An Experimental Framework for Exploiting Vision in Swarm Robotics

Sjriek Alers, Bijan Ranjbar-Sahraei, Stefan May, Karl Tuyls and Gerhard Weiss
Department of Knowledge Engineering
Maastricht University
Email: {sjriek.alers,b.ranjbarsahraei}@maastrichtuniversity.nl,
stefan.may@student.maastrichtuniversity.nl, {k.tuyls, gerhard.weiss}@maastrichtuniversity.nl

Abstract—This paper studies the requirements of a successful vision-based approach in swarm robotic settings. Required features such as landmarks and different patterns are introduced, and appropriate feature detection algorithms are described in detail. The features are designed to be very simple, and providing enough information, while the proposed detection algorithms have considered the very limited resources (i.e., limited storage memory, and limited computational power) of swarm robots. In order to evaluate the performance of the proposed vision approaches and the defined features for the environment, the whole approach is verified by implementation on e-puck robots in a real-world setting.

Keywords—Robot vision systems; Multirobot systems.

I. INTRODUCTION

Natural phenomena have always fascinated and inspired scientists, not only the biologists but also others such as computer scientists. One of the interesting phenomena in nature is the behavior seen in colonies of social insects such as ants and bees. These insects have evolved over a long period of time and display a behavior that is highly suitable for addressing the complex tasks that they face. Therefore, over the recent years an increasing interest is seen among researchers for creating artificial systems that mimic such behavior for accomplishing the complex tasks that humans face in their life [1], [2], [3].

The phenomenon that intelligent behavior emerges from a collection of simple interactions among agents which are relative simple as well, is generally referred with the term Swarm Intelligence (SI) [4]. The best known example for emergence of Swarm Intelligence among social insects is the foraging behavior of ants. In ant foraging, ants deposit pheromones on their path during traveling. Using this path they are able to navigate between the nest and food [5]. A slightly different foraging behavior can be seen among honeybees. Instead of using pheromones to navigate through an unknown environment, honeybees use a strategy called *Path Integration*, in combination with landmark navigation [6]. With the aim to transfer such social behaviors to embodied systems, many researchers are investigating the foraging behavior of ants and bees, by using robots in real environments. However, Foraging is the task of locating and acquiring resources in an unknown environment, which is quite a difficult task, in terms of localization and detection of environmental localization, specially for simple robots in a distributed swarm. The foraging task can be seen as an abstract representation for many other advanced tasks, such as patrolling and routing. Therefore, a successful embodied implementation of distributed foraging can result in promising applications in, e.g., security patrolling, monitoring of environments, exploration of hazardous environments, search and rescue, and crisis management situations.

Getting motivation from the mentioned potential applications of distributed coordination and following the previous work [7], [8], in which we mainly relied on random exploration methods and infrared sensor data for obstacle detection, this paper is focusing on using vision for detecting key environmental features. These features then can be used as waypoints to navigate in an unknown environment, locate other entities, and detect modifications made in the environment.

For this purpose, we explore several visual features that can be used for acquiring information from the environment by a robot with limited computation abilities, and equipped with a simple camera. For detecting key locations in the environment (e.g., corners in a maze), we investigate the usage of specific landmarks for these locations. Each landmark consists of an upper ring with a solid color, so that it can be detected from a distance, and on the lower part a unique barcode for keeping track of the landmark numbers. Furthermore we explore the possibility to detect markers with an even higher data density: QR-codes. The challenge in the detection of these two-dimensional codes, lies in analyzing and processing the camera data with the limited processing and memory resources that are available in our robotic platform. Finally, the most common feature already available in every robotic swarm setting is the robot itself. It's always favorable to detect the relative location and orientation of other robots in respect of one's position. Therefore, the available LEDs on the robot provide a very good feature for robot detection from a distance. Moreover, we have designed specific gradient patterns for nearby robot detection, which can conclude to a very accurate orientation detection.

Authors believe that the proposed environmental features defined in this paper, in combination with the detection algorithms which are included as well, can provide an experimental framework for any kind of swarm robotic experiment with simple robots (e.g., [9], [7], [8], [10], [11]) as illustrated in Fig. 1.

The remainder of the paper is structured as follows: The physical setup and designed software are described in Section II. The main features used in this paper are defined in Section III, and the techniques for detection of each feature is described in Section IV. A real-world demonstration of this work is
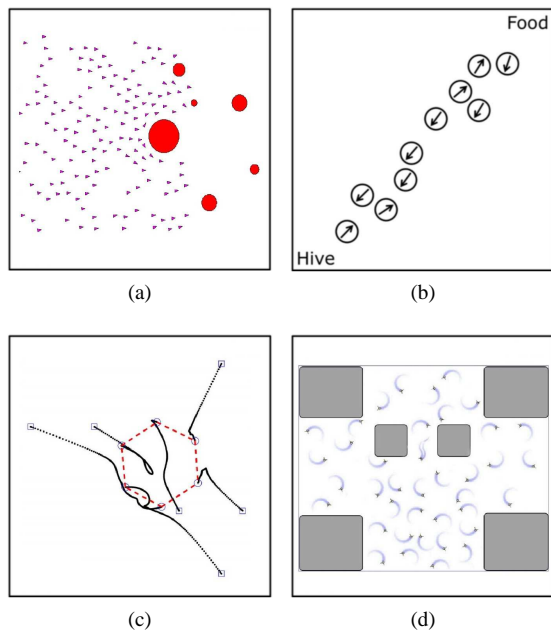
Figure 1. Different swarm robotic applications which require visual feature detection: (a) Flocking in multi-agent systems [9]. (b) Bee-inspire foraging [7]. (c) Formation control in multi-agent systems [10]. (d) StiCo: Stigmergic coverage in complex environments [11].

described in Section V and can also be found in [12]. Finally, in Section VI we will give the concluding remarks and future work.

II. PHYSICAL SETUP AND DESIGNED SOFTWARE

The e-puck robot is a small platform for educational and research purposes, developed by the EPFL University [13]. This robot is efficiently used in numerous projects in the domain of swarm robotics and swarm intelligence (e.g., [7], [8], [14], [13]).

The main features of the e-puck robot include, but are not limited to; a robust design, flexibility for a large spectrum of educational activities, compact size, and rich on-board accessibilities (e.g., microphones, accelerometer, camera).

In this section, first the hardware specifications of the e-puck are briefly introduced, then the developed software which is designed for monitoring the e-puck camera during its image processing and feature detection tasks is described.

A. Hardware Specifications

The e-puck hardware consists of different sensor types for detecting visible or Infra Red (IR) light, sound, acceleration, et cetera. The motors are the only actuators which are available in e-puck. A microprocessor of PIC family with 8 KB RAM memory, assist the robot to get data from it's sensors, analyze it, and perform actions. The main hardware elements, which are involved in our experiments are listed in Table I.

As listed in the table, the on-board camera of the e-puck has a resolution of 640×480 pixels. It is placed at the front of the e-puck, 2.7 cm above the floor. With this camera, objects that are placed on the floor can be detected at a minimum distance of 7.4 cm. The camera angle is approximately $40°$,

TABLE I. E-PUCK TECHNICAL SPECIFICATION

Element	Technical information
Processor	dsPIC30F6014A @ 60 MHz (15 MIPS), 16-bit microcontroller with DSP core
Memory	RAM: 8KB Flash: 144 KB
Motors	2 stepper motors with a 50:1 reduction gear
Camera	VGA color camera with resolution of 640x480 pixels
LEDs	8 red LEDs on the ring, green LEDs on the body, 1 high intensity red LED in the front
Wireless Communication	Bluetooth for robot-computer and robot-robot communications Infrared for robot-robot communication

and at this minimum distance, objects of 5.1 cm width can be fully seen.

Remark 1: *Although, we have a VGA camera, the on-board processing and storage of the e-puck robot is not adequate for dealing with all of the camera data. A gray-scale image of size 640×480 needs at least 307.2 KB to store the image. However, based on the technical details of Table I, the e-puck robot has a RAM of size 8 KB. Analysis, and storage of sub-parts of the image helps to overcome this limitation. In following sections of this paper, we address the issue of how to split an image into informative sub-parts.*

B. Software

In order to monitor the e-puck in real-time, and for debugging the image-processing algorithms, a Java-based software application is developed. This software, shown in Fig. 2, communicates with the e-puck via Bluetooth. It receives text messages from the e-puck, which are reports of intermediate statuses of the e-puck (e.g., "found something", "driving to the landmark", "code read", "searching"). At the same time, the program also receives the captured image from the robot. Logging all of the data, storing the text messages and captured images, as well as the filtered and segmented images, makes both real-time and offline debugging very easy. Finally, it should be mentioned that time stamps are always attached both to the captured images, and stored text messages.

Figure 2. Developed software for monitoring the e-puck: (1) The required controls to establish the connection with the e-puck. (2) The real-time captured image. (3) Log statements (4) Archive of the last 20 captured pictures.

III. FEATURE DEFINITION

Defining a collection of detectable features is, due to the limited resources of a simple robot, an important task that is a part of the main scope in this paper. Different objects in the environment (e.g., pieces of wood, balls, walls, floor, and robots) and many available patterns (e.g., different colors, checkerboard and barcodes) can be considered as environmental features, however their detection via a robot with limited capabilities might be computationally complex, and or not adequately robust to environmental disturbances (e.g., light variations and distance variations).

Generally, we define the required environmental features into two main categories: *far features*, and *close features*. For the far features bright lights (e.g., from a red LED) or specific relative large areas with solid colors, should be considered. Moreover, these features should be recognizable from different directions, which makes cylindrical shapes more favorable. However, for nearby features, the patterns which can store higher amount of information are required (e.g., a one-dimensional barcode or two-dimensional codes which can store digital information). By considering the mentioned constraints, a collection of the most appropriate environmental features will be introduced in this section.

A. Landmark

Most important features for an environment are the landmarks. Robots can use landmarks in many various missions, like localization, mapping, exploration, etc cetera. These features should be recognizable from different directions, and also from a distance. Landmarks should provide useful information to the robots (e.g., their exact location), therefore, we introduce a cylindrical tube, as shown in Fig. 3, which is a combination of a colored ring and a barcode.

At the top of the cylinder a colored ring is denoted which is easily detectable from a distance. For our setup, purple is chosen as ring color, as purple is a color that does not exist in any other objects in our environment. To differentiate the landmarks, an EAN-8 (European Article Number) barcode was selected, containing an ID consisting of 8 digits, including a control number. The EAN-8 barcode is printed vertically below the purple block, surrounding the whole cylinder.

B. QR-Code

Although, landmarks are very useful in terms of being detectable from a distance, we need a smaller pattern which can be mounted on walls, and also directly on robots for providing more dense information (e.g., specific ID of a robot,

wall orders). Therefore, we use a two-dimensional Quick Response code (QR) which is developed as a universal data storage standard. These QR-codes can store a higher data density, then the EAN-8 barcode.

The only disadvantage of the QR-code is the complexity of its pattern. In general, the pattern is comprised of several parts: At the top left, the top right and the lower left corner an orientation pattern is placed. It is a square of size 9×9 modules. The fourth corner does not contain this pattern, which makes detection of QR-code angle easier. In most swarm robotic applications, the orientation of the QR-code can be fixed, so the orientation check can be ignored during image-processing, decreasing the computational complexity drastically.

Different versions of QR-codes have different sizes. The smallest size is Version 1, which has a size of 21×21 modules. For each version, the size is increased by 4 modules in each direction. Between the three orientation patterns there are timing pattern lines with strict changing modules of black and white at row 6 and column 6. Every QR-code from Version 2 and higher, contain position adjustment patterns at specific points. In Fig. 4 the structure for QR-code Version 3 is given, in which the black and white parts are fixed.

C. Robot Detection

A very important feature which will be available in the environment of any swarm robotic application, is the robot itself. Inherently, the robots contain various information, like their position, orientation, and their identifier, which can be very useful for the other individuals to know. Therefore, the ability to detect other robots relative orientation and location, is very convenient for implementation of many complex swarm algorithms (e.g., [7], [8]).

In practice, a good way for detecting other robots with a camera, is detecting the robots light sources (e.g., on-board LEDs). As such a light source has a good contrast to the other parts of the environment, it can be detected from a far distance even on low resolution captured images.

To determine the orientation of the robots, based on it's visual features (e.g., the wheels and body of the robots) is a really complex task. Therefore, we propose to add a black-white pattern comprised of two slopes as shown in Fig. 5. Computing the exact orientation of a robot by using this pattern is easily implementable.

Based on the standard size of an e-puck robot, the pattern should have a total height of 33 mm and consists of two black bars separated by a white bar on top. All the bars are 3 mm

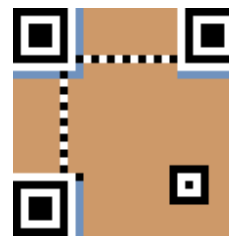Figure 3.   Example for a landmark

Figure 4.   Structure of a QR-code Version 3, displaying orientation and timing patterns

Figure 5.   Body pattern of e-puck for orientation detection

in height. A sloped pattern, in the form of a black triangle is added to the bottom of the pattern.

Finally it should be mentioned that, in addition to orientation measurement, distance measurement also becomes possible by using this specific pattern.

IV.   FEATURE DETECTION

In the previous section, we introduced four main features: *landmark*, *QR-code*, *Robot LED*, and *Robot body pattern*. The main approach for detection of these features is to first use basic filters for highlighting the required information (e.g., purple color or edges in the image) and then zooming into the informative part of the image for reading it in more details. The most important factor in designing each detection algorithm, is to use the least possible memory and computation power. In the following subsections, these techniques for detection of each feature will be described.

Remark 2: *It should be mentioned that all of the required thresholds which will be used in following subsections are computed based on practical experiments and with real-time calibrations. However, describing these experiments in detail is beyond the scope of this paper.*

A. Landmark

The landmark contains a purple ring and an EAN-8 code. The landmarks are designed to be taller than robots. Therefore, finding the purple ring of each landmark, limits the scanning area of the image to the upper half of the camera view.

Detection of an area with a specific color is a basic task. In the first step a color filter with the specific color is applied on the image. Resulting in a grayscale image with bright values for the colors which match the color the best. To avoid errors where single pixels fit to the color, the image is blurred with a Gaussian algorithm [15]. Afterward the image is split into a binary black/white image with a fixed threshold. This procedure is illustrated in Fig. 6.
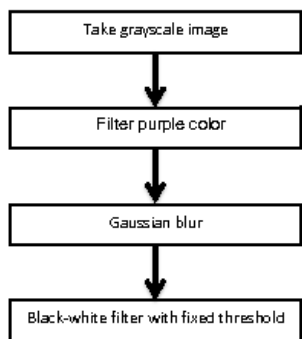

Figure 6.   The required pre-processing procedure for detection of a specific color (e.g., purple).

After this pre-processing phase, a group-finding algorithm [15] is applied on the image, and the largest group, is considered as the purple ring.

Consequently, the exact position of purple ring in the image can help to estimate its distance to the robot. The higher the purple ring is, the further the distance should be. This estimated distance is used to find the appropriate distance to start reading the barcode. As soon as the required distance, in which the barcode is readable is reached, the required scanning area is determined (i.e., the area under the purple ring).

Barcodes are one-dimensional, this simplifies the scanning process and makes the whole procedure faster. Therefore, addressing the issue described in Remark 1, the robot prepares a grayscale image with low resolution in width but high resolution in height (i.e., zooming into an area of 4 pixels in width and 80 pixels in height).

The pre-processing for the EAN-8 barcode is done by using a halftone filter [15] with a threshold calculated by an average of the pixels intensity. Afterward, all patterns of form black-white-black, as shown in Fig. 7 are located. Based on the EAN-8 standards, at least three occurrences should be detected for the start, center and end of barcode.

After this validation check, the part of the image containing the code is transformed into the 67 bits representing the barcode. Each bit is defined by the average of the pixels it represented. For each seven bits the best corresponding match to a data character is determined. As a last step, the control-character is calculated out of the seven data-characters.

B. QR-Code

Detection of QR-codes is more complex than detection of one-dimensional barcodes. We assume that the QR-code is fully visible in the camera frame, as a partial QR-code cannot be decoded. As the QR-code needs a resolution as high as possible, first a black-white image is filtered out of the initial captured image. For finding the three orientation markers of the QR-code, a pattern finding algorithm is used, which looks for a black-white-$3\times$black-white-black transition on each column. The detectable pattern looks like the center line in Fig. 8a. As soon as the pattern is found, the same pattern is located in the rows. The results should be similar to Fig. 8b. The orientation marker validation is passed, if both found regions have nearly the same size and center.

A QR-code is comprised of a collection of modules, each black or white. In order to determine the number of pixels which construct a single module, the size of the orientation modules, and their distance to each other can be used. For example, the pattern shown in Fig. 8 consists of 7 modules, so we can divide the number of pixels in this pattern by 7 to compute the size of a single module. Moreover, to improve

Figure 7.   Structure of EAN-8 barcode, with the black-white-black pattern in the beginning, middle, and end.
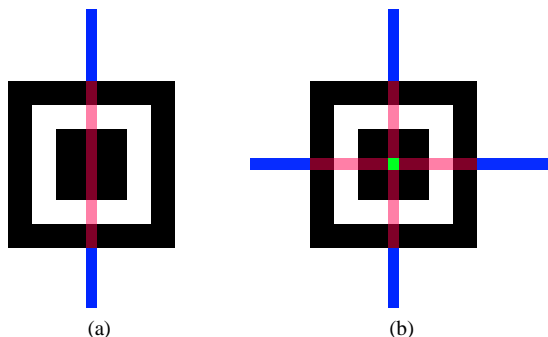
Figure 8. Detection of QR-code orientation markers: (a) pattern detected in vertical alignment. (b) pattern detected in horizontal alignment.

the estimation of the module size, the distance between two patterns can also be used. Each version has a size of $21 + n \cdot 4$ modules, where n is the version of QR-code. Therefore we can calculate the version and get a more exact value for the module size. The decoding process of the QR-code after its structure has been extracted is described in [16].

The most challenging problem with an QR-code image, is that the image has to be stored in a high enough resolution, for being decodable. However, addressing the issue mentioned in Remark 1, the memory on the e-puck is limited for storing additional information to 4 KB for which each bit can store one pixel, as the rest of the memory is used for running the algorithms. To have some error tolerance, there should be at least four pixels describing one module. Therefore, we can find the biggest detectable size for QR-code with following equation:

$$4000 \times 8 = modules^2 + (4 \cdot modules)^2 \qquad (1)$$

in which the left side shows number of available bits, and on the right side, the first and second terms show the number of required bits for storing the QR-code and image itself. This equation concludes to the fact that width and height of the QR-code should not exceed 43×43 modules. The QR-code version which fits into 43×43 modules is Version 6, which is 41×41. In practice we also need memory for the detection algorithms and internal calculations, so the QR-code Version 5 which contains 37×37 modules, is used in our experiments.

C. Robot Detection

An other robot is generally detected in two different steps. First, the detection from a distance is done by searching for the red LEDS, and second, when nearby, the body pattern (Fig. 5) which consists of two black ramps around the robot, is scanned for measuring the exact orientation of robot.

1) LED Detection: The LEDs are mounted above the camera on the e-puck. Therefore only the upper half of the image has to be scanned, which results in a higher usable resolution of the relevant parts of 20×80 gray-scale pixels. First, a black-white filter with a fixed threshold is used. The threshold is chosen to be higher that ambient light, and less that the brightness of an LED. The next filter is a Gaussian blur filter, which is used to combine light groups that are very close to each other, and dismisses single pixels that are falsely
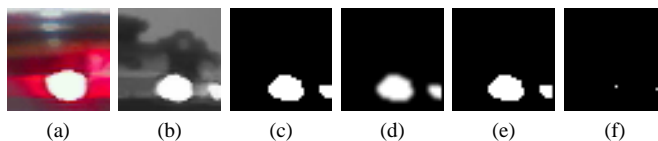

Figure 9. Different steps of pre-processing for LED detection.

recognized. Afterwards, a black-white filter is used, this time with an average threshold. All LEDs are now highlighted.

As it is not possible with this simple detection technique to really determine between a red LED, and LEDs from another color, some improvements should be applied. Therefore, after detecting the light sources, the camera zooms in on each group center (zooming is a built-in feature of the e-puck camera). The zooming ratio depends on the amount of pixels that belong to each group. In Fig. 9 an image of a zoomed in LED is shown. There is a bright center visible with red at the left and the right, but not at top or bottom. This is a result of the surrounding border of the e-puck. To verify that the LED is a red one, both sides of the detected light source, starting from the center are scanned for a red color. The color is checked, by converting the image into the HSL color space and comparing the Hue value, as the lightness and saturation are very unstable. We consider a light source as an LED, if more than 50% of the height of the bright center contains a red surrounding.

2) Body Pattern Detection: If the robot is located close enough to the camera, the body pattern detection can be activated. To get the highest probability to detect the e-puck, the image should have a high resolution, but still work fast. The maximum image size which fits into memory and leaves enough space for the other required operations, is 80×40 pixels in gray-scale. As pre-processing step, a black-white filter with average threshold is applied on the image.

Subsequently, for each column of the image a pattern with one white, and one black module is located. For all locations where the pattern fits, a check is performed if the repetitive white and black modules have approximately a size of 5. If this holds, the column is stored as a part of the pattern. Fig. 10a shows a captured image from an epuck, and Fig. 10b highlights the parts of image which are extracted as body pattern according to this technique.

For rejecting wrong detections, only modules with at least three detected neighboring results are accepted. Afterward, the center of the e-puck is determined by searching the location where most left and right results are found and dividing their x-coordinates by 2. The orientation of robot can be easily measured by computing the length of middle white module, and comparing this size, with the size of the whole pattern.
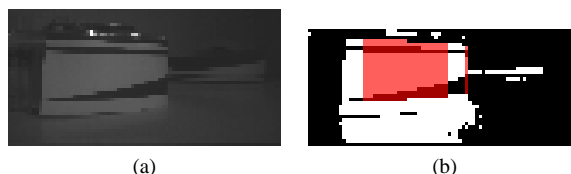

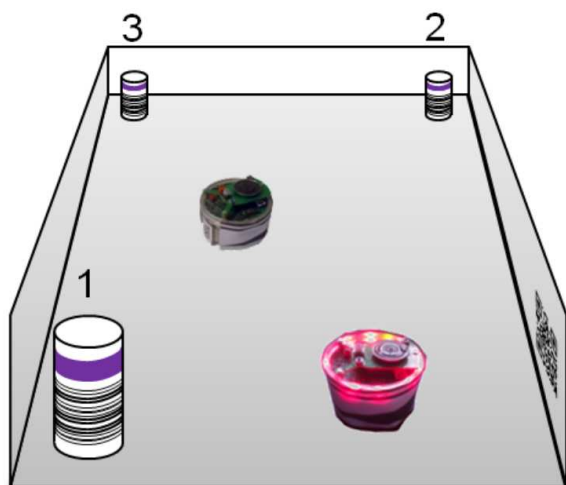Figure 10. Body patter detection (a) Initial image. (b) the detected pattern is highlighted.

Figure 11. Designed scenario for validation of proposed approach

V. REAL WORLD DEMONSTRATION

To examine the proposed approach in a real scenario, an environment as shown in Fig. 11 is set up: A white floor of 40×40 cm^2 is surrounded with white walls. Three landmarks are placed in three corners, and in the fourth corner a QR-Code is attached to the wall. Two e-pucks are placed in this environment. One is stationary, with all of the red LEDs on, and a body pattern around it. The second robot uses the vision-based detection algorithms for detecting the features of the environment.

In this scenario, the robot has to first located landmark #1, continue to #2 and then drive to #3 in the correct order. For each landmark it has to approach it, read the barcode, and after validating the number find the other robot. By using the other robots orientation, it should move in the environment till both robots are facing each other from the front. Finally, the QR-code mounted on the wall is detected, and the code will be extracted.

A video of this performed experiment can be found in [12], including the preprocessed image data sent from the robot.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a feature detection approach based on robot vision, which can be useful for swarm robotic experiments. The e-puck robot was chosen as the main platform for doing experiments. This robot is equipped with a VGA camera, but has limited resources for storing data, and also in performing computations. Therefore, different possible environmental features were introduced, and described accurately. Afterward, required image processing techniques for detection of each feature were described in detail. Finally, a general demonstration was set up to show the applicability of the proposed approach in a real-world robotic experiments.

The feature detection approaches developed in this paper, provided a framework for exploiting vision in various multi-robot scenarios. However, the performance of this framework was not sufficiently evaluated. Therefore, as a future work, authors will examine the introduced techniques accurately, by applying them in different experimental conditions (e.g., light or observation distance variations). Besides, the framework will be applied to a real swarm of robots (e.g., 7 e-puck robots). Finally, extending the framework by introducing other environmental features such as glowing artificial pheromones is a part of this project.

REFERENCES

[1] F. Dressler and O. B. Akan, "A survey on bio-inspired networking," *Computer Networks*, vol. 54, no. 6, pp. 881 – 900, 2010.

[2] D. Floreano and C. Mattiussi, *Bio-inspired artificial intelligence: theories, methods, and technologies*. The MIT Press, 2008.

[3] N. Franceschini, F. Ruffier, and J. Serres, "A bio-inspired flying robot sheds light on insect piloting abilities," *Current Biology*, vol. 17, no. 4, pp. 329–335, 2007.

[4] J. Kennedy, "Swarm intelligence," *Handbook of nature-inspired and innovative computing*, pp. 187–219, 2006.

[5] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851–871, 2000.

[6] N. P.-P. M. Lemmens, *Bee-inspired Distributed Optimization*. Maastricht University, 2011.

[7] S. Alers, D. Bloembergen, D. Hennes, S. de Jong, M. Kaisers, N. Lemmens, K. Tuyls, and G. Weiss, "Bee-inspired foraging in an embodied swarm (demonstration)," in *Proceedings of the Tenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2011)*, pp. 1311–1312, 2011.

[8] N. Lemmens, S. Alers, and K. Tuyls, "Bee-inspired foraging in a real-life autonomous robot collective," in *Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011)*, pp. 459–460, 2011.

[9] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[10] B. Ranjbar-Sahraei, F. Shabaninia, A. Nemati, and S. Stan, "A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 8, pp. 3124–3134, 2012.

[11] B. Ranjbar-Sahraei, G. Weiss, and A. Nakisaee, "A multi-robot coverage approach based on stigmergic communication," in *Multiagent System Technologies*, vol. 7598 of *Lecture Notes in Computer Science*, pp. 126–138, Springer, 2012.

[12] Swarmlab, Maastricht University, "Demonstration of an experimental framework for exploiting vision in swarm robotics." http://swarmlab.unimaas.nl/papers/adaptive-2013-demo/.

[13] F. Mondada, M. Bonani, *et al.*, "The e-puck, a robot designed for education in engineering," in *9th Conference on Autonomous Robot Systems and Competitions*, vol. 1, pp. 59–65, IPCB: Instituto Politcnico de Castelo Branco, 2009.

[14] A. Breitenmoser, M. Schwager, J. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4982–4989, 2010.

[15] R. Gonzalez and R. Woods, *Digital image processing*. Prentice Hall Upper Saddle River, NJ, 2002.

[16] "ISO/IEC 18004:2000, information technology, automatic identification and data capture techniques, bar code symbology, QR code," 2000.