6

Visual Analysis of Robot and Animal Colonies

E. Martínez and A.P. del Pobil
Robotic Intelligence Lab, Jaume-I University Castellón, Spain
Interaction Science Dept., Sungkyunkwan University, Seoul, S. Korea

1. Introduction

Micro-robotics calls for the development of tracking systems in order to study the movement of each micro-robot in a colony to answer questions about **what** they are doing and **where** and **when** they act (see Fig. 1). Moreover, micro-robots can be used to emulate social insect behaviour (Camazine et al., 2001) and study them through tracking experiments involving several miniature robots on a desktop table. Thus, principles of self-organization in these colonies, which were studied so far by analysis of a tremendous amount of insect trajectories and manual event counting, are now better understood by biologists thanks to robotics research.

Fig. 1. Analogy between social insects (left) and a micro-robot colony (right)

Although this approach is only recently increasing its popularity, computer vision systems for tracking moving targets are widely used in many applications such as smart surveillance, virtual reality, advanced user interfaces, motion analysis and model-based image coding (Gavrila, 1999). Surveillance systems seek to automatically identify people, objects or events of interest in different kind of environments (Russ, 1998; Toyama et al. 1999; Haritaoglu et al., 2000; Radke et al., 2005). However, this problem is not easy to solve. First of all, it is not viable to tag each colony member under study. On the one hand, the tag selection process can be difficult since tags must be very small in some cases and, therefore, it might not be possible to detect them in an image. Furthermore, tags can become ambiguous when a swarm is composed of many individuals. On the other hand, tagging can alter individual behaviour. So, an application for tracking unmarked object has been developed. A new problem arises: how to identify the same object in two consecutive *frames*. SwisTrack (Correll et al., 2006) is a previous work following this approach which we try to improve. It is a platform-independent, easy-to-use and robust tracking software developed to study robot swarms and behavioural biology. It is part of the European project LEURRE (http://leurre.ulb.ac.be.2006) focused on building and controlling mixed societies composed

of animals and artificial embedded agents. In preliminary case studies towards this aim (Caprari et al., 2005), an Insbot team has been introduced into a swarm of cockroaches and allowed for modification of the natural behaviour of the swarm. We will show that our application achieves robust performance in object identification and tracking without the need of a strong intervention by the user.
As the structure of the designed method (see Fig. 2), this paper is organized as follows: the visual segmentation and detection of objects are described in Sec. 2 and 3. In Sec. 4 we outline the tracking problem and its solution. Experimental results are given in Sec. 5 and discussed in Sec. 6.

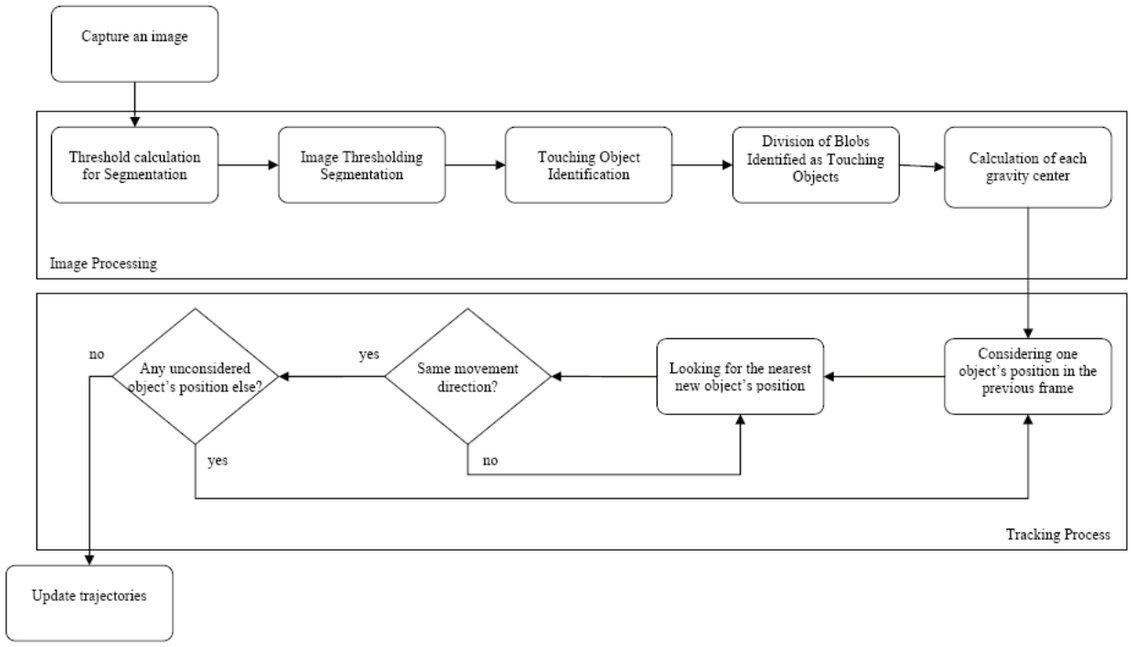


Fig. 2. Flowchart of the whole designed method

2. Object Segmentation

A common element of any surveillance systems is a module that is capable of identifying the set of pixels which represents all the individuals under study in each captured image. There are several techniques to carry out this task. For example, the background modeling approach (Toyama et al., 1999; Haritaoglu et al., 2000) allows to model dynamic factors such as blinking of screens, shadows, mirror images on the glass windows or small variations in illumination due to flickering of light sources. Pixels are classified as background or foreground depending on the fitting of their values with the built model. As a drawback, this method is not capable of adapting to sudden illumination changes, and we ruled it out with the aim of developing a more robust surveillance application in the presence of variation of lighting conditions. On the other hand, most of the alternative techniques are developed by gray-level image processing so that if available images are in color, it is necessary to convert them to gray-level. As our system has color images as input, the binary image resulting from the segmentation process can be obtained from a combination of three

binary images (each color channel generates a gray-level image which is segmented by the selected method obtaining a different binary image), or from a gray-level image obtained directly from the color one.

Thus, the first step is to convert the captured color image to a gray-level one. For such preprocessing, the Hue-Saturation-Intensity (*HSI*) system is used since it encodes color information by separating an overall intensity value *I* from two values encoding *chromaticity* - hue *H* and saturation *S*. *HSI* might also provide better support for computer vision algorithms because it is amenable to normalization for lighting and focus on the two chromaticy parameters that are more associated with the intrinsic character of a surface rather than the lightning source. Thus, the resulting gray-level image is only built from the *intensity* value. Derivation of *HSI* coordinates from *RGB* coordinates is a common process in computer vision (Shapiro and Stockman, 2001).

Once the gray-scale image is available, a segmentation process has to be applied on it. Although frame difference is the easiest and fastest method to detect moving objects in an image, it fails when the objects are steady. This problem could be solved by taking a reference image with no objects and subtracting it from the new ones. Nevertheless, a little illumination change might make the whole process fail, thus, an alternative algorithm should be chosen. In our case, the corresponding binary image is obtained from an input gray-scale by thresholding operations as in *Swistrack*. This technique defines a range of brightness values in the original image: pixel value greater than a threshold (or lower, depending on its definition) belongs to the foreground and the rest of pixels are classified as background. The drawback of this method is the correct determination of the threshold. In *Swistrack* two different kinds of reference images are used, depending on the mode in which the system is operating:

- **Static background**: the background image is captured at the beginning of the experiment and is not updated at any time;
- **Running average**: the reference image is built as the running average of all video *frames* processed until that moment

The threshold is fixed in both cases and represents the minimum difference required to classify a pixel as foreground. It is important to note that the fixed threshold is sensitive to changes in lighting conditions, especially in the first operation mode in which the reference image is not updated during all the experiment. Although the running average is more capable of dealing with illumination changes, it might consider objects that stop moving for a long period of time as part of the background, without detecting their presence in the scene. We have implemented a method for automatically calculating the threshold based on histogram properties by updating its value in each frame, in order to adapt it to variations of the lighting conditions. This provides an advantage over the *Swistrack* method, as significant intensity differences between the objects to be tracked and the background are not necessary.

After the threshold setting, two consecutive morphological operations are applied on the binary image resulting from the segmentation process. These steps are required to erase isolated points or lines caused by different dynamic factors such as, for example, changes induced by camera motion, sensor noise, non-uniform attenuation, blinking of lights or atmospheric absorption. A 3x3 erode filter is used to delete these artefacts and then a 3x3 expand filter is applied to recover the foreground region. A result of the whole process can be observed in Figure 3, where a group of micro-robots is seen from above. As it can be seen,

although the lighting conditions are not good in some places, the designed application is capable of detecting all visible micro-robots in the image. However, the pixels due to light reflexions on the arena are not removed from the image. This issue will be solved by means of the implemented labeling method described in next section.

Fig. 3. An image captured by the system camera (left), the binary image obtained by the segmentation process (center) and the resulting binary image after applying two morphological operations (right)

3. Object Identification

The aim of this stage is to obtain a labeled image in which each label identifies one colony member. This can be a difficult task when objects are touching one another, because an identified *blob* contains all objects with, at least, one point in common. The method implemented to achieve the above goal can be divided into three steps:

1. Labeling of the identified *blobs* in the input binary image. A row-by-row labeling method (Shapiro and Stockman, 2001) is used to reduce the computational cost of the whole process. The binary image is scanned twice: the first time to tag each foreground pixel based on the labels of its neighbors and to establish equivalences between different labels; the second, crossed scan, will unify tags which belong to the same *blob*
2. Classification of the labeled *blobs* as targets to be tracked or as bad-segmented pixels, and detection of collisions inside a *blob* identified as a tracking target. All targets are assumed as not touching in the first captured image. The number of targets to be tracked is specified by the user, and the application calculates the minimal and maximal size allowed from the first captured image. This knowledge, together with the number of the detected *blobs* in each frame, allows to define a series of criteria to determine when a *blob* represents more than one object, and to reject all *blobs* that do not identify target objects but are instead the result of a bad segmentation, as it occurs with the set of noisy pixels in the example of Fig. 3
3. Segmentation of *blobs* that represent groups of more than one object to be tracked. This step is explained in more detail in the next section

As seen above, it is possible that the same *blob* identifies more than one object to be tracked. Thus, it is important to detect these situations and split up the *blob* in the corresponding objects. There are two different tasks to be performed: determining the number of touching objects inside the same *blob* and splitting them up.

The first goal is achieved through several criteria which are relationships between *blobs* and object sizes. They can be easily set up by the application, assuming that no target objects are touching in the first captured image. Thus, our application calculates the maximum and

minimum dimensions of the visual objects from the first captured scene and the criteria remain set. This step is important because perceived object size can vary with the distance between the arena and the camera and if an object size in pixels is directly related to its real size, as in *Swistrack*, a calibration error can be introduced in all calculations. Due to this error, the application might fail the tracking process. So, the only parameter our application needs, which is requested to the user, is the number of objects to be tracked.

The next step is to split up the different objects which compose one *blob*. As it is difficult to identify several objects at the same time, we have studied three different, possible situations assuming that only two objects are touching. Thus, our application split up any complex *blob* in two different parts: one target object and another *blob* which can be again composed of more than one micro-robot. If the new *blob* represents several micro-robots, the split-up process is recursively applied until the obtained *blob* is composed of only one target object. The three different cases that have been studied are the following:

1. two objects touching only in one point;
2. two objects sharing one side, that is, they are horizontally or vertically aligned;
3. two objects touching in several points which do not correspond to their sides

In the first case, a contour method is used. A chain code is calculated by considering that the contact pixel will be visited twice. The method takes into account the contour irregularities due to the segmentation process and it applies different criteria to determine the correct contact point as shown in Fig. 4.

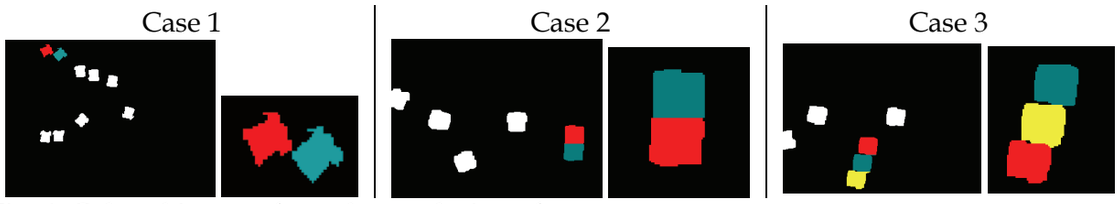
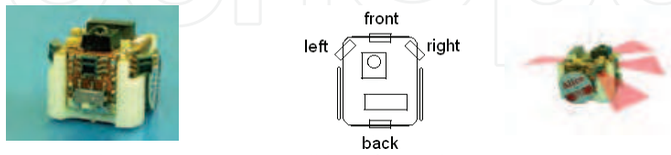


Fig. 4. Colored images from the resolution of contact cases

The second case, when two objects share one side, has been considered because of the micro-robots used in our experiments, the *Alice*2002 (Caprari and Siegwart, 2005) (see Fig. 5). These micro-robots can be seen from above as boxes and two *Alice*2002 can share one side in any moment. It is important to note that this case does not apply to micro-robots that do not have a shape in which a side can be shared. A method based on dimension criteria is used to determine the common side, and it estimates their splitting line as shown in Fig. 4.



(a) A micro-robot *Alice* 2002 (b) Infrared proximity sensors

Fig. 5. Micro-robot *Alice*2002

Finally, the most general and complex case is when an object and another *blob* compose a bigger *blob*, and the contact between them is through several pixels which do not correspond to an object side. Therefore, the designed method is based on holes inside *blobs*. Again, this might be the result of a bad segmentation. For this reason, a set of criteria were defined for

detecting when a hole is due to bad segmentation and when it is a hole between two different objects. As it can be seen in Fig. 4, the designed method provides successful results.

4. Tracking Micro-robots

Once each micro-robot is identified as an object, the next step is to match each object with one of those detected in the previous frame, in order to obtain its trajectory.

Object position can be calculated as the geometrical center of gravity of object contour (Correll et al., 2006), but we decided to calculate it as the geometrical center of gravity of a *blob* corresponding to an object. Since the correspondence between an object and a *blob* has been obtained in the previous step, this procedure is easier and faster. The issue now is how to associate each center of gravity with its corresponding object between the new ones. The chosen solution, the *nearest neighbor* method, is an easy one, but its implementation raises several issues (Correll et al., 2006). The different way of dealing with them will determine the successfulness of the application.

The first challenge to solve is the case of an object that is closer to the previous position than the real one (situation already described in a previous work (Correll et al., 2006)). A quadratic assignment problem for minimizing the sum over the distance of all assignments is used in *Swistrack*, but this does not constitute an optimum solution since it fails in some cases. On the contrary, a solution focused on the previous movements of objects is presented here, that is, the matching algorithm associates the information on the nearest neighbor with that regarding the previous movement direction.

It might also be that an object disappears from the scene (Correll et al., 2006), but if the application does not detect all elements specified by the user, then it will fail when this situation occurs, because objects enter or leave the arena, and the system will repeatedly capture another image until it finds all the objects. Another situation (again presented in (Correll et al., 2006)) is when two shared trajectories are divided at the wrong time, but this is not possible in our application because *blobs* are divided into their corresponding objects even though they are touching, as previously explained. Finally, an additional skill of our application is the ability of avoiding un-associated contours.

Overall, our application is capable of solving the four different situations expounded in (Correll et al., 2006) and does not need the help of the user for correcting wrong trajectories.

It is also important to note that the modified nearest neighbor technique is only applied on a small region of the image, not on the whole image, in order to make the application faster. Again, it is not possible to fix the search area size because a delay can be produced during capture process. Thus, our application calculates the search area dimension based on capture time between *frames* and the maximum velocity of tracking objects, which will be requested to the user. This reduces the computational time and guarantees the success of the matching process.

5. Experiments

We first provide a brief overview of the robotic platform used, followed by the experimental results.

The experimental setup is depicted in Figure 6. A camera is pointing downwards to the desktop where the micro-robots are working. Our application operates with monocular

color video images (see Fig. 6). The distance between the camera and the arena can vary from one setup to another and the system calculates the parameter values needed for obtaining the different criteria. It is important to note that the background of the micro-robot workspace is black in all our experiments, while the most common solution is to use a white background.

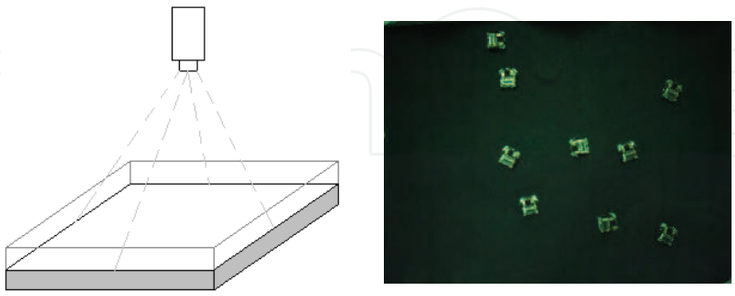


Fig. 6. Experimental setup (left) and a 320x240 captured image by the camera (right)

The objects to be tracked are *Alice*2002 robots, as mentioned above. They are extremely sensitive to external forces and can be very easily damaged if not handled with care. Among their features, we can mention:

- tiny dimensions (22 mm (width) x 21 mm (length) x 20 mm (height))
- small weight (aproximately 11 g)
- infrared proximity sensors (front, right and left) to avoid obstacles
- low consumption (12 - 17 mW)
- autonomy (up to 10 hours thanks to its Ni-MH rechargeable battery (Varta 3/V40H))
- velocity of 40 mm/s

Finally, a Graphical User Interface (GUI) has been developed to check the performance of our application. It is composed of two different windows (see Figure 7(a)): on the left, the user can observe the images taken in real-time and, on the right, a graphical window is showing the different positions of the objects. Each obtained trajectory is drawn in a different color to help the user identify each target. As the duration of the experiments is unknown and the amount of points can be considerable, the application only shows the last seventy object positions to ease tracking of each described trajectory to the user.

Two different experiments have been carried out. The first one is the tracking of three unmarked *Alice*2002 (see Figure 7). Six untagged *Alice*2002 are studied in the second experiment (Fig. 8). Both experiments have been carried out at different times of the day and in different days to test the robustness of our application to different lighting conditions.

As it can be seen in Fig. 7a, all micro-robots to be tracked are not touching in the first stage. Thus, our application can obtain the information it needs: the objects position, i.e., their geometrical center of gravity, and the maximum and minimum size allowed for any object.

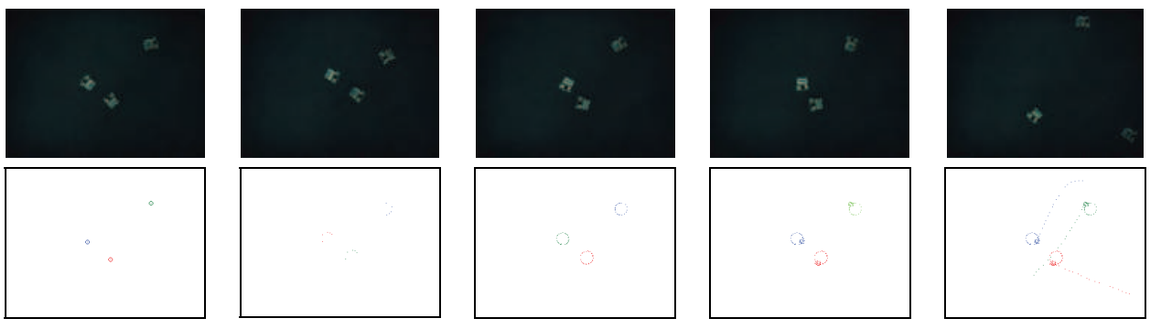
For clarity of representation, objects are highlighted by inscribing them in circles.

Although there are relevant delays between the first and the second *frames* and between the second and the third ones, our application is capable of correctly tracking the objects as shown in Fig. 7b.

To check the system in different situations, we have changed the moving pattern of the objects during the experiment. For the first 25 *frames* all three objects are describing a

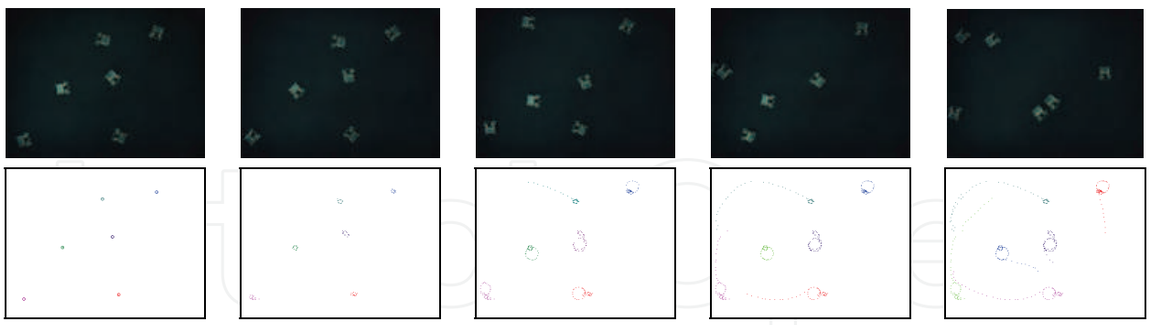
clockwise circular trajectory. A different circular trajectory with smaller radius is described during the next 25 *frames*, as shown in Fig. 7d. Finally, the micro-robots are set in wall following mode, so they describe a straight-line trajectory until they find a wall to follow, as can be observed in Fig. 7e.

The second experiment was similar. In this case, our application had to track six unmarked *Alice*2002. Again, the first captured *frame* (see Fig. 8a) reveals that the experiment starts without collisions between objects. The trajectories described in this experiment are, first, the smallest-radius circular trajectory; then, four of the micro-robots change their trajectory describing a circular trajectory with a larger radius and the two remaining ones go straight ahead looking for a wall to follow. Now, two more micro-robots change to the wall following mode. It is important to note that our application is capable of detecting objects only partially visible as shown in Fig. 8d.



(a) Initial State | (b) After 5 frames | (c) Circular trajectory | (d) Circular trajectory with a different radius | (e) Following wall mode

Fig. 7. Three *Alice*2002 tracking experiment: captured image (up) and trajectory image (down)



(a) First Stage | (b) Circular Trajectory | (c) Change in some trajectories | (d) Another change in some trajectories | (e) Following wall mode

Fig. 8. Six *Alice*2002 tracking experiment

In addition, examples of the use of our split-up method are finally shown in Fig. 9. A one-shot video segment of 10 minute duration is available at
http://www.robot.uji.es/lab/plone/Members/emartine

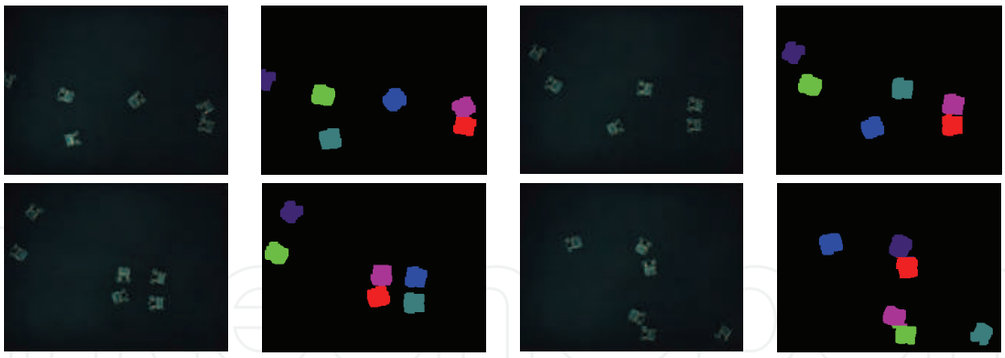


Fig. 9. Collision detection

To conclude this section, a graph is presented (see Fig. 10) which compares the trajectory followed by a member of the studied colony in a multiple-target experiment versus the trajectory obtained by the developed software. As it can be observed, there is no mismatch in data-association thanks to the implemented method to split up blobs when several members are in touch.

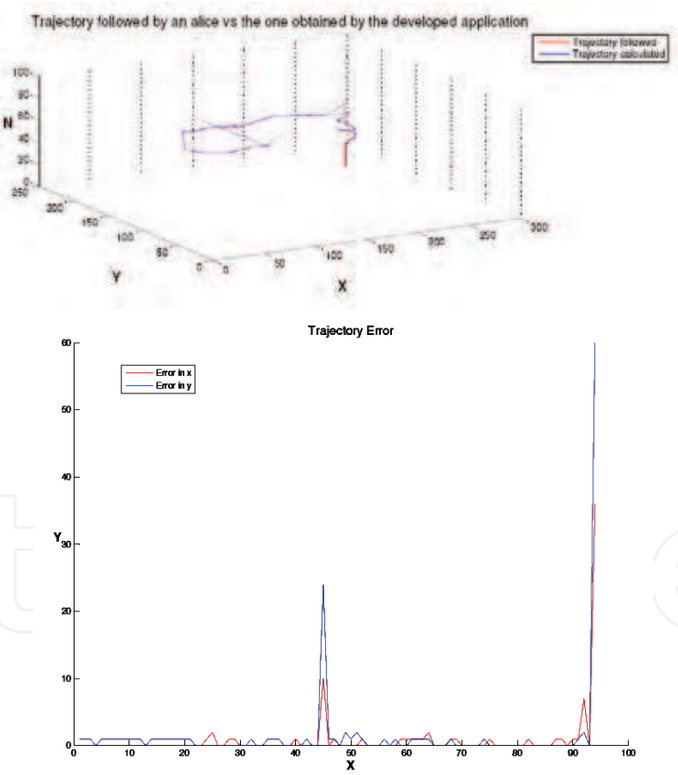


Fig. 10. Trajectory followed by a single target vs individual trajectory obtained by the implemented software (above) and error (below)

6. Conclusions

We have presented a tracking application to study micro-robots or social insect cooperative behavior without the risk of conditioning the results by tagging them. Our system has been compared with previous ones, and namely with *Swistrack*, an application intended to control mixed societies. Although this previous study had the same goal, the authors deal with the tracking problem in a different way. The given results have shown the robustness of our application with regard to lighting conditions. Also, no special illumination is required and performances do not depend on the surrounding objects, as for example it occurs in *Swistrack*.

Our designed method also solves situations in which there are several objects touching one another and it can match an object position in one frame with its position in the next frame. It is also capable of detecting objects even though their velocity is very slow or if they do not move, a case typically difficult for similar methods.

As a further achievement, our application only requires two parameters from the user: the number of target objects and their maximum speed. No thresholds need to be set manually.

Overall, we have designed an application transparent to the user who does not need to know the implementation details to work with it.

So far, our application has only been tested with homogeneous robotic societies. As further research, we plan to test it with mixed societies.

7. Acknowledges

This research was partly supported by the Korea Science and Engineering Foundation under the WCU (World Class University) program funded by the Ministry of Education, Science and Technology, S. Korea, Grant No. R31-2008-000-10062-0), by the European Commission's Seventh Framework Programme FP7/2007-2013 under grant agreement 217077 (EYESHOTS project), by Ministerio de Ciencia e Innovacion (DPI-2008-06636, DPI2004-01920 and FPI grant BES-2005-8860), by Generalitat Valenciana (PROMETEO/2009/052) and by Fundacio Caixa Castello-Bancaixa (P1-1B2008-51)

8. References

Camazine S.; Deneubourg J.L.; Franks N.R.; Sneyd J.; Theraulaz G. and Bonabeau E. (2001). Self-Organization in Biological Systems. *Princeton Studies in Complexity*, Princeton University Press

Caprari G.; Colot A.; Siegwart R.; Halloy J. and Deneubourg J.L. (2005). Building mixed societies of animals and robots. *IEEE Robotics and Automation Magazine*, 12, 2, (58-65)

Caprari G. and Siegwart R. (2005). Mobile micro-robots ready to use: Alice, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3295 – 3300

Correll N.; Sempo G.; Lopez de Meneses Y.; Halloy J.; Deneubourg J.L. and Martinoli A. (2006). Swistrack: A tracking tool for multi-unit robotic and biological systems, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2185-2191

Gavrila D.M. (1999). The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73, 1, (82–98)

Haritaoglu I.; Harwood D. and Davis L.S. (2000). W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 8, (809 – 830)

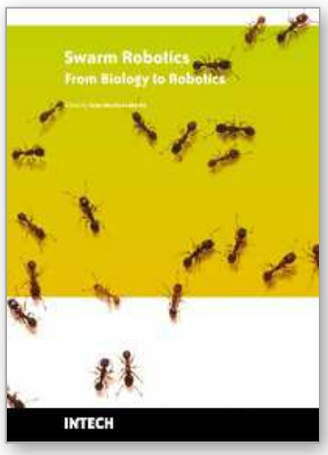
Toyama K., Krum J., Brumitt B., and Meyers B. (1999). Wallflower: Principles and practice of background maintenance, *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pp. 255 – 261, Kerkyra, Greece

Radke R.J.; Andra S.; Al-Kofahi O. and Roysam B. (2005). Image change detection algorithms: A systematic survey. *IEEE Transactions on Image Processing*, 14, 3, (March), (294-307)

Russ J.C. (1998). *The Image Processing Handbook*, CRC Press, third edition

Shapiro L.G. and Stockman G.C. (2001). *Computer Vision*, Prentice Hall, NJ

IntechOpen



Swarm Robotics from Biology to Robotics

Edited by Ester Martinez Martin

ISBN 978-953-307-075-9
Hard cover, 102 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

In nature, it is possible to observe a cooperative behaviour in all animals, since, according to Charles Darwin's theory, every being, from ants to human beings, form groups in which most individuals work for the common good. However, although study of dozens of social species has been done for a century, details of how and why cooperation evolved remain to be worked out. Actually, cooperative behaviour has been studied from different points of view. Swarm robotics is a new approach that emerged on the field of artificial swarm intelligence, as well as the biological studies of insects (i.e. ants and other fields in nature) which coordinate their actions to accomplish tasks that are beyond the capabilities of a single individual. In particular, swarm robotics is focused on the coordination of decentralised, self-organised multi-robot systems in order to describe such a collective behaviour as a consequence of local interactions with one another and with their environment. This book has only provided a partial picture of the field of swarm robotics by focusing on practical applications. The global assessment of the contributions contained in this book is reasonably positive since they highlighted that it is necessary to adapt and remodel biological strategies to cope with the added complexity and problems that arise when robot individuals are considered.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

E. Martinez and A.P. del Pobil (2010). Visual Analysis of Robot and Animal Colonies, Swarm Robotics from Biology to Robotics, Ester Martinez Martin (Ed.), ISBN: 978-953-307-075-9, InTech, Available from: http://www.intechopen.com/books/swarm-robotics-from-biology-to-robotics/visual-analysis-of-robot-and-animal-colonies

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen