DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch111

Evaluation of 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

Visual feature detection with limited resources of simple robots is an essential requirement for swarm robotic systems. Robots need to localize their position, to determine their orientation, and need to be able to acquire extra information from their surrounding environment using their sensors, while their computational and storage capabilities might be very limited. This paper evaluates the performance of an experimental framework, in which environmental elements such as landmarks and QR-codes are considered as key visual features. The performance is evaluated for environmental light disturbances and distance variations and feature detection speed is thoroughly examined. The applicability of the approach is shown in a real robot scenario by using e-puck robots. Finally, the results of applying the approach to a completely different setting, i.e., simulation of pheromones using glowing trail detection, are presented. These results indicate the broad applicability range of the developed feature detection techniques.

Introduction

In recent years there has been a rapidly growing interest in using teams of mobile robots for achieving complex tasks such as environmental coverage (Cortes et al., 2004) and exploration (Burgard et al., 2005). This interest is mainly motivated by the broad spectrum of potential civilian, industrial and military applications of multi-robot systems. Triggered by this interest, today, development of practical approaches for multi-robot problems is a well established topic in multi-robot research (e.g., (Hennes et al., 2012; Ranjbar-Sahraei et al., 2013)).

A natural phenomenon with high relevance to practically applicable multi-robot approaches is the foraging behavior of ants. In ant foraging, ants deposit pheromones on their path, while they are looking for either food or nest, which in long term establishes a path between these two locations (Dorigo et al., 2000). 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. These strategies turn out to be highly effective in solving distributed optimization problems (Lemmens, 2011). Although investigation of foraging behavior of ants and bees is very interesting, the task of locating and acquiring resources in an unknown environment is quite a difficult task in practice (in particular with robots that have limited resources). Considering that the foraging task can be seen as an abstract representation for many other advanced tasks, such as patrolling and routing. 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 (Alers et al., 2011; Lemmens et al., 2011), which mainly was relied on random exploration methods and infrared sensor data for obstacle detection, authors have recently introduced a framework for simple swarm robotic systems, which exploits vision in robots with very limited resources to extract information from landmarks and environmental patterns (Alers et al., 2013). These features are used as waypoints to navigate in an unknown environment, locate other entities, and detect modifications made in the environment. Although the previous paper describes the framework in detail, its performance in different environmental conditions, and various scenarios is not studied/compared yet. Therefore, this paper focuses on evaluation of this framework in different environmental settings (various light intensities, and detection distances). The detection speed is also deeply studied in various scenarios. In parallel, a new environmental feature, the glowing trails, is described; the developed approach is adapted to this feature, and the results are illustrated.

The remainder of the paper is structured as follows: First related work is briefly reviewed, then the vision-based framework for robots with limited resources is introduced. As the main contribution of this paper we address the influence of external variables, like environmental lighting conditions and viewing angles, on the detection performance of specific features. Also the reliability of detection of the various features are evaluated, and the overall performance regarding time and memory consumption in respect to us-

ability in real robot scenarios is investigated. Afterwards, the newly introduced feature, the glowing trails, and its application is described. Finally, two different types of swarm robotic implementations of this framework are illustrated, which can also be found online in (Swarmlab, Maastricht University, 2013a,b). The concluding remarks are included at the end of the paper.

Related Work

Robotic systems use vision for accomplishing various tasks ranging from mobile robot navigation (DeSouza and Kak, 2002) to industrial applications (Gonzalez and Safabakhsh, 1982). However, most of the research in this field is focused on the image processing techniques which need a centralized unit to deal with the computations and memory storage tasks. For instance, (Winters et al., 2000) used a Pentium II 350MHz PC and (Chen and Birchfield, 2006) took advantage of a Dell Inspiron 700m laptop with 1.6 GHz CPU for vision-based mobile robot navigation. (Baeten and De Schutter, 2002) used a vision-based approach for accurate and fast task execution, while all of the computations were carried out by a digital signal processing module, serialized with a host computer.

When vision is required for decentralized units (e.g., robots in a swarm) either, each agent is equipped with relatively powerful resources (e.g., in (Quinlan et al., 2003) the Sony's AIBO robots are equipped with 576 MHz CPU and 64 MB RAM), or has a centralized unit with an overhead camera which processes the image, and sends the required data to the robots (e.g., in (Ranjbar-Sahraei et al., 2012a) a host computer denotes the exact position of robots). Alternatively, (Slusn y et al., 2009) used a swarm of e-puck robots in which each robot takes an image individually, sends it to a centralized processing unit via Bluetooth, and receives the required data back from that server.

In contrast to the above mentioned works, being able to process images on a robot with very limited resources is a mandatory requirement for swarm robotic systems. In these systems using centralized units is impossible (due to the complexity and high amount of data). Equipping robots with high capabilities can be very expensive, although recent development of mobile phones can argue against this. Simple micro controllers will always be more cost effective. Therefore, in this paper we use the e-puck robot camera and its internal resources (i.e., a 60 MHz CPU and 8KB RAM) for detection of different features in the environment (e.g., barcodes, and QR-codes).

A Vision-based Framework for Swarm Scenarios

In (Alers et al., 2013) we explored several visual features that can be used for acquiring information from the environment by a robot with limited computational abilities, equipped with a camera. For detecting key locations in the environment (e.g., corners in a maze), we investigated 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, as can be seen in Fig. 1a. Furthermore, we explored the possibility to detect markers with an even higher data density: QR-codes, as in Fig. 1b. 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, we explored the most common feature already available in every swarm robotic setting: the presence of other robots. It's always favorable to detect the relative distance and orientation to 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, see Fig. 1c. Moreover, we designed a specific gradient pattern for nearby robot detection, as shown in Fig. 1d, which can result in a very accurate orientation and distance detection.

(a) (b) (c) (d)

Figure 1: Detectable features presented in (Alers et al., 2013) (a) Landmarks with barcode. (b) QR-code level 3. (c) Robot LEDs. (d) Robot orientation pattern.

Performance Evaluation

In this section the features, which were briefly addressed in the previous sections (described in more detail in (Alers et al., 2013)), are evaluated for their usability in real-world settings.

We start by evaluating several image filter techniques that are used to transform the captured image into a more suitable format. After this transition we run several utility functions, to cluster pixels or detect specific patterns. These filters and utility functions are the very basis of the feature detection and will run as a pre-processing step on every captured frame before the image is passed through to the actual feature detectors. Then we will describe the environmental influences and corrections needed for optimal performance under different circumstances. We also evaluate the detection performance based on distance variations of the detectable objects. Finally we give an overview of the time that is needed to detect each feature.

Filter and Utilities

To see how applicable the filters are in a real robot setup it is important to know whether the filters can be used in

real-time for the specific setup, and whether for this particular setup all filters can be implemented due to the memory constraints of the platform. Therefore, in this subsection we provide time and memory allocation measurements of the provided filters, such as grayscale, Hue, several forms of Halftoning, and Gaussian blur filters. We also provide this information for the *(*group detection), *(*pattern finding) and *(*Hough transformation) utilities, which are described in more detail in (May, 2013). Furthermore, we provide overall detection-time measurements of the robot camera, that is needed to capture images.

Filters The various types of filters are the core elements of our feature detection. During the detection of a specific feature, often several filters are needed, and sometimes a single filter is used multiple times in one process. To measure the performance and increase the time measurement accuracies, we run the filters 1000 times repeatedly on one picture, and then divide the overall time by 1000 which gives us the running time for one specific filter. As the filters work independent of the data there should be no influence of the actual image to the measurement results. In Table 1 the average time that each filter needs to process a single image with a resolution of 40x40 pixels is listed, this table also lists the memory requirements for each filter. Looking at the memory requirements it should be noted, that only 8000 bytes are available, and a simple filter iteration can already consume a considerable amount of memory (e.g., a single run of the Halftoning histogram filter requires 3.4% of the total memory).

Filter	Time	Memory
grayscale	17.9ms	6 Byte
Hue filter	31.7ms	20 Byte
Halftoning threshold	3.3ms	2 Byte
Halftoning average	5.9ms	8 Byte
Halftoning midpoint	6.9ms	8 Byte
Halftoning histogram	18.8ms	272 Byte
Gaussian blur	24.4ms	7 Byte

Table 1: Resources used by filters

As can be seen in Table 1, the Hue filter is the most time consuming, this is due to the fact that it converts every pixel to the HSL colorspace. The Halftoning histogram filter is the most memory consuming. On the other hand the Halftoning filter with a specific threshold is the fastest and lowest memory consuming filter, as it needs no pre-calculation and can directly process the image.

Utilities The performance of the utilities described in (May, 2013), are more difficult to measure, as they depend on a various number of parameters. Therefore, each utility is evaluated in a separate part. Similar to filters, all utilities, except for the Hough transformation, run 1000 times on a single image to make a more accurate time measurement.

I Group Detection

The group detection algorithm locates all clusters of pixels that are non black, and determines their groups center points. The run-time of this algorithm depends on the amount of clusters that are detected, and on the amount of pixels that are included in a group. As can be seen in Table 2, the performance is highly dependent on the number of pixels per group. All processed images have a resolution of 40x40 pixels.

Groups	Pixel per group	Time	Memory
1	1	4.1ms	15 Byte
	9	6.1ms	41 Byte
	25	10.8ms	123 Byte
2	1	4.9ms	15 Byte
	9	8.8ms	41 Byte
	25	16.1ms	123 Byte
3	1	5.3ms	15 Byte
	9	12.5ms	41 Byte
	25	21.1ms	123 Byte
10	160	398.6 ms	15 Byte

Table 2: Resources used by group detection algorithm

Based, on the results in Table 2, this group detection algorithm should only be used in situations, where time is not a critical factor, as we usually deal with detection of about 10 groups.

II Pattern Finder

There are two pattern characteristics which can influence the performance of the pattern finding algorithm; first the length of the pattern, and second the number of reoccurrences of the pattern in the image. Due to the fact that the pattern finder searches only a single column or row at each iteration, we have to make sure that this row or column is as long as possible. According to the specifications of the camera, the maximum length in vertical alignment is 480 pixels and in horizontal alignment 640 pixels. The latter direction is chosen for the performance measurements. The pattern itself has no influence on the performance, as the image is stored in run-length encoding, but the number of color changes in the pattern do influence the results.

As can be seen in Table 3, the time and the memory consumption mainly depend on the size of the pattern. The second parameter gives the reoccurrence of the pattern in the detection area. In practice this means that the algorithm needs up to 42.3ms, to scan every line of a halftone image with 96x96 pixel. From this we can

Pattern length	Matches	Time	Memory
3	1	2.8ms	33 Byte
	3	2.9ms	33 Byte
	5	2.9ms	33 Byte
9	1	4.0ms	45 Byte
	3	4.0ms	45 Byte
	5	4.1ms	45 Byte
15	1	4.8ms	57 Byte
	3	5.0ms	57 Byte
	5	5.1ms	57 Byte

Table 3: Resources used by pattern detection algorithm

conclude that it is possible to process even large images in an appropriate amount of time.

III Hough Transformation

We implemented two versions of the Hough transformation, the standard one and the fast transformation. The standard Hough transformation is a time consuming algorithm, which can detect multiple lines. Due to the limitations of the memory of the e-puck, the classical Hough Transformation can not run at the same precision as the Fast Hough transformation, which can only detect one line. Unlike the other utilities presented, the Hough transformation is independent of the image data, it always needs the same amount of time and memory. All tests were done with a grayscale image of 40x40 pixel.

Algorithm	Time	Memory
Hough Transformation	2721.3ms	1800 Byte
Fast Hough Transformation	121.2ms	180 Byte

Table 4: Resources used by Hough Transformation

As can be clearly noted from Table 4, the Fast Hough transformation, reduces the run-time requirements by a factor of approximately 22, and the Memory requirements with a factor 10.

Camera In addition to the different filters and utilities, the performance and speed of the camera itself, on the overall performance of the system is measured. Pictures are captured in color, grayscale and halftone mode, and the capture time is computed by measuring the overall time required for capturing 1000 images and calculating the average as shown in Table 5. For color and grayscale images the same picture size is chosen. The advantage of halftone images is their higher data resolution, to accommodate this the resolution of the halftone pictures is higher than the one for color and grayscale.

Just capturing a single image takes much more time than running a single algorithm needed to process the image.

Mode	Size	Time	Memory
Colored	40x40 pixel	153.1 ms	3200 Byte
Grayscale	40x40 pixel	80.7 ms	1600 Byte
Halftone	96x96 pixel	1734.2 ms	2752 Byte

Table 5: Resources used by camera for different modes (i.e., color, gray-scale, halftone)

This can be a downside when a situation requires higher capture details, which can be done by taking multiple images and adding the processed data together. This will lead to higher accuracy, but at the cost of processing time.

Environmental Light Condition

Light is a part of the environment which is uncontrollable in the real world. Depending on where, and how bright the light sources are, the images taken by the robot differ in their contrast and brightness. The robot's internal camera calibration tries to eliminate the influence of the light, and tries to return the same image for different brightnesses. With decreasing light intensity, it is more complicated for the auto correction to fulfill its task. To evaluate the light conditions we have built a box, which has white walls and is closed on all sides except for the top. This box, as can be seen in Fig. 1c, contains the robot, and the top is closed with a laptop monitor. The monitor acts as a controllable light source and has a constant light distribution of 300 cd/m^2. In all our experiments we place de detectable feature inside this box and decrease the light intensity in 10 steps of 30 cd/m^2, from full brightness till complete darkness.

Landmark In the first test landmarks are placed in the box. As shown in Fig. 2, the brightness of images is not that much influenced by the light condition. The first five images have almost the same brightness, but the noise density increases. Only in the last image, when the light intensity is completely reduced, the landmark is not recognizable any more.

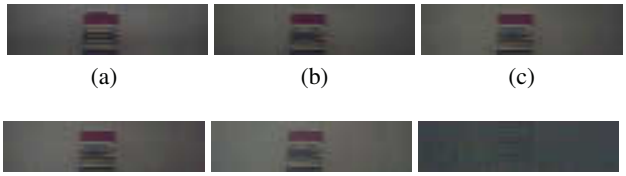
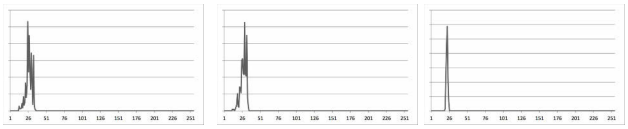


Figure 2: Detection of landmark for different brightnesses (a) 100% (b) 80% (c) 60% (d) 40% (e) 20% (f) 0%

From the histogram values in Fig. 3 it can be seen that the colors are all moving to the same area. This effect is due to the automatic light gain correction of the e-puck camera. It

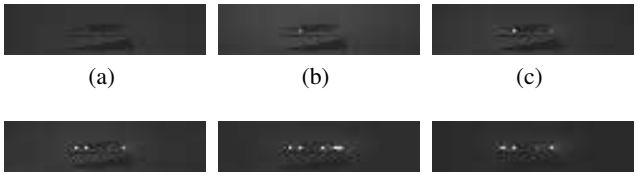
influences the brightness in a way, that the average value of all pixels is always the same.



(a) (b) (c)

Figure 3: Grayscale histogram of landmark with brightness of the environmental light at (a) 100% (b) 60% (c) 0%

Robot Detection In order to examine how the e-puck camera handles the LEDs of another e-puck, we show in Fig. 4 the influence of environmental light on a captured image. Unlike in the images of the landmarks in previous experiment, there are significant differences in these images of this experiment. At 100% brightness, the LEDs cannot be differed from the surrounding environment (see Fig. 4a), but the body pattern is clearly detectable. With low light conditions, as can be seen in Fig. 4f, the LEDs are clearly distinguishable from the remaining image, but the body pattern cannot be detected anymore.



(a) (b) (c)

(d) (e) (f)

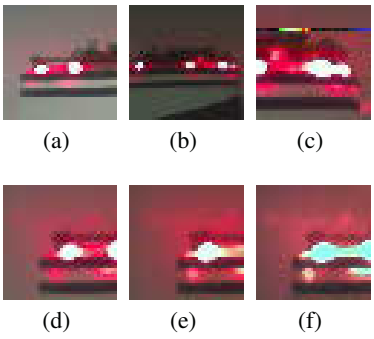
Figure 4: Images taken from an e-puck with brightness at (a) 100% (b) 80% (c) 60% (d) 40% (e) 20% (f) 0%

This effect can also be seen, when the camera zooms in into one of the single LEDs, that was detected in the overall scene. In Fig. 5a the white part of the LED fills only a small part of the image and the single LEDs can be clearly distinguished, while in Fig. 5f the two LEDs are forming one big part.

The influence of the environmental light on the detection of LEDs is very important. In bright environments the probability of detecting this feature is much smaller. In a dark environment, other features, like the body pattern, cannot be detected anymore. An optimal condition could be a light setting similar to the one in Fig. 4d, where the LEDs can be clearly differentiated from the environment and the body pattern still can be detected.

Feature Distance

In this subsection, we address how reliable the e-puck can detect specific features, and how the detection rate is influenced by the distance to a specific feature. For this test, we placed the robot in a white environment in front of only one detectable feature. The e-puck has to detect the feature and calculate its estimated position. The test is repeated 100 times with different angles and distances to the feature.



(a) (b) (c)

(d) (e) (f)

Figure 5: Zooming in to a detected LED of an e-puck with brightness at (a) 100% (b) 80% (c) 60% (d) 40% (e) 20% (f) 0%

Color Block In Table 6 the results of running the color block detection algorithms are shown. During the test distance varies from 20cm to 80cm. Distances below 20cm are not taken into account, because the colored block is not visible in such distances.

Distance	Detection rate
20 cm	96 %
25 cm	97 %
30 cm	96 %
35 cm	96 %
40 cm	89 %
45 cm	78 %
50 cm	79 %
55 cm	68 %
60 cm	51 %
65 cm	38 %
70 cm	15 %
75 cm	7 %
80 cm	1 %

Table 6: Test results for colored block of landmarks

EAN-8 codes The EAN-8 code is a part of a landmark. As the colored block is also part of the landmark, the EAN-8 detection can benefit from this and it's relative good detection rate. Each time a purple block is detected the robot can be sure that the EAN-8 barcode is located below this block. However, the exact range of the EAN-8 code still has to be detected. The correctness of detecting and decoding the barcode depends on the distance from which the e-puck reads the EAN-8 code, as can be seen in Table 7. Values below

20 cm are not tested as this is the minimum distance to see the whole EAN-8 bar code. Table 7 also shows the correctness of the data after decoding, which is much lower than just detecting a valid pattern.

Distance	Detection rate	Correctness
20 cm	93 %	68 %
25 cm	89 %	58 %
30 cm	70 %	32 %
35 cm	58 %	5 %

Table 7: Test results for EAN-8 codes of landmarks

QR-Codes QR-codes provide high density information, but are quite complex to read. For example, one dimensional bar codes just have to be scanned in a line over the image, two-dimensional codes have to be transformed so that they fit in a fixed rectangle, only then it is possible to process them further. Hence, the detection of the outer shape is very important, but also often fails. The higher the QR-version, and thus the number of modules inside the code, the more exact the shape has to be determined. In Table 8 the detection rate of different versions at their optimal distances are listed. All measurements are done from a position right in front of the pattern. The error correction level is set to H, the highest possible level. The correctness is only determined in the cases where the code was correctly detected.

Robot Detection In practice, the detection of other robots is done with two different algorithms. First the program tries to detect the body pattern of the robot. When this algorithm does not detect any robot, robot localization bases on the LEDs is performed. In Table 9 the detection rate at different distances is listed, as well as the distance estimation. As the LED based detection does not return any distance estimation the value in the third column is only calculated if the e-puck is detected by the body pattern recognition algorithm.

Overall Performance

Considering all of the in-detail examination, the performance of a feature detection algorithm is a combination of the processing time of the involving filters, utilities, and also image capturing time. The exact time depends on how often each step has to be executed, and if the objects are recognized by the detection algorithms. In Table 10 the overall processing times for the different features are presented.

Version	QR code size	Detection rate	Correctness
1	21x21	80 %	53 %
2	25x25	48 %	5 %
3	29x29	23 %	0 %

Table 8: Light tests of QR-code detection

Distance	Detection rate	Distance estimation
10 cm	100 %	95 %
15 cm	100 %	79 %
20 cm	75 %	60 %
25 cm	84 %	—
30 cm	58 %	—
35 cm	61 %	—
40 cm	52 %	—
45 cm	39 %	—
50 cm	20 %	—
55 cm	9 %	—

Table 9: Distance tests of Robot detection

Algorithm	Time
Colored block	204.3 ms
EAN-8 code	324.1 ms
Single LED	900.0 ms
Three LEDs	2334.8 ms
Body pattern	198 ms
QR-code	3253.0 ms

Table 10: Required time for detection of the features

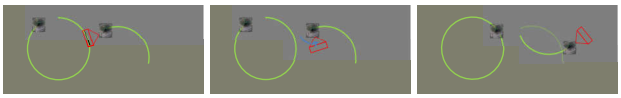
From these overall feature detection times we can conclude that detecting static objects such as colored blocks and EAN-8 codes in a real swarm robot scenarios is doable. Even detecting a QR-code is possible, as long as the location of this code in the environment is known, as searching for a QR-code, using vision, requires high processing time, and memory.

Glowing Trails

As an extension to our current framework we introduce a new feature, the glowing trail. Inspired by nature, in which insects use chemicals for indirect communication (known as Stigmergy), researchers are interested in applying stigmergy in multi-robot systems, as well. However, in-field deployment of an indirect communication requires manipulating environment which is not a trivial task. Researchers have recently used a few techniques for accomplishing this task. For instance, chemical materials has been proposed by (Fujisawa et al., 2008). Due to difficulties in implementation and limited extendibility, this approach didn't provide sufficient applicability in swarm scenarios.

As an alternative, glowing trails exploited by (Kronemann and Hafner, 2010), inspired on (Alers and Hu, 2009), and further extended for swarm robotic scenarios by authors (Ranjbar-Sahraei et al., 2013) are easy to set up in laboratory environment and still very efficient. These glowing trails can help robots to communicate indirectly to achieve their goals (e.g., environmental coverage, intruder tracking, etc.). A simple indirect communication is shown in Fig. 6 in which robots announce their territory border by putting

pheromones on the borders (Ranjbar-Sahraei et al., 2012b).



(a) (b) (c)

Figure 6: Using stigmergic communication for efficient area coverage proposed by (Ranjbar-Sahraei et al., 2012b) (a) robot on right hand side detected the glowing trail. (b) robot changes its circling direction. (c) robots establish separate territories

For detection of the glowing trails, in contrast to the simple method used in (Kronemann and Hafner, 2010), in which photo-sensors were used to detect glowing trails, the e-puck vision approach as described above can be used. Therefore, we take advantage from the developed techniques for color filtering and pattern recognition which are designed based on the limited resources of an e-puck robot, and still powerful enough to extract information from the trails.

The new glowing trail feature can be seen in Fig. 7. The detected grayscale image is converted to a black-white image with a fixed threshold. The amount of white pixels is determined to see if there is any trail in the image. When the image has more than 1% of white pixels, a Fast Hough Transformation (Gonzalez and Woods, 2002) is performed to determine the direction of the trail, see Fig. 7(b).

For using glowing trails, the floor should be covered by phosphorescent material which absorbs UV light and re-emits the absorbed light at a lower intensity for up to several minutes after the original excitation. Robots should also be equipped with UV-LEDs to emit light to the glowing materials.

(a) (b)

Figure 7: New introduced feature: (a) Initial image from a glowing trail received by e-puck camera. (b) Filtered image with a red directional line determined by Hough-Transformation (May, 2013)

Real World Evaluation

To test the proposed features and algorithms under real conditions we used this framework to implement two different swarm approaches. One approach focuses on using these features and algorithms in path optimization problems as in (Alers et al., 2011), the other swarm approach focuses on area coverage with glowing trails as in (Ranjbar-Sahraei et al., 2012b).

For the path optimization approach we described and implemented our framework in (Alers et al., 2013). We examined the proposed approach in a real scenario, in an environment as shown in Fig. 8. A video of this performed experiment can be found online in (Swarmlab, Maastricht University, 2013b), including the intermediate image data from the robot.

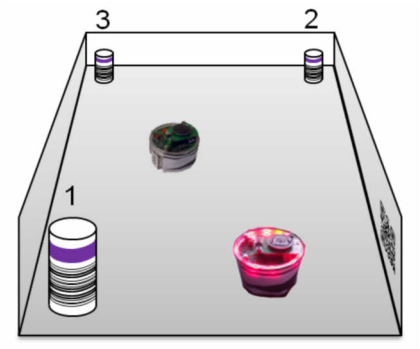


Figure 8: Scenario for validation of proposed approach

For the coverage approach, we used the new glowing trails framework extension, introduced in previous section. In (Ranjbar-Sahraei et al., 2013), we demonstrated an implementation of this approach, the results of applying this vision-based trail detection on a real swarm of e-puck robots is shown in Figs 9a- 9c. A video of the performed experiments can be found in (Swarmlab, Maastricht University, 2013a).

(a) initial (b) intermediate (c) final

Figure 9: Vision-based detection of glowing trails approach

Discussion, Conclusions & Future Research

In this paper we presented an in-depth study of a vision-based feature detection framework for multi-robot scenarios. This study covered a complete range of performance evaluations, ranging from measuring detection rate for different environmental brightnesses and detection distances to detection accuracy for different features. Furthermore, various experiments on a real e-puck robot were performed to measure the required time for different tasks such as applying gray-scale filter, halftone filter, group detection, and pattern finding algorithms.

From the overall performance measurements we can conclude that detection of objects that are not too complex is easily doable. However, the detection time, and the required memory increases drastically when more complex objects are chosen as features. Moreover, we showed that the environmental light variation doesn't affect the detection of the

features, except in the extremes when it is either too bright or too dark. The detection of other robots via their LEDs is doable in a static situation, but in most swarm scenarios the robots move continuously in the environment, which makes detection of moving robots an open research question in our research. Moreover, for the newly introduced glowing trail feature we demonstrated a working dynamic multi-robot scenario, which encourages us to investigate applications in dynamical swarm-optimization settings. Finally, we as the main future work, we are working on integration of the proposed techniques with te bee-inspired foraging algorithms, where the big challenge in this is to fit all foraging and vision algorithms within the limited memory of the robot.

References

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

Alers, S. and Hu, J. (2009). Admoveo: A robotic platform for teaching creative programming to designers. In *Proceedings of the 4th International Conference on E-Learning and Games: Learning by Playing. Game-based Education System Design and Development (Edutainment 2009)*, Edutainment '09, pages 410–421, Berlin, Heidelberg. Springer-Verlag.

Alers, S., Ranjbar-Sahraei, B., May, S., Tuyls, K., and Weiss, G. (2013). An experimental framework for exploiting vision in swarm robotics. In *ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications*.

Baeten, J. and De Schutter, J. (2002). Hybrid vision/force control at corners in planar robotic-contour following. *Mechatronics, IEEE/ASME Transactions on*, 7(2):143–151.

Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. (2005). Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376–386.

Chen, Z. and Birchfield, S. T. (2006). Qualitative vision-based mobile robot navigation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2686–2692. IEEE.

Cortes, J., Martinez, S., Karatas, T., and Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255.

DeSouza, G. N. and Kak, A. C. (2002). Vision for mobile robot navigation: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237–267.

Dorigo, M., Bonabeau, E., and Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871.

Fujisawa, R., Imamura, H., Hashimoto, T., and Matsuno, F. (2008). Communication using pheromone field for multiple robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1391 –1396.

Gonzalez, R. and Woods, R. (2002). *Digital image processing*. Prentice Hall Upper Saddle River, NJ.

Gonzalez, R. C. and Safabakhsh, R. (1982). Computer vision techniques for industrial applications and robot control. *Computer*, 15(12):17–32.

Hennes, D., Claes, D., Meeussen, W., and Tuyls, K. (2012). Multi-robot collision avoidance with localization uncertainty. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 147–154. International Foundation for Autonomous Agents and Multiagent Systems.

Kronemann, M. L. and Hafner, V. V. (2010). Lumibots making emergence graspable in a swarm of robots. In *DIS 2010, The ACM Designing Interactive Systems Conference*, pages 408–411, Aarhus. ISBN 978-1-4503-0103-9.

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

Lemmens, N. P.-P. M. (2011). *Bee-inspired Distributed Optimization*. Maastricht University.

May, S. (2013). E-puck vision: Detecting key features with limited resources. Master's thesis, Maastricht University.

Quinlan, M. J., Chalup, S. K., and Middleton, R. H. (2003). Techniques for improving vision and locomotion on the sony aibo robot. In *Proceedings of the 2003 Australasian Conference on Robotics and Automation*.

Ranjbar-Sahraei, B., Alers, S., Tuyls, K., and Weiss, G. (2013). Stico in action (demonstration). In *Proceedings of the twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Ranjbar-Sahraei, B., Shabaninia, F., Nemati, A., and Stan, S. (2012a). A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems. *Industrial Electronics, IEEE Transactions on*, 59(8):3124–3134.

Ranjbar-Sahraei, B., Weiss, G., and Nakisaee, A. (2012b). A multi-robot coverage approach based on stigmergic communication. In *Multiagent System Technologies*, volume 7598 of *Lecture Notes in Computer Science*, pages 126–138. Springer.

Slusn y, S., Neruda, R., and Vidnerová, P. (2009). Localization with a low-cost robot. In *ITAT*, volume 584 of *CEUR Workshop Proceedings*, pages 77–80.

Swarmlab, Maastricht University (2013a). Demonstration of an experimental framework for exploiting vision in stigmergic communication. http://swarmlab.unimaas.nl/stico/indoor-experiments/.

Swarmlab, Maastricht University (2013b). Demonstration of an experimental framework for exploiting vision in swarm robotics. http://swarmlab.unimaas.nl/papers/adaptive-2013-demo/.

Winters, N., Gaspar, J., Lacey, G., and Santos-Victor, J. (2000). Omni-directional vision for robot navigation. In *Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on*, pages 21–28. IEEE.