SwisTrack: A Tracking Tool for Multi-Unit Robotic and Biological Systems

Nikolaus Correll*, Gregory Sempo†, Yuri Lopez de Meneses‡, José Halloy†,
Jean-Louis Deneubourg†, and Alcherio Martinoli*
*Swarm-Intelligent Systems Group (SWIS), École Polytechnique Fédérale Lausanne
†Unit of Social Ecology (USE), Université Libre de Bruxelles
‡Laboratoire de Production Microtechnique (LPM), École Polytechnique Fédérale Lausanne

***Abstract*— Tracking of miniature robotic platforms involves major challenges in image recognition and data association. We present our 3-year effort into developing the platform-independent, easy-to-use, and robust tracking software *SwisTrack*, which is tailored to research in swarm robotics and behavioral biology. We demonstrate the software and algorithms abilities using two case studies, tracking of a swarm of cockroaches, and a swarm-robotic inspection task, while outlining hard problems in tracking and data-association of marker-less objects.**
Tracking accuracy of a moving robot with respect to camera noise and the calibration model are calculated experimentally.
Its open, platform-independent architecture, and easy-to-use interfaces (Matlab™, Java™, and C++), allowing for (distributed) post-processing of trajectory data online, make the software highly adaptive to particular research projects without changes to the source code. *SwisTrack* is publicly available on *Sourceforge.net* under the OSI Adaptive License and contributions from the robotics and biology community are encouraged.

I. Introduction

Swarm robotics research has seen substantial growth in recent years, and research involving 20 or more miniature robots on a desktop table becomes more and more feasible (see for instance [1], [2]). At the same time, behavioral biologists are better understanding principles of self-organization in social insects [3], often by analysis of a tremendous amount of insect trajectories, and manual event counting.
The European project LEURRE [4] is a project on building and controlling mixed societies composed of animals and artificial embedded agents. As a preliminary case study towards a general methodology for controlling mixed animal/robot societies, a team of Insbots [5] has been successfully introduced into a swarm of cockroaches, and allowed for modification of the natural behavior of the swarm. This self-organized process is highly probabilistic, and thus conclusions can only be drawn after analyzing a large number of experiments (typically 10 to several hundred of 3h duration each) involving up to 30 individuals, a work currently in progress using *SwisTrack*, the software tool described in this paper. Similarly, for the understanding of emergence of collective intelligence from individual behavior in self-organized robotic systems, processing of entire batches of experimental data is necessary, and *SwisTrack* has been successfully applied in such framework as well (for instance in [2] or [6]).
We note that *Swistrack* allows for tracking unmarked objects. Marker-less techniques are of paramount importance in behavioral studies, especially on insects, in order to avoid biased behavior through tagging. On the other hand, markers are not always applicable on robotic platforms either, in particular when the pixel-per-object ratio is low (as it is the case in arenas that are much larger than the individual robots).

A. Related Work

While miniaturization of robotic platforms [1], [5], is only recently promoting swarm-robotics research, and hence makes development of tracking systems necessary, demand for tracking living societies, such as social insects [7]–[9], mammals [10], and pedestrians [11], [12], is much older, and research from these areas might find its application also for swarm robotics tasks.
Due to the difficulties involved and due to the specialization of the task, there are only few tracking packages commercially available, and all that are known to the authors are not applicable to tracking of large numbers of untagged individuals, and do not provide access to their source code.

II. Software Architecture

SwisTrack is written in C++, and uses the freely available WxWidgets [13] API for platform independent Graphical User Interface (GUI) and networking (via TCP/IP) functionalities. The tracking core, and image and video processing, as well as access to a suite of USB Webcams is provided by the Open Source toolbox OpenCV [14] from Intel Inc. Cameras using the IEEE 1394 "Firewire" standard are supported by the 1394 Camera Toolkit [15] (MS Windows only) developed at Carnegie Mellon University, Pittsburgh, USA.
Besides storing trajectory information to disk, *SwisTrack* is able to serve data and to receive commands via a TCP/IP networking connection, making the separation between tracking and post-processing code simple (either by post-processing trajectory information stored to the hard-disk, or online by interacting with the *SwisTrack* TCP/IP interface). As a side-effect, the computational load for post-processing can be distributed on other computers. *SwisTrack* is distributed with skeleton clients for Matlab, C++, and Java that allow for fetching tracking data (blobs as well as trajectory information) and remote controlling *SwisTrack*.

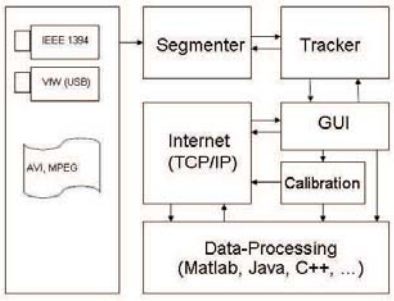

Fig. 1. Block-Diagram showing the object-oriented structure of *SwisTrack*, and its functionality. Typically, data processing is implemented by the user.

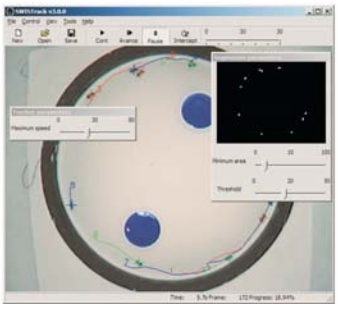

Fig. 2. Tracking of mixed animal/robot societies. Illustration of panels allowing for adjusting of segmenting and tracking parameters. The arena is populated with cockroaches and one Insbot.

The object-oriented architecture and the different, well isolated, function blocks are depicted in the block-diagram in Figure 1. The segmenting and tracking processes are detailed in Section III and IV respectively. A screen-shot of *SwisTrack* engaged in tracking of a swarm of cockroaches and an Insbot, as well as the segmenting and tracking panels are shown in Figure 2.

III. Image Segmentation

Segmenting the image into potential tracking targets (blobs) and background information is a preliminary step for all further operation. There is a broad range of techniques for this purpose, ranging from simple background subtraction, to color segmentation (as used in [7], for instance), to using probabilistic appearance models of the tracking targets (e.g., [9]). However, with increasing complexity, generality is however as the characteristics of tracking objects need to be known in advance (ranging from color-profiles up to probabilistic appearance models), while the computational cost is increased. Parameter-based segmentation gets increasingly difficult for heterogeneous swarms as in the LEURRE project, where the segmentation process would need to distinguish between cockroaches and Insbots. Also, we note that every conceivable parametrization is strongly dependent on environmental conditions (e.g., lighting conditions), rendering the calibration/adaptation process difficult.

For these reasons, and because it appeared to be sufficient for most tasks, only simple background subtraction is implemented in *SwisTrack*. Every frame of the video feed is subtracted from a background image which is either supplied by the user or can be estimated and updated online. Using a user supplied threshold value (only one color plane is considered for background subtraction, typically the brightness channel in a CMYK video), objects having a sufficiently different color from the background can be distinguished. For scenarios where lighting conditions are unstable (for instance when the arena cannot be completely shielded), or the background is changing, *SwisTrack* can estimate the background by calculating the running average, which serves as finite impulse response filter (FIR). This algorithm clearly reaches its limitations, when the objects move only little, which is the case for aggregating cockroaches. Then, targets merge with the background, and the tracking algorithm will most likely pick-up objects that are passing by, and lose the original, resting target.

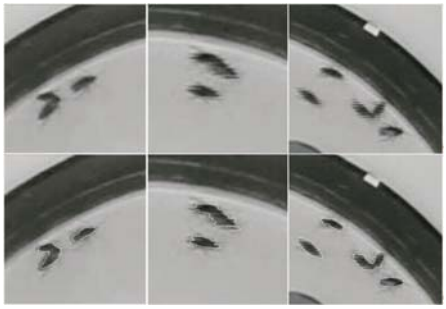

Fig. 3. Poor resolution and contrast, as well as overlapping objects complicate image segmentation and data-association. Video feed (top row), and contours after segmentation (bottom row). The number of cockroaches in the middle column is difficult to estimate even by the human eye.

A. Contour Retrieving

The background subtraction process leads to a binary image, highlighting regions that contain potential objects. Using standard methods (see the OpenCV reference manual for further information [14]), the contours of those connected components can be retrieved. If appropriate, contours can be filtered using different criteria (for instance minimal and maximal size, or morphological features). *SwisTrack* allows for specifying the minimal and maximal sizes of potential targets. However, this needs to be used with caution as the size of contours might vary drastically if objects are too close together or even overlapping. The process is illustrated in Figure 3 for cockroach tracking. Note the different sizes and shapes of the different cockroaches, as well as objects being overlapped or very close together, making data association solely based on the video feed impossible even for the human observer.

B. Accuracy and Repeatability

An object's position is calculated as the geometrical center of gravity of the object's contour, and thus leads to sub-pixel measurements. Due to the sub-pixel resolution and noise of the camera's CCD, objects seem to be moving, even if they are static with respect to the camera. For quantifying this effect, we recorded the position of two Alice robots [1] that were positioned off-center in an arena of 0.98m diameter. The mean

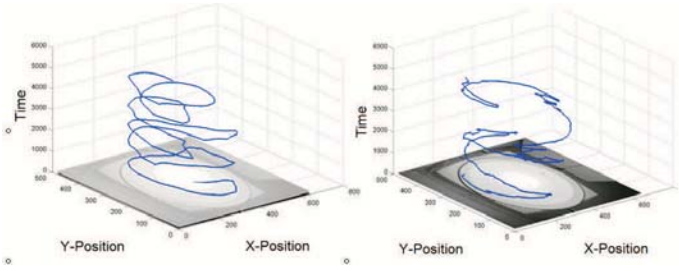

Fig. 4. Trajectory of a single target (left) vs. individual trajectory in a multi-target environment leading to data-association mismatch (right).

error calculates to 0.1777 pixels (standard deviation 0.1240 pixels).

IV. Data Association

We consider the data association step as the most difficult part of the tracking process. Having the information from image segmentation of consecutive frames, the problem is to associate potential targets (blobs) with a potential trajectory. Here, we outline a number of pitfalls that make data association a hard problem, and how they are tackled in *SwisTrack*. If two objects are for instance colliding, it is not clear afterwards which trajectory needs to be associated with which object (see for instance Figure 4). Such problems cannot be resolved with a simple nearest neighbor approach, and a threshold for the maximal allowed distance that an object can move per time-step ("distance gate" [16]). Note, that some of the scenarios detailed below do not apply when robots are equipped with markers, and none of them apply when robots are equipped with unique markers (geometry or color encoded). Also, note that data association becomes even more complicated if agents are allowed to leave or join the arena during tracking as in [7], [9], [11], which is not considered here. The most prominent problem in the literature ([16] and references therein) is to avoid non-optimal solutions, which arise from a greedy implementation of the nearest neighbor approach. Such a situation is depicted in Figure 5, a. Assuming *Object1* chooses first, selection of the nearest neighbor will lead to wrong association. This can be resolved by solving a quadratic assignment problem that would minimize the sum over the cost (distance) of all assignments [16]. This approach however does not help for solving the problem depicted in Figure 5, b. There, both trajectories pick the closest object, which minimizes the distance for both, but might lead to the (wrong) conclusion that both objects are performing a U-turn (see Figure 4, right, for an exemplar trajectory).

Data association becomes even harder when objects are missing in the scene due to noise, reflections, lighting conditions, etc. Such a situation is depicted in Figure 5, c. There, *Object1* is invisible for some reasons in one time-step, and thus trajectory one and two share *Object2* (which might well be if the two objects are close together). On re-appearance of *Object1*, *Object1* is considered too far away (due to the distance gate), and both objects stick to *Object2*.

Finally, when two objects merge (partial overlap or being

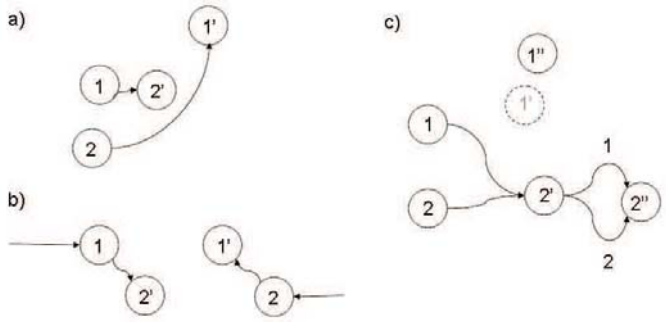

Fig. 5. Potential problems when associating trajectories with targets. Targets are indicated by circles, whereas trajectories are given by arrows. The apostrophe indicates frame indices (time-steps). See main text for comments.

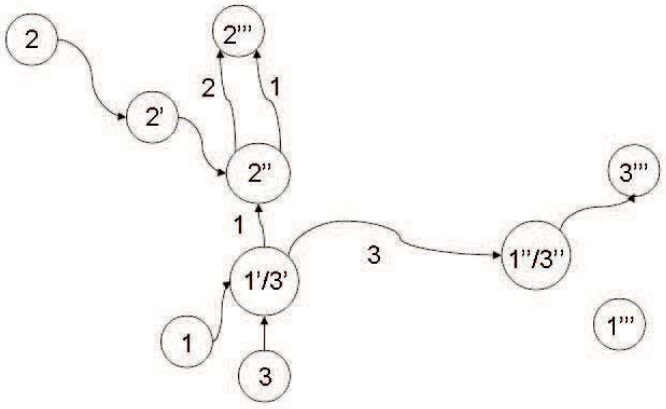

Fig. 6. Shared trajectories can split at the wrong time, leading to data-association mismatch.

too close together) as trajectory one and three do in Figure 6, it might happen that those trajectories split prematurely due to another object (here *Object2*) passing by. In this case, trajectory one might go with contour 2, which represents in fact only a single target, while trajectory 3 continues tracking *Object1* and *Object3*. Upon physical split of *Object1* and *Object3*, *Object1* remains un-associated. In *SwisTrack*, the problems outlined above are resolved as follows. We assume the number of objects to be known and not changing during the entire process (which is reasonable for swarm robotics and insect experiments for which *SwisTrack* was designed). Every trajectory chooses the contour closest to it, even if this contour is already associated with another trajectory. In a second step, every un-associated contour chooses the closest trajectory that is in competition with another trajectory, i.e. shares a contour, and is within range. By this, a maximum number of trajectories and contours will be associated. In order to resolve the problem depicted in Figure 6, *SwisTrack* continuously checks for "free" contours (as 1''' in this example) that fulfill certain criteria that are atypical for noise, such as moving in one direction for more than one second, and associates them with the closest trajectory that shares a contour. Additionally, *SwisTrack* allows the user to manually intervene at any time for re-associating

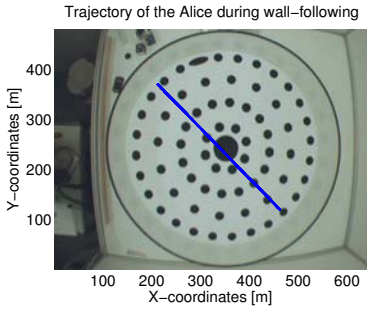

Fig. 7. Experimental setup showing the circular arena and the calibration pattern. The trajectory along the stick (not shown), is superimposed.

trajectories using the GUI.

V. Calibration

SwisTrack comes with a built-in calibration routine, which requires a 2D dimensional calibration target of known dimension. The pattern can be either rectangular or round with an arbitrary number of points. A round pattern (Figure 7) which can be printed on A0 paper (96cm diameter) is distributed with *SwisTrack*, and is an exemplar in this paper for giving an estimate of the accuracy that can be achieved using simple off-the-shelf hardware and *SwisTrack* without modification.

A. Camera Model

Most camera calibration routines were developed for 3D calibration, and are numerically unstable if the calibration target is coplanar with the image plane. Therefore, a 3D object needs to be presented under different angles of view. For our applications however, where objects move in a 2D arena that is usually (approximately) co-planar with an overhead camera, simpler models are applicable. Assuming the image coordinates of N points on a calibration target being u_i, and v_i, and the real world coordinates are given as x_i, and y_i. A suitable representation is then for instance

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} u^2 \\ v^2 \\ u \\ v \\ uv \end{pmatrix}^T \begin{pmatrix} a_1 & a_6 & 0 \\ a_2 & a_7 & 0 \\ a_3 & a_8 & 0 \\ a_4 & a_9 & 0 \\ a_5 & a_{10} & 0 \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & -\cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} dx \\ dy \\ 1 \end{pmatrix} \tag{1}$$

where $a_1 ... a_{10}$ are parameters that can be estimated by a least-squares fit of a suitable number of world-image coordinate pairs (at least 10), and dx, dy, and α are the parameters for simple linear translation and rotation respectively.

SwisTrack is able to recognize a presented calibration pattern, and hence calculates the transformation matrix by solving a least-squares problem.

For testing the performance of the system, we designed a simple experiment. An Alice robot [1] was programmed in wall-following mode in an arena of approximately 98cm diameter, which was cut in half by a straight stick (Figure 7). The scenario was recorded by a Unibrain Fire-I 400 camera (industrial case, resolution of 640x480, frame rate of 30Hz for monochrome images), with a focal length of 2.5cm, mounted approximately 1m above the arena. The experiment lasted 1h, which allowed the Alice to follow the outline of the stick 45 times on its entire length. We first calculated the static calibration error by marking 9 equidistant points

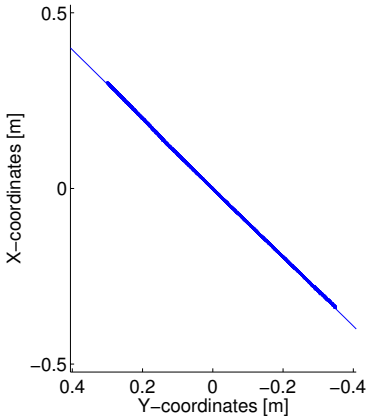

Fig. 8. Mean trajectory along the stick in world coordinates (after calibration)

(11.5cm) along a line cutting the arena into half (similar to the stick), and recording 750 measurements (50s at 15Hz). Hereby, the points were placed in between the calibration points, so that the calibration error can be assumed to be maximal. The average measurement error due to camera noise calculates to 0.17mm (standard deviation 0.03mm), whereas the average absolute calibration error calculates to 2.3mm (standard deviation 1.4mm), measured from the centre of each point. For testing the dynamic calibration error, we discarded all data points, where the Alice was following the arena wall, as the arena is not exactly circular ($\pm 1cm$), and therefore not suitable as reference. The effect discussed in Section III-B (camera noise) seems now to be more prominent for moving objects (mean error and standard deviation are roughly larger by a factor of two than in the static case), which is due to the fact that PD-controller used for wall-following might oscillate slightly (sensor noise and transient behavior of the controller). The data points were transformed in world coordinates using (1), and are depicted in Figure 8. The data was fitted by a straight line parameterized as $y = mx + c$, which is superimposed in the Figure.

We now calculated the distance from each calibrated trajectory point to the average line, which we consider to be the calibration error. The mean error along the straight part of the Alice trajectory is plotted in Figure 9, and ranges from 0.5mm to 2.5mm (standard deviation around 0.5mm). The actual calibration error (Figure 9) increases for objects farther than around 20cm from the center. This is an artifact of the chosen calibration pattern as well as due to the fact that the calibration points are in a plane that is around 2cm lower than the Alice. With increasing distance from the camera axis that is orthogonal to the arena, the perspective changes and the geometrical center of gravity of the Alice contour drifts towards the arena center, yielding a slightly higher error for these regions.

VI. Case Studies

Both case studies are using the same camera (Unibrain Fire-I 400, see Section V-A), however using a focal length of

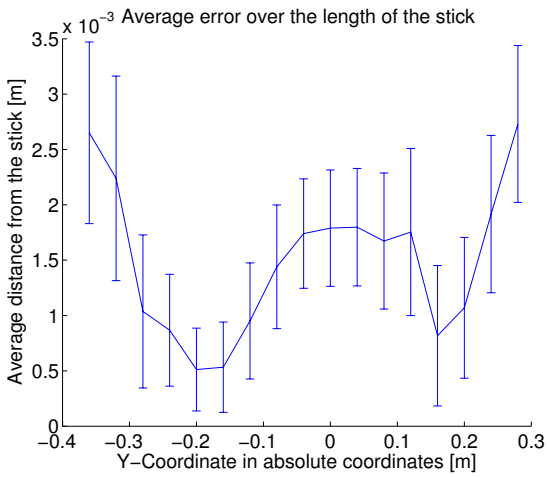

Fig. 9. Average calibration error over the length of the straight part of the Alice trajectory.

Fig. 10. Experimental setup for the inspection task.

Fig. 11. Experimental setup as seen by the camera (640x480, $f = 4$mm).

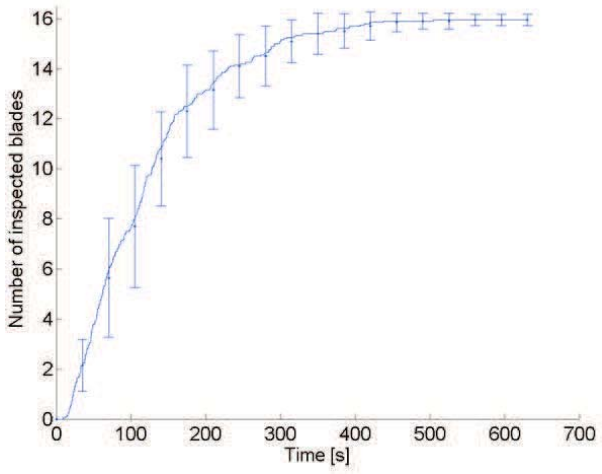

Fig. 12. Mean inspection progress over 20 experiments with 20 miniature robots inspecting a scenario with 16 objects.

$f = 4$mm. Increasing the focal length decreases the field of view, whereas the distortion will be reduced, and hence the calibration error decreased.

We notice that both case studies presented here do not make use of the calibration routine as the applied metrics (area coverage and presence of individuals in certain areas) can be applied in image space, which is best practice whenever possible, as uncertainties introduced by the calibration process can be avoided.

A. Swarm Robotic Inspection

The swarm robotic inspection scenario has been presented in [2]. The aim of this case study is to contribute to the development of a general methodology for designing and modeling swarm robotics systems. In [2], a swarm of up to 20 miniature robots inspects blades in an abstracted model of a jet turbine engine (see Figure 10 for the experimental setup). As the case study is only concerned with the analysis of the interactions within the swarm and its general motion, inspection is assumed to be successful after every blade has been circumnavigated at least once by at least one robot.

The camera has been adjusted such that the CCD fits the rectangular arena as well as possible (Figure 11), leading to an approximate rate of 480 pixels / 1m. By this the Alice robots in use were represented by approximately 10x10 pixel squares. The maximum speed of the robots is around 4cm/s, yielding around 1.33 mm displacement per frame at a frame rate of 30Hz, corresponding to roughly 0.5 pixels/frame. *SwisTrack* was used to record the robots' trajectories that were analyzed for determining task completion. In Figure 12, we summarize the mean inspection progress over 20 experiments involving 20 Alice robots each. The mean duration of an experiment was around 250s.

B. Tracking of Insect Societies

The experimental setup (see [17] for a similar scenario) is depicted in Figure 13 and shows an arena of 0.98m diameter. The arena is populated with a swarm of cockroaches. Within the arena, two shelters of 10cm diameter are hovering at a height of 5cm, being supported from invisible wires. Cockroaches have a size of up to 4cm (1cm width), which leads to approximately 25×5 pixel squares for the area in image space. Unlike a miniature robot, cockroaches can reach speeds up to 1m/s, leading to displacements of up to 16 pixels per frame. An exemplar view through the camera for an experiment involving 10 cockroaches and 2 shelters is provided in Figure 14. The shelters are translucent red, and thus allow the cockroaches still to be visible (although at reduced contrast) for the experimenter and the camera, while

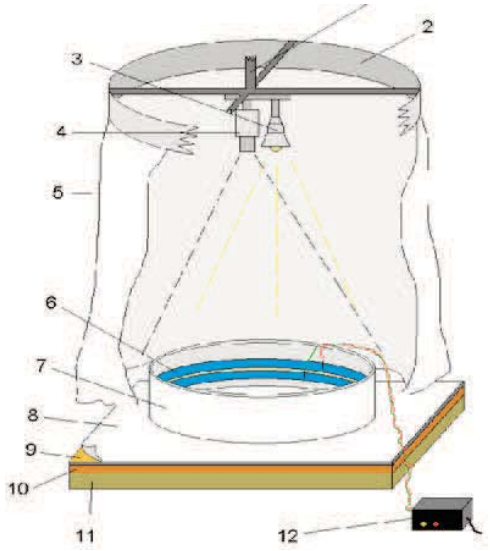

Fig. 13. Experimental setup (drawing S. Portha).

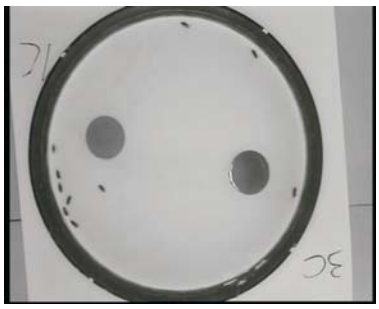

Fig. 14. Experimental setup as seen by the camera (640x480, $f = 4$mm). 10 cockroaches and 2 shelters.

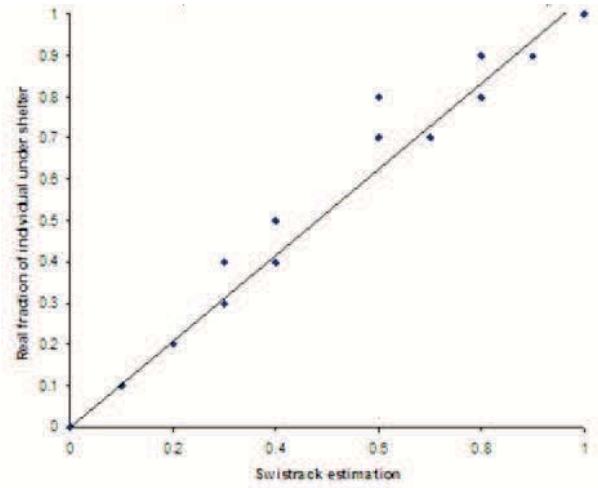

Fig. 15. Estimate of the fraction of individuals (x-axis) under shelter vs. manual counting (y-axis). Experiment duration 180min, 10 individuals.

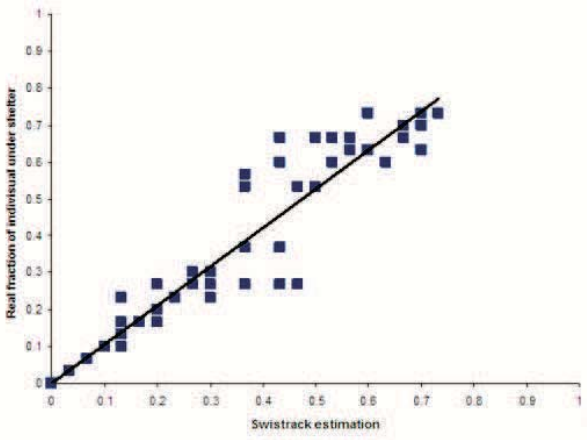

Fig. 16. Estimate of individuals under shelter vs. manual counting (experiment duration 180min, 30 individuals).

providing a "dark" resting spot to the cockroaches that are less sensitive to red light. *SwisTrack* is used in this experiment for counting the evolution of the number of individuals under each shelter. In *SwisTrack*, an arbitrary number of regions of interest (for instance the shelters) can be specified using a binary bitmap (BMP format), and the number of objects within this area is displayed and stored to disc (see also Figure 2). The experimental error introduced by the vision system for an experiment of 180min length involving 10 cockroaches and 2 shelters has been recorded in Figure 15, by comparing the numbers provided by the software with a manual count that was performed every 5 minutes during 180 minutes total (72 data points, numbers are normalized with the total number of objects). Results for the same experiment, but involving a swarm of 30 individuals, are depicted in Figure 16. In both Figures, the Y-axis represents the ratio of individuals under a shelter to the whole swarm counted by a human experimenter and serves as ground-truth data (notice however that manual counting might also show a certain error). The X-axis represents the *SwisTrack* estimate for the same measure. The observed error is mainly due to data association problems, which become more difficult for occluded as well as overlapped targets, conditions which arise under the shelter, which makes the detection of targets more cumbersome due to reduced contrast. Note that the error seems to be maximal when roughly half of the cockroaches are under the shelter. This can be explained by the fact that the activity under the shelter is maximal for these numbers, as aggregates of higher density lead to reduced activity of the animals (the probability to move is decreasing with increasing number of cockroaches under the shelter [17]), whereas small numbers of individuals are less likely to yield data association problems.

VII. Discussion

Whereas quantifiying the tracking error is difficult for the inspection case study where ground truth data is difficult to obtain, we identify common sources of errors in multi-target tracking in the social insect case study. As described above, uncertainties are already introduced in the segmentation routine, which gets increasingly difficult for higher density of individuals in the environment as this increases the likelihood

for collisions and overlap of objects' contours. Errors during segmentation are indeed responsible for the increasing relative error when tracking 30 individuals (Figure 16) as opposed to 10 individuals (Figure 15), as the density of individuals under a shelter is much lower then. We conjecture that increasing the camera resolution might help to increase the accuracy of segmentation, however at cost of an increasing computational time and a more expansive hardware setup.

Uncertainties arising during the segmentation process need to be resolved by the data association algorithm, which becomes increasingly difficult with increasing speed of the agents (see Section IV for an example) or decreasing framerate. This effect has however not been observed in the specific experiments conducted in this paper. In the inspection case study, the ratio of robot speed (4cm/s) and frame rate (30Hz) is sufficiently low, whereas in the social insect case study, cockroaches tend to move at a similar low speed when under or close to the shelter.

Generally speaking, segmentation and data-association can be improved, but only at cost of reduced generality and additional computation. For instance, it could make sense to use a kinematic model of the object to track. In case of the cockroach experiments, this will translate into taking into account the angle of preferential movement instead of just the simple max-speed criterion. Such a specific model would however need adaptation to a robotic scenario, and vice versa. We note that the data association problem will be relaxed as soon as the robots' contours can no longer merge. This can be achieved for instance by using "hats" equipped with high-contrast markers. This will work, if the robot's speed is slow with respect to the cameras frame rate, so that two robots cannot swap positions within one frame, and if the arena is never occluded due to non-transparent shelters or the experimenter's hand for instance.

In further work, we are interested in better understanding and improving the reliability of the employed data association algorithms, which have so far been only developed empirically. Also, we are interested in making the background subtraction method more robust to changing lighting conditions, which should be feasible without loss of generality.

Finally, we notice that implementation of improved algorithms (image segementation or data association) is straightforward in the framework of *SwisTrack*, either by exploiting the object oriented architecture, which facilitates the replacement of singular functional blocks, or by using the TCP/IP interface together with a high-level programming language.

VIII. Conclusion

Using two case studies, one chosen from swarm robotics research, and the other from behavioral biology, we have outlined hard problems that arise in tracking multiple, mobile, marker-less, miniature objects. We also show that *Swistrack* is an essential tool for studying individual characteristics and capabilities of agents in a swarm. This is the baseline for understanding the transition from individual to collective capabilities, and will eventually be a key tool for optimizing the design of the individuals according to the expected emergent properties at the collective level.

We also showed how to achieve useful tracking results using off-the-shelf components, and considerably simple algorithms, wrapped up in a ready-to-use software package, which enables future research with considerably low overhead.

Acknowledgments

This project was partially sponsored by the European Project Leurre, sponsored by the Future and Emerging Technologies program of the European Community (IST-2001-35506). N. Correll and A. Martinoli are also sponsored by the Swiss National Science Foundation (contract Nr. PP002-68647). The authors would like to thank Christopher Cianci, Julien Nembrini, and Pierre Roduit for their contributions to the *SwisTrack* core.

References

[1] G. Caprari and R. Siegwart, "Mobile micro-robots ready to use: Alice," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005, pp. 3295–3300.

[2] N. Correll and A. Martinoli, "Collective inspection of regular structures using a swarm of miniature robots," in *Proc. of the Int. Symp. on Experimental Robotics 2004, Springer Tracts in Advanced Robotics*, vol. 21, 2006, pp. 375–385.

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

[4] (2006) European project leurre. [Online]. Available: http://leurre.ulb.ac.be

[5] G. Caprari, A. Colot, R. Siegwart, J. Halloy, and J.-L. Deneubourg, "Building mixed societies of animals and robots," *IEEE Robotics & Automation Magazine*, vol. 12, no. 2, pp. 58–65, 2005.

[6] V. Trianni and M. Dorigo, "Self-organisation and communication in groups of simulated and physical robots," *Biological Cybernetics*, 2006, in press.

[7] T. Balch, Z. Khan, and M. Veloso, "Automatically tracking and analyzing the behavior of live insect colonies," in *Proc. of the 5th Int. Conf. on Autonomous Agents*, 2001, pp. 521–528.

[8] D. Reynolds and J. Riley, "Remote-sensing, telemetric and computer-based technologies for investigating insect movement: a survey of existing and potential techniques," *Computers and Electronics in Agriculture*, vol. 35, pp. 271–307, 2002.

[9] Z. Khan, T. Balch, and F. Dellaert, "Mcmc-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[10] L. Noldus, A. Spink, and R. Tegelenbosch, "Computerised video tracking, movement analysis and behaviour recognition in insects," *Computers and Electronics in Agriculture*, vol. 35, pp. 201–227, 2002.

[11] K. Teknomo, Y. Takeyama, and H. Inamura, "Frame-based tracking of multiple objects," in *IEEE Workshop on Multi-object Tracking*, 2001, pp. 11–18.

[12] A. Bulpitt, R. Boyle, and J. Forbes, "Monitoring behavior of individuals in crowded scenes," in *Proc. of Measuring Behavior, 3rd Int. Conf. on Methods and Techniques in Behavioral Research*, 2000, pp. 28–30.

[13] J. Smart, K. Hock, and S. Csomor, *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall, 2005.

[14] (2006) Opencv library. [Online]. Available: http://sourceforge.net/projects/opencvlibrary/"

[15] (2006) 1394 camera driver. [Online]. Available: http://www.cs.cmu.edu/ iwan/1394/

[16] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

[17] J.-M. Amé, J. Halloy, C. Rivault, C. Detrain, and J. L. Deneubourg, "Collegial decision making based on social amplification leads to optimal group formation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 15, pp. 5835–5840, 2006.