Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

Combining peripheral and foveal humanoid vision to detect, pursue, recognize and act

Aleš Ude[1,2,*] Christopher G. Atkeson[1,3,*] Gordon Cheng[1,*]

[1]ATR Computational Neuroscience Lab.
Dept. of Hum. Robot. and Comp. Neurosc.
2-2-2 Hikaridai, Seika-cho, Soraku-gun
Kyoto 619-0288, Japan

[2]Jožef Stefan Institute, Dept.
of Automatics, Biocybernetics
and Robotics, Jamova 39
1000 Ljubljana, Slovenia

[3]Carnegie Mellon University
Robotics Institute
5000 Forbes Avenue, Pittsburgh
PA, 15213, USA

Abstract— **In this paper we present a humanoid system that can integrate information provided by its foveal and peripheral cameras. We use peripheral vision to detect and pursue objects of interest based on simple shape and color models. A detection event triggers the robot to direct its eyes towards the object, thus making a more detailed analysis of the observed objects in higher resolution foveal images feasible. The recognition is based on principal component analysis and is performed while the robot actively pursues the detected object. The classification results are inferred using information from a video stream rather than just a single image. Once the desired object is recognized, the robot reaches for it while ignoring other objects.**

I. Introduction

A robot vision system is humanoid if it (1) possesses an oculomotor system similar to human eyes and (2) if it is capable of simultaneously acquiring and processing images of varying resolution taken from two slightly different viewing directions. Approaches proposed to mimic the foveated structure of biological vision systems include the use of two cameras per eye [10], [1], [2], [6], i. e. a narrow-angle foveal camera and a wide-angle camera for peripheral vision, lenses with space-variant resolution [9], i. e. a very high definition area in the fovea and a coarse resolution in the periphery, and space-variant log-polar sensors [8]. This work takes the first approach and explores the advantage of foveated vision over current approaches which use equal resolution across the visual field. Systems with zoom lenses have some of the advantages of foveated vision, but cannot simultaneously acquire wide angle and high resolution images.

The main idea from the practical standpoint is that a humanoid robot would use peripheral vision to detect and track interesting events and objects. A detection event should trigger saccadic eye motions. After the saccade the robot would start pursuing the area of interest, thus keeping it visible in the high-resolution foveal region of the eyes, assisted by peripheral vision if foveal tracking fails. Finally, high-resolution foveal vision should provide the humanoid with a more detailed description of the detected events and objects, upon which the robot could take further actions.

Much research have been carried out to detect and track objects of interest acquired by a humanoid vision system. Cues such as color, disparities, optical flow and 2-D shape have been used to implement real-time processing of information acquired by such systems. Researchers have typically studied behaviors such as visual attention, vestibulo-ocular reflexes, saccadic movements, smooth pursuit and mimicking of human movements [9], [11], [10], [2], [4], [13]. However, peripheral vision played a dominant or exclusive role in all these systems. Even though any algorithm implemented on log-polar cameras or space variant lenses implies the processing of foveal information, researchers who developed such systems seem to have concentrated on problems that can essentially be solved by using only peripheral vision. One notable exception is the work of Breazeal et al. [2], in which foveal images were used to detect the eyes of people whose faces were first identified by peripheral vision. This is a very specialized problem and the authors heavily relied on the underlying behavioral context to simplify the computations.

Here we describe a system that makes nontrivial use of foveal vision in a task for which foveal vision is well suited, recognition. Objects of interest are first detected and tracked by our real-time visual system using information acquired by peripheral cameras [13]. A detection event triggers the robot to direct its gaze towards the candidate region and the robot starts visually pursuing the object. We do not assume that the detected objects are stationary and we account for the object motion during recognition. Since location and shape can be determined much more accurately in foveal views, we apply principal component analysis (PCA) to images acquired by foveal cameras to recognize the object. As an application domain, we consider the situation in which the person interacting with a humanoid shows an object to the robot who then reacts according to the identity of the shown object.

[*]Aleš Ude's e-mail: aude@atr.co.jp
Christopher G. Atkeson's e-mail: cga@cmu.edu
Gordon Cheng's e-mail: gordon@atr.co.jp
ATR CNS HRCN web address: http://www.cns.atr.co.jp/hrcn/

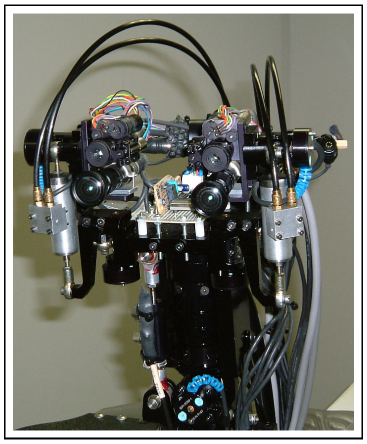

Fig. 1. DB's head with four cameras. Foveal cameras are above peripheral cameras.

For experiments we used humanoid robot DB at ATR. DB is a hydraulic anthropomorphic robot with 30 degrees of freedom (DOFs). Each eye of the robot possesses two DOFs (pan and tilt) and two color cameras: a wide angle camera (100 degrees horizontally) for peripheral vision and a narrow angle camera (24 degrees horizontally) for foveal vision. The foveal camera is located above the peripheral camera and their optical axes are roughly aligned (see Fig. 1). Vision processing was implemented on two state of the art dual processor PCs, one for foveal and one for peripheral vision.

II. Probabilistic Search and Tracking

Our object detector and tracker is implemented in a probabilistic manner. We represent the observed environment by a number of random processes (blobs). Let's denote the probability that a pixel located at $\boldsymbol{u}$ having color intensity $I_{\boldsymbol{u}}$ was generated by the process $\Theta_k,\ k = 1, \dots, K$, by $\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \Theta_k)$. We also introduce an additional outlier process Θ_0, which models the data not captured by other processes. Assuming that every pixel stems from one of the mutually exclusive processes $\Theta_k,\ k = 0, \dots, K$, we can write the probability that color $I_{\boldsymbol{u}}$ was observed at location $\boldsymbol{u}$ using the total probability law

$$\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \boldsymbol{\Theta}) = \sum_{k=0}^{K} \omega_k \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \Theta_k), \tag{1}$$

where ω_k is a prior probability to observe the process Θ_k, $\sum_{k=0}^{K} \omega_k = 1$, and $\boldsymbol{\Theta} = \{\Theta_0, \Theta_1, \dots, \Theta_K\}$. Under these assumptions, the posterior probability that pixel $\boldsymbol{u}$ stems from the l-th process is given by the Bayes' rule

$$p_{\boldsymbol{u},l} = \frac{\omega_l \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \Theta_l)}{\sum_{k=0}^{K} \omega_k \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \Theta_k)}. \tag{2}$$

Ignoring the correlation of assigning neighboring pixels to processes, the overall probability can be approximated by

$$\mathrm{P}(\boldsymbol{I}) = \mathrm{P}(\boldsymbol{I} | \boldsymbol{\Theta}) = \prod_{\boldsymbol{u}} \mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \boldsymbol{\Theta}). \tag{3}$$

At each time step, we would like to determine $(\Theta_1, \dots, \Theta_K, \omega_0, \omega_1, \dots, \omega_K)$ so that likelihood (3) is maximized. Instead of maximizing criterion (3) directly, it is often easier to minimize its negative logarithm (log-likelihood).

Before we can minimize the log-likelihood, we must decide how to model the process distributions Θ_k. Our approach uses shape and color properties to evaluate the probability that a pixel was generated by one of these processes. Assuming that these two properties are independent of each other, we have

$$\mathrm{P}(I_{\boldsymbol{u}}, \boldsymbol{u} | \Theta_l) \sim \mathrm{p}(I_{\boldsymbol{u}} | \Theta_l) \mathrm{p}(\boldsymbol{u} | \Theta_l). \tag{4}$$

In many practical cases the 2-D shape of the tracked objects is roughly ellipsoidal and can be approximated by the center of the object's image $\boldsymbol{x}_l$ and by the covariance matrix $\boldsymbol{\Sigma}_l$ of pixels contained in it. Thus, the shape part of the probability that pixel $\boldsymbol{u}$ belongs to the l-th blob can be characterized by a Gaussian distribution

$$\mathrm{p}(\boldsymbol{u} | \Theta_l) = \frac{1}{2\pi \sqrt{\det(\boldsymbol{\Sigma}_l)}} \exp(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_l) \boldsymbol{\Sigma}_l^{-1} (\boldsymbol{x} - \boldsymbol{x}_l)). \tag{5}$$

For the object's color probability, we assume that it can be modeled by a Gaussian mixture model

$$\mathrm{p}(I_{\boldsymbol{u}} | \Theta_l) = \sum_{k=1}^{K_l} \omega_{l,k} \mathrm{p}(I_{\boldsymbol{u}} | \overline{I_{l,k}}, \boldsymbol{\Gamma_{l,k}}), \tag{6}$$

where $\sum_{k=1}^{K_l} \omega_{l,k} = 1$ and

$$\mathrm{p}(I_{\boldsymbol{u}} | \overline{I_{l,k}}, \boldsymbol{\Gamma}_{l,k}) = \frac{1}{\sqrt{(2\pi)^{2 \text{ or } 3} \det(\boldsymbol{\Gamma}_{l,k})}} \cdot \exp(-\frac{1}{2} (I_{\boldsymbol{u}} - \overline{I_{l,k}}) \boldsymbol{\Gamma}_{l,k}^{-1} (I_{\boldsymbol{u}} - \overline{I_{l,k}})). \tag{7}$$

The outlier process is modeled by a fixed uniform distribution at each image pixel.

The blob and background colors are kept constant in the current version of the tracker. They are learnt off-line. Thus at each tracking step we need to maximize (3) over shape parameters $\{(\boldsymbol{x}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^{K}$ and mixture probabilities $\{\omega_k\}_{k=0}^{K}$. A good iterative approach is provided by an EM-algorithm, in which this is done by first calculating the posterior probabilities $p_{\boldsymbol{u},l}$ (given by (2), (4), (5), (7)) using the current estimate for $\{\Theta_k\}$ and $\{\omega_k\}$ (the expectation step) and then estimating the parameters $\{(\boldsymbol{x}_k, \boldsymbol{\Sigma}_k)\}$ and $\{\omega_k\}$ as if $p_{\boldsymbol{u},l}$ were constants independent of them (the maximization step). The maximization step consists of calculating the weighted mean and covariances of image pixels with $p_{\boldsymbol{u},l}$ being used as weights and reestimation of $\{\omega_k\}$. This process is repeated until convergence.

As we are interested in dynamic scenes captured by cameras in motion, it is necessary that the detection algorithm is also implemented in real-time. The ground

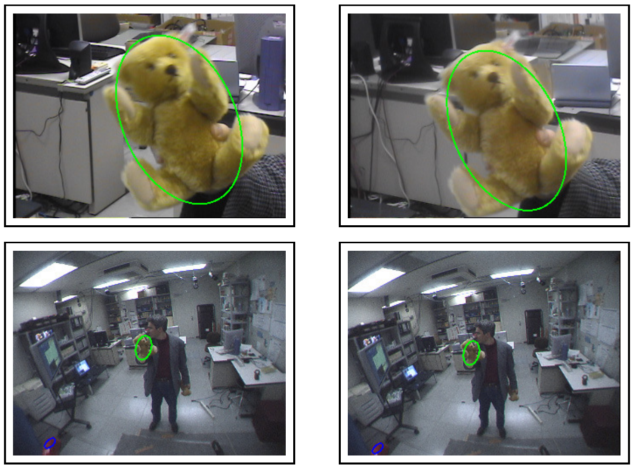

Fig. 2. Tracking in peripheral and foveal images. The green ellipses show the detected locations and shapes.

knowledge for our system is provided by color and shape probability distributions. As it is time consuming to search for ellipsoidal objects in an image, we use color only as a ground knowledge to initialize the tracker. Based on color, the probability that a pixel belongs to the l-th blob is given by (6). Since we do not have any information about the initial blob parameters, we randomly select their shapes and locations in the image. The shape parameters are varied in a controlled way so that 2-D sizes of the generated blobs remain within prespecified limits. Color probabilities (6) are then estimated at each pixel and the tracker is started if the sum of all probabilities within the window exceeds a certain threshold. These thresholds are selected automatically in our system and are different for different objects to account for various illumination properties. More details about these algorithms can be found in [14].

III. Pursuit

Once the object of interest is detected in peripheral images, DB's eyes start to pursue it. The task of the robot at this stage is to bring and keep the position of the object in both peripheral images as close as possible to the center point. This goal is achieved using a set of simplified mappings at all the controlled joints (2 in each eye, 3 in the neck, and 3 in the torso, thus altogether 11 DOFs). Although the proposed mappings are too simple for an open loop control system, they work very well in a closed loop case. Details about this approach can be found in another paper [5]. Since the foveal cameras are rigidly connected to the peripheral cameras and placed above them with roughly aligned optical axes, this method also brings the object close to the center point of foveal images. To account for the offset due to the vertical displacement between the two cameras and bring the object even closer to the center point of foveal images, we introduced a small offset in the vertical direction from the center point of peripheral images. The peripheral cameras are directed in such a way that the object is kept close to the displaced center point instead of the true center point. We determined a fixed offset in an off-line training phase and although theoretically the offset depends on the object depth, this method proved sufficient to keep the object of interest close to the center of foveal images, thus making foveal images suitable for recognition. While the robot attempts to focus on the object, the detector actively searches through the incoming foveal images, which enable us to start tracking as soon as the object appears in the fovea.

Our experiments have shown that we can estimate objects' locations and shapes much more accurately in foveal than in peripheral images (see Fig. 2), which is important for object recognition. However, it is important to keep the information from peripheral images in the loop because the object can quickly disappear from foveal images when its motion is so fast that DB's eyes cannot keep up with it.

IV. Recognition

Object recognition is an important task for humanoid robots. Early approaches to object recognition were implemented predominantly around the 3-D reconstruction paradigm of Marr and Nishihara [7], but many of the recently developed recognition systems made use of viewpoint-dependent models. Most research concentrates on object recognition from a single image, but some results pointing out the importance of temporal information have been published recently, for example [3]. One of the most popular view-based methods is principal component analysis (PCA), which is also referred to as a linear subspace method or eigenspace method. In its most basic form, this method projects the region of interest onto a lower dimensional subspace, which is determined from a number of test images. The distance of the projection vector from known sample vectors is then calculated and the object is classified based on these results. This method has been first proposed for face recognition [12], but it has found numerous other applications afterwards.

A. *Quick Overview of PCA*

The basic idea of principal component analysis is to find a set of vectors that best account for the distribution of object images within the entire image space. The number of vectors needed to represent the object images at sufficient detail is typically much smaller than the dimension of the object images represented by them.

Given a set of training images $\{\boldsymbol{I}_1, \ldots, \boldsymbol{I}_m\}$ (images are considered as column vectors with n entries in this section), the eigenspace decomposition is given by the eigenvectors of the covariance matrix $\boldsymbol{A}\boldsymbol{A}^T$, where $\boldsymbol{A} = [\boldsymbol{I}_1 - \boldsymbol{I}, \ldots, \boldsymbol{I}_m - \boldsymbol{I}]$ and $\boldsymbol{I} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{I}_i$ is the average

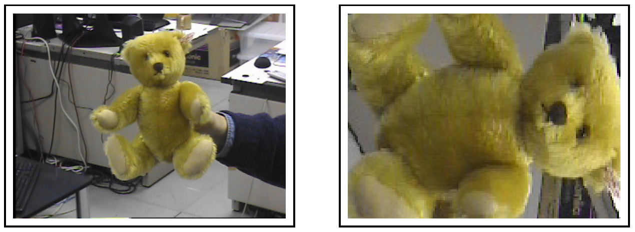

Fig. 3. The original and the warped image

object image. Since $m < n$, only the first m eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ are different from zero. It is more stable to calculate the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ by calculating the singular value decomposition of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. The columns of $\boldsymbol{U}$ associated with the nonzero singular values of $\boldsymbol{A}$ are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ associated with nonzero eigenvalues. There exist efficient and numerically stable methods to calculate the nonzero singular values of $\boldsymbol{A}$ and the associated eigenvectors and we used this approach in our experiments.

A major stumbling block towards a more widespread use of eigenspace methods is the necessity to acquire images of objects at a fixed size and orientation. Eigenspace methods have also been sensitive to the variations in the background behind the observed object, to changes in the illumination conditions, and to occlusions. Our solutions to these problems are described below.

B. Affine Warping

To achieve invariance against the changes in orientation and scale, our recognition system makes use of the results of the blob tracking system. This system determines not only the position but also the shape and orientation of the object in each image. This is due to the EM algorithm from Sec. II, which minimizes the log-likelihood with respect to the position, orientation and shape of the tracked object. This enables us to compute a mapping that transforms the ellipse approximating object shape into an ellipse of a fixed size and with both axes aligned with the coordinate axes of the new image window. The resulting mapping in homogeneous coordinates is given by the following affine transformation:

$$\boldsymbol{A}_i = \begin{bmatrix} 1 & 0 & \frac{w_x}{2} \\ 0 & 1 & \frac{w_y}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{w_x}{2a_i} & 0 & 0 \\ 0 & \frac{w_y}{2b_i} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}(\theta_i)^T & 0 \\ 0 & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & -u_i \\ 0 & 1 & -v_i \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}, \qquad (8)$$

where $\boldsymbol{u}_i = [u_i, v_i]^T$ and θ_i are the estimated position and orientation of the tracked blob at time t_i, a_i and b_i are the half lengths of its major and minor axis, and $w_x \times w_y$ is the fixed size of the window onto which we map the detected ellipse. Fig. 3 demonstrates this process.

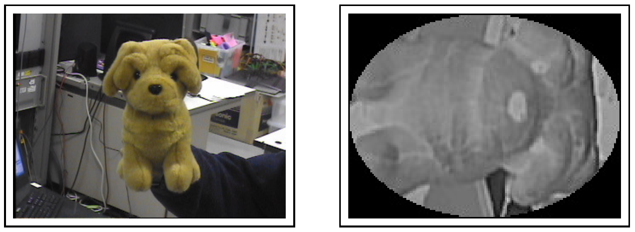

Fig. 4. The original image and the warped LoG filtered image. The black part of the warped image is not used for recognition.

To build the vectors needed for the principal component analysis, we parse through the pixels contained in the new ellipse. The generated vector has a fixed dimension because the ellipse size is fixed. All other pixels in the window should be ignored because with high probability they do not belong to the object. In this way we ensure that the percentage of background pixels that do not belong to the object but still enter the principal component analysis is low, thus ensuring that our method is not sensitive to changes in the background.

C. Robustness to Illumination Changes

It is well known that in their basic form eigenspace methods are sensitive to variations in the lighting conditions. It has been proposed that PCA can be made more robust by filtering the images with edge operators because edge maps are less sensitive to illumination changes. Eigenspace decomposition can be applied to the edge maps instead of the original images. However, edges are localized and even small errors in the calculated blob parameters could cause the recognition process to breakdown.

One way to alleviate this problem is to spread the edges with a membrane function, which is equivalent to the convolution of the edge data with a first order regularization filter [15]. We find it more natural to use the method that has already shown its effectiveness on other similar problems such as the correlation-based stereo matching. In correlation-based stereo matching, images are often first filtered with a LoG (Laplacian of Gaussian) filter:

$$\mathrm{LoG}(x, y) = \frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}. \qquad (9)$$

Unlike in [15], the smoothing operator comes before the edge operator. The parameter σ makes it possible to tune the filter. Unlike stereo matching, where a small σ is often preferable to achieve good localization, it is better to use larger σ for recognition in order to increase the spread of the edges. This makes the approach less sensitive to small errors in position/orientation estimation. Due to the smaller size of the warped images compared to the original images, it is tempting to apply the LoG filter to the warped images and not to the original images. The problem with this approach is that the spatial relationships between pixels are changed in the warped images. This would cause variations in the effect of the LoG filter based on the pixel position, thus spoiling the spatial properties of the filter. We therefore apply the LoG filter first and the affine warping afterwards.

D. Training and Recognition

Our goal is to enable DB to recognize objects shown by a person. In the training phase, the user is expected to present all relevant objects. Since it is impossible to always place the object in the same location in front of the humanoid, the user translates and rotates it around the expected placement position. This generates a rich set of viewpoints for object recognition. We set the number of captured viewpoints to 100 per object. Using the methods described in the previous sections, the object is detected and tracked in the peripheral images so that DB can direct its gaze towards it and start pursuing it. After the object is detected in foveal images, these images are LoG filtered and warped into a normalized shape of Fig. 4. The pixels within the enclosing ellipse of the warped image are used for the principal component analysis. After the set of most significant eigenvectors $\{\mathbf{\Gamma}_k\}$ is determined, we project the training images onto the eigenspace and store the projection results and the average image $\tilde{\boldsymbol{I}} = \sum_{i=1}^{m}$ for the future on-line recognition.

DB's foveal vision is provided by standard NTSC cameras. To avoid dealing with the interlacing effects, we capture foveal images at 30 Hz and at 320×240 pixels resolution. On the other hand, peripheral images used for initial detection, tracking, and pursuit are captured at 60 fields per second and at full resolution. Since the objects are shown to the humanoid from a distance that ensures that the whole object is contained in the foveal image, we cannot expect that the object will cover the whole image. We warp the object onto the window size of 160×120 pixels, which typically causes a slight subsampling compared to its original size in the foveal image. But this is still high definition compared to the size in the peripheral images. The size of our eigenvectors is thus equal to $\pi * 160/2 * 120/2 \approx 15079$. Except for the singular value decomposition needed to calculate the eigenvectors, all other operations are done in real-time by our system. This is obviously not a limitation because there is no reason to insist on real-time computation of the singular value decomposition in the training phase.

Many operations carried out in the recognition phase are the same as in the tracking phase. This includes the detection and tracking in the peripheral images, pursuit of the object with DB's eyes, and detection, tracking, LoG filtering, and affine warping in foveal images. The warped and LoG filtered foveal image is then projected onto the previously computed eigenvectors

$$\omega_k = \mathbf{\Gamma}_k^T(\boldsymbol{I} - \tilde{\boldsymbol{I}}). \tag{10}$$

The resulting projection $\boldsymbol{\omega}$ is compared to the prototypes $\mathbf{\Omega}_i$ generated in the training phase and the solution is given by the class of the closest prototype

$$\arg\min_i \|\boldsymbol{\omega} - \mathbf{\Omega}_i\|. \tag{11}$$

We made several improvements to increase the performance of the recognition system. Firstly, the classification is taken as valid only if the same object has been recognized in left and right foveal image. Secondly, we exploit the dynamic nature of our system and run the recognition process on a time sequence of images. The object is deemed recognized only if the identity of the object does not change over a heuristically chosen time interval. In our experiments we typically used 3 images per second to allow for some interframe motion and waited for two seconds before accepting the recognition result.

For every single image, the above approach selects one of the prototype objects. This is not desirable if an object that does not belong to the database is shown to the robot. But since eigenspace methods enable us to reconstruct the image of the observed object, we can compute the distance between the original image and the reconstructed image [12]

$$\epsilon = \|\boldsymbol{I} - (\tilde{\boldsymbol{I}} + \sum_k \omega_k \mathbf{\Gamma}_k)\|. \tag{12}$$

If this distance exceeds a certain threshold, we consider that the object does not belong to the database and is thus classified as unknown. Unfortunately, the quality of reconstruction can be quite different from object to object and it is difficult to select one threshold for all objects. We therefore introduced an additional (but optional) training phase to select proper thresholds for all objects. In this training phase, the various objects of known identity are shown to the robot and we measure how the reconstruction method performs, i. e. we sample the reconstruction error as given by (12). We then set the threshold ϕ_i for the reconstruction error of object i to

$$\phi_i = (1-\lambda)\frac{1}{n_i}\sum_{k=1}^{n_i}\epsilon_k^i + \lambda\frac{1}{n_0}\sum_{k=1}^{n_0}\epsilon_k^0,\ 0 \le \lambda \le 1, \tag{13}$$

where n_i is the number of occurrences of object i and n_0 is the number of occurrences of objects not belonging to the database. This approach can prevent the system from recognizing unknown objects.

V. Results and Conclusions

We carried out several experiments to test our approach. In all experiments we used 100 training images per object and computed principal component analysis on the combined set of training images (300).

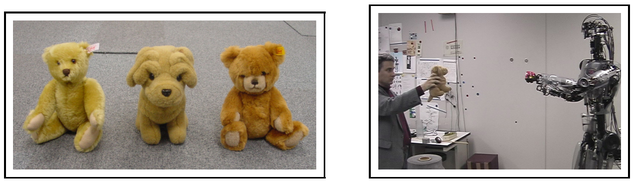

Fig. 5. Objects used in the experiments and DB indicating that he recognized the dog by pointing towards it

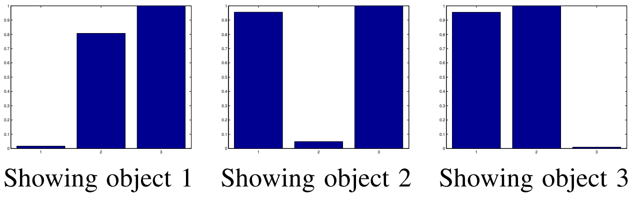

Fig. 6. Discrimination of objects from Fig. 5 shown to the robot with slight movement (see text for explanation)

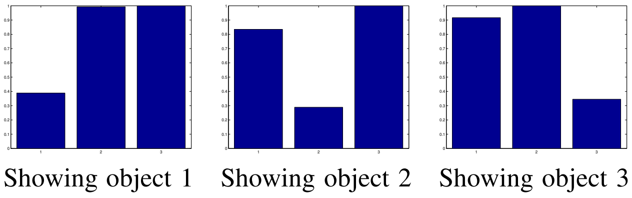

Fig. 7. Discrimination of objects from Fig. 5 shown to the robot with large movement (see text for explanation)

Fig. 6 and 7 show the performance of our method with respect to the amount of object motion. Clearly, it is much more difficult to recognize an object when it is moved quickly and snapshots are taken from various locations in space. In each experiment, one of the objects was moved in front of the robot and 200 snapshots of the moving object were taken. The bar graphs show the normalized average distance of the best prototype for each class (object) from the object image projected onto the eigenspace using (11). A lower score is better. The prototypes associated with the shown object always achieved the lowest score, but as expected the difference decreased with larger movements.

In our interactive experiment, DB was required to distinguish between teddy bears and the toy dog (see Fig. 5). All toys have similar color and a common color model was learnt to detect and track all objects. To indicate successful recognition, DB was asked to point towards the object when recognizing the dog and to do nothing when seeing the teddy bears. The detector and tracker successfully dealt with objects appearing and disappearing from the view and the foveal cameras were able to lock on the shown object (see Fig. 2). There were some false classifications in our interactive experiments when the object was very close to DB's eyes and became too big for the foveal images. However, our dynamic approach uses several snapshots for final identification and was always able to filter out wrong identifications and DB always pointed towards the dog and ignored the teddy bears (see video). All calculations were done in real-time, i. e. at 60 Hz for detection and pursuit and at 30 Hz for recognition.

We conclude that the proposed approach is successful at locating, pursuing and recognizing objects in motion. We have demonstrated for the first time how to integrate peripheral and foveal vision on a humanoid robot to solve these problems in real-time.

Acknowledgement: Atkeson acknowledges the support of the US National Science Foundation (Award 0242482).

VI. REFERENCES

[1] C. G. Atkeson, J. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, and M. Kawato. Using humanoid robots to study human behavior. *IEEE Intelligent Systems*, 15(4):46–56, July/August 2000.

[2] C. Breazeal, A. Edsinger, P. Fitzpatrick, and B. Scassellati. Social constraints on animate vision. *IEEE Trans. Systems, Man, and Cybernetics*, 31(5), July/August 2001.

[3] H. H. Bülthoff, C. Wallraven, and A. Graf. View-based dynamic object recognition based on human perception. In *Proc. Int. Conf. Pattern Recognition, vol. III*, pages 768 – 776, Québec City, Canada, August 2002.

[4] G. Cheng, A. Nagakubo, and Y. Kuniyoshi. Continuous humanoid interaction: An integrated perspective – gaining adaptivity, redundancy, flexibility – in one. *Robotics and Autonomous Systems*, 37:161–183, 2001.

[5] C. Gaskett and G. Cheng. From closed-loop to open-loop visual servo control for humanoid robots. 2003 (pending).

[6] H. Kozima and H. Yano. A robot that learns to communicate with human caregivers. In *Proc. Int. Workshop on Epigenetic Robotics*, Lund, Sweden, 2001.

[7] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. R. Soc. of London, B*, 200:269–294, 1978.

[8] G. Metta, F. Panerai, R. Manzotti, and G. Sandini. Babybot: an artificial developing robotic agent. In *Proc. Sixth Int. Conf. on the Simulation of Adaptive Behaviors (SAB 2000)*, Paris, France, September 2000.

[9] S. Rogeaux and Y. Kuniyoshi. Robust tracking by a humanoid vision system. In *Proc. IAPR First Int. Workshop on Humanoid and Friendly Robotics*, Tsukuba, Japan, 1998.

[10] B. Scassellati. Eye finding via face detection for a foveated, active vision system. In *Proc. Fifteenth Nat. Conf. Artifficial Intelligence (AAAI '98)*, pages 969–976, Madison, Wisconsin, 1998.

[11] T. Shibata, S. Vijayakumar, J. Conradt, and S. Schaal. Biomimetic oculomotor control. *Adaptive Behavior*, 9(3/4):189–208, 2001.

[12] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[13] A. Ude and C. G. Atkeson. Real-time visual system for interaction with a humanoid robot. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 746–751, Maui, Hawaii, October/November 2001.

[14] A. Ude and C. G. Atkeson. Probabilistic detection and tracking at high frame rates using affine warping. In *Proc. Int. Conf. Pattern Recognition, vol. II*, pages 6–9, Québec City, Canada, August 2002.

[15] A. Yilmaz and M. Gökmen. Eigenhill vs. eigenface and eigenedge. *Pattern Recognition*, 34(1), 2001.