An Experimental Set-up For Multi-Robot Applications *

Andaç T. Samiloglu[1,4], Ömer Çayırpunar[2,3], Veysel Gazi[2], and A. Buğra Koku[4]

[1] Baskent University, Mechanical Engineering Department, Bağlıca Kampüsü Eskişehir Yolu 20. km, Bağlıca, 06810 Ankara, TURKEY.
[2] TOBB University of Economics and Technology, Department of Electrical and Electronics Engineering, Söğütözü Cad., No: 43, Söğütözü, 06560 Ankara, TURKEY.
[3] TOBB University of Economics and Technology, Department of Computer Engineering, Söğütözü Cad., No: 43, Söğütözü, 06560 Ankara, TURKEY.
[4] Middle East Technical University, Mechanical Engineering Department, İnönü Bulvarı, Çankaya, Ankara, TURKEY.

Abstract. The objective of this study is to develop an experimental set-up for researchers working on multi-robot systems and for educational purposes in control and robotics courses. The set-up (SwarmCam) consists of mobile robots travelling on a bounded arena, an overhead camera, a PC for processing the images obtained from the camera to determine and if necessary, feedback the global positions and orientations of the robots. We also discuss an experimental application of one of our previous studies on cyclic pursuit of robots.

1 Introduction

In this study we are motivated by the needs on realistic applications of designed and simulated swarm behaviors. There are many studies on swarm robotics that are simulation based and/or performed analytically. However, additional realistic experiments would contribute new insights to these works. Therefore, we designed an experimental set-up to observe the realistic behaviors of robot swarms. This set-up would also be useful for undergraduate and postgraduate educational studies on control systems and robotics.

Many robotic swarm applications typically reject any dependency on a global system such as global positioning. However, if available the global positions and orientations of the robots can be used for development, debugging, and monitoring of swarm robot applications. On the other hand, the local information that a robot may get by its own sensors can be simulated in this set-up, i.e. the relative positions and orientations of robots in a neighborhood of a robot can be derived from the global information and sent to the robots. Therefore,

* This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant No: 104E170 and by the Turkish Academy of Sciences (TÜBA).

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

the swarm applications utilizing only local information of robots can also be studied experimentally by just deriving the local information from the global information. Furthermore, the collective robotic studies which may require global information can utilize this set-up for experimental validations. Even further, the information of the positions and orientations of robots can be recorded for later analysis of the swarm/collective behaviors. One common method of gathering this information is using the odometer of the robots if it is present (in most of the relatively simple mobile robots there is no odometer). However, odometry itself is not a reliable method and odometry errors tend to accumulate over time. Our system which utilizes an overhead camera to determine the positions and orientations of the robots, provides a fast development environment of swarm coordination and control algorithms since it relieves the designer from dealing with low-level odometric estimation and correction. Since we should deal with more than one robot, we also had to develop identification methods to find out which position and orientation belongs to which robot.

There are some studies on the observation of the arena of the swarm by an overhead camera for behavior analysis [1–4]. However, these studies use overhead cameras or marker technology for only observing, visualizing, identification, and/or recording of the behavior/activity of the system. They do not feedback information to the robots.

On the other hand, there are some studies that utilize the overhead camera for position feedback to the members of swarm. Hayes and Dormiani-Tabatabaei used an overhead camera tracking system, combined with a radio LAN among the robots and an external workstation in [5]. They logged position data during the trials, reposition the robots between trials, and emulated the range and bearing sensor signals. Another experimental set-up for robot swarm applications is described in [6]. The authors develop a middleware solution called DISCWorld and describe a prototype system where the precise location information of the robots are extracted by using an overhead camera.

The objective of this study is to build a low cost set-up that can track multiple robots at the same time. In order to allow near-real-time operation, we setup the system such that position and orientation estimation process time is kept as short as possible. The built setup is independent of robots used, hence enabling researchers using different robots may adopt this framework. We also employed an easy to use software environment (Matlab) to facilitate the use of the proposed setup by various researchers with a rapid learning curve. Matlab is a tool that engineering students already learn in other courses and it has specialized functions for image processing and controller development. Therefore, the set-up is easy to use in senior undergraduate and graduate courses as well.

SwarmCam is a single system consisting of 120x180 cm experimental area, 6 E-puck robots with bluetooth interface, logitech USB camera and Matlab as the main image processing (and possibly control) development platform. The positions and orientations of the robots are determined by a labelling system consisting of three small colored dots on the robots. In addition their ID's are determined by a binary coding system consisting of black colored small dots

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

placed on the top of the robots. The system constitutes a very useful platform for hardware in the loop simulations.

2 The Set-Up Structure

The multi-robot experimental set-up is composed of 6 mobile robots (although higher number is also possible), a high quality USB webcam, a high speed computer and an arena (see Figure 1).

Fig. 1. Experimental setup consisting of an arena, robots, PC and overhead camera.

The mobile robots in this set-up should be small enough such that high number of robots may be utilized simultaneously in the experiments. They must have wireless communication modules like bluetooth, wifi, or zigbee for information exchange with the computer and each other. The existence of proximity sensors (IR, US etc.) is preferred for more realistic experiments. In some of our experiments we utilized the E-puck Robot [7]. The E-puck robot is a small (7.0 cm diameter) mobile robot that has powerful microcontroller dsPIC30 (Microchip, PIC microcontroller), 2 stepper motors for differential drive, 8 infrared proximity sensors, bluetooth communication module, and some other sensory units. The robots are programmed such that they set their motor speeds according to the commands supplied by the computer via the bluetooth interface. Another option is to program the robots so that they receive their global position (and/or possibly the relative positions of the neighboring or all the other robots) and have their own internal decision making and control. In addition the abject avoidance

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

logic may be integrated on the robots. If the robots have enough proximity sensors (like ultrasonic, infrared sensors) around their body they may utilize the proximity information of objects (other robots or walls etc.) to avoid collisions.

The overhead camera placed 156 cm above the arena is directly connected to the computer via USB. An image resolution of 640×480 is sufficient for this set-up considering the arena and the robot sizes. The frames grabbed per second (fps) is not a main criteria in the selection of the camera since the image processing unit cannot process more than 3-4 frames per second (the actual robot detection time is 340 ms for the time being). Therefore, 15 or 30 fps of a camera is suitable for this set-up. The optical distortions on the vertices effect the system considerably. Therefore, a camera with high quality lens is essential. We used the webcam QuickCam Pro9000 (Logitech Europe S.A., European Headquarters Moulin du Choc CH - 1122 Romanel-sur-Morges) for grabbing the images of the arena.

The mobile robots move in a bounded arena of size 120×180 cm. The aspect ratio of the arena is designed to be appropriate with the aspect ratio of the camera images which is 4:3. The color of the arena is selected as light gray to be able to distinguish the robot hats from the arena easily. The arena size should be increased with the same aspect ratio for bigger robots or larger area applications.

The image processing, agent behavior algorithms and communication are all performed by Matlab (Mathworks Inc., Natick, MA, USA). Matlab is preferred due to its build in image acquisition and processing toolboxes. Moreover, Matlab is a very common, easy to use, rapid prototyping environment for engineering applications and many scientists and students are familiar with Matlab. However, it is computationally inefficient and might be inappropriate for applications requiring higher fps rates. Therefore, we are also considering developing a software interface with other tools like C#, C++ to have a faster version of the set-up.

The software of the set-up consists of two main parts, robot tracker and robot controller. In the robot tracker part the frames of the arena are grabbed and processed to determine the position, orientation, and identification of the robots. This information set is supplied to the robot controller part that runs functions of behavior of robots. The robot controller part transmits the control signals of the angular and translational speeds to the robots. The resulting angular and translational speeds of the agents are transferred to the agents via wireless communication modules (bluetooth for E-puck robots, around 10ms is consumed per robot to pass the information). The set-up is designed such that one may utilize only the robot tracker part to obtain and analyze the robot behaviors in the case of the robot controllers are embedded on the robots.

The main delay in the system occurs due to image processing (around 340ms per detection of robot pose, orientation, and identification). Therefore, a computer with enough memory to store the images of the arena and high speed central processing unit would result in better system performances (A double core 64bit CPU at 2.4 GHz with 2GB RAM is utilized in our experiments). As mentioned above another option could be to pass the position and orientation

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

information to the robots and let their internal algorithm to calculate the values of the control inputs. That would better model more decentralized and realistic applications.

3 Image Processing Setup/Methods

Depending on the application the image processing system can be used to determine the robot ID's, the global or relative positions, and/or absolute or relative orientations of the robots. Determining the positions of the robots from the overhead images is very simple. However, the problem is to determine which location belongs to which robot. Therefore, additional methods need to be applied to distinguish the robots. In our setup robot hats are designed to find the location, orientations and identification of robots simultaneously. A sample hat is shown in Figure 2(a). The hat has a diameter of 75mm which is slightly larger than the diameter of the E-puck robots. Three circles all having the same color (bright orange) (one placed at the front and the other two placed symmetrically at the rear) are used to find the locations and orientations of robots and the black circles are utilized for the identification of robots.

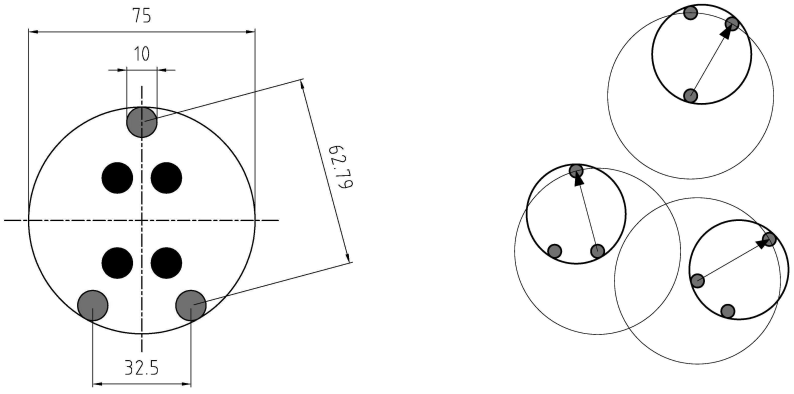


Fig. 2. (a)A sample robot hat used to find the position, orientation and identity of robot. (Dimensions are in mm). (b) Three robots in the arena. The colored dots and the boundaries that other robots should stay out of are shown.

3.1 Determining Robot Locations

First the colored circles are detected in the bitmap images gathered from the camera. Note that most of the cameras supply compressed form of the images (i.e. jpg). However, Matlab reads/converts the images in bitmap format which

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

includes all three color information (8 bits) of the images in three dimensional matrices. A sample configuration of three robots are shown in Figure 2(b).

For distinguishing the colored circles from the rest of the objects in the image we simply use intervals of the color values. In some applications, we also utilize the corresponding HSV (Hue-Saturation-Value) images to find the colored circles. For example, bright purple, bright yellow, and bright green are easy to distinguish colors in HSV format. We get a bitwise matrix (image) by logical operations which outputs 1 for the pixel values in the intervals we set for the Hue, Saturation, and Value (or RGB) of the colors we utilized on the hats and 0 for the other color specifications. In equation (1) the logical operation is shown

$$A = (\underline{H} < Im(:,:,1) < \overline{H})\&(\underline{S} < Im(:,:,2) < \overline{S})\&(\underline{V} < Im(:,:,3) < \overline{V}) \quad (1)$$

where Im is the HSV image of the arena, Im(:,:,i) corresponds to the i^{th} index of the HSV image matrix ($i = 1$ is for Hue, $i = 2$ is for Saturation, and $i = 3$ is for Value) for all columns and rows (":" stands for all of pixel indexes). $\underline{H}$ and $\overline{H}$, $\underline{S}$ and $\overline{S}$, $\underline{V}$ and $\overline{V}$, are the minimum and maximum values of the HSV values of colored circles, respectively. A is the output bitwise matrix that has several objects (let us call the clusters of true valued pixels as objects) on it. Note that there are 8 different logical operations (6 comparison, 2 AND operations) performed on the matrix Im, however in most of the cases we just need three or even two of these operations which needs lower computational efforts that are usually very important in image processing applications. The objects are labelled according to the 8th neighborhood rule with the build in function of Matlab. To eliminate the noise in the binary images some post-processing methods like erosion and dilation may be utilized on the images. The next step is to find the centers of these objects. A simple center of geometry algorithm is run and the positions of these center of geometries is kept in memory. Now, the problem is to find which three points belong to the same robot. For this purpose the distances between the points are utilized. Simply the closest three ones are said to belong to the same robot. This approach is simple and fast but, has one drawback which is when there are robots too close to each other some of the points on these robots may get mixed up and show nonexisting robots. In Figure 2b, some of the bounding circles of robots in which other robots should not travel are shown. It might be also possible to develop a robot identification method which utilizes the previous positions of the robots to overcome this problem; however, so far we have not considered such an approach. Moreover, not allowing the robots to get too close to each other is also good for collision avoidance. This should be guaranteed by the control algorithm. Following the identification process, the positions of the three colored circles of each robot are determined. After that, the positions of the robots are found simply by averaging these three points of each robot.

3.2 Determining the Robot Orientations

Note that the colored circles are placed such that they form an isosceles triangle. The vertex on the intersection of the equal edges is called P_3 and the remaining

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

two points on the other ends of the equal edges are called P_1 and P_2 (see Figure 3(a)). The hat is placed such that the vector from the midpoint of the line connecting the points P_1 and P_2 to the point P_3 is the heading of the robot. The problem here is to determine which point is P_3 (which is also called the heading point). To determine the identification of points we again utilize the distances between the points. The geometry (isosceles triangle) allows us to state that the farthermost point of these three points is the heading point. The remaining two points P_1 and P_2 are identified depending on whether they are on the left or right of the robot. Finding the heading point is sufficient to compute the orientation of the robot. There is no need to determine whether remaining two points are on the left or right hand side of the robot. The average of the points P_1 and P_2 gives the starting point of the orientation vector and P_3 is the end point of this vector. Lastly we compute the unit orientation vector of each robot and store in the memory.

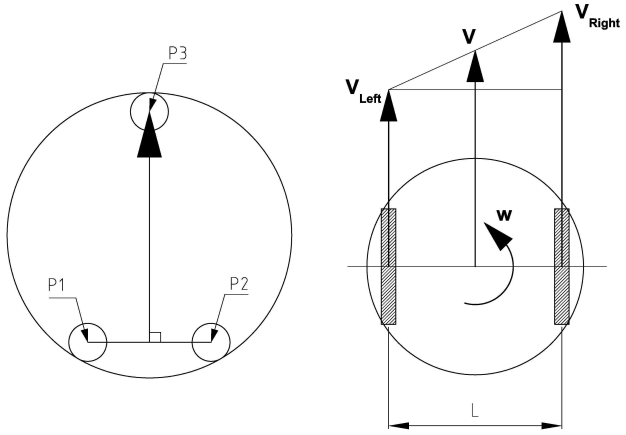
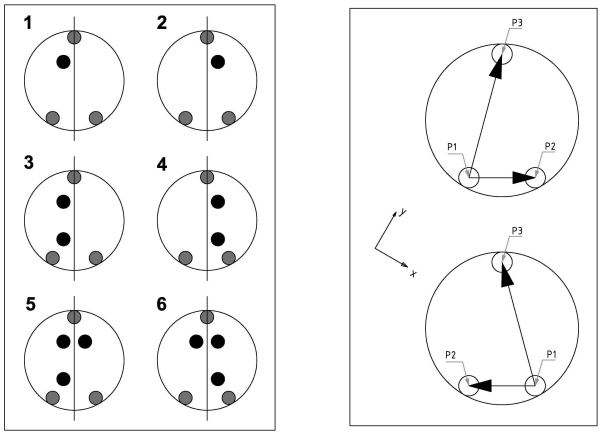


Fig. 3. (a) The colored dots that are used to find the position and orientation of a robot. (b)Speeds of a differential drive robot.

3.3 Determining Robot ID's

As mentioned above for the position and orientation calculations there is no need to find whether P_1 (or P_2) is on the left or right of the robot. However, it is required for the identification of the robots. The regions on the left and right of the robots are utilized to place the black dots in the regions and use an algorithm that checks the number of black dots present in these regions in order to identify the robots. In Figure 4a the black dot placements are shown for 6 different robot hats.

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550



(a) (b)

Fig. 4. (a) Robot Identities. (b) Vectors from P_1 to P_2 and P_1 to P_3 for two different cases: P_1 is on the left for the left figure and P_1 is on the right for the right figure.

At most two black dots are used on the left and right sides of the robots. Here note that we used two methods simultaneously for the identification: (i) the location of the black dots (ii) and the number of black dots on these two regions. It would be easier to identify the robots based only on the number of black dots without dividing the area into regions (such as one dot = robot 1, six dots = robot 6 etc). However, in that case one needs to use as many black dots as the number of robots (i.e., 6 dots for 6 robots). This would result in some problem due to the size of the robots and resolution of the camera when the number of robots increases. By adding the method of dividing regions (left and right in our case) allows us to use less number of black dots (at most 3 for 6 robots) and larger black dots which will be represented with more/enough number of pixels in the image. This approach is better when the number of robots is higher. For example by using 3 dots on each left and right of the regions we can identify 16 ($4^2 = 16$) different robots or with 4 dots, 25 ($5^2 = 25$) different robots. On the other hand, increasing the regions of interest (say top and bottom in addition to left and right-in total 4 regions) will allow one to use less number of larger black dots with enough spacing. Note that the larger the black dots and the larger the spacing between them would give better images that one can process. The black dots are determined in the image matrix similar to the colored ones mentioned in Section 3.1. This time HSV value intervals are set for black. The only problem left is to find the left and right regions of the robots.

For determining the regions on the left and right hand side of robots we need to first find which of the points P_1 and P_2 are on the left (or right). For this purpose we will simply utilize the cross products of the vectors from the point P_1 to P_2 and P_3. If this cross product is negative then P_1 is on the right hand side of the robot (so P_2 is on the left) and if it is positive then P_1 is on the left

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

hand side while P_2 is on the right. The vectors and the global coordinate axes are shown in Figure 4b. Mathematically speaking

$$P_1 \textit{ is on the left if} \quad sign(\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}) > 0 \tag{2a}$$

$$P_1 \textit{ is on the right if} \quad sign(\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}) < 0 \tag{2b}$$

In image processing applications similar to the one discussed here, there are many factors effecting the performance of the system in achieving the goals. One of the main disturbances is the non-unique light source. Some low quality cameras may become very volatile in getting the colored images of the objects. The color values of the objects may change tremendously such that the values may not become stable between the pre-specified threshold values in getting the binary matrices of objects. Therefore, we recommend cameras with auto focus, adjusting luminance of images, and a qualified lens. Otherwise, one may have to calibrate the system (the software) before each experiment. The resolution of the images that we grab from the camera in our setup is 640×480. One would prefer to use higher resolutions which would result in better object detections but with a longer processing time. Image processing time is an important criteria affecting the response time of the system. In fact, the main cause of delays is slow image processing in most of our applications. Therefore, a high capacity/speed memory and processing units of computers and efficient coding (with less memory usage, optimized image processing and appropriate variable types) would result in higher speed.

4 Transmitting Position and Orientation Information

The position and orientation information gathered from the overhead images of the arena can be used for determining the new translational and angular speeds of the robots according to the behavioral algorithms/models investigated in the particular application under consideration. The computing unit (i.e., PC, Laptop) should pass these new speed setting information to each robot. The bluetooth interface is utilized in this setup. Each robot is connected to the master processing unit via bluetooth. However, the bluetooth interface can support at most 7 slaves at the same time. Therefore, the set-up may work for at most 7 robots. For higher number of robots alternative communication units can be: Zigbee and wifi.

The robots we utilized are differential drive robots. They have two stepper motors which can be driven at different speeds. Therefore, speed information of each motor are transmitted via bluetooth. The mathematical relationship between the left and right motor speeds and the translational and angular speeds of the robots can be obtained as

$$V = \frac{V_{right} + V_{left}}{2} = \frac{r}{2}(\omega_R + \omega_L) \quad ; \quad \omega = \frac{V_{right} - V_{left}}{L} = \frac{r}{L}(\omega_R - \omega_L) \tag{3}$$

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

where V and ω are the translational and rotational speeds of the robot, respectively. V_{left} and V_{right} are the left and right wheel speeds, respectively. L is the distance between the left and right wheels. The speeds are shown in Figure 3(b).

5 Experimental Examples

Here we describe an experimental result obtained using the set-up described in the preceding sections. The experiment is performed for testing and verification of the results in [8] where we had studied analytically and via simulations the problem of cyclic pursuit of a swarm of agents. In cyclic pursuit the agents are ordered from 1 to n. Agent i pursues agent $i+1$ modulo n. In other words, the last (n^{th}) agent pursues the first one. In [8] we assumed that the agent dynamics are arbitrated by a finite state machine (FSM) with a sequence of the behaviors: Move towards the pursued agent; Wait for a predetermined time interval; then sense the location of the next agent and move again towards that agent. As a difference from the procedure in [8] in the experimental application the agents were programmed so that they do not stop at the *wait* state, they continue to travel at the last velocity and orientation. In [8] we assumed that each agent has a low-level control which guarantees that the agent reaches the computed way-point in a finite time. However, for the experimental application in this article we had to implement such a low-level controller which will guarantee that the robot moves between two subsequent way points and had opportunity to observe low-level dynamics in the resulting behaviors as well. In [8], the delays and asynchronism were also modelled. We introduced the variables $\tau_{i+1}(t)$ which satisfy $0 \leq \tau_{i+1}(t) \leq t$ in order to represent the delay in the position measurements. In other words, we assumed that at time t agent i knows $z_{i+1}(\tau_{i+1}(t))$ instead of the actual $z_{i+1}(t)$ about the position of agent $i+1$ where $z_i(t) = [x_i(t), y_i(t)]^T \in R^2, i = 1, 2, ..., n$. In other words, $z_{i+1}(\tau_{i+1}(t))$ is the *perceived position* of agent $i+1$ by agent i at time t. The difference between the current time t and the value of the variable $\tau_{i+1}(t)$ is the delay occurring due to the sensory, computing and/or communication processes or other reasons. Moreover, we introduced a set of time indices T^i, $i = 1, 2, ..., n$, at which agent i updates its way-point z_i. The mathematical model of the asynchronous cyclic pursuit is given by

$$\begin{aligned} z_i(t+1) &= (1-p)z_i(t) + p\, z_{i+1}(\tau_{i+1}(t)), \qquad t \in T^i \\ z_i(t+1) &= z_i(t), \qquad t \notin T^i \end{aligned} \tag{4}$$

where p is the gain satisfying $0 < p < 1$. We first studied the convergence of the positions of a multi-agent system in a cyclic pursuit under synchronism ($\tau_{i+1}(t) = t$) and used these results in the proof of the convergence of the model with asynchronism and time delays. Numerical simulations were also performed to verify the theoretical results. In Figure 5a, the paths of 5 agents in cyclic pursuit obtained in the simulation studies in [8] are shown.

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

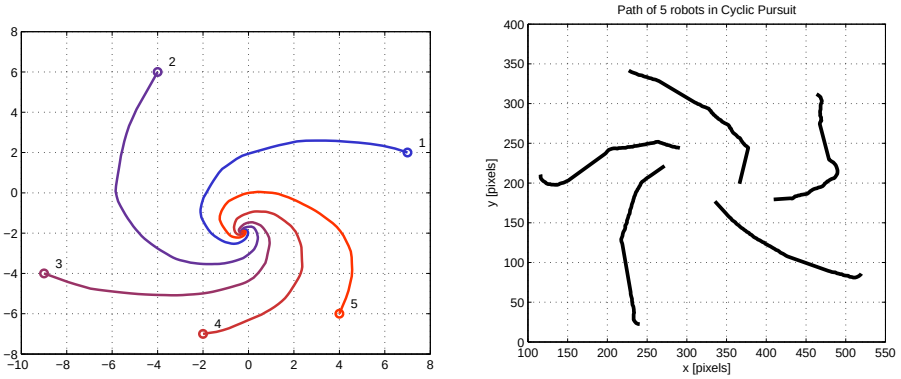


Fig. 5. (a) The result of simulations for cyclic pursuit rendezvous of 5 robots in our previous study [8].(b) Path of 5 E-puck robots in cyclic pursuit obtained in the set-up.

In Figures 6 and 5b the results obtained for the cyclic pursuit of 5 E-puck robots in our set-up are shown. Comparing the Figures 5a and 5b, we observe that the analytical and simulation based results in [8] are also verified by the experimental results. The frames at 1, 100, 150, and 195 time steps are shown in Figure 6. The robots are spread away at the beginning of the simulation. Each robot follows its leader and travels on spiral like path shown in Figure 6. At the end they converge to each other. Note that in these video frames there are additional virtual geometries drawn on and between the robots virtually. The lines show the connection between the follower and the leader. The colored dots on the robots show the left, right, and heading points of the robot hats as mentioned in Section 3.

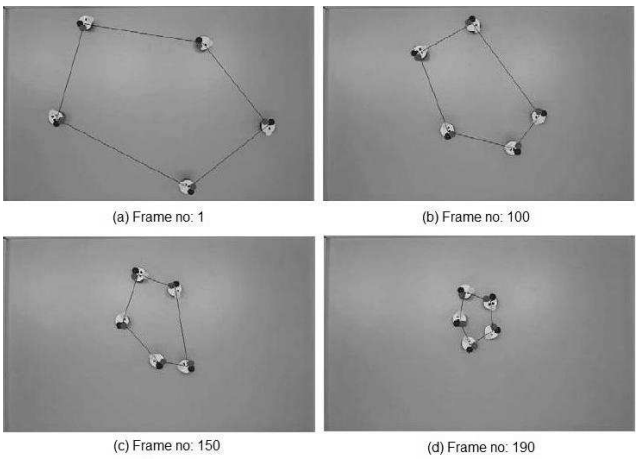


Fig. 6. Cyclic Pursuit of 5 robots.

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550

6 Concluding Remarks

The objective of this study is to develop an experimental set-up for swarm robot applications. The small sized relatively simple mobile robots called E-puck are utilized in the set-up for the time being. However, it can be easily used with other robot platforms as well. The arena is observed by a high quality USB camera connected to a high speed PC. We perform the image processing works in Matlab and feedback the positions and orientations of robots to the behavior algorithms governing the swarm dynamics. The main delays are resulted from the image processing computations. Further optimization of the algorithms for faster response of the system is still possible.

We believe that this test bed is a very useful experimental facility which can be used for testing swarm coordination and control algorithms as well as can be used in both graduate and undergraduate courses.

References

1. N. Correll, G. Sempo, Y.L.d.M.J.H.J.L.D., Martinoli, A.: Swistrack: A tracking tool for multi-unit robotic and biological research. Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (2006) 2185–2191
2. Lucas P. J. J. Noldus, Andrew J. Spink, R.A.J.T.: Computerised video tracking, movement analysis and behaviour recognition in insects. Computers and Electronics in Agriculture **35**(2-3) (2002) 201–227
3. Trifa, V., Cianci, C.M., Guinard, D.: Dynamic control of a robotic swarm using a service-oriented architecture. In: Proceedings of International Symposium on Artificial Life and Robotics, Beppu, Japan (2008)
4. Fiala, M.: Artag, a fiducial marker system using digital techniques. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on **2** (2005) 590–596 vol. 2
5. Hayes, A., Dormiani-Tabatabaei, P.: Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on **4** (2002) 3900–3905 vol.4
6. Hawick, K.A., James, H.A.: Middleware for context sensitive mobile applications. In: ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2003) 133–141
7. E-puck Robots: E-puck robot specifications. Available from http://www.e-puck.org (2008)
8. Şamiloglu, A.T., Gazi, V., Koku, A.B.: Asynchronous cyclic pursuit. In et al., S.N., ed.: Proc. of 9'th Conference on Simulation of Adaptive Behavior (SAB06). Lecture Notes in Artificial Intelligence (LNAI) 4095. Springer Verlag, Berlin Heidelberg (2006) 667–678

Workshop Proceedings of SIMPAR 2008
Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS
Venice(Italy) 2008 November,3-4
ISBN 978-88-95872-01-8
pp. 539-550