Proceedings of IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids 2004)
November 10–12, 2004, Los Angeles, California, USA

OBJECT RECOGNITION ON HUMANOIDS WITH FOVEATED VISION

ALEŠ UDE[1,2,3] and GORDON CHENG[1,2]

[1] *ICORP Computational Brain Project, Japan Science and Technology Agency*
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

[2] *ATR Computational Neuroscience Laboratories*
Department of Humanoid Robotics and Computational Neuroscience
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
aude@atr.jp *gordon@atr.jp*

[3] *Jožef Stefan Institute, Department of Automatics, Biocybernetics and Robotics*
Jamova 39, 1000 Ljubljana, Slovenia

Object recognition requires a robot to perform a number of nontrivial tasks such as finding objects of interest, directing its eyes towards the objects, pursuing them, and identifying the objects once they appear in the fovea. In this paper we describe a system that makes use of foveated vision to solve the problem of object recognition on a humanoid robot. The system employs a biologically motivated object representation scheme based on Gabor kernel functions to represent multiple views of objects. We demonstrate how to utilize support vector machines to identify known objects in foveal images using this representation. A mechanism for visual search is integrated into the system to find a salient region and to place an object of interest in the field of view of foveal cameras. The framework also includes a control scheme for eye movements, which are directed using the results of attentive processing in peripheral images.

Keywords: Humanoid vision; foveated vision; object recognition.

1. Introduction

A robot vision system is humanoid if it firstly possesses an oculomotor system similar to human eyes, and secondly if it is capable of simultaneously acquiring and processing images of varying resolution taken from two slightly different views. Approaches proposed to mimic the foveated structure of biological vision systems include the use of two cameras per eye[2,12,17,18], i. e. a narrow-angle foveal camera and a wide-angle camera for peripheral vision; lenses with space-variant resolution[16], i. e. a very high definition area in the fovea and a coarse resolution in the periphery; and space-variant log-polar sensors[14]. Our work follows the first approach[1] (see also Fig. 1).

Researchers working on humanoids equipped with foveated vision typically studied behaviors such as visual attention, vestibulo-ocular reflexes, saccadic move-

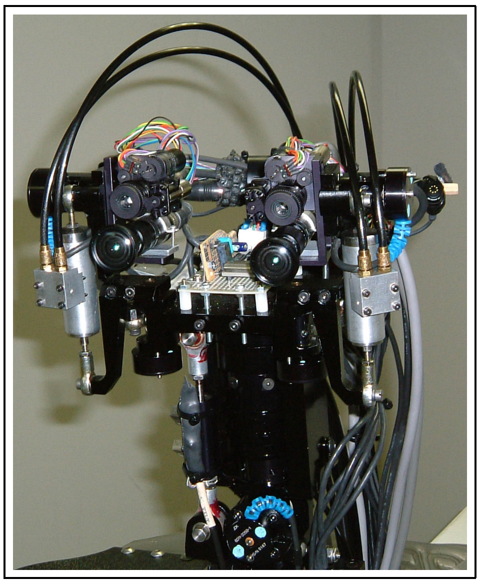

Fig. 1. DB's head. Foveal cameras are above peripheral cameras.

ments, smooth pursuit and mimicking of human movements[16,18,17,2,5,22]. However, peripheral vision played a dominant or exclusive role in all these systems. Even though algorithms implemented on log-polar cameras or space variant lenses imply the processing of foveal images, 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 Scassellati[17], 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 visual processing.

We have recently presented a system that makes nontrivial use of foveal vision in a task for which foveal vision is well suited, recognition[23]. We utilize foveation as follows: our humanoid robot DB relies on peripheral vision to search for interesting areas in visual scenes. The attention system reports about salient regions and triggers saccadic eye movements. After saccade the robot starts pursuing the area of interest, thus keeping it visible in the high-resolution foveal region of the eyes. Finally, high-resolution foveal vision provides the humanoid with a more detailed description of the detected events and objects, upon which the robot can take further actions.

Our initial system employed LoG (Laplacian of the Gaussian) filters at a single, manually selected scale and principal component analysis (PCA) to represent objects. The nearest neighbor approach was used to identify known objects in visual scenes. This system was used successfully in interactive experiments with DB.

To improve its performance, we explored some alternative object representation schemes and classification algorithms. We experimented with representations based on Gabor jets[25], which are constructed by convolving an image with a number of Gabor filters at different scales. Gabor filters are prominent in machine vision because they achieve the best possible joint resolution in 2-D visual space and 2-D Fourier domain[6]. They are often used for feature detection. It is also interesting for humanoid robot vision that receptive field profiles of simple cells in the primary visual cortex of primates can be approximated by 2-D Gabor functions[6]. In this paper we show how support vector machines (SVMs) can be used to classify objects represented by Gabor jets. We also present some more details of a complete system starting from visual search over the generation of eye movements to object recognition.

2. Visual Search and Tracking

To classify an object in the visual scene, the robot must first identify the object's location in the image. Visual search and the role of attention in search has been much discussed in recent literature[26]. Treisman's feature integration theory is one of the most thoroughly studied approaches and resulted – somewhat modified – in several technical implementations, e. g. Itti et al.[10], including some implementations on humanoid robots[2,18]. These implementations are mainly concerned with bottom-up, data-driven processing directed towards the generation of saliency maps. However, many theories of visual search, e. g. guided search, suggest that there are several ways for preattentive processing to guide the deployment of attention[26]. Besides the bottom-up guidance towards salient regions, there is also a top-down guidance based on the needs of the searcher. Here we briefly present our implementation of the top-down search process that bypasses the saliency maps.

2.1. *Top-down Guidance*

The theory that human visual search always relies on accumulating information about objects over time has been recently disputed by Horowitz and Wolfe[9]. The authors showed, in a number of behavioral experiments, that search efficiency is not impaired if the scene is continuously shuffled while the observer is trying to search through it. They concluded that during a visual search episode, no memory is devoted to rejected distractors. Although they acknowledge the existence of inhibition of return (IOR), they argue that IOR has only a very short duration (last 4-6 attended items).

These findings suggest that it does not make sense to implement complex search schemes when a humanoid robot looks for a particular feature or object in an unknown, dynamic and cluttered environment. It is a daunting task to keep and update all the attended positions in memory when the robot moves and the scene changes. We therefore decided to implement the top-down search, which needs to be executed in real-time, in a purely random fashion. Such an approach does not exclude the existence of strategically planned searches such as for example limiting the search to a particular area in the image, but we assume that such searches are not planned in real-time and are based on higher-level knowledge of the scene.

We assign to each object in our object library a number of signal detectors describing object features such as for example color. It has been argued recently that many aspects of visual search can be explained by the signal detection theory[24]. The signal detectors do not need to be tuned to one object only, e. g. objects having the same color are associated with the same detector. Signal detectors can describe the properties of more than one feature and can thus deal with compounded features. We assume that 2-D shapes of objects of interest can be approximated by the second order statistics of pixels contained in their projected images. Since we do not have any information about the location and identity of the objects (apart from that we are looking for objects from the library), we start by randomly selecting the object size, shape and location in the image. If the number of feature detectors is not too large, all of them are evaluated at each location, otherwise we randomly select some of them for evaluation so that real-time processing is still possible. The group of objects associated with the detector is assumed detected if the signal detector exceeds a threshold that is learned in the training phase. The shape parameters are varied in a controlled way so that 2-D sizes of the generated object hypotheses remain within prespecified limits. This implements search at multiple resolutions. To ensure that the processing time is constant, which is necessary to guarantee real-time operation of the system, we map the randomly generated object location onto a window of fixed size. We also implemented a short term inhibition of return by rejecting all newly generated locations that are located within any of the enclosing ellipses of the last 5 randomly selected object hypotheses. A new test location is generated in this case.

2.2. *Tracking*

An object detection event triggers our tracker, which is implemented in a probabilistic manner. It uses shape and color cues to track objects of interest at video rate (60 Hz). What matters for our implementation of object recognition on a humanoid robot is that it can estimate position, orientation, and scale of tracked objects. Details of the algorithm are given elsewhere[21].

3. Generation of Eye Movements and Smooth Pursuit

The main task of the control system is to place a salient region in the field of view of both foveal cameras so that further analysis and eventually object recognition can be carried out. Although the focus of the task is to bring the object into the center of the fovea, the control system uses the view of the object from peripheral cameras as the basis for control. Motion based on information acquired from peripheral images is more reliable because objects can easily be lost from the view of the foveal cameras. Since the foveal cameras are rigidly connected to the peripheral

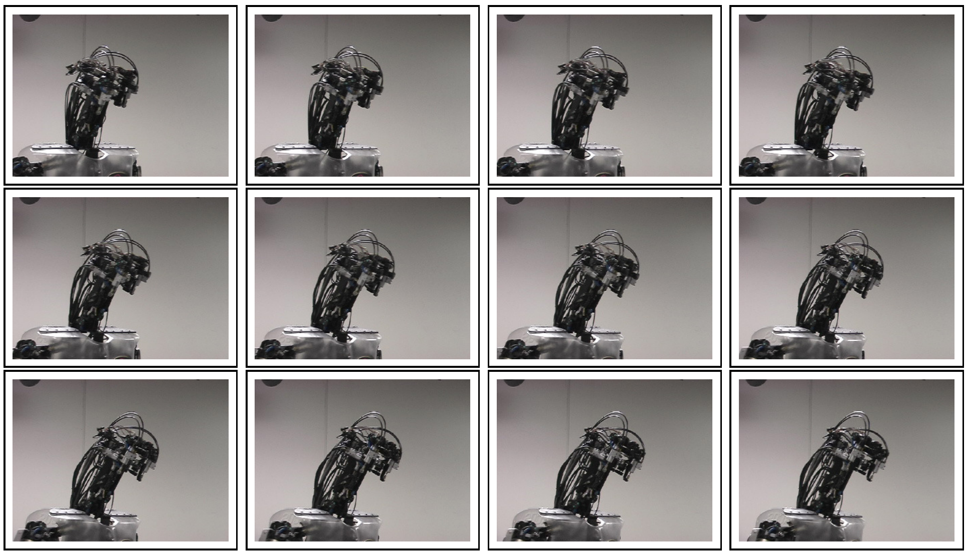

Fig. 2. DB pursuing the toy dog. These images illustrate that the head (3 DOFs) and the torso (3 DOFs) of the robot also move to support the pan and tilt motion of the eyes (4 DOFs), which are primarily responsible for getting the object in the fovea.

cameras and mounted above them with roughly aligned optical axes, the object can be placed in the foveal images by bringing it into a position slightly displaced from the center of peripheral images.

We determined a fixed offset in an off-line training phase and although theoretically the offset depends on the object's 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 (see Fig. 3). While the robot attempts to focus on the object, the detector actively searches through the incoming foveal images, which makes it possible to determine the object's position in foveal images immediately after the object appears in the fovea.

The robot's primary mechanism for maintaining the view of the object of interest is the eye movement: the control system continuously alters the pan and tilt of each eye to keep the object near the center of the corresponding view. Our robot DB has altogether 30 degrees of freedom and other joints can support the eyes to keep the object in the center of the fovea. We implemented supportive head and torso movements and thus use 10 degrees of freedom (4 on the eyes + 3 on the head + 3 on the torso) to maintain the view of the object[7] (see also Fig. 2). The task of the robot head and torso DoFs is to assist the eyes by increasing the viewable area. The complete control system is implemented as a network of PD controllers expressing the assistive relationships. The proposed controllers do not rely on a standard model of robot kinematics and are too simple for an open loop control system. However, they work very well in a closed loop case. Details about this approach can be found in another paper[7].

Our experiments proved that we can estimate objects' locations and shapes much more accurately in foveal than in peripheral images (see Fig. 3), which is essential for object recognition.

4. Object Representation

Early approaches to object recognition in static images were implemented predominantly around the 3-D reconstruction paradigm of Marr[13], but many of the recent recognition systems make use of viewpoint-dependent models. View-based strategies are receiving increasing attention because it has been recognized that 3-D reconstruction is difficult in practice and also because of some psychophysical evidence for these techniques[19].

In a view-based approach, each object is represented by a number of images taken from different viewpoints. These model images are compared to the test images acquired by the robot. However, since a humanoid robot and objects can move in space, objects appear in images at different positions, orientations and scales. To recognize objects, we must first extract the portions of images that contain objects of interest. The extracted images must then be normalized to a fixed size, which enables us to compare the acquired images with the model images using methods like principal component analysis and support vector machines.

4.1. *Normalization through Affine Warping*

A common feature of many blob trackers - including the one implemented by us[21], which was used in the experiments described in this paper - is that they approximate the shape of tracked objects by the second order statistics of pixels that are

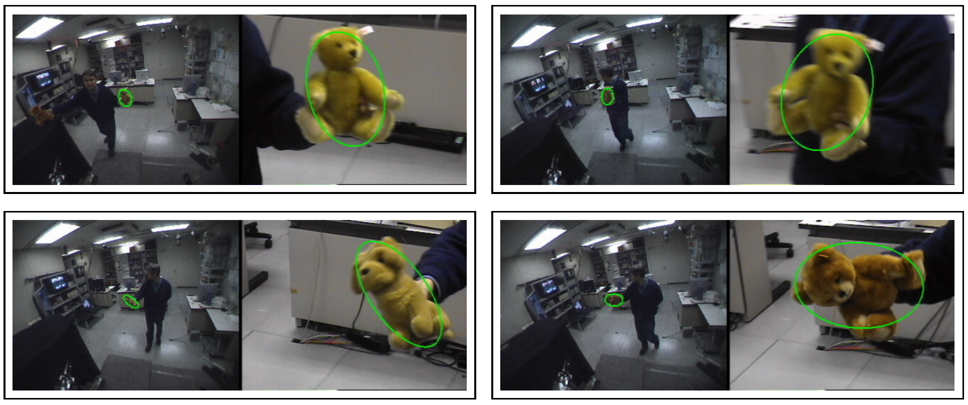

Fig. 3. Tracking in peripheral and foveal images. The ellipses show the detected locations and shapes. It is apparent that foveal images are much better suited for recognition than peripheral images.

probabilistically classified as "blob pixels". By computing the eigenvalue decomposition of the associated covariance matrices, we can estimate the extent of the blobs along their major and minor axes, i. e. calculate the location and shape of ellipses enclosing the blob pixels. As the lengths of both axes can differ significantly, it makes sense to normalize each image along the principal axis directions instead of image coordinate axes and to apply a different scaling factor along each of these directions, taking into account the length of the corresponding axis.

By aligning the object's axes with the coordinate axes, we also achieve invariance against planar rotations. This reduces the number of views that need to be stored to represent an object.

Normalization along the principal axes can be implemented by applying the following transformations: (1) translate the blob so that its center is aligned with the origin of the image, (2) rotate the blob so that its principal directions are aligned with the coordinate axes, (3) scale the blob so that its major and minor axis are as long as the sides of a predefined window, (4) translate the blob so that its center is aligned with the center of the new window. The resulting mapping in homogeneous coordinates is given by the following affine transformation:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & \frac{w_x}{2} \\ 0 & 1 & \frac{w_y}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{w_x}{2a} & 0 & 0 \\ 0 & \frac{w_y}{2b} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}(\theta)^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -u \\ 0 & 1 & -v \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}, \qquad (1)$$

where $\boldsymbol{u} = [u, v]^T$ and θ are the position and orientation of the blob, a and b are the half lengths of its major and minor axis, and w_x and w_y are the predefined width and height of the window onto which we map the window containing the blob.

The process of geometrically transforming the input image by the affine mapping given in Eq. (1) is know as *affine warping*. Since matrix $\boldsymbol{A}$ is invertible, we implemented affine warping by parsing through the pixels of the output window, which is smaller than the input window, and by applying the inverse mapping $\boldsymbol{A}^{-1}$ to each of the pixels in this window. We estimate the associated color intensities at these positions either by a nearest neighbor or cubic interpolation.

4.2. *Gabor Jets*

Early view-based approaches used raw grayscale images as input to the selected classifier, e. g. principal component analysis[20] or support vector machines[15]. Such an approach, however, cannot deal with illumination changes. Some recent recognition systems therefore apply a bank of illumination-insensitive filters to the original images before starting the recognition process. In this way robustness against varying brightness conditions can be improved. We follow the approach of Wiskott et al.[25], who applied a bank of Gabor filters to the incoming images, both in training and in recognition phase. Gabor filters are known to be good edge detectors and are therefore robust against varying brightness. They have limited support both in space and frequency domain, which results in a certain amount of robustness against translation, distortion, rotation, and scaling.

Complex Gabor kernels are defined by

$$\boldsymbol{\Phi}_{\mu,\nu}(\boldsymbol{x}) = \frac{\|\boldsymbol{k}_{\mu,\nu}\|^2}{\sigma^2} \cdot \exp\left(-\frac{\|\boldsymbol{k}_{\mu,\nu}\|^2\|\boldsymbol{x}\|^2}{2\sigma^2}\right) \cdot \left(\exp\left(i\boldsymbol{k}_{\mu,\nu}^T\boldsymbol{x}\right) - \exp\left(-\frac{\sigma^2}{2}\right)\right), \quad (2)$$

where $\boldsymbol{k}_{\mu,\nu} = k_\nu[\cos(\phi_\mu),\ sin(\phi_\mu)]^T$. Gabor jet at pixel $\boldsymbol{x}$ is defined as a set of complex coefficients $\{J_j^{\boldsymbol{x}}\}$ obtained by convolving the image with a number of Gabor kernels at this pixel. Gabor kernels need to be selected so that they sample a number of different wavelengths k_ν and orientations ϕ_μ. Wiskott et al.[25] proposed to use $k_\nu = 2^{-\frac{\nu+2}{2}}$, $\nu = 0, \ldots, 4$, and $\phi_\mu = \mu\frac{\pi}{8}$, $\mu = 0, \ldots, 7$. They proposed to use a similarity function based on scalar products between normalized magnitudes of Gabor jets, i. e. $\{a_j^{\boldsymbol{x}}/\|\boldsymbol{a}^{\boldsymbol{x}}\|\}$, where $a_j^{\boldsymbol{x}}$ is the magnitude of the corresponding complex coefficient $J_j^{\boldsymbol{x}}$, $\boldsymbol{a}^{\boldsymbol{x}} = [a_1^{\boldsymbol{x}}, \ldots, a_n^{\boldsymbol{x}}]^T$, and n is the jet dimension ($n = 40$ in case of Wiskott et al.[25] jets). This collection of normalized magnitudes is called a Gabor jet at pixel $\boldsymbol{x}$ in the rest of this paper.

The calculation of Gabor jets is computationally expensive because the underlying Gabor kernel functions have large support. It is therefore advantageous to compute the convolutions $\boldsymbol{I} * \boldsymbol{\Phi}_{\mu,\nu}$ with the help of Fast Fourier Transform

$$\boldsymbol{I} * \Phi_{\mu,\nu} = \mathcal{F}^{-1}\left(\mathcal{F}(\boldsymbol{I}) \cdot \mathcal{F}(\boldsymbol{\Phi}_{\mu,\nu})\right) \quad (3)$$

It is not necessary to use Gabor jets at every image pixel as an input to the recognition system. Ideally, one would calculate the jets only at important local features. We did not attempt to extract local features because it is often difficult to extract them in a stable manner. Rather we decided to calculate the jets on a regular grid of pixels. We took pixels $\boldsymbol{x}$ appearing in Eq. (2) from a regular grid with the spacing of 4 pixels. Normalized jets $\{a_j^{\boldsymbol{x}}/\|\boldsymbol{a}^{\boldsymbol{x}}\|\}_{j=1}^n$ calculated on this grid and belonging to the ellipse enclosing the object like in Fig. 4 were utilized to build feature vectors that can be used for classification.

It is important to note that we first scale the object images to a fixed size and then apply Gabor filters. In this way we ensure that the size of local structure in the acquired images does not change and consequently we do not need to change the frequencies k_ν of the applied filters.

5. Recognition with Support Vector Machines

Support vector machines (SVMs) are a relatively new classification system rooted in the statistical learning theory[4]. They are considered as state of the art classifiers because they deliver high performance in real-world applications. They have been applied to several problems in computer vision including face recognition[8] and also to more general 3-D object recognition problems[15].

First we consider the problem when we need to distinguish between two objects or between one object and all other objects in the database. A linear SVM for two-class classification problems is given by an optimal hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$, which

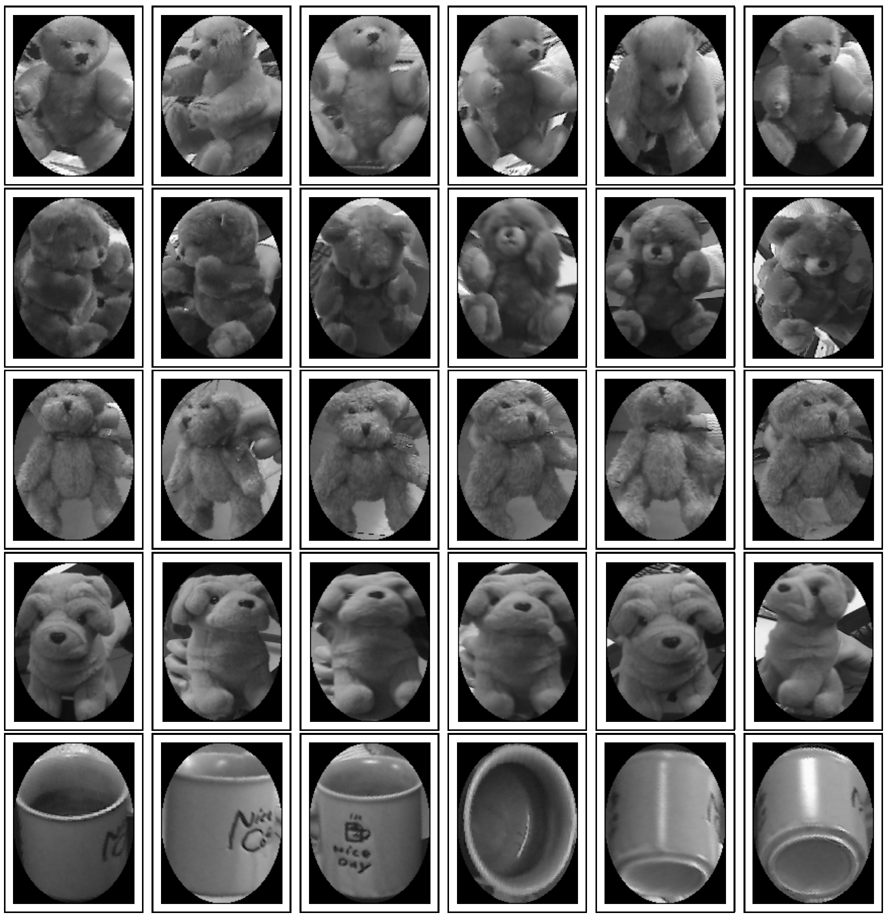

Fig. 4. Training images of the five objects used in statistical experiments. Scaling and planar rotations are accounted for using the results of visual tracking and affine warping. To take care of rotations in depth, we must collect a sufficient amount of typical viewpoints.

separates data points belonging to both classes. The decision function is given by

$$\mathrm{f}(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x} + b). \tag{4}$$

The hyperplane is optimal in the sense that it separates the largest fraction of points from each class, while maximizing the distance from either class to the hyperplane. It can be computed by minimizing

$$\frac{1}{2}\|\boldsymbol{w}\|^2 \tag{5}$$

subject to

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1,\ i = 1, \ldots, N, \tag{6}$$

where y_i is respectively equal to 1 and -1 for positive and negative examples, $\boldsymbol{x}_i$ are the training data (in our case vectors combining the Gabor jets), and N is the number of training images. This is a quadratic programming problem and there exist algorithms that can solve it efficiently[4].

It can be shown that $\boldsymbol{w}$ is given as a linear combination of data points $\boldsymbol{x}_i$, i. e. $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$, $0 \leq \alpha_i$. Most of the coefficients α_i take zero values. Feature vectors corresponding to nonzero α_i are called support vectors. This formulation enables us to rewrite the decision function (4) in terms of dot products between training feature vectors $\boldsymbol{x}_i$ and the feature vector $\boldsymbol{x}$ generated by a test image

$$\mathrm{f}(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b\right). \tag{7}$$

Studying the quadratic programming problem (5) and (6), one can note that it can be solved in such a way that data points $\boldsymbol{x}_i$ appear only in dot products $\boldsymbol{x}_i^T \boldsymbol{x}_j$. This allows the extension of support vector training to the case of nonlinear separating surfaces. The key idea here is to map the original feature points into a higher dimensional space $\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x})$, where the data can be separated by a linear hyperplane. It turns out that if $\boldsymbol{\Phi}$ fulfils certain conditions, then there exists kernel function K so that $\boldsymbol{\Phi}(\boldsymbol{x}_i)^T \boldsymbol{\Phi}(\boldsymbol{x}_j = \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In this case the decision function becomes

$$\mathrm{f}(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \tag{8}$$

Since feature vectors $\boldsymbol{x}$ only appear in dot products both in the training algorithm and in the decision function (4), we only need to use K and never need to know the function $\boldsymbol{\Phi}$. Hence nonlinear SVMs can be calculated using exactly the same basic algorithm, the only difference being that we replace the dot products $\boldsymbol{x}_i^T \boldsymbol{x}_j$ with $\mathrm{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

An important family of kernel functions is given by Gaussian radial basis functions (RBF)

$$\mathrm{K}(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right). \tag{9}$$

In our experiments we used linear SVMs and RBF-based SVMs. The support vector machines were trained using the SVMlight software[11].

We implemented two classification schemes: one versus the rest, where the goal is to determine whether a particular object is in the scene or not, and the tree structure scheme initially proposed by Guo et al.[8], where the goal is to identify an object when multiple choices are allowed (multi-class classification problem). In this second case each support vector machine is trained to distinguish between two objects and the final result is obtained by elimination.

The multi-class recognition problem requires the calculation of many dot products between the training feature vectors and the test feature vector. In this case it is advantageous to reduce the dimensionality of Gabor jet representation. This can be accomplished by applying PCA (or ICA) to the training jets. Projections of the training jets onto the calculated principal components can then be used as input for SVM training.

We also 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 certain period of time. This is based on the assumption that correct classifications are stable whereas misclassifications are not and change as the viewpoint changes. In our experiments we typically used 3 images per second to allow for some interframe motion and waited until the recognition result was stable for at least two seconds before accepting it. Otherwise the robot continues to observe the object until the classification result becomes stable or the person interacting with the robot removes it.

6. Experiments and Conclusions

Experimental results are shown in Tab. 1 - 4. The task here was to determine whether an object was in the scene or not using only one support vector machine (one versus the rest scheme). For each of the five objects in the database (see Fig. 4), the images of the object under consideration were taken as positive examples and the images of all other objects were taken as negative examples. Training images were collected while a user moved the objects in front of the robot. Gabor jets were used as input to the SVM training. We captured the incoming images at 320×240 pixels and the portion of the image containing the tracked object (see Fig. 3) was mapped onto the window with 120×160 pixels (see Fig. 4). Our foveal cameras can capture 30 images per second, but we used only three of them to prevent the collection of too many similar training images. We captured 200 training images per object, hence the data collection process took a bit longer than one minute per object. Each support vector machine was trained using 200 positive examples and 800 negative examples. The performance of the system degraded when we used less training images or a lower resolution target window. For testing we used images taken from a different video acquired by the robot's foveal cameras.

We tested the classification of Gabor jet vectors - both in raw form or processed by principal component analysis - using linear SVMs and SVMs based on radial basis functions. Tab. 1 - 4 show that there was no significant difference between these four approaches. PCA was able to retain the essential properties of feature vectors, but support vector machines were able to classify the feature vectors with (Tab. 1 and 3) or without PCA (Tab. 2 and 4). The method of choice is thus more dependent on other factors such as computing time. Classifiers with PCA require more training time, but they return a classification result faster because feature vectors are smaller in this case and dot products between them can be calculated quicker. This is especially important when using the binary tree structure designed to solve multi-class problems, which require the calculation of many dot products between feature vectors. It, however, remains to be shown if PCA would perform well on large databases. In addition, our initial results show that SVMs without

Table 1. Misclassifications (Linear SVMs, PCA)

	false neg.	false pos.
teddy bear 1	4.5 %	0.5 %
teddy bear 2	7.8 %	0.3 %
teddy bear 3	1.4 %	1.9 %
toy dog	3.9 %	2.7 %
coffee mug	2.1 %	0.7 %
combined	4.0 %	1.2 %

Table 2. Misclassifications (Linear SVMs)

	false neg.	false pos.
teddy bear 1	8.5 %	0.1 %
teddy bear 2	6.4 %	0.3 %
teddy bear 3	2.3 %	0 %
toy dog	3.1 %	0.6 %
coffee mug	2.3 %	0 %
combined	4.5 %	0.2 %

Table 3. Misclassifications (RBF SVMs, PCA)

	false neg.	false pos.
teddy bear 1	7.4 %	0 %
teddy bear 2	8.0 %	0.1 %
teddy bear 3	3.7 %	0 %
toy dog	2.7 %	0.4 %
coffee mug	1.9 %	0 %
combined	4.7 %	0.1 %

Table 4. Misclassifications (RBF SVMs)

	false neg.	false pos.
teddy bear 1	5.8 %	0.2 %
teddy bear 2	6.6 %	0.3 %
teddy bear 3	0.2 %	1.4 %
toy dog	2.9 %	0.4 %
coffee mug	2.5 %	0.1 %
combined	3.7 %	0.5 %

PCA perform better than SVMs with PCA when we reduce the number of training views per object. Currently, our method of choice is RBF-based SVM without PCA.

Our results cannot be compared directly to the results on standard databases for benchmarking object recognition algorithms because here the training sets are far less complete. Some of the errors are caused by the lack of data in our models rather than by a deficient classification approach. In addition, these results were obtained using cameras in motion and automatic figure-ground discrimination routines described in Section 2. Our results show that it is possible to recognize objects without using accurate turntables, which are often used to systematically capture all relevant views. This is especially important from the application point of view because not using a turntable greatly simplifies the training process.

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.

The recognition system is implemented on two dual processor PCs that concurrently process video streams coming from the peripheral and foveal cameras. However, such an architecture will become too limiting for real-time execution once the complexity of the cognitive tasks increases. Therefore, a cluster of processors is being established with the ultimate aim of emulating the human visual system. The cluster can be employed to explore various cognitive architectures, such as the one in this paper. Currently 40 PCs are being used, connected together over a 1 Gbit Ethernet network. The vision processing on each PC can range from the most basic (e. g. color extraction, edge filtering, etc) to higher-level (e. g. visual tracking, recognition, etc). The sophistication can increase quite rapidly simply through connecting the processing outputs of simpler elements to the inputs of more advanced processing elements in a bottom-up manner. Manipulation of the lower-level processes can also be performed in a top-down fashion. Thus, this framework will provide the flexibility to explore a greater range of cognitive architectures.

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), 45–66 (2000).
2. C. Breazeal, A. Edsinger, P. Fitzpatrick, and B. Scassellati, Active Vision for Sociable Robots, *IEEE Trans. Systems, Man, and Cybernetics - Part A: Systems and Humans*, **31**(5), 443–453 (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 (ICPR), vol. III,* Québec City, Canada, 2002, pp. 768–776.
4. C. J. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, **2**(2), 121–167, 1998.
5. 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).
6. J. G. Daugman, Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression, *IEEE Trans. Acoustics, Speech, and Signal Processing*, **36**(7), 1169–1179 (1988).
7. C. Gaskett and G. Cheng, Online learning of a motor map for humanoid robot reaching, in *Proc. 2nd Int. Conf. Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2003.
8. G. Guo, S. Z. Li, and K. L. Chan, Support vector machines for face recognition, *Image and Vision Computing*, **19**, 631–638 (2001).
9. T. S. Horowitz and J. M. Wolfe, Visual search has no memory, *Nature*, **394**, 575–577 (1998).
10. L. Itti, C. Koch, and E. Niebur, Model of Saliency-Based Visual Attention for Rapid Scene Analysis, *IEEE Trans. Pattern Anal. Machine Intell.*, **20**(11), 1254–1259 (1998).
11. T. Joachims, Making Large-Scale SVM Learning Practical, in *Advances in Kernel Methods – Support Vector Learning*, eds. B. Schölkopf, C. J. C. Burges, and A. J. Smola (MIT Press, Cambridge, Mass., 1999), pp. 768–776.
12. H. Kozima and H. Yano, A robot that learns to communicate with human caregivers, in Proc. Int. Workshop on Epigenetic Robotics, Lund, Sweden, 2001.
13. 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).
14. 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, 2000.

15. M. Pontil and A. Verri, Support vector machines for 3D object recognition, *IEEE Trans. Pattern Anal. Machine Intell.*, **20**(6), 637–646 (1998).
16. S. Rogeaux and Y. Kuniyoshi, Robust Tracking by a Humanoid Vision System, *Proc. IAPR First Int. Workshop on Humanoid and Friendly Robotics*, Tsukuba, Japan, 1998.
17. B. Scassellati, Eye Finding via Face Detection for a Foveated, Active Vision System, in *Proc. Fifteenth Nat. Conf. Artificial Intelligence (AAAI '98)*, Madison, Wisconsin, pp. 969–976, 1998.
18. T. Shibata, S. Vijayakumar, J. Conradt, and S. Schaal, Biomimetic Oculomotor Control, *Adaptive Behavior*, **9**(3/4), 189–208 (2001).
19. M. J. Tarr and H. H. Bülthoff, Image-Based Object Recognition in Man, Monkey, and Machine *Cognition*, **67**(1-2), 1–20 (1998).
20. M. Turk and A. Pentland, Eigenfaces for Recognition, *Journal of Cognitive Neuroscience*, **3**(1), 71–86 (1991).
21. A. Ude and C. G. Atkeson, Probabilistic Detection and Tracking at High Frame Rates Using Affine Warping, in in *Proc. Int. Conf. Pattern Recognition (ICPR), vol. III,* Québec City, Canada, 2002, pp. 6–9.
22. A. Ude and C. G. Atkeson, On-line tracking and mimicking of human movements by a humanoid robot, *Advanced Robotics*, **17**, pp. 165–178 (2003).
23. A. Ude, C. G. Atkeson, and G. Cheng, Combining peripheral and foveal humanoid vision to detect, pursue, recognize and act, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Las Vegas, Nevada, 2003, pp. 2173–2178.
24. P. Verghese, Visual Search and Attention: A Signal Detection Theory Approach, *Neuron*, **31**, 523–535 (2001).
25. L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, Face recognition by Elastic Bunch Graph Matching, *IEEE Trans. Pattern Anal. Machine Intell.*, **19**(7), 775–779 (1997).
26. J. M. Wolfe, Moving towards solutions to some enduring controversies in visual search, *Trends in Cognitive Sciences*, **7**(2), 70–76 (2003).