Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Scanning the Environment with Two Independent Cameras - Biologically Motivated Approach

Ofir Avni*, Francesco Borrelli†, Gadi Katzir‡, Ehud Rivlin* and Hector Rotstein§
*Department of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel
†Universita' del Sannio, Benevento, Italy
‡Department of Biology, Oranim - University of Haifa, Tivon, Israel
§Armament Development Authority and
Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel

***Abstract*— In this paper we present a novel method for visual scanning and target tracking by means of independent pan-tilt cameras which mimic the chameleon visual system. We present a systematic and optimization-based approach to the problem, from the high-level to the low-level control.**

In particular, in the first part we develop a new algorithm for scanning the sphere using multiple cameras. The algorithm combines information about the environment and a model of target movement, to perform optimal scanning by means of stochastic dynamic programming.

In the second part we develop a model-based control strategy for target tracking. A switching optimal control strategy based on smooth pursuit and saccades is designed by means of explicit Model Predictive Control (MPC) theory.

We simulated and experimentally validated our theory on a robotic chameleon head composed of two independent Pan-Tilt cameras. The resulting scanning pattern and target tracking has a remarkable resemblance to the one seen in nature by chameleons.

I. Introduction

Environment visual scanning is an important task for leaving creatures and autonomous systems such as robots. For both systems it is critical to visually explore the environment in order to identify relevant events or targets. In nature, solutions to this task are multiple: from immovable eyes on top of a movable neck (e.g. owls), through coupled eye movements as in primates (including humans), to independent eye movements observed in many fishes and in chameleons. If we limit ourselves to animals with perspective camera eyes, we can roughly identify three models which represent three different levels of eye movements vs. neck movements and correspond to the three examples above: the owl-like model, the human-like model and the chameleon-like model. In the world of robotics, the common solutions refer mainly to the first two models, namely using of two fixed cameras or moving them in a coupled manner [1]–[3]. These two models have the advantage of relative ease in calculations, such as estimating depth from stereo, yet they sacrifice the search efficiency by maintaining a large overlap between the two views. Our goal is to develop, model and control a robot head that derives from the third model, that is, a robot which uses two independently moving cameras.

Eye movements in chameleons are unique: the eyes are laterally positioned on the head (Fig. 1 shows a chameleon and our robotic head), and move independently while scanning large sections of the environment. The scanning is done through large independent saccades, which are rapid movements of the eye from one location to another. The range of movement of each eye is almost 180^o in the pan axis and about 90^o in the tilt axis. Once a target, such as an insect is detected, the chameleon fixates first one eye, and then both, on the prey and then shots its very long tongue to capture it [4]. If a target moves while the eyes are fixated, the entire head performs tracking movements and/or the eyes perform coupled saccades. The chameleon is the only vertebrate known to switch between coupled saccades and independent eye saccades [5]. In a complementary work we are studying eye movements in the chameleon. Preliminary results [6] indicate that the chameleon make use of a "negative correlation" for its global scanning strategy. That is, if one eye is scanning forward, the other will mostly be pointing backward, and if one eye points above the horizon, the other will look below it.

Fig. 1. The Robotic Head and a Chameleon Head

We consider a system with multiple pan-tilt cameras. The system' goal is to autonomously scan the environment for targets and once a target is found, to actively track it. The goal can be achieved by decomposing the problem into three tasks. Task A is the environment scanning. This is performed by a high-level algorithm that governs the global scanning method of the cameras. Task B is the image processing which investigates the cameras inputs. This includes for example the detection of targets. Task C is the target tracking. This is performed by an optimal controller based on smooth pursuit and saccades and designed by means of off-line Model Predictive Control (MPC) theory.

In this article we will concentrate on tasks A and C; we

will not discuss computer vision algorithms.

The high-level scanning algorithm performs a continuous scan of the environment looking for targets. The scanning algorithm is based on a model of the target movement and maximizes the number of targets found. Several approaches have been proposed in the literature for such goal; in particular in [7] the authors conduct a search for targets in a built environment with stationary targets.

In this manuscript, we assume a Markovian model both for the entrance of targets into the environment and for the movement of the targets in the environment. We formulate the problem as a dynamic stochastic programming [8] problem and solve it in order to obtain the optimal scan path. Our approach follows the ideas of [9] where the authors deal with coordinating the effort of several autonomous unmanned vehicles in the context of search and rescue. In our work the searcher is able to perform "jumps" between locations, which have additional cost.

The low-level control of both cameras performs two tasks: it moves the cameras from one section to another and it tracks a target once it is found. Active target tracking is one of the fundamental tasks in active vision. In general, in order to achieve a good active target tracking, it is common to combine *smooth pursuit* and *saccades* movements [10], [11]. In smooth pursuit the target is tracked smoothly as long as it is inside a predefined window in the center of the image, while the saccades purpose is to rapidly correct camera orientation when the target is lost or when we wish to move the camera to a new section. In this paper we make use of Model Predictive Control in order to design an active target tracking control policy based on the scheme described above.

Model Predictive Control (MPC) is a control technique. The main idea of MPC is to use the *model* of the plant to *predict* the future evolution of the system [12]. Based on this prediction, at each time step t a certain performance index is optimized under operating constraints with respect to a sequence of future input moves. The first of such optimal moves is the *control* action applied to the plant at time t. At time $t+1$, a new optimization is solved over a shifted prediction horizon. This leads to a feedback control policy.

In practice, solving an optimization problem in real-time might be challenging for processes with fast sampling times. Recently a method was developed to compute the optimization problem off-line for all feasible states. For linear and piecewise-linear systems, the feedback solution deriving from an MPC scheme with a linear or quadratic performance index and linear constraints is shown to be a piecewise linear function of the states. Therefore, MPC resorts to a lookup function evaluation [13], [14]. This has allowed us to use a real-time MPC scheme in order to control the process presented in this paper. We refer the reader to the manual of the Multi-Parametric Toolbox [15] for a quick introduction to the topic.

The rest of the paper is organized as follows. In Section II we present the high-level scanning algorithm, in Section III we present the smooth pursuit controller, the saccade controller and the way they are combined together. In Section IV we present simulations and experimental results. Conclusions are presented in Section V.

II. Optimal Search Method

The scanning algorithm presented in this section optimizes the search path in order to maximize the probability of finding a target and minimize the effort invested in the scan. In order to simplify the computational load we partition the environment into M regions, each region represents a location the system can explore. Time is also divided to discrete steps. Each time step is long enough to perform a search step. A search step comprises the actions of moving the camera to a new location and inspecting this new location by means of visual processing.

In the proposed model, the target is able to move from one region to another or even to appear and disappear from the environment of the system. To model target entering/exiting the system, a special region, region 0, is added to the system. This region represents all the areas in the world that are out of the system' sight, so the system cannot look for targets in that region. A target can appear in the system, by moving from region 0 to one of the other regions, and disappear from the system when moving to region 0. A Markovian model is used to model the movements of targets between regions. The Markovian model is defined by a transition probability matrix A, whose element in row i and column j is denoted by a_{ij}. We denote by X^k a random variable for the location of the target at time step k. Then, the elements of A will be computed as $a_{ij} = P(X^{k+1} = i | X^k = j)$, i.e., the conditional probability for a target to move from region j to region i, assuming the target in region j. We introduce a Probability Distribution Function (PDF) $\mathbb{P}$ which defines for each region j the probability for the target of being in j and which is updated according to Markovian model described by $A\mathbb{P}$.

Even if the target is located in a certain region j, the camera might not find the target when looking in region j. The chances of finding the target, or *detection rate*, is a characteristic of each region, and depends on several factors. For example, in an area with uniform color, detection rate is expected to be high, while in areas with a large variety of color and condense texture the detection rate will be lower. The estimation of detection rate for each region is complex and influenced by several parameters. We assume that the detection rate can be computed by means of computer vision algorithms and denote d_i the detection rate of region i.

Consider a PDF $\mathbb{P}^k$ at time step k defined as discussed previously. If the system inspects region i and fails to find a target then we update the PDF $\mathbb{P}^k$ by means of a function $T_i(\cdot)$ defined next:

$$(T_i(\mathbb{P}^k))_j = \begin{cases} \frac{P_j^k}{1-d_i P_i^k} & j \neq i \\ \\ \frac{(1-d_i)P_i^k}{1-d_i P_i^k} & j = i \end{cases}$$

where $(T_i(\mathbb{P}^k))_j$ denotes the posterior probability that the target is in region j given that no target was found in region i, and P_i^k denotes $\mathbb{P}(X^k = i)$. If the system inspects region i the updated posterior distribution at time step $k+1$ is given by the combination of the posterior update and the Markov process:

$$\mathbb{P}^{k+1} = AT_i(\mathbb{P}^k) \tag{1}$$

During the scanning, the cameras move from one region to a different one. In the proposed approach, we associate a cost to such movement. Therefore, a cost function $c(i,j)$ is defined for each pair (i,j) of regions. This cost reflects the effort needed to drive a camera from region i to region j plus the effort needed to scan region j. A cost of ∞ is set if the camera cannot move between i and j. For example, in the case where each camera guards its own hemisphere, the cost of moving the right camera to the left side will be ∞. On the other hand the system gains a reward R if a target is captured. The value of this reward relative to the cost function will determine the system behavior and is a tuning parameter.

We formulate a dynamic programming problem which weights cost and reward at each time step. The state vector at a given time step k is denoted by $(\mathbb{P}, i)$, where i is the region visited by the system at time $k-1$ and $\mathbb{P}$ is the current posterior probabilities after visiting region i. The state update function is defined by equation (1). We define the dynamic programming recursion in a similar way to [8], [16] where the stage cost captures both the reward for acquiring a target and the effort needed to perform the scan:

$$V_n(\mathbb{P}, i) = \min_j \left[c(i,j) + (R - d_j P_j R) + \right. \\ \left. + (1 - d_j P_j)\, V_{n+1}(AT_j(\mathbb{P}), j)\right] \tag{2}$$

where $V_n(\mathbb{P}, i)$ is the value function (or cost-to-go) at time n.

At each time step n we solve the dynamic programming recursion in (2) backward in time, starting from $V_{n+N} = 0$ in order to compute the optimal cost V_n over the horizon N and the optimal search policy. At the next time step the process is repeated for the same horizon N. During the search, the system acquires data on the different regions. Such data might be used to update A in order to better fit the new set of data.

The procedure illustrated previously can be extended to two or more cameras. The state vector will store the last location of all the cameras and the corresponding posterior probabilities matrices. A cost function will be associated to each camera and the optimization problem will involve all the cameras and the union of the regions which can be reached by all the cameras. In the experimental setup we use two cameras in order to scan the environment. Each camera can scan only a subset of the regions (the intersection of the subsets is not necessarily empty).

III. Smooth Pursuit and Saccade for Autonomous Tracking

In this section we describe the autonomous tracking controller design. In particular, we introduce the smooth pursuit controller, the saccade controller, and the method they are combined together. The smooth pursuit controller is a robust controller and is based on the assumption that the target model is excited by an unknown and bounded disturbance which brings it away from the center of a given tracking window. The saccade controller is a minimum-time optimal controller that takes the opposite approach: it assumes that the target movement is known, without any disturbances, and tries to center the target in the tracking window as fast as it can.

A. Smooth Pursuit Control

A detailed dynamical modeling of a pan-tilt camera has been developed. Because of lack of space, next we introduce an approximated linear, time-invariant (LTI) dynamical model. The following LTI dynamical model combines the camera states and target states:

$$\begin{bmatrix} \mathbf{x_p(t+1)} \\ \boldsymbol{x_t(t+1)} \end{bmatrix} = \begin{bmatrix} A_p & 0 \\ 0 & A_t \end{bmatrix} \begin{bmatrix} \mathbf{x_p(t)} \\ \boldsymbol{x_t(t)} \end{bmatrix} + \begin{bmatrix} B_p \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ B_t \end{bmatrix} v(t)$$

where $\mathbf{x_p} = [x_p, \dot{x}_p]$ is one axis camera' state vector, $\mathbf{x_t} = [x_t, \dot{x}_t]$ is the target state vector in the coordinate system of the camera' axis , $u(t)$ is the input to the camera plant at time t and $v(t)$ is the acceleration that drives the target at time t. The dotted variables refers to the sampled derivative of the corresponding signal.

We apply a linear state transformation by replacing the target states with the tracking error states: $x = \left[x_p, \dot{x_p}, \Delta x, \dot{\Delta x}\right]$, where $\Delta x = x_t - x_p$ and $\dot{\Delta x} = \dot{x}_t - \dot{x}_p$. The resultant model will be compactly written as

$$x(t+1) = Ax(t) + Bu(t) + Ev(t) \tag{3}$$

We define $x_{t+k|t}$ as the predicted state at time $t+k$ starting from state $x_{t|t} = x(t)$ and applying the input sequence $u_t, \ldots, u_{t+k-1}$, and the disturbances sequence $v_t, \ldots, v_{t+k-1}$ to the model (3).

The goal of the smooth pursuit controller is to maintain the target inside a predefined tracking window. The acceleration $v(t)$ that drives the target can be seen as a disturbance that competes against the controller goal. The smooth pursuit controller is designed by solving a min-max game where the disturbance is the opponent and the controller tries to minimize the worst-case performance index while satisfying the constraints for all the possible disturbance. State constraints will include the presence of the target within a pre-specified window and bounds on position, variation and acceleration of the head angles. We define the following min-max optimization problem

$$\begin{aligned} &\min_{\substack{u_t \in \mathbb{U} \\ s.t \quad x_t \in \mathbb{X} \quad \forall v_t \in \mathbb{V}}} \left\{ \|Ru_t\|_p + \max_{v_t \in \mathbb{V}} \left\{ \|Qx_{t+1|t}\|_p + \ldots \right.\right. \\ &\qquad \min_{\substack{u_{t+N-1} \in \mathbb{U} \\ s.t \quad x_{t+N-1} \in \mathbb{X} \quad \forall v_{t+N-1} \in \mathbb{V}}} \left\{ \|Ru_{t+k-1}\|_p + \right. \\ &\left.\left.\left. + \max_{v_{t+N-1} \in \mathbb{V}} \left\{ \|Qx_{t+N|t}\|_p \right\} \right\} \right\} \right\} \end{aligned} \tag{4}$$

where N is the prediction horizon, $Q \in \mathbb{R}^{n\times n}$ defines the weights on the states and $R \in \mathbb{R}^{m\times m}$ defines the weights on the control inputs. $\mathbb{X}$, $\mathbb{Y}$ and $\mathbb{U}$ are polyhedral sets that define constraints on the states and on the input. p can be either 1 or ∞ for a linear optimization problem or 2 for a quadratic optimization problem.

Solving this optimization problem (4) in real-time might be challenging. In the case of a piecewise-linear performance index ($p = 1, \infty$), an off-line solution to problem (4) can be computed [17] as a function of the current state vector x, i.e., $U^*(t) = f(x(t))$, with $U^* = \left[u_t^*, \ldots, u_{t+N-1}^*\right]$. In [17] it is shown that the off-line solution f to problem (4) is a piecewise linear function of the states. That is, the states space is partitioned into polyhedra and in each polyhedron the optimal controller is a linear function of the states. Hence, at each time step, the computational effort to find the optimal control is reduced to searching for the correct polyhedron and computing the corresponding linear function in order to obtain U^*. Once U^* is obtained, the first input signal u_t^* is applied, and the procedure is repeated over a shifted horizon.

B. Saccade Controller

When the smooth pursuit controller fails, or when one prefers to move the camera and scan another section, the saccade controller is used. In both cases we want to complete the task as fast as possible, hence the saccadic motion. For this reason, the saccade controller, in general, requires a more accurate model of the target motion. In our simplified scheme we will assume that there is no disturbance and that the target moves with a constant acceleration.

We design a minimum-time optimal saccade controller by following an approach similar to the one presented in [18]. We formulate the optimization problem

$$\begin{aligned} \min_{U=u_t,\ldots,u_{k+N-1}} J &= \sum_{k=1}^{N} \left\|Qx_{t+k|t}\right\|_p + \left\|Ru_{t+k-1}\right\|_p \\ s.t. & \\ & x_k \in \mathbb{X} \quad k = 1, \ldots, N \\ & y_k \in \mathbb{Y} \quad k = 0, \ldots, N-1 \\ & u_k \in \mathbb{U} \quad k = 0, \ldots, N-1 \\ & x_{t+N|t} \in T_{set} \end{aligned}$$

without disturbances and with a terminal state constraints T_{set}. We solve it off-line for different horizons $N \leq N_{max}$ for a given N_{max}. The off-line solution is piecewise affine [14], i.e, the set of feasible states (states that can be brought to T_{set} in N steps) is partitioned into polyhedra and in each polyhedron the optimal controller is a linear function of the states.

We make use of all the controllers with different horizons in a combined way. At each time step, for a given current state x we look for a feasible controller with the smallest horizon N. Once this has been found we implement the corresponding optimal control policy. If one is not found, the camera is moved towards the target as fast as possible until the state vector enters one of the feasible areas. It should be noted that checking whether or not the state vector is inside a feasible region of a controller is simple: the feasible region of a linear MPC controller is a polyhedron.

The feasible area of the MPC controllers will increase with N. There will be a trade off between the computational complexity and the choice of N_{max}. The use of a long horizon N_{max} allows a wider tracking capability (due to the increase of the feasible area). On the other hand a long prediction horizon requires higher real-time computational capabilities. In our case we choose N_{max} such that its feasible area is wide enough to enforce the state vector to enter it when we move the camera in maximum speed towards the target.

C. Switching between Smooth Pursuit and Saccade

Consider the case where the smooth pursuit controller is active and it becomes infeasible (i.e, the target exits the tracking window). Since the smooth controller is robust to disturbances on the target motion, then an acceleration larger than modeled is acting on the target. In order to apply the saccade controller we estimate the acceleration using the last and current states. With this estimation as target input, we implement the saccade controller as described in the previous section. During the operation of the saccade controller, no image processing is performed, and the control is based on estimated states. This is due to the fact that it is hard to find the target in the image when the relative speed between the camera and the target is high. Any attempt on image processing would fail.This also preserve the ballistic nature of saccades seen in primates.

When the saccade is completed, based on the estimated movement of the target, the system starts to look for the target in the center of the image. If the estimates were correct, the target will be found. If the target is not found, the system switches back to scanning mode.

IV. Simulations and Experiments

The Simulation and Experiments section is divided between the simulations of the high-level scanning algorithm and experiments of the smooth pursuit and saccade controllers.

A. Scan Method Simulations

1) 1D Simulations: We present first simulations for scanning a 1D circle using two cameras. The environment describes a 2D world where the system is in the middle of the circle and the cameras can be directed in angles of $0-360^o$. The circle is divided to 10 regions where each camera can scan its own side and the forward and backward regions, as detailed in Fig. 2. The horizon of the dynamic programming in (2) was set to three. A is set to represent an appearance probability of 0.3 and probability of staying in the same region of 0.63. All regions have exiting probability of 0.3 and transition probability that is divided uniformly between the neighbors. The detection rate of all regions (with the exception of the "no-target" region) is set to 1. The cost function was set to $c(i,j) = (i-j)^2+1$ if the movement was possible, and $c(i,j) = \infty$ if the movement from i to j was not possible. The cost to move to and from

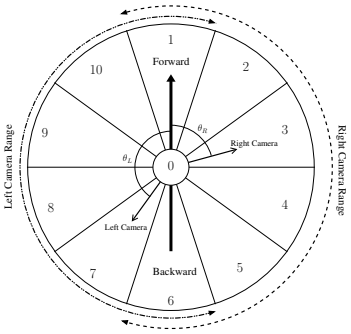


Fig. 2. 1D Simulation Setup - The world is a circle with 10 regions. Region 0 represents the system. Right camera can be directed to regions $[1-6]$. Left camera can be directed to regions $[1, 6-10]$. The wide arrow represents the forward looking direction of the system. The two thinner arrows represents possible locations of the right and left cameras.

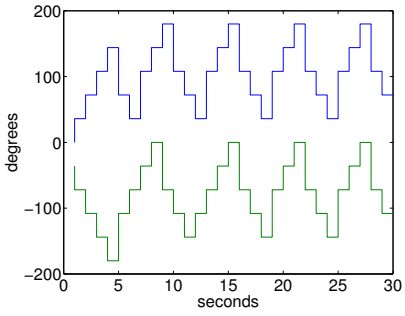
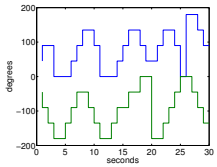


Fig. 3. Simulation Results for Search Method in 1D - the right camera has positive angles and the left camera has negative angles. The cameras start at the same location, directed forward, but develop a negative correlation quickly.

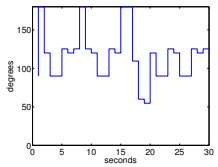
the "no-target" region was also set to infinite, which means our system will always continue to search for targets.

The results are shown in Fig. 3. The y-axis is the angle in degrees of each camera from the forward direction. The right camera has positive angles and the left camera has negative angles. It can be seen that the positions of the cameras are negatively correlated, that is, when one camera is looking forward the other is usually in a backward position and vice-versa.

2) 2D Simulations: In the case of a 3D world we have a 2D scanning area. We present how a system of two cameras that scans the upper hemisphere (in a realistic setup, the lower hemisphere is usually below the ground surface). Each camera can scan half of the hemisphere where front and back regions can be scanned by both the cameras. Simulation settings were similar to those in 1D simulation. The results for the pan axis can be seen in Fig. 4(a) (tilt axis results are not presented due to lack of space). In Fig. 4(b) the angle between the cameras during the search is presented. This angle can range between 0^o to 180^o. An angle of 0^o represents that the cameras point to the same direction, while an angle of 180^o indicates the cameras are directed to opposite directions. As can be seen most of the time the angle between the cameras is larger than 90^o which indicates a negative correlation between the cameras.



(a) Pan Axis



(b) Angle Between Cameras

Fig. 4. Simulation Results for Search Method in 2D - a) Pan axis of the two cameras. Right camera has positive angles and the left camera has negative angles. b) Angle between the cameras. The angle is usually larger than 90^o which indicates a negative correlation pattern.

B. Smooth Pursuit and Saccade Simulations

The smooth pursuit and saccade controllers were first simulated in matlab and then experimentally validated on the robotic head. Below we present the results from the experiments on the robotic head. The head is composed of two pan-tilt units model PTU-D46-17.5 of DirectedPerception. Two flea cameras of PointGrey are mounted on top of the pan-tilt units and are working with a frame rate of 30Hz. The cameras are connected to a PC computer using a firewire link and the controllers of the pan-tilt units are connected to the computer using a serial RS232 cable.

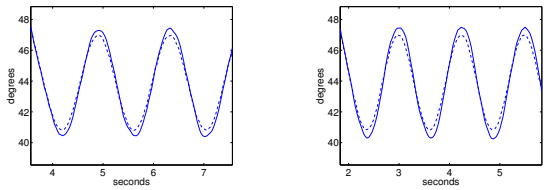
Delays is one of the main concerns when dealing with visually guided active tracking. In our case the image arrives with a delay of almost one time step. During tracking, the target is located using the normalized cross correlation on the predefined window. Once the target is located, the tracking error and tracking error speed are estimated and the control signal is computed by finding the active region of the controller based on the current state. This process is fast enough (about $1msec$ for control computation). Therefore the total delay in the system is considered to be one time step.

In this manuscript we present a simple tracking task. The system tracks successfully a simple target of a black circle. The results for tracking a sinusoidal signal with constant amplitude and different frequencies can be seen in Fig. 5. When the frequency is low, target acceleration is low and the smooth pursuit controller preforms a good tracking. When the frequency is high, the saccade controller is activated due to the high acceleration during the direction change, and drives the target back to the smooth-pursuit tracking window (Fig. 5(d)).

In Fig. 6 we present experimental results for a constant acceleration on the target. The experiments begins with low acceleration phase that brings the target to a base speed. The base speed is maintained for 0.25 seconds then the test acceleration is applied until the target exits the experiment area. As the acceleration becomes larger, the rate of saccades needed to track the target gets higher. With larger accelerations, after the first saccade, the system could no longer find the target (not presented here due to lack of space).

V. Conclusions

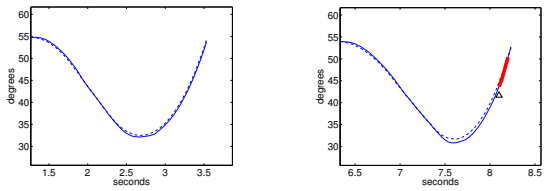
In this paper we have presented a novel method for visual scanning and autonomous target tracking by means



(a) $f = 0.7$Hz (b) $f = 0.8$Hz

(c) $f = 0.9$Hz (d) $f = 1$Hz

Fig. 5. Experimental results for sinusoidal signal - f is the frequency of the signal. The target is represented as a dashed line and the camera in solid line. Saccades are indicated with a thicker red line. The start of a saccade is marked with a triangle directly below or above the line. In sub-figure 5(d) the system uses the saccade controller to bring back the target into the tracking window.



(a) $w = 1170^o/sec^2$ (b) $w = 2050^o/sec^2$

(c) $w = 2340^o/sec^2$ (d) $w = 2920^o/sec^2$

Fig. 6. Experimental results for constant acceleration - w is the constant acceleration applied on the target (in $degrees/sec^2$). The target is represented as a dashed line and the camera in solid line. Saccades are marked as in Fig. 5. As the acceleration is getting larger more saccades are needed to keep track of the target .

of independent pan-tilt cameras which mimic the chameleon visual system. We employed a systematic and optimization-based approach to the problem, both for the the high-level scanning algorithm to the low-level tracking control design. Both algorithms use a model-based approach combining robotic head model and target model to optimally solve the scanning and the tracking problem.

The scheme is flexible and information about roads or traveling paths in the environment can be easily incorporated into the model and change the scanning pattern and tracking algorithm accordingly. Our new scanning algorithm has been simulated with a setting of two cameras that scans the hemisphere, a setup which is similar to those of chameleons. The optimal scanning pattern shows a "negative correlation" between the cameras. This scanning pattern has a distinguishable resemblance to the scanning patterns seen in chameleons in nature. This may suggests that the principles of our scanning strategy are similar to those that governed the evolution of the scanning strategy of the chameleon.

For low level tracking controller we have designed and implemented linear and robust MPC algorithms. The MPC technique has several advantages: (i) It is model based, (ii) it allows to include the system and target physical constraints in the control design, (iii) it is robust to abrupt and bounded changes in target directions. Thanks to most recent development on MPC theory we have computed and implemented the piecewise affine state feedback solution to the MPC problem. The implementation is real-time capable and experimental results have shown good performance.

REFERENCES

[1] A. Bernardino and J. Santos-Victor, "Binocular visual tracking: Integration of perception and control," *IEEE Transactions on Robotics and Automation*, vol. 15(6), pp. 1937–1958, 1999.

[2] S. Vijayakumar, J. Conradt, T. Shibata, and S. Schaal, "Overt visual attention for a humanoid robot," in *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, vol. 4, October 2001, pp. 2332–2337.

[3] M. Asada, T. Tanaka, and K. Hosoda, "Adaptive binocular visual servoing for independently moving target tracking," in *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA'00)*, vol. 3, 2000, pp. 2076–2081.

[4] W. Kirmse, R. Kirmse, and E. Milev, "Visuomotor operation in transition from object fixation to prey shooting in chameleons," *Biological Cybernetics*, vol. 71, no. 3, pp. 209–214, aug 1994.

[5] M. Ott, "Chameleons have independent eye movements but synchronise both eyes during saccadic prey tracking," *Experimental Brain Research*, vol. 139, no. 2, pp. 173–179, July 2001.

[6] O. Avni, G. Katzir, and E. Rivlin, "Eye movements in the chameleon," in Preparation.

[7] H. Lau, S. Huang, and G. Dissanayake, "Optimal search for multiple targets in a built environment," in *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, August 2005, pp. 228–233.

[8] S. M. Ross, *Introduction to Stochastic Dynamic Programming*. Orlando, FL, USA: Academic Press, Inc., 1983.

[9] E.-M. Wong, F. Bourgault, and T. Furukawa, "Multi-vehicle bayesian search for multiple lost targets," in *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA05)*, April 2005.

[10] E. Rivlin and H. Rotstein, "Control of a camera for active vision: Foveal vision, smooth tracking and saccade," *Int. J. Comput. Vision*, vol. 39, no. 2, pp. 81–96, 2000.

[11] O. Sutherland, H. Truong, S. Rougeaux, and A. Zelinsky, "Advancing active vision systems by improved design and control," in *ISER '00: Experimental Robotics VII*. London, UK: Springer-Verlag, 2001, pp. 71–80.

[12] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.

[13] A. Bemporad, F. Borrelli, and M. Morari, "The Explicit Solution of Constrained LP-Based Receding Horizon Control," in *IEEE Conference on Decision and Control*, Sydney, Australia, Dec. 2000.

[14] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[15] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox (MPT)," 2004. [Online]. Available: http://control.ee.ethz.ch/ mpt/

[16] J. N. Eagle, "An optimal search for a moving target when the search path is constrained," *Oper. Res.*, vol. 32, no. 5, pp. 1107–1115, 1984.

[17] A. Bemporad, F. Borrelli, and M. Morari, "Robust Model Predictive Control: Piecewise Linear Explicit Solution," in *European Control Conference*, Porto, Portugal, Sept. 2001, pp. 939–944.

[18] P. Grieder, P. Parrilo, and M. Morari, "Robust Receding Horizon Control - Analysis & Synthesis," in *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003, pp. 941–946.