Machine Vision and Applications (2014) 25:173–184
DOI 10.1007/s00138-013-0520-2

FULL LENGTH PAPER

Active tracking and pursuit under different levels of occlusion: a two-layer approach

Tomer Baum · Idan Izhaki · Ehud Rivlin · Gadi Katzir

Received: 25 May 2012 / Revised: 3 May 2013 / Accepted: 10 May 2013 / Published online: 12 June 2013

Abstract We present an algorithm for real-time, robust, vision-based active tracking and pursuit. The algorithm was designed to overcome problems arising from active vision-based pursuit, such as target occlusion. Our method employs two *layers* to deal with occlusions of different lengths. The first layer is for short- or medium-term occlusions: those where a known method—such as mean shift combined with a Kalman filter—fails. For this layer we designed the hybrid filter for active pursuit (HAP). HAP utilizes a Kalman filter modified to respond to two different modes of action: one in which the target is positively identified and one in which the target identification is uncertain. For long-term occlusions we use the second layer. This layer is a decision algorithm that follows a learning procedure and is based on game theory-related reinforcement (Cesa-Bianchi and Lugosi, Prediction Learning and Games, 2006). The learning process is based on trial and error and is designed to perform adequately with a small number of samples. The algorithm produces a data structure that can be shared among agents or sent to a central control of a multi-agent system. The learning process is designed so that agents perform tasks according to their skills: an efficient agent will pursue targets while an inefficient agent will search for entering targets. These capacities make this system well suited for embedding in a multi-agent control system.

Keywords Bio-inspired computer vision · Hybrid system · Active pursuit · Reinforcement learning · Tracking with occlusion

T. Baum (✉)
Mathematics Department, Technion, Israel Institute of Technology, 32000 Haifa, Israel
e-mail: tomeradibaum@gmail.com

I. Izhaki · E. Rivlin
Computer Science Department,
Technion, Israel Institute of Technology, 32000 Haifa, Israel
e-mail: ehudr@cs.technion.ac.il

I. Izhaki
e-mail: idan192@t2.technion.ac.il

G. Katzir
Department of Evolutionary and Environmental Biology,
University of Haifa, 31905 Mount Carmel, Israel

G. Katzir
Department of Marine Biology, University of Haifa,
31905 Mount Carmel, Israel

1 Introduction

Active tracking and pursuit of targets by autonomous systems is a rapidly developing research area. In this paper we present a real-time algorithm for such a system, with a focus on robust reactions to occlusion. We address the problem of tracking targets with occlusions. This problem can be roughly divided into two sub-problems arising from difference in occlusion's duration. The first sub-problem is overcoming short-term occlusions, i.e., those where the target's characteristics as observed before the occlusion are still relevant. The second sub-problem is selecting a policy for reobtaining a view of the target when the target had enough time to change its motion pattern behind the occlusion with high probability.

Tracking systems must be able to deal with the disappearance and reappearance of targets. Single hypothesis trackers such as Kalman filter-based trackers [1] cope poorly with the disappearance of a target. In [2], for example, a location in the frame that resembles the target is chosen by the mean-shift process. This can cause the tracker to drift away from the target's real location and prevent it from being reidentified when it next appears. Particle filter-based trackers[3] and

trackers that incorporate Sequential Monte Carlo and Kalman filter, e.g. UKF, [4], robustly cope with the problem of target reappearance. However, these multi-hypothesis trackers best suit situations where the target reappears in the current scene. Moreover, these trackers require that the image be large enough to be identified without zooming in. But these requirements are not met in many scenarios, such as when the target moves fast, is far from the camera or when an active tracking system needs to physically pursue the target. The algorithm presented here is designed to cope with these difficulties.

We use two layers of reaction to estimate bearing to an occluded target. The first layer, the hybrid filter for active pursuit (HAP), is designed to cope with medium term occlusions. This would include, for example, those for which a known method—such as mean shift combined with a Kalman filter—fails, while the properties of the target's motion (e.g., velocity and direction) before it disappeared can be considered unchanged. Our design of HAP is a variation on the general concept of the novel approach in [5], where the authors generalized the classic Kalman filter by modeling the arrival of a new observation as a random process. They also found a critical value on the arrival rate beyond which a transition to unbounded error covariance occurs, and they used a random process that indicates, at every time instance, whether the measurement of the Kalman filter is available. However, this indicator has a similar binomial distribution for every t. HAP differs from [5] in that we introduce a new random process indicating that the target has been identified and use a binomial distribution with a parameter that can vary over time. This modification arises from the fact that the probability of identifying a target while in pursuit can change over time.

The second layer is a decision algorithm, embedded in a learning procedure and based on game theory-related reinforcement which is designed to address long occlusions. It is pursuit-evasion oriented [10]. The system decides on reacting by one of the following possible robot reaction policies:

1. Continue its motion towards the place where the target was last spotted.
2. Move towards the predicted target location, using the target's observed motions prior to occlusion.
3. Search for a new target.

The learning process is based on "trial and error" and is designed to perform adequately with a small number of samples. Thus, we require that

1. A wrong choice will not reoccur on a given scenario in the future.
2. A robot showing poor results (i.e., a low rate of success) will not attempt to pursue and will perform mainly the search policy.
3. Each trial should have an effect on the learning process.

The algorithm will have a data structure that can be easily shared among agents, or sent to a central control of a multi-agent system. Moreover, the learning process is designed with an agent performing tasks according to its skills, so that a successful pursuer should pursue while a unsuccessful pursuer should search for entering targets. These capacities make this system suitable for embedding in a multi-agent system [10].

Our method, which is related to a game theoretic approach, is termed "Randomized Prediction with Expert Advice" [6]. "Prediction with Expert Advice" describes a problem where a forecaster attempts to iteratively predict a state so as to minimize a function called "regret" by evaluating the quality of each expert's advice in comparison to the actual state achieved. In the system, when a target is lost, the recent history of its observed data is saved to a parameter vector, which is considered as a new expert. Its *advice* is the latter vector with an addition of one of the three robot strategies mentioned above. After a policy has been applied and its success evaluated, the policy is reconsidered and the quality of all the advice involved in the decision is reevaluated. The algorithm iteratively creates a mapping from the space of all possible target parameter vectors to the set of policies. The learning algorithm is suitable for systems where hidden Markov models **(HMM)** [7] are difficult to construct, for example systems that randomly generate a high noise level from sensors and communication. The data structure built by the learning process is simple and suitable for sharing in a multi-agent system.

To validate the robustness of the method we implement an algorithm based on this method on **NEMALA**, a low-cost, miniature robotic platform designed to function as an agent in a multi-agent system. The possible robot strategies are inspired by the reactions of praying mantises to occlusions. Praying mantises were presented with simulations of prey disappearing on a computer screen. The mantises reacted in one of the following manners:

1. Fixated on the place where the target had been last observed.
2. Continued performing saccadic movements, as if the target was visible and in motion. This behaviour resembles extrapolation.
3. Performed peering movements.

These behaviours were "translated" to the pursuing behaviours of the robot. In this paper we focus on robust estimation of the bearing to the target. We assume that correct estimation of the range to the target is less important because, if high enough velocity is maintained in the direction of the target, it should eventually be intercepted.

2 The first layer: a hybrid Kalman filter for bearing estimation

Our control process comprises two parallel sub-processes, one for controlling direction and one for controlling velocity. The system state is represented by the following state vector:

$$S = [R, \Theta], \tag{1}$$

where R is the range from the robot to the target (in cm) and Θ is the angle (in radians) between the line connecting the location of the robot with the target's location and the principle axis of the camera. The camera cannot move independently. We assume that R and Θ are controlled by two independent processes that aim to minimize R and Θ.

We assume that the pursuer's velocity is higher than the velocity of the target such that if bearing to the target is evaluated correctly, a simple guidance law will lead to its interception. In the simulations, the velocity of the pursuer was constant and higher than the velocity of the evader. We assume that the robot steers to minimize Θ. Control issues are not discussed in this paper as we consider the main challenge of our system to be the estimation of Θ. We follow the approach presented in [5].

2.1 The validity of target identification

The system performs vision-based pursuit as follows: for every new frame we assume that an image-processing stage is performed and that its output $\mathbf{l}(t) = (x(t), y(t))$ is the evaluated location of the target's center of mass in the frame (in pixels). We further assume that the principal point of the camera is at $(x_p(t), y_p(t)) = (0, 0)$. In our experiments we used the mean-shift tracker for the image-processing stage. We assume that if the estimation of $\Theta(t)$ is correct, then

$$x(t) \simeq \frac{\mathrm{FL}}{\mathrm{PS}}\Theta(t), \tag{2}$$

when FL is the focal length and PS is pixel size, both in centimeters.

As mentioned above, we introduce a new random process indicating that the target had been identified; we term this process *Validity Random Process*. For each time instance t, this process is defined by a random variable we termed *Validity Random Variable*, denoted by γ_t. For example, if obtained data indicate that the target is partially occluded, then the probability of $(\gamma_t = 1)$ should be smaller than if data indicate that the target is currently in full view. In the case that HAP fails to reidentify the target, i.e., if the estimated probability of $(\gamma_t = 1)$ is smaller than a certain threshold $th1$, the system switches to the second layer. The *Validity Random Variable* $\gamma(\mathbf{l}(t)) = \gamma_t$ is defined as follows: For every $t = 0, 1, 2 \ldots$,

$$\gamma_t = \begin{cases} 1, & \text{the target is identified in } \mathbf{l}(\mathrm{t}). \\ 0, & \text{else} \end{cases} \tag{3}$$

Target identification is evaluated by setting a threshold on a resemblance function $\mathrm{res}(\mathbf{l}(t))$ that considers the target's expected parameters, obtained from a known target model, and the model of the area of $\mathbf{l}(t)$. Let us denote $P(\gamma_t = 1) = \lambda_t$ where λ_t is also a function of $\mathrm{res}(\mathbf{l}(t))$. We assume that γ_t is distributed s.t $\lambda_t = \mathrm{res}(\mathbf{l}(t))$. We compute λ_t as the resemblance of the target's histogram to an initial model colour histogram. In most simulations we used a function of the inner product of the normalized histograms.

2.2 The dynamic system model

The robot turns at every iteration to obtain $\hat{\Theta} = 0$. We model the dynamics of the Θ as

$$\Theta(t+1) = a\Theta(t) + w(t) \quad t = 0, 1, 2 \ldots \tag{4}$$

$0 \leq a \leq 1$. a is a constant that is in direct proportion to the system's efficiency in keeping the target's image within the captured frame close to the principal point. w_t, is a white noise caused by the tracking process:

$$w(t) \sim N(0, Q). \tag{5}$$

$N(0, Q)$ is a Gaussian distribution with 0 mean and variance $VAR(w(t)) = Q$. We assume that $x(t)$ and $\Theta(t)$ are related as follows:

$$x(t) = \Theta(t) + v(t) \quad t = 0, 1, 2 \ldots \tag{6}$$

We introduce a dynamic bimodal measurement noise covariance

$$v(t) \sim \begin{cases} N(0, R), & \gamma_t = 1 \\ N(0, r(\lambda_t)), & \gamma_t = 0 \end{cases} \tag{7}$$

where $r(\lambda_t) > R$, the variance of the measurement noise in the case that the target is unidentified, monotonically decreases as a function of λ_t. This definition of the measurement noise covariance suits the different characteristics of the two modes (identification and non-identification of a target) in which our system performs.

2.3 The estimation process

We begin with the following definitions:

$$\hat{\Theta}(t+1|t) \triangleq E\left[\Theta(t+1) | (x(1), x(2), \ldots, x(t)), \gamma_{t+1}\right]. \tag{8}$$

$$P_\Theta(k|j) \triangleq E\left[\left(\Theta(t+1) - \hat{\Theta}(t+1|t)\right)^2 | (x(1), x(2), \ldots, x(t)), \gamma_{t+1}\right]. \tag{9}$$

From Eqs. (4) and (6), Θ_{t+1} and x_{t+1} are jointly Gaussian conditioned on previous measurements and γ_{t+1} with

$$E\left(\Theta_{t+1}, x_{t+1} | \left(x(1), x(2), \ldots, x(t)\right), \gamma_{t+1}\right) = \left[\hat{\Theta}(t+1|t), \hat{x}(t+1|t)\right], \tag{10}$$

and

$$COV\left(\Theta_{t+1}, x_{t+1} | (x(1), x(2), \ldots, x(t)), \gamma_{t+1}\right) = \begin{pmatrix} P_\Theta(t+1|t) & P_\Theta(t+1|t) \\ P_\Theta(t+1|t) & P_\Theta(t+1|t) + \gamma_{t+1}R + (1-\gamma_{t+1})r(\lambda_t) \end{pmatrix}. \tag{11}$$

The Kalman filter equations will be

$$\hat{\Theta}(t+1|t) = a\hat{\Theta}(t|t). \tag{12}$$

$$P_\Theta(t+1|t) = a^2 P_\Theta(t|t) + Q. \tag{13}$$

Correction:

$$\hat{\Theta}(t+1|t+1) = \hat{\Theta}(t+1|t) + P_\Theta(t+1|t)T(t) \times \left(x(t+1) - \hat{\Theta}(t+1|t)\right). \tag{14}$$

$$P_\Theta(t+1|t+1) = P_\Theta(t+1|t) - P_\Theta(t+1|t)T(t)P_\Theta(t+1|t), \tag{15}$$

where

$$T(t) = \left(P_\Theta(t+1|t) + \gamma_{t+1}R + (1-\gamma_{t+1})r(\lambda_t)\right)^{-1}. \tag{16}$$

As mentioned earlier in this section,if the value of λ_t falls beneath the threshold th_1, measurements are considered unreliable and the HAP—as ineffective. In the following section we present a decision process for selecting a policy for re-obtaining a view of the target:

3 The second layer: decision using a game theory approach

As noted in the previous section, when the value λ_t decreases below th_1, a decision needs to be made regarding one of the three robot policies described in the introduction. We call this stage the "decision stage," and the algorithm described in this section *Decision with Expert Advice* (DEA). The system's decision depends on a learning process, based on the game theoretic approach known as "prediction with expert advice" [6]. In our algorithm, we keep an evolving set of learned data in a data structure, consisting of a matrix denoted as *the advisors' matrix*, and additional variables.

3.1 The decision stage

In the decision stage, two processes occur: (i) an update of the data structure and (ii) a policy decision. Selected parameters of the available data, such as the target's last Θs and recent history of λ_t are kept as a new row in the advisors' matrix. We call such a set of parameters *expert*, denoted by **xp**. In addition we add to each row i three variables:

- $\mathbf{q}_i$—weight, assigned an initial value 1, indicating the quality of the advice.
- $\mathbf{pnn}_i$—the line number of the advice followed (Eq. 18).
- $\mathbf{P}_i$—The chosen policy number (1,2, or 3, as defined in Sect. 1).

The last two variables are updated after a policy decision has been made. An *advice*, $\mathbf{adv}_i$, is the set $\{\mathbf{xp}_i, \mathbf{q}_i, \mathbf{pnn}_i, \mathbf{P}_i\}$.

The policy decision. Define the advice distance

$$d_\Delta(i,j) = (xp_i - xp_j)\mathbf{\Delta}(xp_i - xp_j)^{\mathrm{T}}, \tag{17}$$

where $\mathbf{\Delta}$ is a positive semidefinite matrix.
We define, for the last row, a *preceding nearest neighbor* (*pnn*), as

$$pnn = argmin_{\{j:j<i\}}\{d_\Delta(i,j)/q_j\}. \tag{18}$$

The *pnn* is the advice index we use to set the current policy,

$$\mathbf{P}_i = \mathbf{P}_{pnn}. \tag{19}$$

3.2 The evaluation stage

Following the decision stage, the system carries out the chosen policy. The evaluation stage takes place only if one of the active policies (1 or 2) was chosen. Let T denote the frame when the decision was made. A policy choice is called "successful" if the target is reidentified.

Updating the learned data. The advisors' matrix is updated at this stage only in case of failure. In this case an additional row $(i+1)$ added.

If the target is successfully identified, the weights are updated as follows:

$$updated\ \mathbf{q}_{pnn} = \alpha\mathbf{q}_{pnn}$$
$$updated\ \mathbf{q}_i = \beta\mathbf{q}_i. \tag{20}$$

The latter equations express our wish to strengthen correct advisors. In the case of failure, the weights are updated as follows:

$$updated\ \mathbf{q}_{pnn} = \psi\mathbf{q}_{pnn}$$
$$updated\ \mathbf{q}_i = \delta\mathbf{q}_i$$
$$updated\ \mathbf{q}_{i+1} = \epsilon\mathbf{q}_i \tag{21}$$

In this case, the policies of i and $i+1$ are also updated as follows:

$$updated\ \mathbf{P}_i = \begin{cases} 1 & \mathbf{P}_i = 2 \\ 2 & \mathbf{P}_i = 1 \end{cases} \tag{22}$$

$$updated\ \mathbf{P}_{i+1} = 3, \tag{23}$$

where $\alpha \leq \beta \leq \delta$, $\epsilon < 1$ and $\psi < 1$.

β is the "increment" in the weight of a correct policy decision and thus it is the highest. δ is relatively large, to ensure that an error will result in a trial of a different policy if a target with similar parameters appears. α is the increment in a weight of a correct advisor. ψ is a "penalty" for a mistaken advisor. ϵ is small due to the fact that the algorithm produces policy 3 when both active policies have produced failures. If the value of ϵ is smaller, the system will be more likely to choose active pursuit policies 1 and 2.

The correct values for these parameters should be set through experiments. The algorithm described above enables rapid decision making upon the arrival of a new target. The obtained advisors' matrix, via Sect. 3.1, implies the mapping of the possible parameters of the target vector (**PTP**) to the possible reaction policies. We term this mapping the *output mapping*. We first assume that the size of the set **PTP** is finite and second, that a real mapping, maximizing the probability of regaining a target for each set of target parameters, is available. We term this mapping the *ground-truth mapping* (**GT**). We define the correctness $Crct$ of an output mapping M in relation to the ground-truth mapping **GT** as follows:

$$\text{True}_{\mathbf{GT}}(\mathbf{M}) = \{v \in \mathbf{PTP} : \mathbf{M}(v) = \mathbf{GT}(v)\}. \tag{24}$$

Then,

$$\text{Crct}_{\mathbf{GT}}(\mathbf{M}) = \frac{|True_{\mathbf{GT}}(\mathbf{M})|}{|\mathbf{PTP}|}. \tag{25}$$

The correctness is the optimization function we try to maximize. If the real mapping has a few connected regions, then the algorithm above may produce a mapping that contains too many connected regions, resulting in a decrease in its correctness with time. We address this problem by segmenting the learning process: we pre-define the segment size N. For each N arriving targets, the obtained advisors' matrix A_i is stored. Each such advisors' matrix is weighted as 1 upon its creation and its weight is updated through time in accordance with its success in evaluating policies for each target. A decision as to the policy for a newly arrived target is now made by a weighted voting of the different advisors' matrices. We term this procedure weighted temporal mappings (WTM).

4 Experiments

4.1 Simulations for the estimation process

To evaluate the performance of HAP we created a simulator in MATLAB. We compared the performance of a mean-shift tracker (MS), a mean-shift tracker combined with a Kalman filter tracker (MS-KF), and a mean-shift combined with HAP. Initial identification of a target was either performed manually by a user in the simulations or automatically by histogram comparison in NEMALA. We conducted four types of comparisons:

1. Comparing the performance of the three trackers in an artificial movie of a small square moving behind a larger rectangular surface.
2. Comparing the performance of the three trackers in real movies.
3. Comparing the performance of the three trackers implemented in a simulation of a robot equipped with a single camera, when pursuing an evader.
4. Comparing the performance of the three trackers on a real movie, with changing virtual occlusion duration.

In these simulations, measurement covariance for the MS-KF tracker was equal to the measurement covariance of the HAP tracker when the target was identified, i.e., when $\gamma = 1$. When $\gamma = 0$, the HAP tracker uses measurement covariance achieved by the calibration process (see below). In all of the experiments we used a zero process noise covariance, as the tracker should not produce any process noise.

Simulation of a basic movement with occlusion. We created a video clip of a grey square moving behind a larger rectangular surface (Fig. 1). Each movie frame was 800×800 pixels colored white. The target size 80×80 pixels, coloured with grey level 127 (in a 0–255 grey-level scale) (Fig. 1). The occluding rectangle was black with randomly generated grey noise (Fig. 2). The noise was Gaussian distributed with mean and std equaling 127. Values below 0 were considered 0 and values over 255 were considered as 255. The height of the rectangle was 300 pixels and width changed between experiments. The square moved at a velocity of 20 pixels per frame and performed random turns.We compared the performance of MS, MS-KF and HAP on 100 randomly generated clips.

As predicted, the MS tracker could not overcome a full occlusion, the MS-KF success rate dropped when the occluding rectangle was of 105 pixels or more, while the HAP success rate remained high up to 200 pixels (Fig. 3).

We also compared the performance of MS, MS-KF and HAP on 150 randomly generated clips with parameters such as velocity and direction changed. HAP produced an average distance of ca. 22 pixels between the centre of mass and the estimated center of mass while MS-KF produced A distance of ca. 119 pixels and MS produced a distance of ca. 318 pixels.

Tracking in a realistic movie clip. In order to evaluate the performance of the algorithm in a realistic scenario, we compared the tracking performance of MS, MS-KF and HAP on a real movie clip. The movie clip is of a car filmed from a helicopter (Figs. 4, 5). Our measure of success was the duration

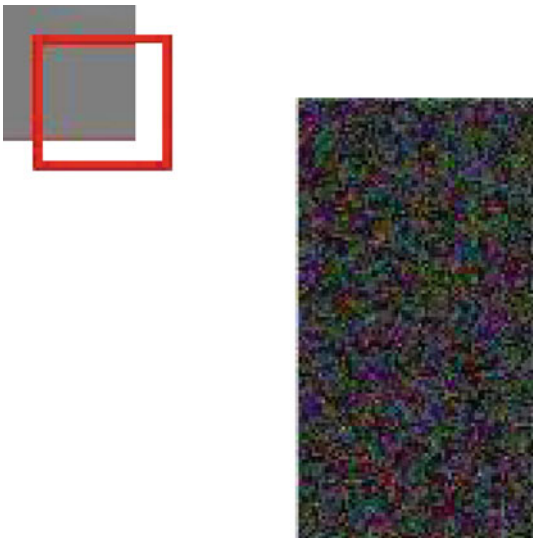

Fig. 1 A frame from an artificial simulation clip. The *red frame* indicates where the MS identifies the target. In the case that the moving *grey square* is surrounded by the *red frame*, the tracker identifies it perfectly (color figure online)

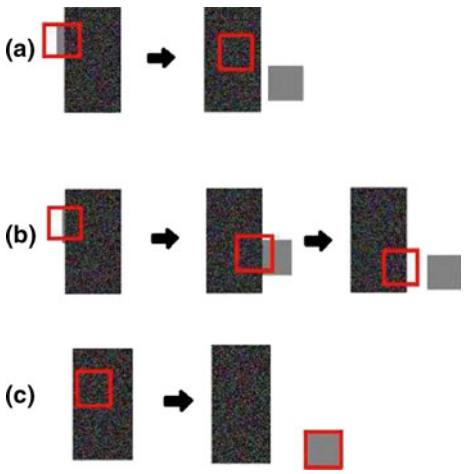

Fig. 2 Simulation on a rectangle of size 125 pixels (ca. 1.5 times the occluded target). **a** MS fails when the target is fully occluded. **b** MS-KF fails to continue with motion of the target after the target reappears. **c** HAP successfully overcomes target occlusion

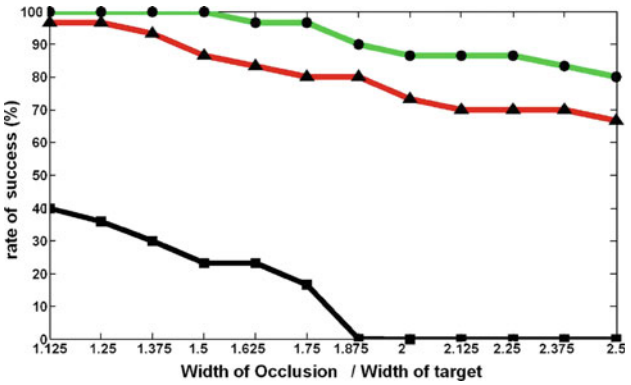

Fig. 3 Results of the simulations. MS (*squares*) generally fails to overcome occlusions, while HAP (*circles*) outperforms MS-KF (*triangles*). When occlusion is of width of 2.5 times the width of the target, HAP overcame the occlusions in 80 % of the experiments while the MS-KF success rate was 66.6 %

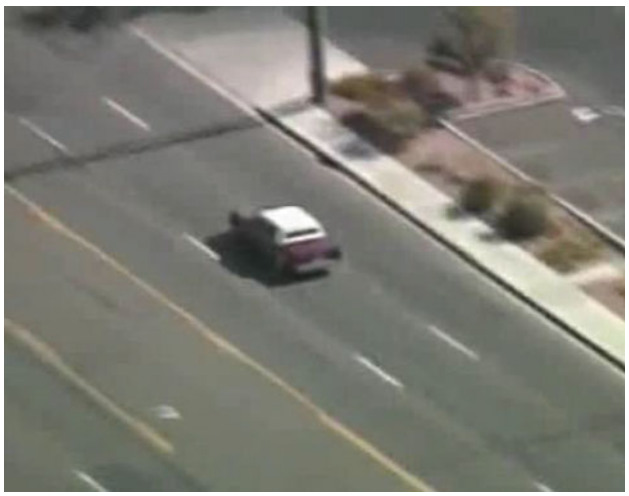

Fig. 4 A frame from the beginning of the movie clip. An evading car filmed from a helicopter. During the movie the image of the car continuously gets smaller, as the car moves away from the camera

of correct tracking, as identified by us. As in the previous experiments, we found a value for the measurement noise covariance that maximizes the performance of the MS-KF tracker.

Video duration was 15 s. The MS tracked the target for 8 s, and a large occlusion by a tree caused the tracker to lose the target. The MS-KF lost the target after 4 s. A change in the motion of the photographing camera caused the MS-KF tracker to lose the target. Compared with the other algorithms, HAP tracked for longest duration, never losing the target.

In two additional movies, we demonstrate the robustness of HAP. In a video clip taken from ABC news, a black car viewed from a helicopter is blocked by large tree and then drives through the shadow of the tree. The movie is filmed at 25 frames per second. In the first frame, the tracker frame was set on the front of the car. The target moved through the shadow of the tree for seven frames, during which the tracker maintained its focus on the front of the car, which was then hidden by the tree's shadow. This occlusion lasted for 32 frames, after which the tracker regained the front of the car. The shadow affected the view of the car for an additional 30 frames during which the tracker maintained its view of the car but the frame was now locked on the front window. HAP succeeded to overcome the occlusion. In a second video, we tracked the shoe of one of the authors while walking, with the author's view blocked by a large black cover. The author was hidden for seven frames (at 15 fps). HAP succeeded to overcome this occlusion as well. We also performed the

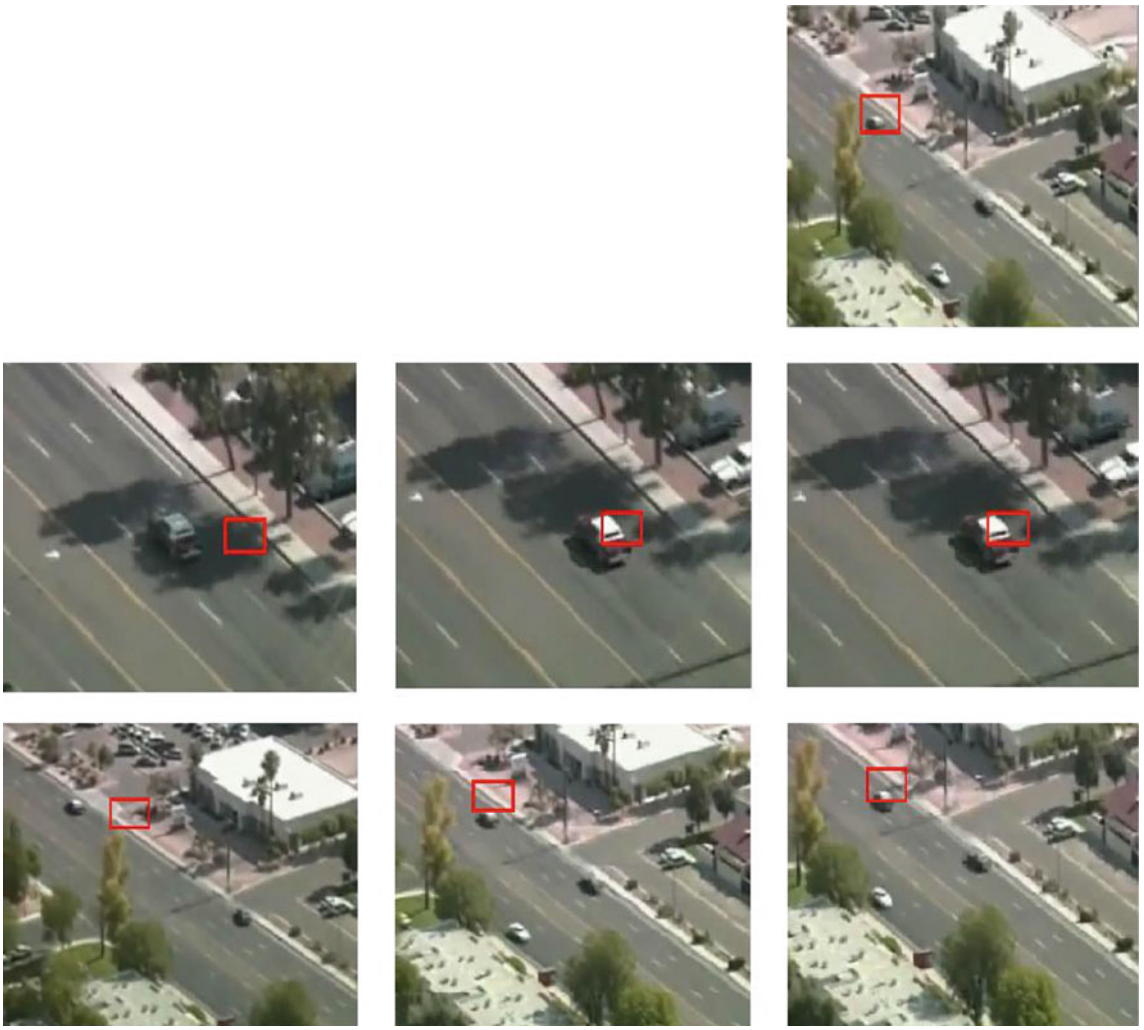

Fig. 5 A comparison of the performance of HAP, MS-KF and MS (respectively *top*, *middle* and *bottom rows*) on a real video clip. The sequences are at ca. 0.5 s intervals viewed from *right* to *left*

following experiment: we filmed a black ball moving in a room, then added a black rectangular occlusion (Fig. 6). We measured the distance between the ball centre as identified by a user and the centre the algorithm found. Results appear in Fig. 7.

Simulation of a pursuit. As noted earlier, the main purpose of HAP is to overcome difficulties that are typical to vision-based pursuit. To validate our claims, we created the visual pursuit simulator (VPS). The VPS simulates a vision-based pursuit on a plane. Both the pursuer and the evader were simulated black spheres and the background was white, to make the image processing easier, as we were not interested in evaluating the image processing performance of our system.

We graphically presented the pursuit by means of overview from the pursuer's point of view (Fig. 8). We assumed that the pursuer's camera, sampling at 30 Hz, views an angle of 60 ° and that the initial distance between the pursuer and the evader was 75 cm. The pursuer moved at a velocity of 80 cm/s while the evader moved at 60 cm/s. We randomly generated a path in which the evader performed frequent direction change. We assumed that the pursuer maneuvers itself to keep the evader located on the optical axes of the camera. We assumed that the pursuer can turn at a speed of 90 °/s. We randomly created 100 evading paths for each of the trackers, MS, MS-KF and HAP, with the target disappearing for 5, 10 and 15 s (total of 900 runs). After 100 frames, we checked whether the target had been regained. The results indicate that HAP outperforms both the MS and MS-KF (Fig. 9).

Calibration process. We used the following calibration process: MS tracking was performed on a test movie, to find a correct value for $th2$. $th2$ is the threshold for the resemblance function. A resemblance value lower than $th2$ implies $\gamma = 0$. A lower than correct threshold means that HAP may fail to identify occlusion and will continue estimating based on false observations. For a target that moves with fixed velocity this

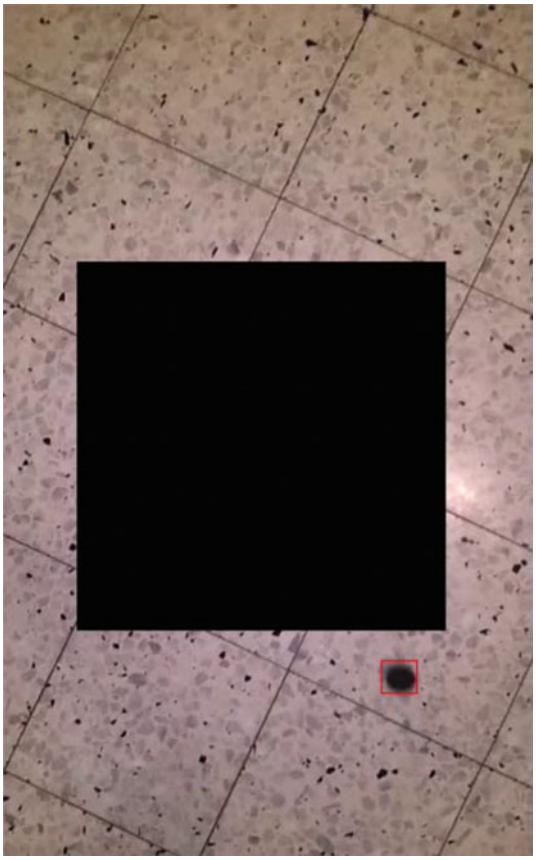

Fig. 6 A frame from a movie of a ball rolling on a floor with a synthetic occlusion

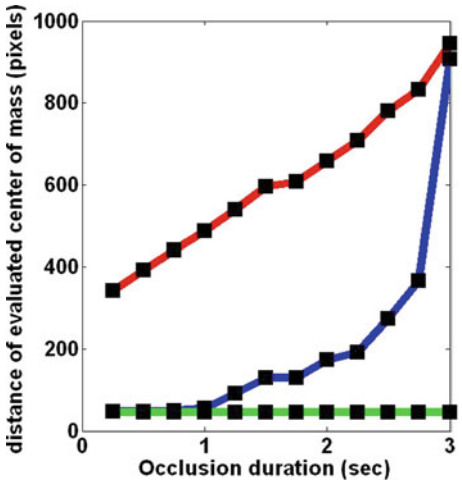

Fig. 7 A real movie with a synthetic occlusion of changing duration: a comparison of performance of HAP (*green*), MS-KF (*blue*) and MS (*red*). Samples are marked with *squares*. Frame size was $1,280 \times 720$ pixels (color figure online)

may cause HAP to stop following the target. In case of a higher than correct threshold HAP may ignore correct observations, which may cause the target to be lost if it changes its velocity or direction.

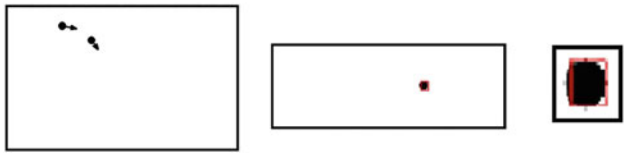

Fig. 8 Graphical representation of the pursuit in the VPS. Overview (*left*), captured frame from the pursuer camera (*middle*), and enlargement of the evader's area in the camera (*right*). The *red rectangular frame* indicates where the MS located it in the captured frame (color figure online)

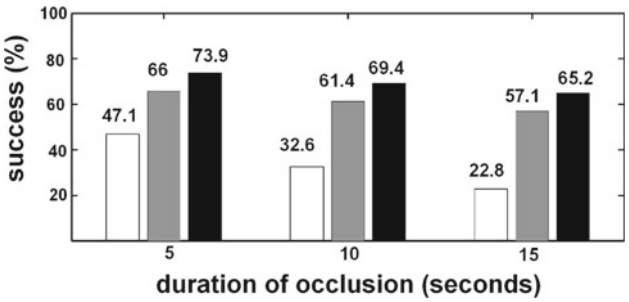

Fig. 9 A comparison of success rate of the three trackers MS (*white*) MS-KF (*grey*) and HAP. MS fails to overcome occlusions in most cases. HAP outperforms MS-KF consistently by ca. 8 %

λ_t was computed as the inner product of the histograms of the target model and the potential target. The values of $th2$ varied between 0.5 and 0.2, depending on the experiment. For MS-KF, before each experiment, we performed a series of simulations in order to find a value for the measurement noise covariance that maximizes the success rate of the algorithm. This was also used as the measurement noise covariance in HAP when a target was considered as identified, i.e., $\gamma = 1$. For HAP we performed an additional calibration process to identify the correct value for a matrix B. In our experiments the measurement covariance of HAP with occlusion was identified was $B * (1/\lambda_t - 1)$.

In realistic scenarios a calibration process should be performed on scene. The process can be a supervised learning process, with a user identifying whether a target is identified or lost. The process can also be an unsupervised learning process that identifies events with drastic decrease in the resemblance function as occlusions. Simulation movies are available at https://www.dropbox.com/sh/nkk51je1c0qotiv/qzjQPckOzk.

4.2 Decision simulations

We compared the performance of the DEA, WTM and Q-learning [9] algorithms. Correctness ($Crct$, Eq. 25) of the three algorithms was examined on two-dimensional ground-truth mappings. For each ground-truth mapping we performed 100 simulations, with 100 consecutive targets appear-

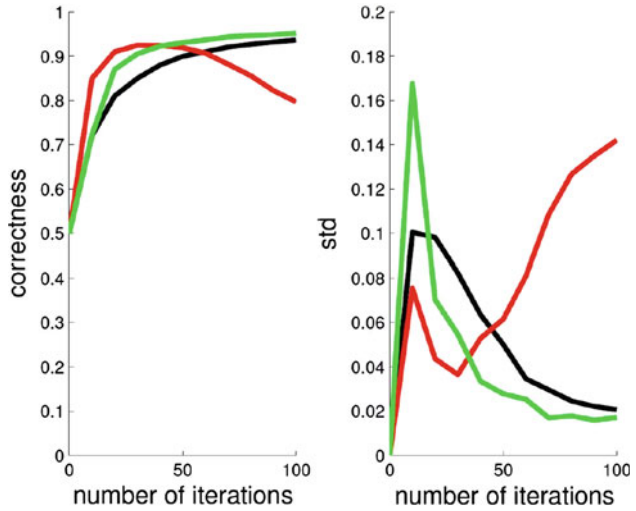

Fig. 10 The performance of the three algorithms on GT1, with $\psi = 0.8$. *Green* DEA, *red* DEA with WTM, and *black* Q-learning on **GT1**. *Left* correctness, *right* standard deviation of the correctness (color figure online)

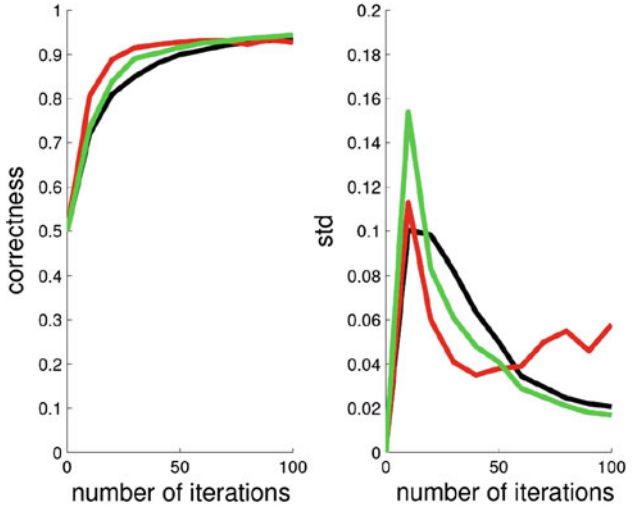

Fig. 11 The performance of the three algorithms on GT1, with $\psi = 0.45$. *Green* DEA, *red* DEA with WTM, and *black* Q-learning on **GT1**. *Left* correctness, *right* standard deviation of the correctness (color figure online)

ing in each. We computed the correctness and the standard deviation of the correctness over time. One dimension represented velocity (100 possible values) and the second represented λ_t of a target at the moment of disappearance (100 possible values). Thus the PTP size was 10,000.

The parameter values were $\alpha = 1.2, \beta = 1.2, \delta = 1.35, \epsilon = 0.65$. ψ was either 0.8 or 0.45. The first set of ground-truth mappings (**GT1**) represented a decision space in which a low-velocity target (<50) acquires policy 1; otherwise, policy 2 is applied. We tested our algorithm on these ground truths with $\psi = 0.8$, and we ran the same simulation with $\psi = 0.45$. When we used $\psi = 0.8$, DEA achieved a high degree of correctness (91 %) within 20 iterations. However, in the following iterations it added connectivity regions to the mapping it created, which led to a decrease in correctness over time. Adding WTM solved this problem, but the correctness rate increased at a slower rate (Fig. 10). When we used $\psi = 0.45$, DEA outperformed both Q-learning and WTM (Fig. 11). Figure 12 shows selected mappings obtained after running 100 iterations, with **GT1** shown on the right. In the tests on GT2 and GT3 we used $\psi = 0.45$.

The second set of ground-truth mappings (**GT2**) were randomly generated to produce complicated mappings with many connected regions of policy 1 (Fig. 14). DEA outperforms the other algorithms in learning these complicated mappings (Figs. 13, 14, 15, 16).

GT3 were examples of a ground truth where policy 3 is also chosen. This set of ground-truth mappings represents scenarios where the robot is less capable of a successful pursuit. In these ground-truth mappings DEA outperforms the other algorithms (Figs. 13, 15, 16).

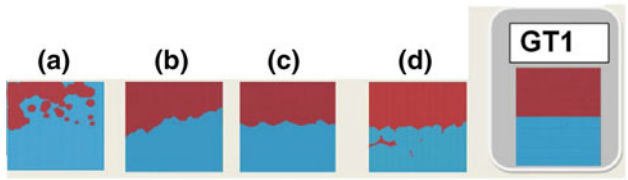

Fig. 12 Example of learned mappings for GT1: *Red* policy 1, *blue* policy 2. **a** DEA with $\psi = 0.8$, **b** DEA with WTM, **c** Q-learning, **d** DEA with $\psi = 0.45$. Resemblance of the learned mapping to the ground truth is an indication of the quality of the learning process (color figure online)

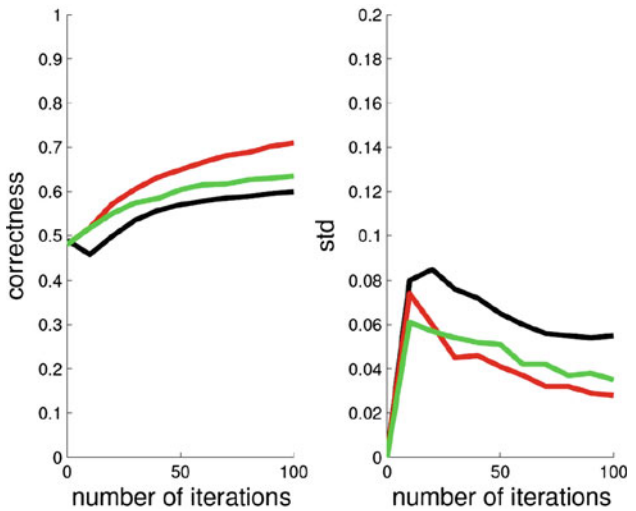

Fig. 13 The performance of the three algorithms on ground truth GT2. *Green* DEA, *red* DEA with WTM, and *black* Q-learning. *Left* correctness, *right* standard deviation of the correctness (color figure online)

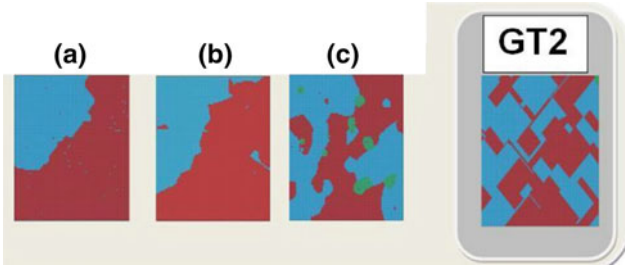

Fig. 14 Example of learned mappings for GT2: *Red* policy 1, *blue* policy 2, *green* policy 3; **a** Q-learning, **b** DEA with WTM, **c** DEA. Resemblance of the learned mapping to the ground truth is an indication of the quality of the learning process (color figure online)

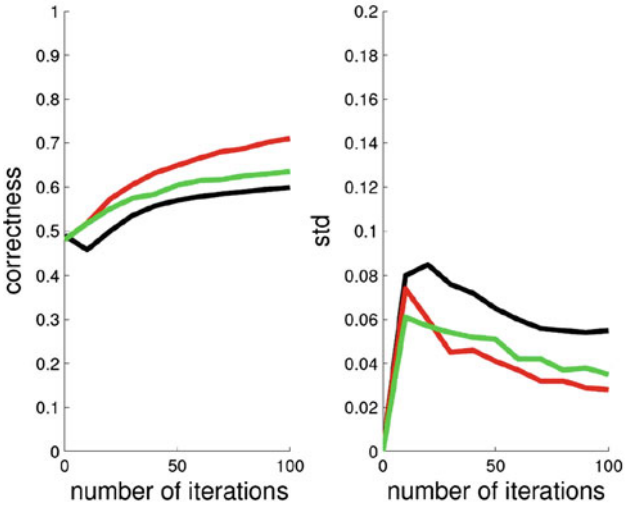

Fig. 15 The performance of the three algorithms on ground truth GT3. *Green* DEA, *red* DEA with WTM, and *black* Q-learning algorithm. *Left* correctness, *right* standard deviation of the correctness (color figure online)

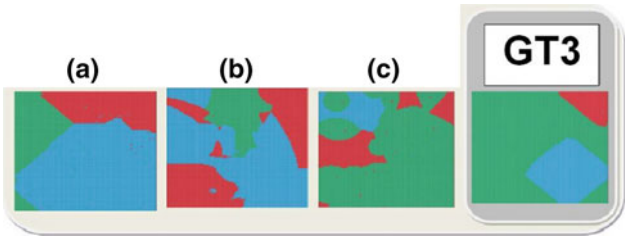

Fig. 16 Example of learned mappings for GT3: *Red* policy 1, *blue* policy 2, *green* policy 3. **a** Q-learning, **b** DEA with WTM, **c** DEA. Resemblance of the learned mapping to the ground truth is an indication of the quality of the learning process (color figure online)

4.3 Implementation of the control algorithm in a miniature robotic system

We examined HAP on a low-cost miniature robot (Fig. 17), designed in our lab. The robot has two electric 5V actuators and is equipped with a camera with a wide lens. The robot has computational capabilities (it is equipped with a PIC18

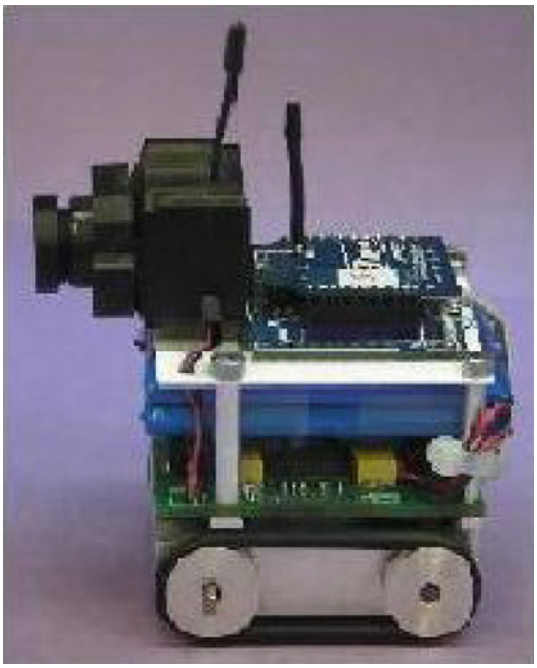

Fig. 17 The miniature single camera robotic system

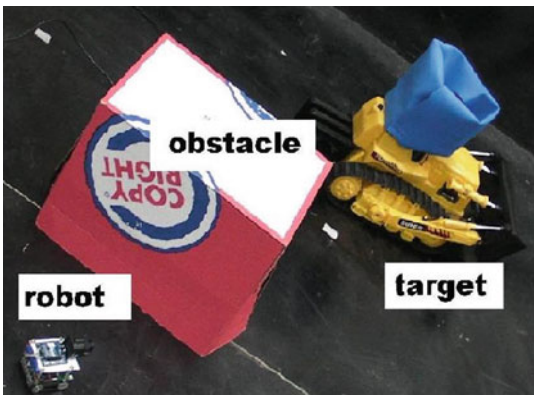

Fig. 18 The experimental setup with a target partially obscured from the robot

microcontroller), yet in these experiments, data processing was performed on a dual-core PC computer using mutex for synchronization between the threads. Control data were transmitted using a wireless connection through a serial port at 38,400 bps. The camera transmitted movie frames to the central unit through an analog communication channel. The setup (Fig. 18) consisted of

1. The obstacles: red cardboard boxes (30 × 20 × 25 cm).
2. The target: a yellow and blue miniature bulldozer trying to avoid the robot.
3. A scene: a rectangular black wooden arena.

The system experienced noise from changing lighting conditions, as well as from communication and control problems (Fig. 19).

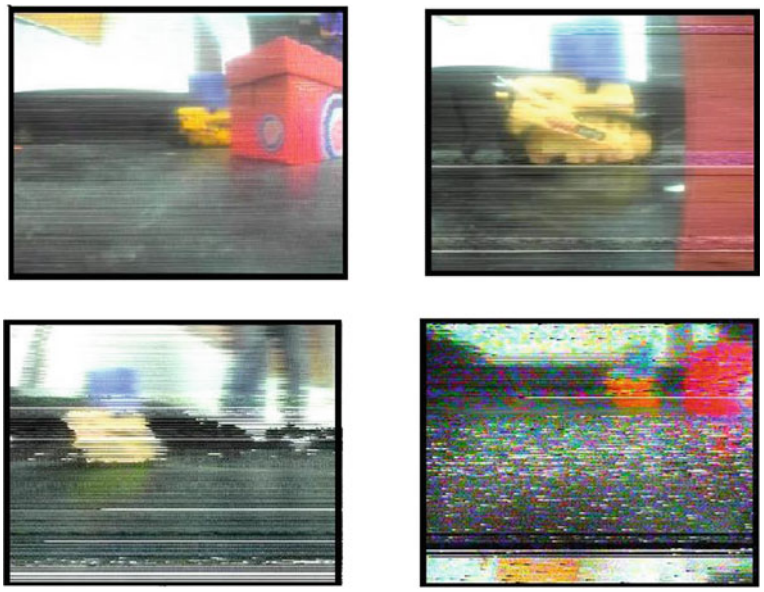

Fig. 19 Disturbances in captured frames. *Top row*: *left* the target and an obstacle as they appear in a captured frame; *right* radial distortion apparent at the edge of an obstacle. *Bottom row* disturbances in captured frames due to problems in communication

Another difficulty was disturbances in the control communication, which increased in the proximity of obstacles.

The control of the robot. The robot functioned in three alternative operation modes:

Search

In this mode, the robot turns around "looking" for a valid target. A frame is considered to contain a valid target based on two main criteria:

1. The probability of pixels with colours similar to the expected colours of the target is higher than a certain threshold.
2. The target's image structure, obtained by computing connected components of the yellow and blue pixel groups, is similar to a precomputed one.

A search stage terminates by initializing a pursuit. If a target has not been identified after a full rotation (360°), the robot searches for an obstacle to encircle in order to reach currently unviewed areas.

Pursuit

A basic pursuit module was programmed to chase the center of mass of the target. The Kalman filter-based control was implemented in the observation module, designed to produce the current centre of mass of the target. The Kalman filter affects the robot's control by manipulating the centre of mass.

Encircle an obstacle

Encircling an object was achieved as follows. For each movie frame we have computed the percentage of pixels with colour close to that of the obstacle. The robot was driven so as to keep this percentage fixed.

Results of the NEMALA experiments. In these experiments we tested the robustness of the HAP algorithm. The robot managed to overcome occlusions of a few seconds and to change its covariance as in HAP. After the need to move into the decision phase was identified, the robot performed reaction policy 2, that is, it moved towards the predicted target location by linearly decreasing Θ in the estimation process. As the robot turned to zero Θ it preformed an arc-motion towards the area where the target was supposed to reappear. We found that the presented system overcomes occlusions when the target continues its motion behind the occluding object.

5 Conclusions and future works

We present here the two-layered approach for active vision-based pursuit, focused on overcoming occlusions. Different simulations were performed to examine the characteristics of the system.

The first layer we present is the Hybrid filter for Active Pursuit (HAP), a Kalman-based filter for coping with short-term or medium-term occlusions, i.e., those where a known method such as mean-shift combined with Kalman filter fails. Our estimation method functioned in a noisy environment and was tested successfully on a prototype of a robotic system. HAP performance was tested in tasks of tracking in a realistic movie scene and on artificial movies created in Matlab. We constructed a realistic system, based on HAP algorithm, which performed pursuit of a moving target.

A second layer, for occlusions of long duration, was also presented. This layer is a decision algorithm that follows a learning procedure based on game theory-related reinforcement. The learning process is based on "trial and error" and is

designed to perform adequately with a small number of samples. We compared the performance of the present algorithm with a known learning algorithm (Q-learning) and showed improved results. The algorithm produced a data structure that can be shared among agents or sent to a central control of a multi-agent system.

Future work includes testing on real multi-agent systems and study of the parameters of the learning process. As this algorithm is designed for a multi-agent system, several agents should be incorporated into a better learning algorithm and boosting methods [8] should be considered.

Acknowledgments The authors deeply thank Orly and Daniel Ghosalker for their contribution to the machine learning section, Daniel Sigalov for his constructive remarks, and Amir Geva for the programming of NEMALA. Gadi Katzir was supported during the writing by an ISF grant. Tomer Baum thanks Boreh Olam for everything.

References

1. Bar-Shalom, Y., Li, X.R., Kirubarajan, T.: Estimation with Applications to Tracking and Navigation. Wiley, New York (2001)
2. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. **2**, 142–151 (2000)
3. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., Nordlund, P.J.: Particle filters for positioning, navigation and tracking. IEEE Trans. Signal Process. **50**(2), 425–437 (2002)
4. Van Der Merwe, R., Wan, E.: The square-root unscented Kalman filter for state and parameter-estimation. IEEE Int. Conf. Acoust. Speech Signal Process. **6**, 3461–3464 (2001)
5. Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M.I., Sastry, S.S.: Kalman filtering with intermittent observations. IEEE Trans. Autom. Control **49**(9), 1453–1464 (2004)
6. Cesa-Bianchi, N., Lugosi, G.: Prediction Learning and Games. Cambridge University Press, Cambridge (2006)
7. Rabiner, L.R., Juang, B.H.: An introduction to hidden Markov models. IEEE Acoust. Speech Signal Process. Mag. **3**(1), 4–16 (1986)
8. Schapire, R.E.: A brief introduction to boosting. Int. Joint Conf. Artif. Intell. **16**(2), 1401–1406 (1999)
9. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**, 279–292 (1992)
10. Durham, W., Franchi, A., Bullo, F.: Distributed pursuit-evasion without mapping or global localization via local frontiers. Auton. Robots **32**, 81–95 (2012)

Author Biographies

Tomer Baum was born in Haifa (Israel) in 1976. He received a BA (2002) and MA (2005) in mathematics from the Technion, Israel Institution for Technology. He is currently a researcher in the Intelligent Systems Lab in the Technion and conducts research on computational aspects of compound eye systems.

Idan Izhaki was born in Tel-Aviv (Israel) in 1987 and has received a bachelor degree in software engineering of computer science from the Technion, Israel Institute of Technology in 2012. He is currently a software engineer at Intel, Israel (Haifa).

Ehud Rivlin received a B.Sc and M.Sc degrees in computer science and a M.B.A degree from the Hebrew University in Jerusalem, and a Ph.D from the University of Maryland. Currently, he is a Professor in the Computer Science Department at the Technion, Israel Institute of Technology. His current research interests are in machine vision and robot navigation.

Gadi Katzir received his BSc (Biology) and MSc (Zoology) from the Hebrew University of Jerusalem, Israel. He received his PhD in Ethology from Cambridge University (UK). His research is conducted at the University of Haifa and Oranim, centres on visually guided behaviour in birds, reptiles, fishes and insects.