem using camera vision and analyzes their defects that motivated this research. One approach is to use the area of the sphere in the camera frame to
track the leader target. The other is to use the positions of three features in the camera frame to locate the leader target and then achieve the leader-follower tracking task. Chapter 2 also justifies a key assumption to this thesis, that one camera installed on the follower can determine the position of a sphere ball installed on the leader.

Chapter 3 describes our method for tracking one sphere ball using camera vision and a proportional controller in the leader-follower tracking system. In this chapter, we first describe the algorithm to locate the sphere ball using camera vision, then provides the model of a differential drive robot, and finally describe the control algorithm to implement the follower to track the sphere target.

Chapter 4 describes the simulation and implementation of our leader-follower tracking algorithm described in Chapter 3. The chapter also provides the information of the hardware and software used in the research.

Finally, Chapter 5 includes a conclusion and a brief discussion of future works.
CHAPTER 2
ANALYSIS OF TWO EXISTING APPROACHES FOR OBJECTS’ POSITION USING
CAMERA VISION

2.1 Introduction

Our goal is to build a leader-follower tracking system using camera vision. To track the leader, the follower needs to obtain leader information using its own camera, such as estimated velocity and angular velocity of the leader, the relative position of the leader and follower, and the bearing angle between the headings of the follower and the leader. Since the velocity and angular velocity of the leader and the bearing angle can be inferred from the relative position, the relative position is an key quantity to estimate for leader-follower tracking algorithm. Through the geometry relationship between cameras on the follower and features on the leader, the relative position of robots can be calculated, as shown in these papers [2]-[6]. Motivated by implementation concerns of this project, we try to develop a tracking algorithm with minimal hardware and feature usage. In particular, when drones and robots search a disaster zone, they need to load batteries, sensors, microcontrollers, and antennas so that there is little space left for cameras or multiple well-separated features on their frames. Meanwhile, the lighter weight can prolong the running time of drones and robots. As a result, we aim to use the minimal number of cameras and features to complete the tracking task.

In paper [5][4], Davison and Murray calculate the object’s position using two cameras and one feature. In paper [6], Fang et al describe an approach to calculate the objects’ location using one camera and four features on one plane. In consideration of the load limitation of robots and the implementation inconveniences, the number of cameras and features in paper [5],[6]doesn’t match our requirement of using a minimal number of cameras and features in our research.
In paper [7], an object’s relative distance, which is scaled by the area of one feature (a ball) in the camera frame, is obtained using one camera and one feature (a ball). When the area of the ball decreases in the camera frame, it means the ball is moving away from the camera, or the ball is approaching the camera. This approach didn’t provide the relative position between the camera and the ball. In this thesis, we call the method location-feature-area. In paper [8], Chen and Jia use one camera and three features to estimate the leader’s position. The location method uses the positions of three features in one camera view. This approach is referred to as location-feature-position in this thesis. Although the location-feature-area and location-feature-position better match with our requirement compared to [5] and [6], they are still not suitable for our research. As the location-feature-area method cannot provide the relative position between the camera and the ball, we cannot build an advanced controller to improve the performance of the leader-follower tracking system for our research. In addition, as the requirements of multiple features in the location-feature-position method are hard to achieve (described in Section 2.3), we cannot implement this approach for our research. We conducted simulation and analysis for these two approaches, which further demonstrate show their defects if being applied in this research. Motivated by problems of these two approaches, we aim to develop an approach to track the leader using only one camera and one feature.

The remainder of this chapter is organized as followers. Chapter 2.2 describes the projection model of a camera. Chapter 2.3 describes and evaluates the location-feature-area approach. Chapter 2.4 describes and evaluates the location-feature-position approach. Chapter 2.5 provides a brief conclusion and discusses a key assumption that we use for target’ location tracking using camera vision.

2.2 Camera Projection Model
This section describes the camera projection model that is used to locate an object’s position. This model represents the geometric relationship between an image point and its relative point in the camera coordinates. As shown in Figure 1.(a), O is the original point in the camera’s coordinates; Z axis is the optical axis of the camera and perpendicular to the image plane at point o; o is the original point in the image coordinates; the distance between O and image plane is f, in fact, f is the camera’s focal length; point P (X, Y, Z) in the camera coordinates projects to point p (x, y) on the image plane.

![Figure 1 Illustration of camera projection](image)

According to book [9], point P projects to point p via:

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} = \begin{bmatrix}
  \frac{f}{z} & 0 & 0 \\
  0 & \frac{f}{z} & 0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  X \\
  Y \\
  1
\end{bmatrix}
\]  

(2-1)

Then point p projects on the CCD (charged coupled device) sensor plane and it can be expressed in the pixel coordinate via:

\[
\begin{bmatrix}
  m \\
  n \\
  1
\end{bmatrix} = \begin{bmatrix}
  \frac{1}{\alpha_u} & 0 & m_o \\
  0 & \frac{1}{\alpha_v} & n_o \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]  

(2-2)

where \((m_o, n_o)\) is the principle point in pixel coordinate, \(\alpha_u\) and \(\alpha_v\) are one pixel physical dimensions in the horizontal direction u and vertical direction v (see Figure 1).
In view of (2-1) and (2-2), the above matrix is rewritten as:

\[
\begin{bmatrix}
m \\
n \\
1
\end{bmatrix} = \begin{bmatrix}
1/\alpha_u & 0 & m_o \\
0 & 1/\alpha_v & n_o \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
f/2 \\
z/2 \\
z
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
1
\end{bmatrix}
\]

(2-3)

\[
\begin{bmatrix}
m \\
n \\
1
\end{bmatrix} = \begin{bmatrix}
f_x/2 \\
z/2 \\
z
\end{bmatrix} \begin{bmatrix}
m_o \\
n_o \\
1
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
1
\end{bmatrix}
\]

(2-4)

where \(f_x = f/\alpha_u\), \(f_y = f/\alpha_v\). In addition, \(f_x, f_y, m_o, n_o\) are the intrinsic parameters and can be obtained by the camera calibration procedure.

2.3 The Location-feature-area Method

The section reviews the location-feature-area method which was implemented on the CMUcam5 Pixy sensor (Pixy) [7] in Figure 2. This sensor was designed by Charmed Labs for color detection. It consists of a microcontroller NXP LPC4330 and an image sensor Omnivision OV9715. Pixy possesses the following features. 1) It has a fast speed to process digital images. Its speed can arrive 50 times per second. 2) It is convenient for users to use. Pixy has a friendly interface which works with Raspberry Pi, Arduino, and BeagleBone. Pixy provides its library for these microcontrollers so that users don’t need to knowledge principles of pattern recognition or compile codes to detect a target. The detection procedure of Pixy is in the following. Through the software PixyMon that Charmed Labs offers, users decide which color to be identified. When the target color is detected, Pixy draws the target with a straight rectangle. For instance, in Figure 3, the target color is red, and the Pixy draws...
out the target with a square contour. Users just call functions from Pixy’s library to obtain the area $S$ and center location $(x_u(t), y_v(t))$ of the contour in the camera frame. Meanwhile, the bearing angle $\beta(t)$ between the camera and the heading of the robot can be obtained.

In paper [7], a mobile robot tracks a ball using a proportional controller:

$$V_F(t) = S_d - S(t)v_{gain}$$

$$d_v(t) = ((\beta_d - \beta(t)) + (\beta_d - \beta(t))V_f(t))d_{gain}$$

$$V_l(t) = V_f(t) + d_v(t)$$

$$V_r(t) = V_f(t) - d_v(t)$$

Where $S_d$ is the desired size of the ball, $\beta_d$ is the desired bearing angular, $v_{gain}$ and $d_{gain}$ are the feedback gains, $d_v(t)$ is a steering differential, $V_l(t)$ is the velocity of left wheel, $V_r(t)$ is the velocity of right wheel.

And the controller for the camera to tracking the ball is shown in the following equations

$$e_x(t) = x_{u0} - x_u(t)$$
\[ e_y(t) = y_u(t) - y_{uo} \]
\[ V_{cp}(t) = e_x(t)pGain + e_x(t)dGain \]
\[ V_{ct}(t) = e_y(t)pGain + e_y(t)dGain \]

where \((x_{uo}, y_{uo})\) is the intrinsic center of the camera, the derivatives of \(e_x(t)\) and \(e_y(t)\) are \(\dot{e}_x\) and \(\dot{e}_y\) \(pGain\) and \(dGain\) are the feedback gains, \(V_{cp}(t)\) and \(V_{ct}(t)\) are the velocities of the pan servo and the tilt servo. In order to simulate the leader-follower tracking system using the location-feature-area method, we build the code to simulate in MATLAB.

<table>
<thead>
<tr>
<th>The pseudocode for leader-follower tracking system using the location-feature-area method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1:</strong> Initialization</td>
</tr>
<tr>
<td>1.1 ((x_f(1), y_f(1)) = (50,25), V_f(1) = 0, W_f(1) = 0, \beta_d = 0, S_d = 400, (x_{uo}, y_{uo}) = (160,120)) and the total time steps (t_{total} = 1000)</td>
</tr>
<tr>
<td>1.2 (\textbf{for } t = 1: t_{total})</td>
</tr>
<tr>
<td><strong>Step 2:</strong> Object detection</td>
</tr>
<tr>
<td>2.1 According to the ball’s projection in the pixel frame, one obtains (S) and ((x_u, y_v))</td>
</tr>
<tr>
<td><strong>Step 3:</strong> Calculate the velocities of follower</td>
</tr>
<tr>
<td>3.1 (\textbf{if } S(t) = S_d \text{ &amp;&amp; } \beta(t) = \beta_d)</td>
</tr>
<tr>
<td>3.2 (V_{lf}(t) \leftarrow 0, V_{rf}(t) \leftarrow 0;)</td>
</tr>
<tr>
<td>3.3 (\textbf{else})</td>
</tr>
<tr>
<td>3.4 (\text{Calculate } V_f(t) \text{ and } d_v(t) \text{ with equation (2-6);})</td>
</tr>
<tr>
<td>3.5 (V_{lf}(t) \leftarrow V_f(t) + d_v(t);)</td>
</tr>
<tr>
<td>3.6 (V_{rf}(t) \leftarrow V_f(t) - d_v(t);)</td>
</tr>
<tr>
<td>3.7 (\textbf{end})</td>
</tr>
<tr>
<td><strong>Step 4:</strong> Update the position of the follower</td>
</tr>
<tr>
<td>4.1 (\beta(t + 1) = \beta(t) + \frac{V_r - V_l}{b} \delta; \quad %\delta \text{ is the sampling time, } b \text{ is the distance between the centers of the left wheel and right wheel})</td>
</tr>
<tr>
<td>4.2 (x_f(t + 1) = x_f(t) + V_r + V_l \cos(\beta(t + 1))\delta;)</td>
</tr>
<tr>
<td>4.3 (y_f(t + 1) = y_f(t) + \frac{V_r + V_l}{2} \sin(\beta(t + 1))\delta;)</td>
</tr>
<tr>
<td><strong>Step 5:</strong> Update the bearing angular (\beta(t + 1))</td>
</tr>
<tr>
<td>5.1 (\textbf{if } (x_u, y_v) = (x_{uo}, y_{uo}))</td>
</tr>
<tr>
<td>5.2 (V_{cp} \leftarrow 0, V_{ct} \leftarrow 0;)</td>
</tr>
<tr>
<td>5.3 (\textbf{else})</td>
</tr>
<tr>
<td>5.4 (V_{cp} \leftarrow e_x pGain + e_x dGain ;)</td>
</tr>
<tr>
<td>5.5 (V_{ct} \leftarrow e_y pGain + e_y dGain;)</td>
</tr>
</tbody>
</table>
The result of the simulation is shown in Figure 5. In order to simulate the real scenario, an arbitrary trajectory of the leader is created using the smooth-turn mobile model \([10]\) in Figure 4. The starting points of the leader target and follower robot are \((51, 26)\) and \((51, 25)\). As shown in Figures 4, 5, although the follower follows the ball to move, the follower cannot achieve the desired tracking performance. The bearing angle \(\beta(t)\) is more than 0.5 in the last part of the simulation. In the implementation of Pixy, some defects of location-feature-area are observed: 1) It cannot estimate the relative position between the camera and the leader ball, and such it cannot implement an advanced controller to improve the performance of leader-follower tracking problem. 2) The camera cannot distinguish the detected target from objects with a similar color. As a result, the camera may lose the identified target easily. Sometimes,

<table>
<thead>
<tr>
<th>5.6</th>
<th><strong>end</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>(\beta(t + 1) = \beta(t) + V_c \delta; ) go to back step 2</td>
</tr>
<tr>
<td></td>
<td><strong>end</strong></td>
</tr>
</tbody>
</table>

![Figure 4](image_url)

Figure 4 The trajectories of follower and leader. The follower tracks the leader using the location-feature-area method.
Figure 5 Simulation result. $V_F$ is the follower’s velocity. $x_u$ is the position of the ball in the camera view. $\beta$ is the beading angular between the camera and the heading of the robot.

Figure 6 Target detection in Pixy. The ball is the detected target and the rectangle is noise. (a) The noise is identified as a target. (b) The ball and noise are identified as a target.

Pixy may draw the target with noise and then calculate the area of contour in a wrong way (see Figure 6). Due to these defects, this algorithm does not perform well.
2.4 The Location-Feature-Position Method

This section reviews and analyzes the location-feature-position method. In this algorithm [6], the altitudes of three features are not equal to zero in the camera coordinate, the height of one feature must be positive in the camera coordinate, and these features are not on one perpendicular plane of the follower robot’s motion plane. Also, the camera must always detect three separate features. According to the camera projection model and geometric analysis between the leader, camera and the desired position of the follower [8], the following formation can calculate the bearing angular $\beta$ between the headings of the follower robot and the leader target:

$$a_{ij}\sin\Delta + b_{ij}\cos\Delta = c_{ij} \tag{2-7}$$

where

$$\Delta = -\varphi + \varphi^* + \beta$$

$$a_{ij} = -x_i x_i^* - y_i y_i^* + y_j y_i^* + x_j x_i^*$$

$$b_{ij} = -x_i y_j^* + y_i x_j^* - y_j x_i^* + x_j y_i^*$$

$$c_{ij} = -x_i y_j + y_i x_j - x_i^* y_j^* + y_i^* x_j^* \tag{2-8}$$

$\varphi$ is the bearing angular between the camera and the heading of the follower robot, $\varphi^*$ is the desired bearing angular between the camera and the heading of the follower robot, $x_i = \frac{x_i}{z_i}, y_i = \frac{y_i}{z_i}$ and $(X_i, Y_i, Z_i)$ is the position of one feature $Q_i$ in the camera frame, $x_i^* = X_i^*/Z_i^*, y_i^* = Y_i^*/Z_i^*$ and $(X_i^*, Y_i^*, Z_i^*)$ is the desired position of the feature $Q_i$ in the camera frame, $x_j = \frac{x_j}{z_j}, y_j = \frac{y_j}{z_j}$ and $(X_j, Y_j, Z_j)$ is the position of one feature $Q_j$ in the camera frame, $x_j^* = X_j^*/Z_j^*, y_j^* = Y_j^*/Z_j^*$ and $(X_j^*, Y_j^*, Z_j^*)$ is the desired position of the feature $Q_j$ in the camera frame. When the number of features is 3, the following formation can be obtained from (2-1):

$$[\sin\Delta \cos\Delta] = (C^T C)^{-1} C^T D \tag{2-9}$$

where
According to equation (2-3), the bearing angular $\beta$ is

$$\beta = \text{atan2}(\sin\Delta, \cos\Delta) + \varphi - \varphi^*$$

(2-11)

As the camera cannot measure the location error $(X_e, Y_e)$ between the current location and desired location of the follower robot, the paper [8] defines new variables $(x_e, y_e) = (X_e/Z_l, Y_e/Z_l)$ where $Z_l$ is the altitude of one feature in the camera feature. $(x_e, y_e)$ is measureable through the camera projection model. In the location-feature-position method, any one feature of the three features can be chosen as the position of the leader. The target feature is denoted as $Q_1(X_1, Y_1, Z_1)$. Through the geometric analysis [8], $(x_e, y_e)$ is calculated by the following equation:

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = R(\beta) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + R(\varphi) \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

(2-12)

where $x_d = \frac{x_d}{Z_1}$, $y_d = \frac{y_d}{Z_1}$ and $(X_d, Y_d)$ is the desired position of the follower corresponding to the leader; $x_1 = \frac{x_1}{Z_1}$, $y_1 = \frac{y_1}{Z_1}$; $R()$ is a rotation matrix:

$$\begin{bmatrix} \cos() & -\sin() \\ \sin() & \cos() \end{bmatrix}.$$  

The procedure of the leader-follower tracking system using the location-feature-position method is shown in Figure 7. In this procedure, $v_f$ is the velocity of the follower, $w_f$ is the angular velocity of the follower, $(X_f, Y_f)$ is the position of the follower in the world coordinates, $w_c$ is the angular velocity of the camera, and $e_x$ is the error between the current position and the desired position of $Q_1$ at the horizontal direction in the pixel coordinates. The follower controller $f(x_e, y_e, \beta)$ and pan camera controller $C(e_x, w_f)$ are backstepping controller which builds a stabilizing controllers by a recursive method.
In order to simulate the leader-follower tracking system using the location-feature-position method, we build the following codes to simulate in MATLAB.

**Step 1:** Initialization

1.1 \[ X_d = 0.5, Y_d = 0.3, v_f = 0, w_f = 0, X_f = 49.50, Y_f = 24, m_1^* = 419.299 \]

the total time steps \( t_{total} = 1000 \), \( \varphi^* = \frac{\pi}{6} \); the orientation angular of the follower \( f_d = \pi/2 \)

Note: \( m_1^* \) is the desired position of the feature \( Q_1 \) in the pixel coordinates;

**Step 2:** Object detection

2.1 \[ \beta \leftarrow \text{atan2}(\sin \Delta, \cos \Delta) + \varphi - \varphi^* \]

2.2 \[ \begin{bmatrix} X_e \\ Y_e \end{bmatrix} \leftarrow R(\beta) \begin{bmatrix} X_d \\ Y_d \end{bmatrix} + R(\varphi) \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} \]

2.3 \[ e_x \leftarrow m_1 - m_1^* \text{; } \% m_1 \text{ is the position of feature } Q_1 \text{ at the horizontal direction of the pixel coordinates.} \]

**Step 3:** Update the position of the follower

3.1 \[ v_f, w_f \leftarrow f(x_e, y_e, \beta); \]

3.2 \[ f_d \leftarrow f_{d\text{pre}} + w_f \delta \text{; } \% \delta \text{ is the sampling time and } \delta = 0.1s, f_{d\text{pre}} \text{ is the } f_d \text{ at the last step time.} \]

3.3 \[ X_f \leftarrow X_{f\text{pre}} + v_f \cos(f_d)\delta \text{; } \% X_{f\text{pre}} \text{ is the } X_f \text{ at the last step time;} \]

3.4 \[ Y_f \leftarrow Y_{f\text{pre}} + v_f \sin(f_d)\delta \text{; } \% Y_{f\text{pre}} \text{ is the } Y_f \text{ at the last step time;} \]

**Step 4:** Update the bearing angular \( \varphi \)

4.1 \[ w_c \leftarrow C(e_x, w_f) \]

4.2 \[ \varphi \leftarrow \varphi_{\text{pre}} + w_c \delta \text{; } \% \varphi_{\text{pre}} \text{ is the } \varphi \text{ at the last step time;} \]

4.3 Go back **Step 2**

**end**
As shown in Figures 8 and 9, the errors $x_e, y_e$ tend to become zero, and the follower robot tracks the leader target well. To compare it with the location-feature-area method, the location-feature-position method can measure the bearing angular $\beta$ and the relative position error $(x_e, y_e)$. Furthermore, the follower robot can track the leader target by a relative position $(x_d, y_d)$. In addition, an advanced controller for the leader-follower tracking system is built using the bearing angle $\beta$ and position error $(x_e, y_e)$.

In consideration of the requirements of features, three feature were used on the leader robot. If these features are small ones, the camera on the drone cannot catch them. If the features are large ones, they become close to each other and hence the camera cannot distinguish them as three separate features. Moreover, when we implement this method on a drone and a robot, it is possible that the height of one feature is zero in the camera frame. In this case, the camera cannot measure the position of the leader. As such, it’s difficult to implement the leader-follower tracking system using the method location-feature-position for our drone-robot tracking problem.

Figure 8 Trajectories of the follower and leader.
2.5 Conclusions

The location-feature-area method uses the area of one ball and the bearing angle $\beta$ to scale the leader target’s location in the camera frame to address the leader-follower tracking problem. However, the relative position of the follower and leader cannot be obtained. The location-feature-position method uses three features to calculate the target’s relative position. Due to the requirement of three features, it is still unrealistic to implement in our research. Therefore, it is necessary to find a new effective approach to tracking using camera vision. The method can provide the accurate relative position of a detected object corresponding to the camera. Meanwhile, the approach only needs to use one camera and one feature. In this thesis, we use one camera and one feature to locate the target and then build a controller to address the leader-follower tracking problem. In Chapter 3, we describe how to locate a target using one camera and one ball.

Figure 9 Simulation result. $e_x$ is the image error, $x_e$ and $y_e$ are the formation error.
CHAPTER 3
LEADER-FOLLOWER TRACKING ALGORITHM BASED ON A SINGLE SPHERE FEATURE

3.1 Introduction

In this section, we build a leader-follower tracking system with one camera installed on the follower and one sphere placed on the leader. In this tracking system, the leader’s position need to be measured using a leader-follower tracking algorithm. Through the camera measurement and the geometry analysis between the follower and the leader, we can obtain the leader’s location. Based on the estimated leader’s position, we can build a control system on the follower to track the leader. As shown in Figure 10, we expect the follower to track the leader with the distance $L$ and bearing angular $\lambda$. $L$ is the distance between the camera’s center $O_l$ and the sphere’s center $O_c$. $\lambda$ is the bearing angular between the heading of the follower and $O_cO_l$. In order to keep the feature in the camera view, we also build a P-controller for a pan camera which just moves on its pan direction.

![Diagram of leader-follower tracking](image)

Figure 10 The formation of leading-follower tracking.
In this section, we first show how to use one camera to measure the sphere’s relative position with respect to the follower and calculate the bearing angle $\beta$ between the directions of the follower and the leader. Then, we build P-controllers that allow the follower to track the leader based on parameters that the camera measures.

3.2 Estimation of Sphere’s Position in the Camera Coordinate System

In this section, we focus on the procedure to measure the leader’s position using the camera vision in the camera’s coordinates. First, we show the projection of a sphere on the image plane. Then we briefly describe the relationship between the image coordinate and the camera coordinate and introduce the procedures to transfer the sphere’s location from the image coordinate to the camera coordinate.

3.2.1 Projection of a Sphere on the Image Plane

Figure 11 shows the projection of a sphere on the image plane, as marked in yellow. In fact, the projection is an ellipse which is the intersection plane of the image plane and the green cone. In this figure, $O_s$ is the sphere’s center with location $(X_s, Y_s, Z_s)$ in the camera’s coordinate. $R_s$ is the radius of the sphere, and $\gamma$ is the bearing angle between the vector $O_cO_s$ and the principal axis $X_c$, where $O_c$ is the optical center.

As shown in Figure 10, the outline of the sphere’s projection (yellow ellipse) is the projection of the green circle, which is the intersection of the sphere and the green cone tangent to the sphere. In paper [11], the ellipse is expressed by the following function:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \quad (3-1)$$

Where $a \approx \frac{Z_s}{R_s}, b \approx \frac{Z_s}{R_s}\cos(\delta)$. The area $A$ of the ellipse is $A = \pi ab$.
Next, let us show the calculation of the sphere’s center location \((X_s, Y_s, Z_s)\).

### 3.2.2 Relationship between the Image Coordinates and the Camera Coordinates

Suppose \((m_i, n_i)\) is the location of a point in the image coordinate, then it has the following relationship with its corresponding location \((X_i, Y_i, Z_i)\) in the camera coordinate \(X_cY_cZ_c\) (see the Figure 11), according to the camera projection model:

\[
m_i = f_y \frac{Y_i}{X_i} + m_o \tag{3-2}
\]

\[
n_i = f_z \frac{Z_i}{X_i} + n_o \tag{3-3}
\]

Where \(f_y = f/\alpha_u\) and \(f_z = f/\alpha_v\), \(f\) is the focal length, \(\alpha_u, \alpha_v\) are the pixel dimensions, and \((m_o, n_o)\) is the location of the principal point of the camera in the image coordinates.

### 3.2.3 Estimation of the Sphere Position from an Image

The procedures introduced in paper [11] estimates the location of sphere’s center \((X_s, Y_s, Z_s)\) in the camera coordinates are shown as follows.
1) Estimate the center of the sphere’s projection \((m_s, n_s)\), and its area \(A\).

2) According to equations 3-2,3-3, it’s easy to obtain

\[
\frac{y_s}{x_s} = \frac{m_s - m_o}{f_y} \tag{3-4}
\]

\[
\frac{z_s}{x_s} = \frac{n_s - n_o}{f_z} \tag{3-5}
\]

3) Denote \(\frac{y_s}{x_s}\) as \(y_s\), and \(\frac{z_s}{x_s}\) as \(z_s\), then the bearing angle \(\gamma = -\arctan \sqrt{y_s^2 + z_s^2}\).

4) Calculate the area \(A\) of the sphere’s ellipse projection by \(A = N / (f_y f_z)\), where \(N\) is the number of pixels in this area.

5) Compute \(X_s = R_s \sqrt{\frac{\pi}{(A \cos \gamma)}}\). Then \(Y_s = y_s X_s\), \(Z_s = z_s X_s\).

Figure 12 Leader-follower formation.
Table 1 Definitions of Notations used in Figure 12

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_l )</td>
<td>the heading of the leader</td>
</tr>
<tr>
<td>( Y_f, Y_f' )</td>
<td>the heading of the follower at ( T_0, T_1 )</td>
</tr>
<tr>
<td>( O_f )</td>
<td>the center of the follower</td>
</tr>
<tr>
<td>( O_c, O_c' )</td>
<td>the optical center of the camera</td>
</tr>
<tr>
<td>( d )</td>
<td>the distance between ( O_f ) and ( O_c )</td>
</tr>
<tr>
<td>( O_l )</td>
<td>the center of the leader at ( T_0 )</td>
</tr>
<tr>
<td>( O_l' )</td>
<td>the center of the leader at ( T_1 )</td>
</tr>
<tr>
<td>( OXY )</td>
<td>The world coordinates</td>
</tr>
<tr>
<td>( O_cX_fY_f )</td>
<td>robot frame of the follower at ( T_0 )</td>
</tr>
<tr>
<td>( O_c'X_f'Y_f' )</td>
<td>robot frame of the follower at ( T_1 )</td>
</tr>
<tr>
<td>( O_cX_cY_c )</td>
<td>camera frame at ( T_0 )</td>
</tr>
<tr>
<td>( O_c'X_c'Y_c' )</td>
<td>camera frame at ( T_1 )</td>
</tr>
<tr>
<td>( \alpha_0, \alpha_1 )</td>
<td>the orientation of the camera corresponding to the follower at time ( T_0 ) and ( T_1 )</td>
</tr>
<tr>
<td>( O_lY_l )</td>
<td>the direction of the leader</td>
</tr>
<tr>
<td>( O_lO_l' )</td>
<td>Direction vector of the leader’s motion</td>
</tr>
<tr>
<td>( X_{c0}, Y_{c0}, Z_{c0} )</td>
<td>the camera coordinate of ( O_l ) at ( T_0 )</td>
</tr>
<tr>
<td>( X_{c1}, Y_{c1}, Z_{c1} )</td>
<td>the camera coordinate of ( O_l' ) at ( T_1 )</td>
</tr>
</tbody>
</table>

3.3 Estimation the Bearing Angle \( \beta \) and the Relative Position of the Sphere

In this section, we introduce the process to estimate the bearing angle \( \beta \) and the relative position of the sphere in the follower’s coordinates. In order to measure \( \beta \), we need to measure the relative position of the sphere and transfer the positions of the sphere at time \( T_1 \) and \( T_0 \) into the follower’s coordinate \( O_cX_f'Y_f' \) (see Figure 12).

3.3.1 Transfer Sphere’s Position at Time \( T_0 \) to the Follower’s Coordinates \( O_cX_f'Y_f' \)

In this section, we show the transformation of the sphere’s location from the image coordinates to the follower’s coordinate \( O_cX_f'Y_f' \) at \( T_0 \). Denote the location of the sphere and its area obtained by the camera at time \( T_0 \) as \((m_{t0}, n_{t0})\) and \(A_0\), respectively. According to Section 3.2, we can get the sphere’s center location \((X_{c0}, Y_{c0}, Z_{c0})\) in the camera coordinates.
In order to transfer \((X_{c0}, Y_{c0})\) to the coordinate \(O^*_cX^*_fY^*_f\), we first need to transfer it to the coordinate \(O_cX_fY_f\). This can be realized through coordinate space transformation as follows.

\[
\begin{bmatrix}
  X'_{c0} \\
  Y'_{c0}
\end{bmatrix} = R(\pi/2 - \alpha_0) \begin{bmatrix}
  X_{c0} \\
  Y_{c0}
\end{bmatrix}
\]

Where \(\alpha_0\) is the bearing angle among \(X_c\) and the direction \(Y_f\) at time \(T_0\), \((X'_{c0}, Y'_{c0})\) is the location of the sphere’s center in \(O_cX_fY_f\). \(R\) is the rotation matrix shown as follows

\[
R(\cdot) = \begin{bmatrix}
  \cos(\cdot) & -\sin(\cdot) \\
  \sin(\cdot) & \cos(\cdot)
\end{bmatrix}
\]

Now we can calculate the location of the sphere’s center in the coordinate \(O^*_cX^*_fY^*_f\). In order to achieve this, we first need to obtain the moving direction of the follower from \(T_0\) to \(T_1\), which is approximated as the direction of the vector \(O_cO^*_c\) in \(OXY\) (see Figure 12). Suppose the velocity and angle velocity of the follower are \(v\) and \(w\), which can be obtained by encoders. The direction angular of the follower in \(OXY\) is \(\theta\). Then, \(O_cO^*_c = (\Delta x, \Delta y)\) can be obtained by following equations

\[
\Delta x = (v\cos(\theta + w\Delta T) + dw\cos(\theta + w\Delta T)) \cdot \Delta T \tag{3-8}
\]

\[
\Delta y = (v\sin(\theta + w\Delta T) + dw\sin(\theta + w\Delta T)) \cdot \Delta T \tag{3-9}
\]

where \(\Delta T = T_1 - T_0\) and \(d\) is the distance between the center of the follower and the camera. Then we can get the location \((X_0, Y_0)\) of sphere’s center \(O_t\) in the coordinate \(O^*_cX^*_fY^*_f\) as follows

\[
\begin{bmatrix}
  X_0 \\
  Y_0
\end{bmatrix} = R(\pi/2 - \alpha_0 - w\Delta T) \begin{bmatrix}
  X_{c0} \\
  Y_{c0}
\end{bmatrix} - R(\pi/2 - \theta - w\Delta T) \begin{bmatrix}
  \Delta x \\
  \Delta y
\end{bmatrix} \tag{3-10}
\]

3.3.2 Transfer Sphere’s Location at Time \(T_1\) To The Follower’s Coordinate \(O^*_cX^*_fY^*_f\)

In this section, we introduce the transformation of the sphere’s location \(O_t\) from the image coordinate to the follower’s coordinate \(O^*_cX^*_fY^*_f\). Denoting the position of the sphere and its area obtained by the camera at time \(T_1\) as \((m_{t1}, n_{t1})\) and \(A_1\) respectively, we can follow the procedures introduced in Section 3 to calculate the sphere’s center location \((X_{c1}, Y_{c1}, Z_{c1})\). Then the sphere’s center location in the follower’s coordinate \(O^*_cX^*_fY^*_f\), denoted as \((X_1, Y_1)\), can be
obtained through simple coordinate space transformation. The mathematical equation is shown as follows.

\[
\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = R(\pi/2 - \alpha_1) \begin{bmatrix} X_{c1} \\ Y_{c1} \end{bmatrix}
\]  

(3-11)

where \( \alpha_1 \) is the bearing angle between axes \( X^*_c \) and \( Y^*_f \) at time \( T_1 \) (see Figure 12).

3.3.3 Estimation of the Bearing Angle \( \beta \)

Having obtained the location of the sphere expressed in the follower’s coordinate \( O_c^*X^*_fY^*_f \) at \( T_0 \) and \( T_1 \), we can now calculate the bearing angle \( \beta \) between the follower’s direction \( Y^*_f \) and the motion vector \( O_lO^*_l \) of the leader, using the following equation:

\[
\beta = \begin{cases} 
\arctan \left( \frac{X_1-X_0}{Y_1-Y_0} \right) & \text{if } Y_1 - Y_0 > 0 \\
\arctan \left( \frac{X_1-X_0}{Y_1-Y_0} \right) + \pi & \text{if } Y_1 - Y_0 < 0 \text{ and } X_1 - X_0 \geq 0 \\
\arctan \left( \frac{X_1-X_0}{Y_1-Y_0} \right) - \pi & \text{if } Y_1 - Y_0 < 0 \text{ and } X_1 - X_0 < 0 \\
\frac{\pi}{2} & \text{if } Y_1 - Y_0 = 0 \text{ and } X_1 - X_0 > 0 \\
-\frac{\pi}{2} & \text{if } Y_1 - Y_0 = 0 \text{ and } X_1 - X_0 < 0 \\
0 & \text{if } Y_1 - Y_0 = 0 \text{ and } X_1 - X_0 > 0 
\end{cases}
\]  

(3-12)

3.4 Controller Design

In this section, we design P-controllers for the leader-follower tracking system using camera vision. As shown in Figure 12, we expect the follower tracks the leader with the distance \( L \) and bearing angular \( \lambda \). \( L \) is the distance between the camera’s center \( O_l \) and the sphere’s center \( O_c \). \( \lambda \) is the bearing angular between the heading of the follower and \( O_cO_l \). In the process of leader-follower tracking system, the follower need to adjust its velocity \( v_f \) and angular velocity \( w_f \) to keep itself at the desired position corresponding to the leader. The driver model of the robot can influence how to adjust \( v_f \) and \( w_f \) so that it is necessary to comprehend the kinematics of the follower robot. In this project, we apply the controller algorithm on a
differential driver robot with DC motors. Therefore, Section 3.4.1 introduces the differential drive kinematics and DC motor model. Then, Section 3.4.2 focuses on the P-controller design.

3.4.1 The Differential Drive Kinematics and DC Motor Model

The differential drive kinematics model [13] is:

\[ w(t) = \frac{v_r(t) - v_l(t)}{b} \]  \hspace{1cm} (3-13)

\[ v(t) = \frac{v_r(t) + v_l(t)}{2} \]  \hspace{1cm} (3-14)

where \( w(t) \) is the angular velocity of the robot, \( v(t) \) is the velocity of the robot, \( v_r(t) \) and \( v_l(t) \) are the velocity of the wheel at the right side and left side, \( b \) is the distance from the left-side wheel to the right-side wheel. According to Equations 3-13 and 3-14, we can obtain the following equations:

\[ v_r(t) = \frac{2v(t) + bw(t)}{2} \]  \hspace{1cm} (3-15)

\[ v_l(t) = \frac{2v(t) - bw(t)}{2} \]  \hspace{1cm} (3-16)

Assuming the radius of the wheel is \( R \), we can get the angular velocity of the wheels are:

\[ w_r(t) = \frac{v_r(t)}{R} \]  \hspace{1cm} (3-17)

\[ w_l(t) = \frac{v_l(t)}{R} \]  \hspace{1cm} (3-18)

where \( w_r(t) \) is the angular velocity of the wheel at the right side, \( w_l(t) \) is the angular velocity of the right side.

The DC motor model [14] is

\[ v_m = e + R_a i_a + L_a \frac{di_a}{dt} \]  \hspace{1cm} (3-19)

\[ J \frac{dw_m}{dt} + B_l w_m = T_e - T_l \]  \hspace{1cm} (3-20)

\[ T_e = K_b i_a \]  \hspace{1cm} (3-21)

where \( v_m \) is the input voltage of the motor, \( e \) is the induced emf, \( R_a \) is the resistance of the motor, \( i_a \) is the current of the motor, \( L_a \) is a self–inductance, \( w_m \) is the motor’s angular speed,
J is the inertia moment of the motor, $B_l$ is the viscous friction coefficient, $T_e$ is the air gap torque of the motor, $T_l$ is the lode toque, $K_b$ is the induced emf constant. To transform equation (3-19, 3-20) into their Laplace equations, we can get the following equations

$$I_a(s) = \frac{V_m(s) - K_b w_m(s)}{R_a + sL_a} \quad (3-22)$$

$$w_m(s) = \frac{K_b I_a(s) - T_l(s)}{(B_l + sJ)} \quad (3-23)$$

Formulations 3-22 and 3-23 indicate that the voltage of motor can control the angular velocity of a wheel. In fact, PWM (pulse-width modulation) is often used to adjust the voltage of a motor. Therefore, we can use the value of PWM to control the angular velocity of a motor. Since the angular velocity $w_l, w_f$ decide the velocity and direction of the follower, we can control the value of PWM to control the follower’s movement.

### 3.4.2 Control Design for the Follower and Camera

Our goal is that the follower tracks the leader with the desired bearing $\lambda_d$ and desired length $L_d$. According to function 3-11, the camera measures the sphere’s position $(X(k), Y(k))$ at time $k$ in the follower’s coordinate. The distance $L(k)$ between the camera and the sphere is:

$$L(k) = \sqrt{X^2(k) + Y^2(k)} \quad (3-24)$$

The bearing angular $\lambda(k)$ between the heading of the follower and the camera’s direction is:

$$\lambda(k) = atan2(X(k), Y(k)) \quad (3-25)$$

Setting the setpoint $(L_d, \lambda_d)$, we can build a P-controller algorithm for leader-follower tracking system:

$$v_f(k) = vgain(L(k) - L_d) \quad (3-26)$$

$$w_f(k) = wgain(\lambda_d - \lambda(k)) \quad (3-27)$$

where $vgain$ and $wgain$ are the gain of the P-controller. According to Equations from 3-13 to 3-18, we obtain the desired angular velocities $w_r(k), w_l(k)$ of the follower’s wheels. The
current angular velocities $w_{rc}(k), w_{lc}(k)$ are measured by encoders installed on the shaft of the motors. Likewise, we design a new P control based on the error between $w_r, w_l$ and $w_{rc}, w_{lc}$ in order to make the rotational speeds of motors keep pace with their desired rotational speeds. The P-controller algorithm for motors is:

$$pwm_r(k) = prgain(w_r(k) - w_{rc}(k))$$  \(3-28\)

$$pwm_l(k) = plgain(w_l(k) - w_{lc}(k))$$  \(3-29\)

where $prgain$ and $plgain$ are the gains of the P-controller. Then we obtain the input PWM of the motors:

$$PWM_r(k) = PWM_r(k - 1) + pwm_r(k)$$  \(3-30\)

$$PWM_l(k) = PWM_l(k - 1) + pwm_l(k)$$  \(3-31\)

where $PWM_r$ and $PWM_l$ are the input PWMs of motors at the time $k$, $PWM_r(k - 1)$ and $PWM_l(k - 1)$ are the input PWM of motors at the time $k - 1$. In the whole process of leader-follower tracking system, the camera needs to track the sphere on the leader and keep the ball in the camera view. Otherwise, the camera will lose the target and cannot estimate the relative position of the leader. As a result, it is necessary to build a controller to make the camera keep pace with the ball’s movement. We also build a P-controller algorithm for the camera which moves on its pan direction. The desired position of the ball is the center point of the camera view in the pixel coordinates. Denote the center point of the camera view is $m_o$ at the pan direction of the camera. The P-control of the camera is designed as follows:

$$w_c(k) = cgain(m_o - m(k))$$  \(3-32\)

where $cgain$ is the feedback gain, $m(k)$ is the position of the ball in the camera frame at the time $k$, and $w_c(k)$ is the angular velocity of the camera.

3.5 Conclusion

In this chapter, we apply one camera and one ball to locate the leader’s position corresponding with the follower. Meanwhile, through transferring the positions of the leader
into the follower’s coordinates, we obtain the bearing angular $\beta$ between the headings of the follower and leader. Then, we apply the information from the camera to build P-controllers for the leader-follower tracking system. In the next chapter, we focus on the simulation and implementation for leader-follower tracking system using camera vision.
CHAPTER 4
SIMULATION AND IMPLEMENTATION

In this section, we focus on the simulation and implementation of the leader-follower tracking system using camera vision. This section is organized as follows. Chapter 4.1 includes the procedure and results of simulation studies using MATLAB. Chapter 4.2 includes the hardware and software designs utilized in this research. Chapter 4.3 describe the procedure to detect an object using a Pi camera. Chapter 4.4 includes the procedure and results of the implementation for the leader-following tracking system.

4.1 Simulation

According to Chapter 3, we design a diagram in Appendix 1 to achieve the leader-following tracking task using one camera and one ball. In this figure, \((x_f, y_f)\) is the position of the follower, \(e_L = L_d - L(k)\), and \(e_\lambda = \lambda_d - \lambda(k)\). Furthermore, to simulate the leader-follower tracking system using one camera and one ball, we build the following codes to simulate in MATLAB.

<table>
<thead>
<tr>
<th>Step 1: Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 (L_d = 1.414) m, (\lambda_d = 0.7) radian, (m_0 = 243.942), (n_o = 158.441)</td>
</tr>
<tr>
<td>(f_y = 497.157), (f_z = 498.684), (R_x = 0.06) m, (v_f(1) = 0), (w_f(1) = 0),</td>
</tr>
<tr>
<td>(PWM_f(1) = 0), (PWM_r(1) = 0), (x_f(1) = 50), (y_f(1) = 25), (total = 1000)</td>
</tr>
<tr>
<td>(\phi(1) = 0), (\theta(1) = \pi/2);</td>
</tr>
<tr>
<td>Note: (R) is the radium of the ball;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 2: Object detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2 For (k = 1): total</td>
</tr>
<tr>
<td>2.1 The camera obtains the position ((m, n)) of the ball in the pixel coordinate;</td>
</tr>
<tr>
<td>2.2 The camera measures the ball’s position ((X_{c0}, Y_{c0}, Z_{c0})) in the camera Coordinates</td>
</tr>
<tr>
<td>2.3 [ \begin{bmatrix} X(K) \ Y(K) \end{bmatrix} \left(R(\pi/2 - \phi(k)) \begin{bmatrix} X_{c0} \ Y_{c0} \end{bmatrix} \right) ]</td>
</tr>
<tr>
<td>2.4 (L(k) \leftarrow 2\sqrt{X^2(k) + Y^2(k)})</td>
</tr>
<tr>
<td>(\lambda(k) \leftarrow \text{atan2}(X(k), Y(k)))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 3: Calculate the velocity and angular velocity of the follower</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 (e_L \leftarrow L_d - L(k))</td>
</tr>
<tr>
<td>(e_\lambda \leftarrow \lambda_d - \lambda(k))</td>
</tr>
</tbody>
</table>
Through P-controller for the follower, calculate \( v_f, w_f \)

**Step 4:** Update the position of the follower

| 3.2 | 4.1 \( \theta(k+1) = \theta(k) + w_f \delta \); \% \( \delta \) is the sampling time |
| 3.2 | 4.2 \( x_f(k+1) \leftarrow x_f(k) + v_f \cos(\theta(k+1)) \delta \) |
| 3.2 | 4.3 \( y_f(k+1) \leftarrow y_f(k) + v_f \sin(\theta(k+1)) \delta \) |

**Step 5:** Update the bearing angular \( \phi \)

| 5.1 | \( e_m \leftarrow m_o - m \) |
| 5.2 | \( w_c \leftarrow e_m \text{c}g \text{i}a \) |
| 5.3 | \( \phi(k+1) \leftarrow \phi(k) + w_c \delta \) |
| 5.4 | Go back step 2 |

As shown in Figure 13, the follower tracked the leader to move. In this simulation, the trajectory of the leader robot is created by the smooth-turn model [10]. The initial position and heading angle of the follower robot are \((50,25)\) and \(\pi/2\). The initial position and heading angle of the sphere ball are \((51,26)\) and \(\pi/2\). Figure 14 shows that \(e_L, e_\lambda\) and \(m\) are close to zero, the camera tracks the ball and keep the ball around the center of the camera view. As a result, the follower tracks the leader at the desired position.

![The trajectories of the follower and leader](image.png)

*Figure 13* The trajectories of the follower and leader.
4.2 Hardware and Software Designs

This section introduces the hardware and software designs which are used in the implementation.

4.2.1 Raspberry Pi 2

Raspberry Pi 2 (see Figure 15) is a cheap and powerful microcontroller with a 900MHz quad-code CPU and 1 GB RAM. Raspberry Pi 2 has excellent capability to calculate data and deal with image information. Furthermore, Raspberry Pi 2 have a friendly development environment. Users can build a project in the Linux system of Raspberry using Pi 2 Python, c, c++ and Java. Raspberry Pi 2 also supports the openCV library which is an open source library for computer vision and machine learning. In addition, there are 4 USB ports and 40 GPIO pins on Raspberry Pi 2. These interfaces make Raspberry Pi 2 connect conveniently with other devices, such as camera, sensors. In this research, Raspberry Pi 2 is used to detect the target and locate the position of the target in the camera frame.

Figure 14 Simulation result. $e_x, e_L$ the formation error; L The relative distance between the camera and the ball; m the position of the ball in the camera view.
4.2.2 Arduino UNO

Arduino UNO is a user-friendly microcontroller built on the ship ATmega328P. As shown in Figure 16, it just has 16 I/O pins which include 6 PWM output and 16M clock speed. However, the UNO can work with lots of sensors and other devices through UART, I2C. To control motors’ speed, Arduino UNO sends a value in the range (0~255) to PWM output pin. In addition, the PWM output from Arduino UNO is more stable than that from Raspberry Pi 2. In the Arduino forum, there are rich materials to help people to study and use this microcontroller. In this project, the UNO play a big role. It calculates the
wheels’ speed with encoders which are installed on the motors’ shafts. It calculates the target’s position for the adjustment of PWM output which decides the velocities of motors. It sends the angular velocity of the camera servo to 16 channel PWM module which keeps the camera to track the target.

4.2.3 16 Channel PWM /Servo Shield

As shown in Figure 17, it is a 16-channel PWM/Servo Shield. The shield assists microcontroller to control servos by a stable PWM output. The 16 channel PWM /Servo Shield has the following advantages. It can control 16 channel and up to 992 servos using I2C. The shield is convenient to get commands from microcontrollers such as Arduino, Raspberry Pi through I2C. It can automatically control servos without keeping communication with microcontrollers so that microcontrollers can save time to execute other tasks. In this project, the shield is used to control camera’s servo. The reason is Raspberry PI 2 cannot offer a stable PWM output, and Arduino needs to process data as fast as possible.

![16-channel PWM/Servo Shield](image)

**Figure 17** 16-channel PWM/Servo Shield.
4.2.4 PI Camera

PI camera (see Figure 18) is an assistant device for Raspberry PI 2. The parameters of PI camera can be adjusted by Raspberry PI 2 though python command such as video formats, picture formats, and exposure modes. To compare with other webcams, PI camera is a cheaper and an excellent camera. The sensor resolution for a PI camera achieves up to 2592*1944, and its video modes can change in 1080p30, 720p60, and 640*480p60/90. PI camera is suitable to work with openCV and Raspberry PI 2. In this project, the PI camera catches the image of the target and sends the image information to Raspberry PI 2.

4.2.3 Rover 5 Motor Driver Board

As shown in Figure 19, it is a Rover 5 Motor Driver Board. The motor driver designed by Dagu can drive four motors with microcontrollers, such as Arduino, Raspberry Pi. The driver board can change the direction and speed of motors. In this research, the driver board connects with Arduino to control the direction and speed of wheels.

4.2.4 Lynxmotion Quadrature Motor Encoder
As shown in Figure 20, it is a Lynxmotion Quadrature Motor Encoder. It includes two output signals. The encoder has 100 cycles per revolution and 400 quadrature counts per revolution. In this research, the motor encoder accounts the speed of wheels and sends the speed data to Arduino to calculate the velocity and angular velocity of the follower robot.

4.2.5 OpenCV

OpenCV stands for Open Source Computer Vision Library, which includes 2500 optimized algorithm on computer vision and machine learning. It supports the current main operation systems like Windows, Linux, and Mac OS. Meanwhile, it supports the following languages: C++, C, Python, Java and MATLAB. It is convenient for the user to use the algorithms of openCV on vision application so that users don’t need to build a complex algorithm on the computer vision and machine learning every time. Also, the openCV is a free software. In fact, many users and companies are using openCV to develop projects and applications such as robot’s navigation and face recognition. In this project, the openCV plays the main role to identify the sphere target from the camera view. The process of target detection will be described in the following section.

4.3 Detecting the Target Using Raspberry Pi 2 and OpenCV

This section focuses on the process of target detection using Raspberry Pi 2 and openCV. Raspberry Pi 2 receives image information from Pi camera and then calls for algorithms from openCV to implement these goals: color detection, smooth image, contours of images, shape recognition, locating the target’s center, and area calculation in the camera frame. The remainder of this section elaborates this procedure.

4.3.1 Color Detection
Color detection refers to distinguishing an object by color. Apparently, an image consists of color components. To detect a target with a special color, it should separate an image into color components and then select the detected color components. In this research, an image is recorded by a Pi camera and sent to Raspberry Pi in an array of color components. The color element is presented by RGB color model as shown in Figure 21.a. RGB stands for red, green, and blue. In RGB color model, any color component consists of the values of these three colors. RGB color model is used to display images on electronic devices, such as a computer, phone, and tablet. However, it is difficult for people to dictate the RGB value of a color component. Therefore, it needs to convert the color space of the image into HSV color space shown in Figure 21.b. HSV stands for hue, saturation and value. The HSV color model is converted from RGB color model. To compare it with RGB color model, it is easier for people to describe a color component in HSV color model. Likewise, this conversation from RGB to HSV makes the separation of color components easily.

![Figure 21 Illustration of RGB and HSV color space](image)

(a) RGB color space. (b) HSV color space.

In the implementation, the detected color is red and its HSV range is from [105, 65, 65] to [125, 255, 255]. Furthermore, the image needs to be transformed from 3-dimension to 1-dimension in order to reduce the calculation load to smoothen an image in next step. The pixel
color will be white in the detected HSV range, or the pixel color will be black. Therefore, the color image changes into a gray image. As shown in Figure 22, Our target is the red ball. The color in the detected range changes into white and other becomes black. Obviously, the color detection cannot distinguish the target from objects with the similar colors.

![Figure 22 Illustration of color detection. (a) The target ball and the noise rectangle in the normal picture. (b) The result of color detection.](image)

4.3.2 Smooth image

As shown in Figure 22, the edges of the white blocks are irregular. Noises produce edges similar to a gear edge, and impacts correct measurement of the positions and areas of these blocks. Therefore, it is necessary to remove noises in the gray image. There are four

![Figure 23 Illustration of Median Filtering. The edges of blocks become smooth.](image)
methods to reduce noise in openCV: averaging, Gaussian Filtering, Median Filtering, and Bilateral Filtering. In this research, the Median Filtering is chosen to blur an image and reduce noise. The result of the smoothened image in Median Filtering is shown in Figure 23. Clearly, from the comparison between Figure 22 and Figure 23, the edges of blocks in Figure 23 are very smooth.

4.3.3 Contours in Images

Contours mean boundaries of the detected object. The contour is an essential requirement to recognize and analyze detected shapes in openCV. The method of contours is to distinguish while block from the black background. As shown in Figure 24, the contours present the shapes of the detected target and noise.

![Illustration of contours in the image](image)

(a) Picture from camera. (b) contours of objects.

4.3.4 Shape Recognition

The shape recognition is a useful approach to distinguish the detected target from noise. Although the contours of blocks have been drawn in Figure 24, it doesn’t yet distinguish the detected sphere ball from noise. As the contour of identified target is close to a circle, next step is to compare contours with a circle and then obtain the shape ratios, which means the unlike ratio between two shapes. The unlike ratio between two shapes. In this research, the contours with shape ratio less than 0.01 identify the targets to be detected. As shown in Figure 26, the shape ratio of the approximate circle is less than 0.01, and the shape ratio of noise is almost
0.1. Sometimes, noises cause objects to have similar shapes with the shape to be detected. It is difficult to select the target from noise just according to the shape ratio. The solution will be described in the following section.

4.3.5 Position and Area of the Target

Thought the algorithm of the image moments in the openCV library, the position and area of the contour are easy to be calculated. We still need to eliminate some noise which is similar with the shape of the detected target. In fact, though color detection and shape recognition, the area of noise is almost less than that of the detected target. Therefore, the detected target can be enumerated by the areas of blocks.

4.4 Implementation Result

The section focuses on the implementation result of the leader-follower tacking system using one camera and one ball. In Chapter 3, the P-controllers of the follower and camera has been built. In Section 4.3, the approach of target detection has been described. According to the diagram of hardware (see Figure 26), we build and test the hardware system. The desired
length $L_d$ is 60 cm, and the desired angular $\lambda_d$ is zero. The trajectories of the follower and the ball are shown in Figure 27.

Figure 26 Illustration of the hardware connect in the implementation.

Figure 27 The trajectories of the leader and follower. The red line is the trajectory. The red ball is the detected target.
Figure 28 reports the implementation result. However, the follower tracks the ball to move and achieve the leader-follower tacking task using one camera and one ball.

According to the Figure 28, the follower tracks the ball to move. But we need to adjust the feedback gains to improve the performance of the leader-follower tracking system.
CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this research, we developed a method to locate a sphere using camera vision. This approach is used to estimate the relative position and bearing angular between the headings of the leader and follower. This research also builds P-controllers to realize the leader-follower tracking task using camera vision. The simulation result validates our algorithm. Then, we implemented this algorithm in a real robot system. We develop a system for the target detection by Pi camera, Raspberry Pi 2 and openCV.

The results of the simulation and implementation indicate the viability of the location method using one camera and one sphere. Meanwhile, we used our location method and built a controller for leader-follower tracking system with reasonable performance.

Our location approach just uses one camera and one ball and can provide the relative position between the camera and detected target. It is convenient for users to apply this approach on the leader-follower tracking system.

5.2 Future Work

We need still to improve the performance of the leader-follower tracking system using camera vision. The authors will design an advanced controller to replace the simply P-controller, such as the backstepping controller and adaptive controller, to achieve a better tracking performance. Finally, we achieve this task that drone tracks the robot on the ground as well as possible.
APPENDIX

DIAGRAM OF LEADER-FOLLOWER TRACKING
Figure 13 The diagram of the leader-follower tracking.
REFERENCES


