An Experiment on Vision-Based Leader-Following for Multiagent Systems

Dzung Tran*, Si Dang†, Tansel Yucelen‡, and Sriram Chellappan§

Addressing problems related to multiagent systems like formation control, flocking, and leader-following in the absence of networked communications is a challenging task. In this paper, we consider a simple-yet-effective vision-based methodology for solving the leader-following problem as applied to a group of ground mobile robots. Specifically, each robot is equipped with a camera module to recognize and estimate the distance to its leader, then the proposed control architecture allows each robot to adjust itself for tracking and maintaining the desired distance. Finally, the proposed control algorithm is implemented on ground robot platforms with low-cost camera modules for experimental validation.

I Introduction

Multiagent systems have received enormous attention from multidisciplinary research communities in the past two decades. Most papers on this topic consider agents having the capability to exchange information with their neighbors. This is understandable owing to the fact that contemporary technology allows such ease of communication. However, there are many scenarios where information exchange is inhibited or inoperative. For such cases, agents are expected to rely only on their on-board sensors to perceive the situation and environment for accomplishing their missions; yet, solving multiagent problems in the absence of communication is a challenging task. One of the most useful and affordable sensors that can be equipped for each agent for such tasks is camera.

There have been a number of studies on solving multiagent system problems with vision, ranging from using omnidirectional cameras for tracking leader(s) with different techniques like optical flow, visual servoing, etc. (see, for example, Refs. 1–4 and references therein) to the trend with pinhole camera models (see, for example, Refs. 5–7 and references therein). In particular, the techniques used in Refs. 6 and 7 require some feature object mounting on top of the leader robot couple with the known intrinsic parameters of the camera to estimate the distance to the leader and to adjust the movement accordingly. In this paper, we consider a vision-based methodology for solving the leader-following problem as applied to a group of ground mobile robots. Specifically, taking advantage of the developments in computer vision, we train the follower

*D. Tran is a Graduate Research Assistant of the Mechanical Engineering Department and a Member of the Laboratory for Autonomy, Control, Information, and Systems (LACIS, `http:/http://lacis.eng.usf.edu/`) at the University of South Florida, Tampa, Florida 33620, United States of America (emails: `dtran3@mail.usf.edu`).

†S. Dang is an Undergraduate Student of the Computer Science and Engineering Department at the University of South Florida, Tampa, Florida 33620, United States of America (emails: `sidang@mail.usf.edu`).

‡T. Yucelen is an Assistant Professor of the Mechanical Engineering Department and the Director of the Laboratory for Autonomy, Control, Information, and Systems (LACIS, `http://http://lacis.eng.usf.edu/`) at the University of South Florida, Tampa, Florida 33620, United States of America (email: `yucelen@usf.edu`).

§S. Chellappan is an Associate Professor of the Computer Science and Engineering Department at the University of South Florida, Tampa, Florida 33620, United States of America (email: `sriramc@usf.edu`).

¶This work was supported in part by The Florida High Tech Corridor Council. Any opinions, conclusions and recommendations in this paper are those of the authors alone, and not necessarily those of funding agency.

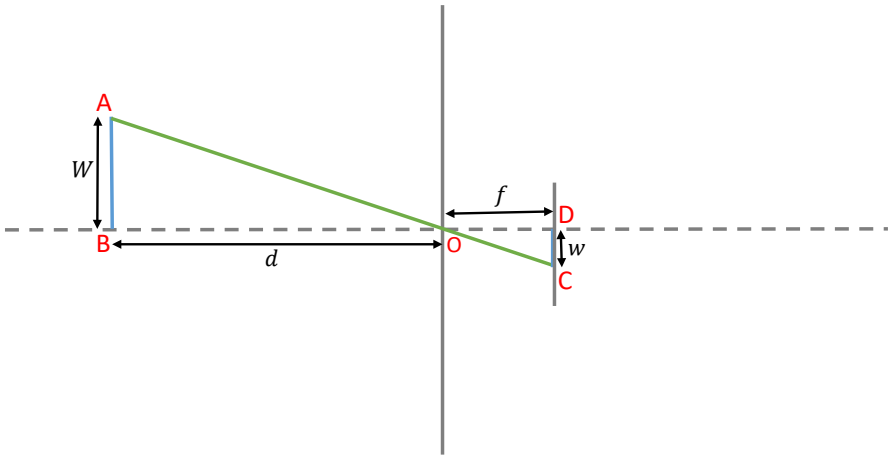


Figure 1. The pinhole camera model.

to recognize and locate the leader robot on the image frames. We then introduce a simple procedure to calibrate the cameras and obtain the estimated distance to the leader without the need of feature objects. The control algorithm to track the leader is also presented. Finally, the proposed architecture is implemented on a group of Khepera IV ground mobile robots equipped with a Raspberry Pi and a low-cost camera module for experimental validation.

II Distance Estimation Method and Control Architecture

In this paper, we consider that there exists a leader agent in the robot swarm that is tracking a dynamic target and each follower has the capability to recognize and choose a neighboring agent as its leader to track and maintain the desired distance. Each robot uses the information obtained from its camera module to achieve this objective. To this end, we first discuss the vision-based methodology for each follower robot to estimate the distance to its leader and then the control algorithm to achieve and maintain the desired distance.

A Calibration Procedure and Distance Estimation Method

For estimating the distance from the follower to the leader, we consider the pinhole camera model as illustrated in Figure 1. Clearly, the triangles AOB and COD are similar. Therefore, the ratio given by

$$\frac{W}{d} = \frac{w}{f}, \tag{1}$$

holds, where W is the width (or diameter) of the leader robot, d is the distance from the leader robot to the camera (or the follower robot), f is the focal length of the camera, and w is the width (or diameter) of the leader robot on the image. The ratio on both sides of (1) is unitless; and hence, the pair W and d shares the same units so does the pair w and f. In this case, we choose the unit for W and d is in millimeters (mm), while the unit for w and f is in pixels. Here, we consider that a follower is able to recognize and draw a tight bounding box around its leader[a] (see Figure 2(a)). Thus, the diameter w of the leader robot on the image can be obtained by the width (in pixels) of the bounding box. Usually, the focal length f of the camera is given in millimeter; however, the focal length can be also calculated in pixels by the following calibration

[a]Methods to achieve this are discussed in the next section.

(a) The leader robot (Khepera IV) is detected in a frame with the bounding box around it.

(b) The new image coordinate considered in this paper.

Figure 2. A frame obtained from video stream of a follower robot.

procedure:

i) Measure the diameter W of the leader robot in millimeters.

ii) Place the leader robot a known distance d (in millimeters) away from the camera.

iii) Obtain the diameter w of the leader robot in pixels from the bounding box (e.g., see Figure 2(a)).

iv) Calculate the focal length f in pixels using the formula $f = \frac{wd}{W}$.

This calibration procedure just needs to be done once. After the focal length in pixels is found and w is obtained from the width of the bounding box in each frame, we can estimate the distance d by rearranging (1) as

$$d = \frac{Wf}{w}. \tag{2}$$

The Algorithm 1 below summarizes the process to estimate the distance from the leader. Note that the "Leader_Model" in Algorithm 1 is the model that helps the follower to detect the leader robot. We briefly discuss how to obtain it in the next section.

Algorithm 1

function DistanceEstimation(Leader_Model) **returns** DistanceFromLeader
    Get a frame from the video stream
    Use Leader_Model to find the leader(s) on the frame and draw the bounding box for each recognized leader
    If detect multiple leaders on the frame, then choose the closest one (biggest bounding box) for tracking
    Obtain the diameter w (in pixels) of the leader from the bounding box
    DistanceFromLeader ← Estimate the distance d_{est} from the leader using (2)

B Control Method

In this subsection, we discuss the control method for the robot to track and maintain the desired distance from its leader. We first note that the robot platform used in this paper is a differential drive robot and the size of each image frame is fixed, say, a pixels width by b pixels height (we further assume that a and b are even numbers). In addition, the image coordinate system in OpenCV considers the top left corner as $(0, 0)$ position (i.e., the origin), x values are increasing toward the right, and y values are increasing toward the

Figure 3. Khepera IV mobile robot with Raspberry Pi 3 and camera on top.

bottom. However, without loss of generality, we transform the image coordinate into a new one as shown in Figure 5(b), where the origin is located at $(a/2, b)$ of the original image coordinate system. In other words, in this paper when we mention a point (x, y) on the image, we mean its location in this new coordinate system unless stated otherwise. Since the camera module is mounted on top of the robot and aligned with the heading direction, the y-axis indicates the current heading direction of the follower robot. Let (x_c, y_c) be the centroid of the bounding box on the leader robot and ψ be the angle defined by the vector pointing to the point (x_c, y_c) and the x-axis. Now, our objective is not only maintaining the desired distance d_r from the leader robot but also keeping the centroid of the bounding box around the leader robot on the y-axis (i.e., the desired value of heading angle $\psi_d = 90°$). For this purpose, the Algorithm 2 below summarizes the control method. The Algorithm 2 is straightforward, but there are a couple of main points that need to be discussed here. First, in Line 7, $\dot{e}_d$ and $\dot{e}_\psi$ can be easily obtained by passing e_d and e_ψ to a command filter (see, for example, Ref. 8). Second, the linear velocity V and angular velocity ω of the follower robot in Lines 8 and 9 are obtained from the proportional-derivative control laws. Finally, once V and ω are available, the left and right wheel speeds (v_l and v_r) can be calculated (see, for example, Ref. 9) and sent to the robot for execution.

Algorithm 2

1: **function** ControlSignalGeneration(d_r,$\psi_d = 90°$) **returns** (v_l, v_r)
2: $d_{est} \leftarrow$ DistanceEstimation(Leader_Model)
3: Obtain the centroid (x_c, y_c) of the bounding box
4: $\psi \leftarrow \text{atan2}(y_c, x_c)$
5: $e_d \leftarrow d_r - d_{est}$
6: $e_\psi \leftarrow \psi_d - \psi$
7: Obtain $\dot{e}_d$ and $\dot{e}_\psi$
8: $V \leftarrow K_{p_v} e_d + K_{d_v} \dot{e}_d$
9: $\omega \leftarrow K_{p_\psi} e_\psi + K_{d_\psi} \dot{e}_\psi$
10: $(v_l, v_r) \leftarrow (V, w)$

III Experiment Setup and Object Detection

A Experiment Setup

Khepera IV ground mobile robot in Ref. 10 together with a Raspberry Pi 3 model B+ in Ref. 11 and camera module V2 in Ref. 12 mounting on top of the robot (see Figure 3) are used as the experimental platform in

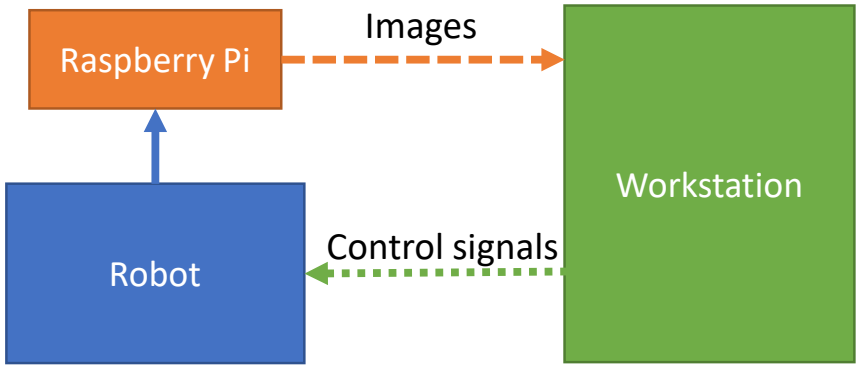


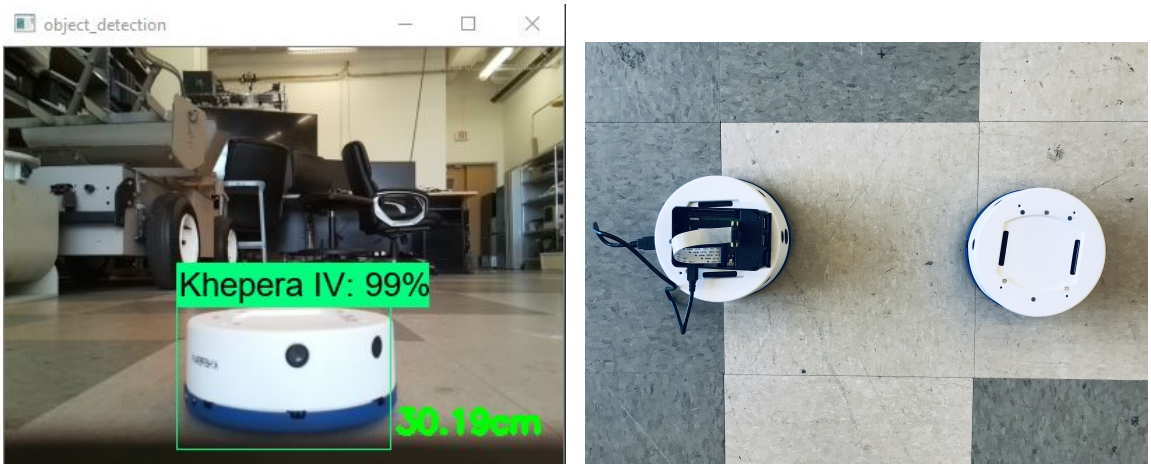
Figure 4. The communication between the Khepera IV robot, Raspberry Pi 3 and the workstation.

this paper. The Khepera IV ground mobile robot has an on-board camera, but the quality is not sufficient for the implementation considered in this paper, so it is not used here. The Raspberry Pi 3 model B+ is currently used to stream the video over a local network so that a workstation can access to the video stream and then use it for processing images and generating control signals[b]. The control signals are sent to the robot through User Datagram Protocol (UDP). The diagram for communication between the Khepera IV robot, Raspberry Pi 3, and the workstation is illustrated in Figure 4.

B Object Detection

After receiving the video stream from the robot, the workstation needs to process the images and finds the leader within the images in order to generate the corresponding control signals for the robot's action. For this purpose, the Tensorflow Object Detection API on Python environment is utilized. A tutorial for utilizing Tensorflow Object Detection API can be found at Ref. 13. For the first step, about 100 images of the Khepera IV robot from different angles are collected. They are then scaled down to the same resolution 800×600. Among the collected images, 80% of them are used for training and the rest is reserved for testing purposes. In the second step, we need to label the data; that is, we need to mark the position of the leader robot on each image in the training set. Here, we utilize LabelImg[14] as the tool for this task. The tool allows us to draw a bounding box around the leader robot on each image, and a corresponding .xml file with the position of the object is generated. Subsequently, a Python script is utilized to gather information from the xml files and generate two csv files (one for training and one for testing), which are then used to generate the corresponding TFRecord files. These files are the input data for training the object detector. We then utilize the faster_rcnn_inception as the Tensorflow detection model for training. After the training, the leader robot can be recognized in real-time from a video stream at the rate of five frames per seconds as illustrated in Figure 2.

[b]In the near future, we will investigate to implement vision processing and generating control signals on the Raspberry Pi itself in the distributed manner without the need of the workstation.



(a) The follower's view. (b) The top view.

Figure 5. The follower's view and top view during the experiment.

IV Experimental Results

This section discusses the results of implementing Algorithm 1 for estimating the distance from the leader and Algorithm 2 for tracking the leader. For this purpose, we set up an experiment with one leader and one follower. Specifically, the leader is remote-controlled by an operator and the objective of the follower is to recognize and follow the leader while maintaining a distance of 30 cm away from the leader. The video of the experiment is available at `https://youtu.be/bCmi09flFLg`. As shown in the video and illustrated in Figure 5, the follower consistently recognizes the leader and is able to draw a bounding box around the leader. In addition, we note that each floor tile has the dimension of 30×30 cm, hence Figure 5 also shows that the estimated distance from the leader is quite accurate. In the video, the follower is not only able to keep the leader within its sight but also track the leader. In particular, when the leader is not in motion, the follower is able to get closer and maintain a distance of 30 cm away from the leader.

V Conclusion

In this paper, we showed that through using a trained model for recognizing the leader robot, we could estimate the distance and implement the controller for the follower to track the leader. In addition, the experiment showed that under the proposed control architecture the follower can recognize and track the leader efficiently. Future work will focus on improving the performance of object detection and implementing visual odometry on the robots so that the trajectories of the robots during the mission are recorded.

References

[1]A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 813–825, 2002.

[2]N. Cowan, O. Shakerina, R. Vidal, and S. Sastry, "Vision-based follow-the-leader," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1796–1801.

[3]R. Vidal, O. Shakernia, and S. Sastry, "Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation," in *ICRA*, 2003, pp. 584–589.

[4]G. L. Mariottini, F. Morbidi, D. Prattichizzo, N. Vander Valk, N. Michael, G. Pappas, and K. Daniilidis, "Vision-based localization for leader–follower formation control," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1431–1438, 2009.

[5]Y. Watanabe, A. Calise, and E. Johnson, "Vision-based obstacle avoidance for uavs," in *AIAA guidance, navigation and control conference and exhibit*, 2007, p. 6829.

[6]B. Fidan, V. Gazi, S. Zhai, N. Cen, and E. Karataş, "Single-view distance-estimation-based formation control of robotic

swarms," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 12, pp. 5781–5791, 2012.

[7]H. Wang, D. Guo, X. Liang, W. Chen, G. Hu, and K. K. Leang, "Adaptive vision-based leader–follower formation control of mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 2893–2902, 2016.

[8]J. Farrell, M. Polycarpou, and M. Sharma, "On-line approximation based control of uncertain nonlinear systems with magnitude, rate and bandwidth constraints on the states and actuators," in *Proceedings of the 2004 American control conference*, vol. 3. IEEE, 2004, pp. 2557–2562.

[9]R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[10]"Khepera IV," https://www.k-team.com/khepera-iv, [Accessed 30-May-2019].

[11]"Raspberry Pi 3 Model B+," https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, [Accessed 30-May-2019].

[12]"Camera module v2," https://www.raspberrypi.org/products/camera-module-v2/, [Accessed 30-May-2019].

[13]G. Tanner, "Creating your own object detector with the tensorflow object detection api," https://gilberttanner.com/blog/creating-your-own-objectdetector, [Accessed 20-November-2019].

[14]Tzutalin, "Labelimg," https://github.com/tzutalin/labelImg, [Accessed 20-November-2019].