16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

A TRACKING FRAMEWORK FOR LABORATORY EXPERIMENTS WITH MULTIPLE TARGETS AND OCCLUSION

Rafael H. C. de Souza, and Neucimar J. Leite

Institute of Computing, P.O.Box 6174
University of Campinas - UNICAMP
13084-971, Campinas, SP, Brazil
{rafaelh.souza@students., neucimar@}ic.unicamp.b

ABSTRACT

The automatic tracking of components allows a quick and consistent analysis of parameters related to biological experiments. The register of parameters is done by a system that analyzes a sequence of images and computes a number of descriptors characterizing the behavior of each target. Our objective is to create a framework for tracking in biological experiments, with multiple targets that may suffer occlusion. Such framework must be capable of being adapted to other classes of tracking. Here we present the framework, a new method of cells segmentation, and an optimization method for occlusion solving.

1. INTRODUCTION

Animals are used in a great variety of experiments related to medical research, learning process modeling and even procedures whose application in humans would be ethically prohibited [7]. In these experiments, parameters like trajectory, distance, average speed, or movement pattern must be registred. Generally, the parameters are computed by a manual or semi-manual method. In such cases, a tracking system would have the advantages of being more consistent and quick in the analysis of the obtained parameters. Besides, a tracking system is much more efficient than a human in the collection of many parameters, such as speed, distance, and trajectory of the targets. A tracking system for biology experiments must be capable of computing a number of descriptors of movement and calculating the parameters of each kind of experiment. Tracking systems are specially suited for three kinds of experiments: the register of rare behaviors (that may happen with an interval of hours), the register of behaviors that long for many hours (like the night behavior of an animal) and spatial parameters, like speed and distance.

The problem of tracking a single target in a video is relatively simple. A much more difficult problem is the tracking of multiple targets, since there may be interactions between them, leading to occlusions which happen, as defined in this project, when two or more targets get too close so the tracking algorithms can not distinguish them. It can generate inconsistency, leading to a incorrect tracking process. So, a tracking system must be able to tracker multple targets at the same time while beeing able to avoid or correct the inconsistency caused by the occlusions.

In this work, we will develop a tracking framework for multiple targets with occlusion. We will present some algorithms applied to biology experiment which are adaptable to other classes of tracking. We also introduce a simple segmentation method for cells, and an optimization approach for the occlusion solving.

2. RELATED WORK

Few articles deal with the problem of tracking animals in biology experiments. For example, the work in [2] describes a method very simple method for collecting parameters concerned with the Morris water maze (*MWM*) [10]. Basically, it searches for the head of the rat, modeled as a 3×3 matrix.

In [1], a system for tracking of multiple ants in laboratorial experiments is presented. Its algorithms, however, have segmentation problems due to their background subtraction algorithm. When an ant stops moving, for example, it is considered background. Besides, this work does not deal with the occlusion problems.

In [5, 6] an algorithm for tracking of multiple rats with occlusion is presented. This approach, however, supposes that the unique kind of occlusion is by contact, when two rats are too close. It does not consider the case in which the rats are one over another. Besides, methods for parameter register are not defined.

As an example of commercial system we have the *Ethovision*. According to [9], it is a generic system that can be applied for many different types of laboratorial experiments. However, it collects only few parameters concerning to each different experiment. Furthermore, the tracking method consists in a simple background subtraction with manual thresholdings as segmentation method. This kind o segmentation can be a problem in some tracking cases, since the user is not allowed to modify the system due to its commercial nature. Another commercial system that worths mentioning is the *Water 2020* [4] which was designed for the *MWM* experimental tests only.

3. MODEL

In this section we present our framework model, its main definitions and assumptions. Section 3.1 introduces the classes of tracking tackled by this framework and the occlusion definition. The system modules are discussed in section 3.2.

3.1 Tracking and Occlusion Definition

This project started as the definition of a method for tracking animals in the MWM [8]. In this experiment, the video represents a sequence of images containing the targets, filmed from the top, which should be tracked from a *2D* plane.

A target is an object that may either move or not to be detected by some tracking algorithm. A tracker object is an instance of a tracking algorithm assigned to a target. At each frame, each target must be tracked by one tracker object. Due to the occlusion problem, to be discussed next, two trackers may change their targets. However, all targets must be tracked by at least a tracker in all frames – no target can be lost. The trackers may change their targets only during an occlusion event. Finally, the targets do not disappear from the image limits, and no target appears after the first frame of the video experiment. Given an image and the position of each target, two targets are in occlusion if they can not be separated, given some criteria. For example, we can use the fact that two targets are in occlusion if their geometrical centers are too close or if, in case a segmentation procedure is used, their segmented regions merge.

3.2 System modules

The system is composed by the following modules (figure 1):

- *Video manager module*: It is responsible for reading a video, obtaining its frames, and loading these frames into the system data structures.

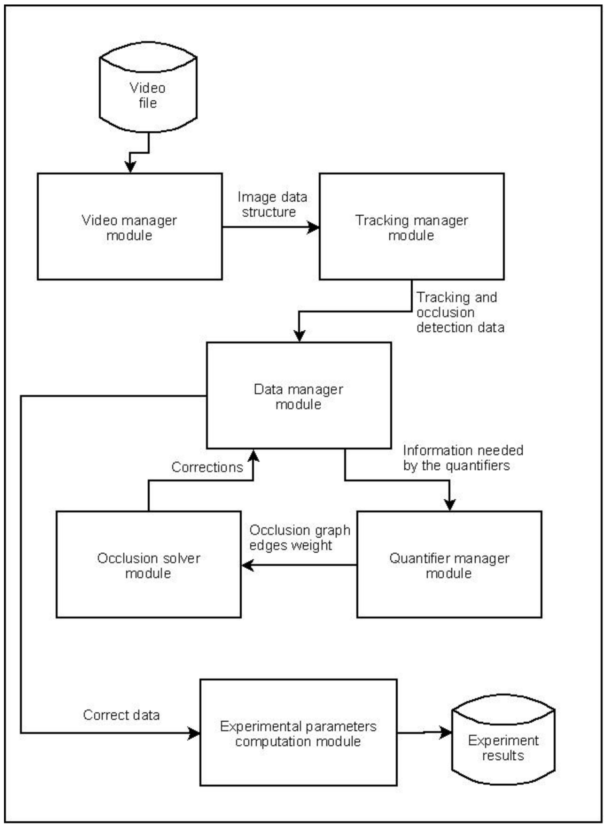
Video file
Video manager module
Image data structure
Tracking manager module
Tracking and occlusion detection data
Data manager module
Information needed by the quantifiers
Corrections
Occlusion solver module
Occlusion graph edges weight
Quantifier manager module
Correct data
Experimental parameters computation module
Experiment results

Figure 1: System diagram.

- *Tracker manager module*: It carries out the tracking of the targets and detects possible occlusions. This module is not yet responsible for the occlusion solving.
- *Data manager module*: It stores all data generated by the tracker module. It also makes available methods for manipulating these data.
- *Occlusion solver module*: It solves the occlusion problems, correcting the errors of the data stored in the data manager module. Section 4.2.1 presents the heuristic for facing these problems.
- *Quantifier manager module* : It manages a set of algorithms used to help the occlusion solver module and combines its results.
- *Parameters computation module*: It generates a set of parameters of interest for the experiment being conducted.

3.2.1 The occlusion graph

The occlusion graph G is a structure used in many steps of the occlusion solving process. It is a directed graph, subdivided into different layers. Let V_i be the set of vertices in the layer l_i. In the first layer, there is one vertex v_i for each target $t \in T = \{t_1, t_2, ..., t_k\}$. The layer l_i has one vertex for each group of targets in occlusion in the frame f_i (a non occluded target t belongs to an occlusion group whose only member is t). The edge (u, v), $u \in V_i$ and $v \in V_{i+1}$, indicates that any target represented by u in frame f_i is in the group represented by v in the frame f_{i+i}. So, the edges represent the merging and splitting information of the groups. Thus, two edges e_i, e_j, such that $e_i = (u, v)$ and $e_j = (w, v)$, indicate a group merging. In the same way, two edges e_i, e_j, such that $e_i = (u, v)$ and $e_j = (u, w)$, indicate a group splitting. A tracking of a target t_i is defined as a path p_i from the initial layer l_0 until the last layer l_n. An occlusion solution is defined as a set of paths $P = \{p_1, \ldots, p_k\}$ respecting the vertex capacity restriction. Each vertex of V has an associated capacity. This capacity is the number of targets in the occlusion group represented by the vertex. If a vertex $v \in V$ represents k targets, no more than k paths may pass through this vertex, otherwise v would represent more than k targets, yielding a contradiction (since each path represents a target motion).

Each path has a weight defined by the weight of his edges. A weight of an edge $e \in p$ depends on the subpath $p_s \in p$ used to reach e. The quantifier manage module (section 3.2.3) is responsible for the calculation of this weight.

The figure 2 shows an example of the steps in the generation of an occlusion graph. The first six frames show the tracking of four targets. In the following six frames, each vertex represents a target, and each edge represents an occlusion. The final frame of figure 2 shows the generated occlusion graph.

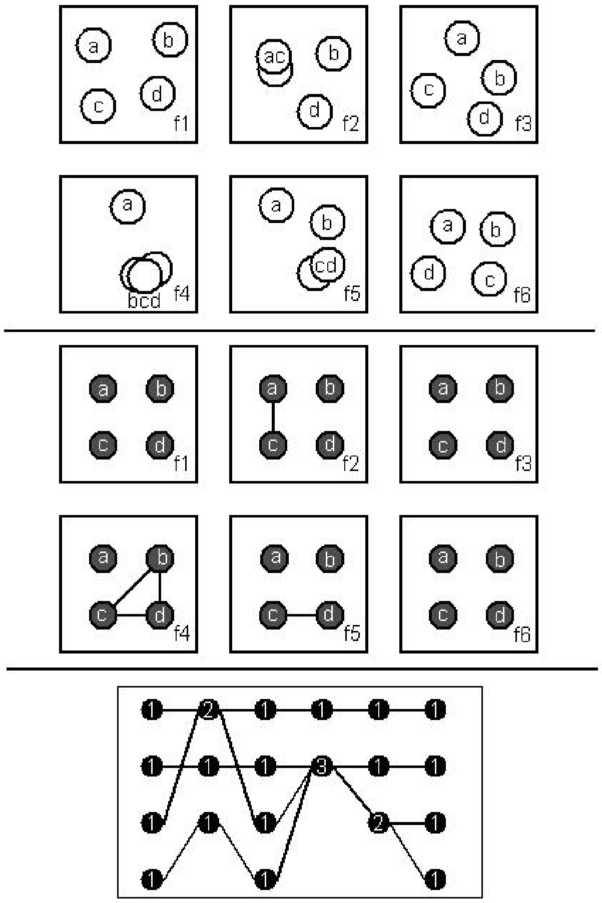

Figure 2: Steps in the creation of an occlusion graph.

3.2.2 Occlusion solver

Given the occlusion graph G, described in the previous section, the occlusion solver module calculates the set of paths P constituting a tracking solution whose weight is maximal. Note, that this problem is *NP-Hard*. Since the occlusion graph is generally quite large, this module considers a heuristic approach to obtain a good final solution. This problem regards finding a set of paths in a graph whose edges have a variable weight. As shown in [3], the *Ant Colony* is the state of the art method for dealing with this kind of problem. Thus, the occlusion solver module in figure 1 implements a heuristic based on the Ant Colony algorithm. Section 4.2.1 defines the implementation of such a heuristic.

3.2.3 Quantifier manager module

The edges of the occlusion graph have an associated weight function. The weight $w(e|p_i)$ of the edge $e = (u_k, v_w)$, $u \in l_j$,

$v \in l_{j+1}$, given a path p_i from the first layer to the vertex u_k, represents the probability of the target t_i being in the group r_k, in the frame f_j, going to the group r_w, in the frame f_{j+1}. The algorithm responsible for this weight calculation is a *quantifier algorithm*.

A quantifier algorithm may calculate the weights using a variety of different information. For instance, it may use the information of the previous trajectory of the target in positions represented by the path p_i, its color histogram, and its shape. As mentioned in section 3.1, these information must be capable of distinguishing all the targets. The implemented quantifier algorithms are presented in the section 4.2.2. The quantifier manager module has a set of these quantifier algorithms. Besides, it must implement a method for combining the results of different quantifiers in order to calculate more accurately the corresponding weights (section 4.2.2).

4. ALGORITHMS

In this sections we present the main algorithms of the framework in figure 1. A new segmentation method is introduced in section 4.1. Occlusion related algorithms are presented in section 4.2.

4.1 Watershed basins threshold

Determining a global threshold value may be impossible in many image processing applications. If we consider, for example, the case of tracking wandering cells (section 5.1), it is impossible to determine a global threshold value to be used as a segmentation threshold due to the nature of the images histograms. In such a case, only the biggest cells (which have the darkest gray values) are segmented while many others can be defined as background. For our cells video, a local threshold method is applied. First, the images are filtered with a morphological closing operation in order to remove small irrelevant minima. Then, the watershed lines of all remaining local minima are calculated, and the corresponding image is subdivided into catchment basins. Let each basin be a region. A statistical threshold algorithm is applied, yielding a different threshold value for each one of these regions. Figure 3 shows the watershed lines of an original image, the watershed basins thresholding approach and, for the sake of comparison, the result of the segmentation by considering a global threshold.

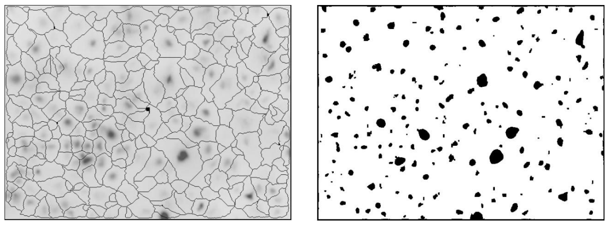

Figure 3: Original image and its watershed lines, watershed-based thresholding, and a global thresholding of a group of cells.

4.2 The occlusion related algorithms

In this section we describe the occlusion solving related algorithms. Section 4.2.1 presents a new heuristic for solving our occlusion problems based on the *Ant Colony* metaheuristic. Section 4.2.2 presents the *quantifiers*, algorithms used to solve the occlusion problem.

4.2.1 Occlusion solving by an Ant Colony heuristic

As discussed in section 3.2.1, the best solution for the occlusion problem is a maximum weight set of paths from the first to the last layer of the occlusion graph, respecting the flow capacity on the set of vertices. Though it is not the scope of this paper, it is worth mentioning that finding this set of paths is a *NP-Hard* problem and, in such a case, we should consider a heuristic-based method as a good solution to the problem.

In cases occlusion problems, a solution can be defined by a set of paths in a graph whose edges weight may vary along the graph. This specific characteristic suits the *Ant Colony* metaheuristic application [3], which was chosen here to approach the considered problem. Indeed, this problem has a very simple formulation as an *Ant Colony* heuristic and, as mentioned before, it is the state of the art method to solve problems in graphs where the edges weight changes in time. The basics of an *Ant Colony* heuristic is described in [3]. In order to use such an heuristic, we must define: how ants build their path in the graph; how to compute the probability of an edge being chosen; and how to update the pheromone of the edges. For this purpose, each ant has the following informations: an id i; a path p_i, containing all the edges already traversed by the ant; and a path $p^o pt_i$, the best path found in all iterations by the ant.

The graph used by the heuristic is identical to the occlusion graph except for the pheromone values in the edges and the best set of paths found by the ants in all iterations. Let $F = (f_1, f_2, \ldots f_{n_t})$ be the set of ants, where n_t is the number of targets being tracked. The ants from F build their path, ordered by the value of their best path found (the ant with the best path builds first). An ant f_i builds his path edge by edge. The edges are chosen randomly with different probabilities between them. Let f_i be in the vertex u, the probability of the edge e_{uv} being chosen is defined by:

$$p(e_{uv}|path_i) = \frac{\tau_{uvi}^{\alpha} \cdot \eta\left(e_{uv}\right)^{\beta}}{\Sigma_{w \in Ngb(u)} \tau_{uwi}^{\alpha} \cdot \eta\left(e_{uw}\right)^{\beta}} \tag{1}$$

Where τ_{uvi} is the pheromone value of the edge e_{uv} associated to the ant i (the pheromone of an ant influences itself only), $\eta\left(uv\right)$ is a function that assigns a heuristic value to the edge e_{uv}. The values α and β are parameters (both set to 2.0 in this project), $Ngb\left(u\right)$ is the set of neighbor vertices of u. In this problem, $\eta\left(e_{uv}\right)$ is zero if the vertex v already has its flow capacity used by other paths or if there is no valid path (w.r.t the flow capacity) from v to some vertex from the last layer of the graph, otherwise, $\eta\left(uv\right) = wgt\left(e_{uv}|path_i\right)$. Finally, we must define the pheromone update method. The pheromone update used in our implementation is based on a variation of the classical *Ant Colony* metaheuristic called *Ant Colony System*. Here, we have two types of update: *local update* and *offline update*. The *local update* is performed by all ants each time one of them traverses an edge. The *offline update* is performed one time per iteration only in the edges in the set of best paths found during the last iteration. The *local update* is defined as:

$$\tau_{uvi} = (1-\varphi) \cdot \tau_{uvi} + \varphi \cdot \tau_0, \tag{2}$$

where the parameters φ and τ_0 are, respectively, the local rate of pheromone evaporation and the initial value of the pheromones. The *offline update* is defined as:

$$\begin{aligned} \tau_{uvi} &= (1-\varphi) \cdot \Delta\tau_{uvi} + \varphi \cdot \tau_0, \\ &\quad \text{if } e_{u,v} \in p_i \\ &= \tau_{uvi}, \\ &\quad \text{otherwise ,} \end{aligned}$$

where ρ is the offline rate of pheromone evaporation and $\Delta\tau_(uvi)$ is a heuristic value indicating how good is the path found by the ant f_i. In our implementation, $\Delta\tau_(uvi)$ is the weight of the path p_i. The normal approach for this problem would be the use of a greedy heuristic. In this case, a greedy heuristic would have the advantage of being far quicker than the *Ant Colony*. However, the *Ant Colony* heuristic has a key advantage: it avoids the locally optimal solution and tends to find the global optimal one. This algorithm, as defined here, is performed *off-line*, i.e., performed after the targets were tracked in all video frames. However, the algorithm may be easily adapted to work *on-line*. In order to execute quicker, the occlusion graph may be constructed layer by layer, and the *ant colony* heuristic used to find partial solutions which, in turn, may finally be considered to solve the occlusion problem.

4.2.2 *Quantifiers*

The quantifiers have the most important role in the occlusion solving. They compute the value $w_k(e|p)$,i.e., the probability of the target k which traversed the path p being in the occlusion group represented by u. For this purpose, the quantifiers use all the information gathered by the tracker manager module.

Each type of quantifier considers a specific kind of information to treat the corresponding occlusions. Sometimes, however, one kind of information alone is not enough to distinguish the targets. For example, if two of them have nearly the same size, an area quantifier probably will not be able to tell them apart. So, in this project, we define a *quantifier combination* which constitutes also a quantifier. In this sense, we define a combined probability as:

$$p_k(e|p) = \Sigma_{i=1}^{m} \alpha_i \cdot wgt_{q_i}(e_{u_n v}|p) \tag{3}$$

Thus, the probability computed by our quantifier combinator is a linear combination of the results of the quantifiers q from the set Q. The calculation of the set $A = \{\alpha_1, \dots, \alpha_m\}$ is done by the *consensus rate* of the quantifiers. This *consensus rate* is based on the idea that if a quantifier gives equal probabilities to all the edges of a vertex, it does not add useful information for the solution of the occlusion event as when it gives a high probability to an specific edge. The *consensus rate* of a quantifier is defined below.

$$avg_i = \frac{1}{|Ngb\,(u_n)\,|} \Sigma_{w \in Ngb(u_n)} wgt_{q_i}(e_{u_n w}|p)$$

$$\alpha_i^2 = \frac{1}{|Ngb\,(u_n)\,|} \Sigma_{w \in Ngb(u_n)} \left[wgt_{q_i}(e_{u_n w}|p) - avg_i\right]^2 \tag{4}$$

So, the *consensus rate* of the quantifier q_i is given by the standard deviation of the probabilities of the candidate edges. The set A is normalized so that it sums to 1.0.

5. RESULTS

In this section we illustrate some of our experimental results. We have evaluated two aspects of the framework, namely, whether the implemented trackers respect the assumptions of section 3.1 and the correctness of the occlusion solution. Section 5.1 presents the results of the implemented trackers and section 5.2 shows some results of the occlusion solving heuristic.

5.1 Trackers results

In this section, we discuss the result of the tracking module through some example videos taken from real experiments. For the tests we used videos from 5 different experiments, with the following characteristics.

1. Open field: In this field, a rodent wanders in a pool. The video has 717 frames.
2. Morris water maze: A rodent swims in a pool filled with water. The video has 1947 frames.
3. 3 wandering ants: 3 ants coming from a hole wander in a restricted area. The video has 1598 frames.
4. 20 wandering: 20 ants wander in a restricted area. The video has 299 frames.
5. Wandering cells: A great number of cells wander in a neural tissue. 32 of them are tracked. The video has only 40 frames with a very low quality and sample rate.

Figures 4, 5, and 6 show some results of the video tracking. In all the above experiments, except for the last one, the targets were correctly tracked. In the last video, some cells could not be tracked due to the great amount of noise and the low sample rate of the frames. Nevertheless, the biggest cells were correctly tracked which shows that the tracker algorithms following the initial assumptions of the framework are feasible and really promising.

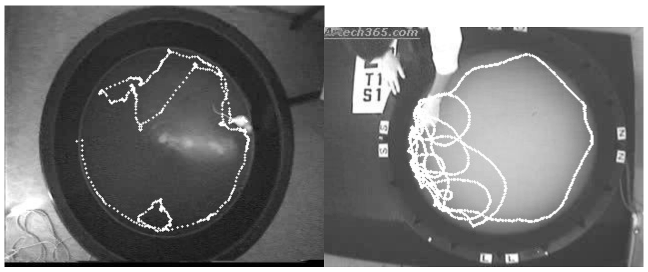

Figure 4: Tracking results for the open field and the Morris water maze experiments.

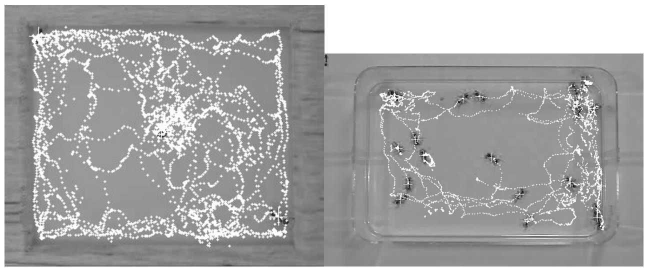
Figure 5: Tracking results for the experiments with ants.

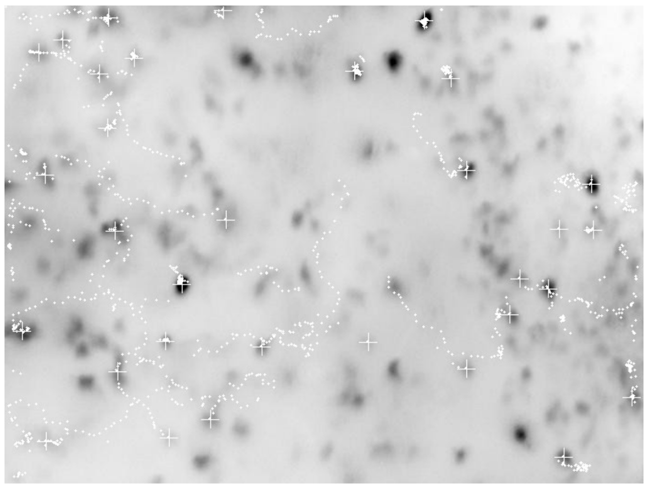
Figure 6: Tracking result for the wandering cells experiment.

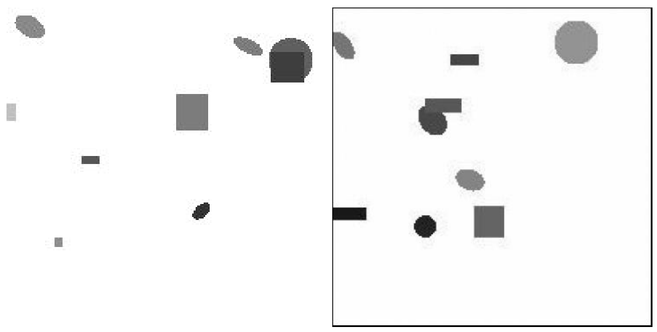

Figure 7: Two samples of the synthesized images.

Video	# of frames	# of targets	AC ERROR	GH ERROR
1	100	9	0%	30%
2	50	9	0%	0%
3	70	8	0%	16%
4	100	9	14%	28%
5	80	9	10%	20%
6	100	8	12.5%	50%

5.2 Occlusion solution

For the occlusion solving tests, the ground truth values for each target in each video would be necessary. However, such information is not available for none of the videos we are using in this project. Also, registering all positions manually would demand too much time. For example, for the multiple target videos used in this project, nearly $10,000$ positions should be registered manually. Besides, this great number of manual work could be error prone. In order to avoid this, we developed a test case synthesizer. This synthesizer generates a video of regions with different shapes, sizes, colors and motion patterns. All parameters (except the color, for the sake of an easy segmentation) are corrupted by noise. Figure 7 illustrates the generated video frames. In order to show that the *ant colony heuristic* is an efficient method that can be associated to occlusion solving, we generated 6 synthesized videos and compared the ant colony with a greedy heuristic. This greedy approach works exactly as the *ant colony*, except that only the edges having the greatest probabilities are chosen, not taking into account the pheromone values. The results are shown in table 5.2. There, the number of frames and number of targets of each video is listed. The number of occlusions not-solved by the *ant colony* heuristic is shown in the column *AC*. The number of occlusions not-solved by the greedy heuristic is shown in the column *GH*. As we can see from this table, the *ant colony* heuristic always found better occlusion solutions than the greedy heuristic. The reason is that the greedy heuristic finds only local optimal solution, while the *ant Colony* method does a better exploration of the solution space. In video 4, 5, and 6, the *ant colony* heuristic failed to solve one occlusion due to the calculation of the weight function of the quantifier module. This errors happened due to problems of finding a combination of quantifiers that generates a function whose maximum is the correct occlusion solving. This is a subject of future research.

6. CONCLUSION

In this work we proposed a framework for the tracking of components in video images used in laboratorial experiments. This framework intends to be general enough to be adaptable to a broad class of experiments. We discussed the implementation of some algorithms, including methods of segmentation and tracking of targets, related to the modules of our proposed system, and a novel approach for occlusion solving based on combinatorial optimization. The framework has been used in different kinds of biological experiments, thus showing his applicability for this class of problems.

As future work, the framework must be able to deal with targets appearing and disappearing, since many biology experiments (e.g. experiments with bacteria, which can get divided by mitosis) have this characteristic. We must develop new kinds of quantifier algorithms to deal with a greater number of experiments. Also, we must develop new methods for the quantifier combination, through machine learning methods, probably the most suitable for this kind of problem.

The presented framework does not execute iteratively, since all tracking must be performed first, and then the occlusions are solved. However, it is not hard to adapt this framework to execute the tracking and occlusion solution iteratively and in real time. The tracker module already works in real time, even with multiple targets being tracked. The occlusion graph could be constructed layer by layer, with partial occlusion solutions being calculated in each iteration. The Ant Colony Metaheuristic has a very useful characteristic in this case: if a good solution is avaliable for a graph and the graph has some modifications (in this case, the addition of one new layer), a new good solution can be calculated using few iterations. So, research can be done for a real-time framework.

7. ACKNOWLEDGEMENTS

The authors wish to acknowledge support from MCT/CNPq and FAPESP of Brazil under grants 472402/2007-2 and 07/52015 - 05/03534-0, respectively. We also would like to thanks professor Francesco Langone, from the Institute of Biology - UNICAMP, for providing the test videos.

REFERENCES

[1] T. Balch, Z. Khan, and M. Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 521–528, New York, NY, USA, 2001. ACM Press.

[2] S. F. Barrett. An improved morris water maze tracking algorithm for psychophysical studies. *Biomed Sci Instrum*, 8:36–263, 2000.

[3] M. Dorigo and G. D. Caro. *New ideas in optimization*, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[4] H. Image. Water 2020 — software specifications. *http://www.hvsimage.com/information/specifications/*, 2004.

[5] Z. Kalafatic. Model-based tracking of laboratory animals. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 2, pages 22–24, 2003.

[6] Z. Kalafatic, S. Ribaric, and V. Stanisavljevic. A system for tracking laboratory animals based on optical flow and active contours. In *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, pages 334–339, 2001.

[7] P. Martin and P. Bateson. *Measuring Behaviour: an introductory guide*. Cambridge University Press, 1988.

[8] R. G. Morris. Spatial localization does not require the presence of local cues, learning and motivation. *Learning and Motivation*, 12:239–260, 1981.

[9] N. I. Technology. Noldus ethovision 3.0 — technical specifications. *http://www.noldus.com/download/ev_ techn_ specs_ v30.pdf*, 2002.

[10] Wikipedia. Morris water maze — wikipedia, the free encyclopedia, 2007. [Online; accessed 1-June-2007].