Adaptive object classification: a mobile robot case study

University of Glasgow, Department of Aerospace Engineering
Internal Report 9607

Eric Gillies

May 1996

Abstract

An important task for any autonomous agent is to classify any objects that it encounters in its environment. A particular type of object may be useful to the agent, for example, or alternatively, the positions of recognizable objects may be used as local landmarks in a cognitive map of the environment. Biological organisms are adept at classifying new or familiar objects. It is apparent that biological organisms often have *self organizing* methods of classification and that they *specialize* in recognizing particular groups of objects. Biological organisms also often incorporate information from their own movements; rather than relying solely on sense information alone. In this paper, these three observations inspire an adaptive scheme in which a mobile robot learns to recognize examples from a group of complex objects. The autonomous agent moves around each object to determine the object shape. The important features of the shape are then extracted by a self-organizing neural layer. Each object is then represented by a small number of feature amplitudes making object classification more simple. The paper describes the scheme applied to a small skid-steer mobile robot equipped with infra-red proximity sensors.

1 Introduction

It is useful for a mobile, autonomous exploration robot to be able to classify any objects that it encounters [1] [2]. The robot's task may be to identify a 'useful' or 'interesting' object for example. Alternatively, the relative positions of successfully classified objects can be used by the autonomous agent as local *landmarks* in a cognitive map of its environment [3]. The agent may determine its own position, or determine a path towards a goal point by consideration of this map or arrangement of obstacles. Many animals navigate with reference to local landmarks and can easily recognize and classify important objects in their environment [1] [4]. In contrast, mobile robots are often unable to characterize complex objects without using significant computing power and complex sensor arrangements. Moreover, robotic agents often classify objects erroneously if the orientation of the object or the robot's sensory activation pattern is different from the 'memorized' version [1]. Biological systems are adept at either shifting their mental image of the object or shifting their orientation until the object is matched to a memory [1] [4].

The robot perceptual process is augmented if information is processed from both the input sensors (eyes or touch sensors etc.) and also the physical movements of the agent [1] — for example, a stationary robot equipped only with a few simple proximity sensors can only characterize an object very roughly (only a crude measure of the object size is determined by the proximity sensors); whereas, if the robot displays a wall following behaviour, the robot may be able to determine the object shape by inference of the path it takes while moving around the obstacle [5] (this process may be similar to the one used by leaf cutting insects to determine a suitable leaf shape [6]). Path integration, or *dead reckoning*, is a common method of navigation amongst animals and insects [7] [8].

There is a large amount of literature concerned with complex (and expensive) visual recognition systems [2],[9][10]. In contrast, there has been relatively little work on intelligently aquiring and classifying object shapes by simply moving a robot around or over the object [5]. However, for a simple autonomous agent, it is appropriate that any algorithm for learning to classify objects is *self-organizing*: it is difficult to predetermine what significant features of the information gathered by the agent are important for classification. No supervisor is present to label each object encountered by the agent. Also, an algorithm for recognition of *arbitrary* objects is very complex; whereas an algorithm for recognition of a distinct class of objects is simpler [11]. Therefore, most animal recognition systems specialize in recognizing a particular kind of object and can be relatively simple as a consequence (humans, for example, seem to specialize in the recognition of other human faces [11]). It is therefore appropriate that the robot recognition system *specializes.* (Industrial robots, for example, encounter a specific set of industrial components; planetary exploration robots encounter geological formations etc.).

Algorithms for classification operate most efficiently when only a few variables describe each object. If the important *features* of each object are used, to describe the object, the number of variables used in the object description is dramatically decreased [11] [12]. A linearly optimal method for pre-processing input data is proper orthogonal decomposition [13] [11]. This decomposition selects, *without bias*, an orthogonal set of features from the object data set. Each object can be characterized by the smallest possible set of feature 'amplitudes'. Furthermore, the proper orthogonal decomposition process is easily encompassed within a self-organizing neural network framework [12]. The process of orthogonal decomposition (analogous to determination of *principal components* [12]) is thought to be used by simple animals when orienting themselves with their environment (for example, rodents and human toddlers are thought to use principal component analysis to re-orient their mental maps of their environment to a previously encountered orientation [4]).

Thus, an efficient algorithm for object classification can be constructed for a mobile robot which uses its simple sensors in conjunction with a wall following algorithm to measure the shape of each object it encounters. A self-organizing algorithm for characterizing the important features of each encountered object is then realized by proper orthogonal decomposition using a single layer neural network. A small supervised (or unsupervised) neural network can be used to determine which class each encountered object belongs to [14].

The autonomous robot used in this case study is a *Khepera* robot [15]. This robot is a cylindrical (55mm dia.) skid-steer type platform and is equipped with eight infra-red proximity sensors and two stepper motors (10 pulses per mm of advancement— giving a wheel position resolution of 0.08mm). The robot is therefore capable of negotiating a wide range of corners and consequently well suited to the wall following task. An MC68331 microcontroller with 256K of RAM and 512K of ROM manages all the sensor readings, motor control and serial input/output routines. Autonomous behaviour can be coded either on the robot microcontroller or on a workstation which communicates via a serial line to the robot [15]. A schematic layout of the *Khepera* robot is shown in figure 1. In this experiment, the agent processes were coded in C and run from a workstation.

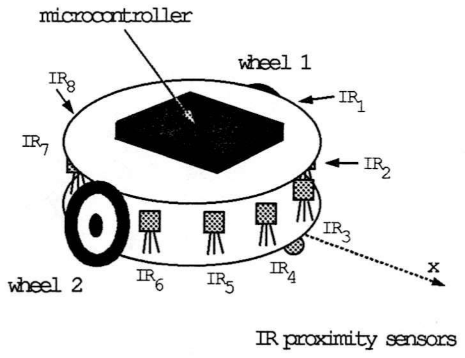

Figure 1: Layout of the Khepera robot

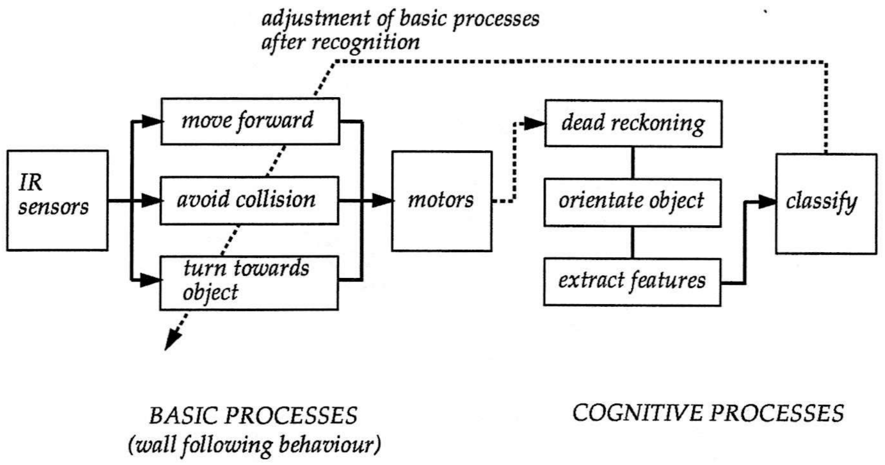

Figure 2: Basic robot processes. The emergent behaviour is wall following and object recognition

2 Agent processes

A block diagram of the robot processes is shown in figure 2. The robot processes comprise two different groups: a set of basic behaviour processes relating sensor readings to robot movements; and a set of computational (or cognitive) processes which form an internal representation of an encountered object and classify the object. A wall following behaviour emerges from the three basic processes. An internal representation (or map) of an encountered object is formed by a dead reckoning scheme which computes the agent's position— thus inferring the object shape from the path taken by the agent while wall following. The 'orientate object', 'extract features' and 'classify' procedures take this shape information and determine the object type. As depicted in figure 2, the output of the classification algorithm may be used to change the parameters governing the basic processes and hence change the observed movement behaviour of the robot (for example, the robot may be instructed to avoid a certain type of object etc.).

The three basic behaviour processes run in parallel and translate readings from the IR sensors into speed values for the robot motors. The first basic behaviour required of the agent is to *explore* its environment. The most efficient method for

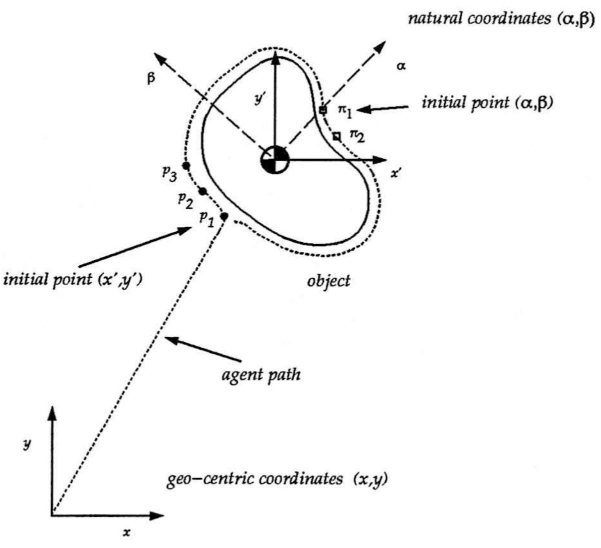

Figure 3: Agent axis system

searching an unknown world is to move forwards in a straight line [16]. Exploration is therefore implemented by giving left and right hand motors a constant value—

$$\omega_{i_{explore}} = \chi, \qquad i = 1,2 \tag{1}$$

where χ is a constant value. The second basic process is to avoid collisions. An optimal solution is the familiar Braitenberg algorithm,

$$\omega_{i_{avoid}} = \sum_{j=1}^{8} I_j B_{j+8i} \qquad i = 1,2 \tag{2}$$

where $\mathbf{B} = [4,4,6,-18,-15,-4,-5,-3,-5,-15,-18,6,4,4,3,5]$. To successfully classify and recognize objects the agent must also be 'interested' in any detected obstacles— the agent must therefore turn towards objects.

$$\omega_{i_{object}} = (3-2i)(k - I_6) \qquad i = 1,2 \tag{3}$$

where k is a constant dependent on the maximum infra-red sensor reading. This process causes the agent to turn towards the right when the agent detects an obstacle to the right. A wall following behaviour emerges from the three basic processes: this behaviour causes the robot to trace out the perimeter of any objects that it encounters— thus allowing the shape of the object to be inferred. Algorithm (3) also ensures that the robot always orbits an obstacle in a clockwise direction: characterization of the object shape is simpler if the data is collected in a consistent way. The total values passed to the two motors are

$$\omega_{i_{total}} = \zeta\omega_{i_{explore}} + \eta\omega_{i_{avoid}} + \theta\omega_{i_{object}} \qquad i = 1,2 \tag{4}$$

The constants η, ζ and θ are selected so that the wall following is stable enough to orbit the desired class of objects in the environment. The values may be changed by the output of the recognition process— a particular type of object may warrant pure avoidance for example.

While the agent is moving, it executes a dead reckoning, or *path integration*, process that takes values from the motor speeds and calculates the robot position relative to its initial orientation and position. The path integration algorithm is obtained by integration of the kinematic equations of motion for the skid-steer platform [17] [7]. At the start of exploration, the agent initiates a geo-centric x, y Cartesian frame depicted in figure 3. Once the agent encounters an object (at a contact point $\vec{p}_1$ in the figure) successive points, each a constant arc length apart, are stored in an 'object vector' $\vec{\phi}$. Once the agent completes an orbit (ie. when $||\vec{p}_i - \vec{p}_1|| \approx 0$) the centroid of the object is calculated as

$$x_c = \frac{1}{N}\sum_{i=1}^{N} p_{x_i} \qquad y_c = \frac{1}{N}\sum_{i=1}^{N} p_{y_i} \tag{5}$$

where the object comprises N points $\vec{p}_i = (p_{x_i}, p_{y_i})$. The position of the object in the environment may be stored as (x_c, y_c). Information pertinent to the shape of the object is stored in the object vector $\vec{\phi}$. The cognitive processes of orientating the cognitive map of each object (in a consistent way), and of feature extraction are described in the next two sections.

3 Orientation

Different approach directions of the agent to an object result in different initial contact points ($\vec{p}_1$ in figure 3) and therfore different object vectors. Hence, the agent's classification of the object depends on the direction that the object is approached from. This dependence is undesirable: the agent should be able to classify an object correctly no matter what orientation the object is in. Animals either perform physical movements such that the image of the object corresponds with a memory; or shift their mental map until the object is recognized [1] [4]. A more natural coordinate basis, dependent on the object shape alone, is required for representation of each object. It has been suggested that many animals use a principal components analysis in order to orient themselves to a familiar object or environment [4]. Therefore, an appropriate method for selecting a natural coordinate basis for each object is to select the principal axes from the bi-variate data set (Cartesian x, y coordinates) that constitutes each object vector [12]. The centroid coordinates of each object are removed from each object vector (such that $\sum p_{x_i} = \sum p_{y_i} = 0$). Each of the N measurement points in the object vector is of the form

$$\vec{p}_i = \begin{bmatrix} p_{x_i} \\ p_{y_i} \end{bmatrix} \tag{6}$$

The first principal component of the object vector corresponds with the largest eigenvalue of the 2D correlation matrix [12]

$$\mathbf{R} = \frac{1}{N}\sum_{i=1}^{N} \vec{p}_i \vec{p}_i^{\,T} \tag{7}$$

which can be calculated simply from,

$$\lambda_1 = \frac{(R_{11} + R_{22}) + \sqrt{(R_{11} + R_{22})^2 - 4(R_{11}R_{22} - R_{12}^2)}}{2} \tag{8}$$

The proposed first principal axis is then parallel with

$$\vec{\alpha} = \begin{bmatrix} 1 \\ \frac{(\lambda_1 - R_{11})}{R_{12}} \end{bmatrix} \tag{9}$$

The arithmetic involved in this computation is simple and can be performed rapidly either by using (8) directly or an approximation to (8) afforded by a multi-layer perceptron. Alternatively, a single, self-organizing, linear neuron will converge to the first principal axis (using an appropriate Hebbian training algorithm). However, for 2D problems, this method takes longer to converge than simply calculating (8).

The principal axis of the object may be aligned with $\vec{a}$ or $-\vec{a}$. The agent map of the object will therfore fall into either of two orientations (180^o out of phase). This is much better than the multitude of orientations that would result, from multiple approaches to the object, had no principal axis been calculated. A simple algorithm proves effective to refine the orientation of the agent object map further: the positive principal axis direction is taken as the direction with the largest crossing point of the object edge with the axis. This algorithm works well in the majority of cases, but is confused when the crossings of the object edge with the axis have equal positive and negative values. The 'initial point' in the re-oriented object vector is taken as the largest crossing of the first principal axis.

To perform meaningful comparisons between objects it is also necessary that every object vector has the same dimension. This is coded on the agent as a simple linear interpolation: the measured and re-oriented object vector is interpolated to form a new object vector of constant size.

4 Feature extraction

An autonomous agent may infer the shape of an obstacle by analysis of the path taken while moving in an orbit around the obstacle. The Cartesian coordinate of the agent is a scalar function of the arc length travelled along the edge of the object (as represented in figure 3). Each object encountered by the agent can be represented as an $2N$ dimensional 'object vector' $\vec{\phi}$ of concatenated *natural* coordinates:

$$\vec{\phi} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_N \\ y_N \end{bmatrix} \tag{10}$$

where N is the number of position measurements recorded by the robot during its orbit. The measured coordinates are with respect to the object centroid, such that $\sum x_i = \sum y_i = 0$.

The environment is assumed to contain M distinct types of object. Therefore, the shapes of the objects in the environment may be described completely by $2MN$ numbers— the space describing the object shapes is of dimension $2MN$ (a full description of the environment also requires a further $3M$ numbers: the Cartesian position of each object centroid and the orientation of each object to a geo-centric frame). For large M there is nothing particularly efficient or natural about this coordinate frame or its Fourier transform [11].

However, if the 'useful' or 'important' objects in the agent's environment all belong to a specific class of objects then it is reasonable to assume that a smaller number of dimensions can adequately describe the environment. Animals specialize in recognizing examples from well defined classes of objects (for example: humans find it easy to recognize other human faces, leaf cutting insects specialize at recognizing leaf shapes etc. [11] [6]) and animals often perform their recognition process instantaneously [1]. It is therefore likely that biological systems exploit a much more efficient (or alternatively low-dimensional) description of important objects in their environment.

It is therfore appropriate to transform the high dimensional data space, representative of the object shapes, into a low dimensional, *readily identifiable feature space* [12]. Proper orthogonal decomposition (POD) is based on the Karhunen-Lóeve expansion and selects just such a natural coordinate basis [13]. The POD basis is optimal in that it selects, *without bias*, the smallest linear sub-space capable of representing the observed object data [11]. The POD basis is also biologically plausible: the POD uses only linear transformations (yet it makes no assumptions about the linearity of the problem of interest) and can be re-formulated into a self-organizing neural network format. Moreover, the POD basis is a natural extension of the bi-variate principal components analysis used to orient the object in the agent's internal representation.

4.1 Proper Orthogonal Decomposition

It is required to select a characteristic unit vector $\vec{\psi}$ which has a 'direction' in phase space as close as possible to the collection of object vectors, $\{\vec{\phi}\}$ in the sense that, $(\vec{\psi} \cdot \vec{\phi})$ is maximized. To characterize objects from an environment containing an ensemble of objects it is necessary to find a vector which is best correlated with every member of the ensemble [13]. It is natural to concentrate on departures from the mean shape. The object vectors are adjusted again so that the new object vector $\vec{\varphi}_i$ is calculated by

$$\vec{\varphi}_i = \vec{\phi}_i - \langle\vec{\phi}\rangle \qquad (11)$$

where the braces denote an ensemble average. Hence $E\{\vec{\varphi}\} = 0$. Therefore the best statistical measure over which to maximize $(\vec{\psi} \cdot \vec{\varphi})$ is the mean square

$$E\{(\vec{\psi} \cdot \vec{\varphi}_i)^2\} = \lambda \geq 0 \qquad (12)$$

Maximization of (12) produces the best correlated vector to the ensemble of velocity field realizations in a mean square sense. It is readily shown that extremal $\vec{\psi}$ correspond to eigensolutions of the algebraic eigenproblem [11]

$$\mathbf{R}\vec{\psi} = \lambda\vec{\psi} \qquad (13)$$

where $\mathbf{R}$ is the environment average, spatial correlation matrix of object vectors,

$$\mathbf{R} = E\{\vec{\varphi}\vec{\varphi}^T\} \qquad (14)$$

The matrix eigenproblem defined by (13) yields an *orthonormal* set of vectors that characterizes the spatial structure of the *collection of objects forming the environment.* The eigenvectors, or modes, can be recognized as 'directions' in $\Re^{2N}$ along which the variance of the discretized object shapes have local maxima. The 'modal amplitude' is calculated by,

$$A_i = (\vec{\psi} \cdot \vec{\varphi}) \qquad (15)$$

The eigenvalue λ_i can be interpreted as the *probability* that any encountered object will fall along the direction $\vec{\psi}_i$.

Solution of the eigenproblem (13) is a difficult task if the number of spatial measurement points, N, is large— the problem is of order $(2 \times N)^2$ for two-dimensional objects. The effective order of the problem can be reduced to an M^2 eigenproblem, however, using the method of 'snapshots' [11]. However, the entire POD process is equivalent to the evolution of a single layer of linear neurons trained by a Hebbian, self-organizing algorithm [12].

4.2 Self-organizing, autonomous, feature extraction

It is appropriate, on the grounds of autonomy and also biological plausibility, that the algorithm for feature extraction be self-organizing. Consider a singel layer of P linear neurons. The input vector to the layer is the zero ensemble mean object vector, $\vec{\varphi}_i$. The output of any particular neuron in the network is

$$A_j(n) = \sum_{i=1}^{P} w_{ji}(n)\varphi_{ni} \tag{16}$$

The synaptic weights are labelled so that w_{ji} corresponds to the j'th neuron connected to the i'th input. The network is trained with the *self-organizing Hebbian* algorithm

$$w_{ji}(n+1) = w_{ji}(n) + \eta \left(A_j(n)\varphi_{ni} - A_j(n) \sum_{k=1}^{j} w_{ki}(n) A_k(n) \right) \tag{17}$$

where η is a very small learning rate. This algorithm corresponds to self-amplification of the network weights when there is agreement between the pre-synaptic and post-synaptic signals and competition between the weights of the network. As n approaches infinity, the equilibrium condition $\vec{w} \to \vec{\psi}$ satisfies the eigenproblem

$$\mathbf{R}\vec{\psi} = \lambda\vec{\psi} \tag{18}$$

which is the same eigenproblem as in the POD mode calculation [12].

This algorithm only converges if the ensemble average of the object vectors is exactly zero. It is therfore important to extract the mean-object vector (as in (11)). It is important that the extraction of the mean-object vector is also an autonomous process. The continuously calculated average object is obtained using

$$\langle\vec{\phi}_k\rangle = \frac{k-1}{k}\langle\vec{\phi}_{k-1}\rangle + \frac{1}{k}\vec{\phi}_k \tag{19}$$

where k is the number of objects so far encountered. The k'th input vector to the network is thus given by

$$\vec{\varphi}_k = \vec{\phi}_k - \langle\vec{\phi}_k\rangle \tag{20}$$

Thus, when the agent encounters its first object, $\vec{\phi}_1$, the mean object is calculated as $\vec{\phi}_1$ and the input to the network is $\vec{0}$. However, as the number of encountered objects increases, the vector $\langle\vec{\phi}_k\rangle$ converges to the mean-object vector as required.

The converged weights of this linear network form an optimal basis for representation of the encountered objects. The network is arranged such that the eigenvalues of each mode are in monotonically decreasing order. Each eigenvalue is a measure of how important the associated mode, or feature, is in the environment. For M objects, there are only M non-zero eigenvalues of the correlation. However, if only $K < M$ eigenvalues are large, the self-organizing network only need contain K neurons. The POD basis therefore undergoes a dimensionality reduction.

Any object in the environment can then be characterized by $K < M$ outputs of the linear network— rather than the $2N$ numbers needed for a Cartesian coordinate description. The POD basis is of most use when the objects to be characterized form a well defined class [12]. This is a feature of many animal recognition systems. Also, if a new object is encountered which did not form part of the characterization ensemble, then the POD modes are still useful: the modes are still an orthonormal axis system on which the new object can be projected. If the new object belongs to a similar class to the other objects in the characterization ensemble then the POD projection will still be efficient for representation of the new object.

4.3 Classification

After encountering the M distinct object shapes in the environment the agent computes M features. The corresponding eigenvalues relate the probability of encountering each feature in the environment. Only the features of significant importance are retained: if the eigenvalue is below a certain percentage probability then the corresponding feature is not stored. This process results in the agent storing $K < M$ vectors (the important POD modes). When an object is encountered by the agent (17) is used to calculate a K–dimensional vector of mode amplitudes. This small dimension vector is used by a classification algorithm to determine the object class.

The classification algorithm may be a self-organizing neural classifier or a hybrid supervised/unsupervised network (eg. a Kohonen SOFM and a supervised linear classifier) [12] or a traditional classification algorithm [14]. In order to examine the performance of the agent's self-organizing feature extraction algorithm, a traditional classifier is used in this case study. A neural classification scheme involves its own issues of convergence and accuracy which may confuse the analysis of the feature extraction performance of the agent.

Once the feature extraction algorithm of the agent has converged then a set of exemplar mode amplitudes is measured and labelled for each object (using the agent controlled directly via a serial line). Traditional classification is coded by computing the 'distance' (in K–dimensional space) between each exemplar amplitude vector and an encountered object. The best match is selected and if this distance is less than a small threshold value then the agent outputs the corresponding label attached to the exemplar amplitude vector. If the distance exceeds the threshold the agent outputs a value corresponding to non-classification. This classification algorithm may be used to change the observed behaviour of the robot on encountering a specific object. For example, the θ parameter in (4) may be set to zero (causing the agent to turn away from the object) for a certain class is output.

5 Results

The basic robot processes, orientation algorithm and self-organizing feature extraction network were coded in C and run on a work-station communicating with the agent. The agent was set in a flat 2m$\times$2m domain in which variously shaped obstacles with vertical walls (each approximately 3–4 times wider than the agent) were placed. In total, nine different object shapes were placed one at a time in the environment. This manual process was used to avoid the experimental difficulties with setting the agent 'free' in a very large domain. Using a large, obstacle-filled, environment is impractical with the agent processes running on a work-station communicating with the agent-body via a serial line.

5.1 Object measurements

The basic behaviour parameters were adjusted to give a stable wall following behaviour. The nine object shapes corresponded roughly to the outlines of the first nine letters of the alphabet. These shapes constitute a convenient set of obstacles— the letters are significantly complex; yet easily recognizable by a human observer. The letter shapes contain several common features so the POD process was expected to perform well on such an ensemble. Typical agent maps of the outlines of the nine shapes are shown in figure 4.

Figure 5 shows the actual walls of an 'A' shaped object (solid lines) and the inferred wall positions calculated by the agent. The object centroid is marked with cross hairs. The agent and object are both portrayed in figure 6. Each object was measured at points a constant arc length apart (corresponding to $\frac{1}{3}$ wheel

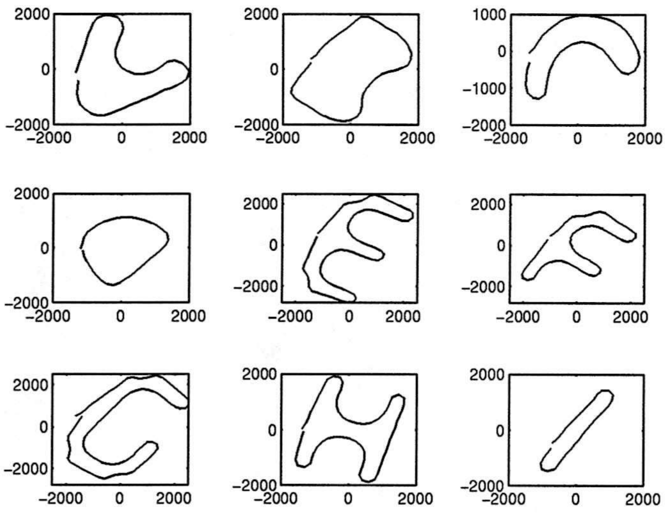
Figure 4: Outlines of the nine shapes found in the agent environment

rotations)— hence, each object was initially represented by a different number of points dependent on the size of the object perimeter. (Measuring the points at a constant distance interval, as opposed to a time interval, proved to increase the reliability of the object orientation routine). Once oriented correctly, by the agent, the object vectors were linearly interpolated to have 100 points. *Only one object vector (and also one mean-object vector) is stored by the agent at any one time.*

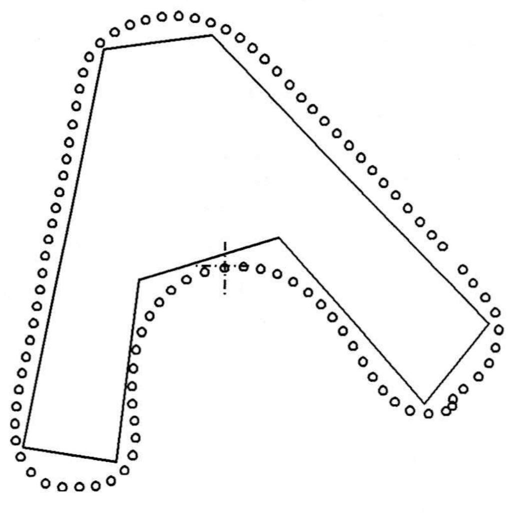
Figure 5: Actual and inferred outline of the 'A' shape

5.2 Object orientation

The object orientation process performed well on the nine objects. For each of the nine objects, ten different approaches (from different directions) were undertaken by the agent. Table 1 shows the success of the orientation algorithm. The percentages

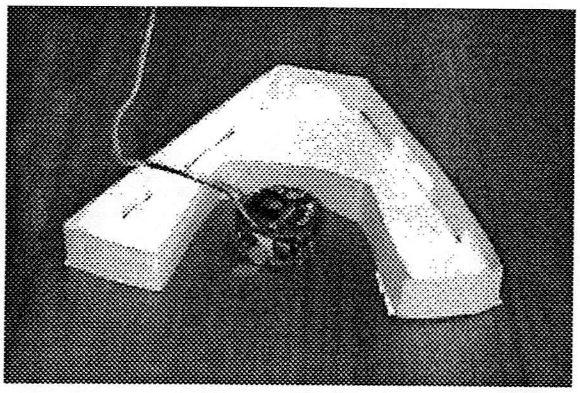
Figure 6: The agent and 'A' shaped object

in the table are the percentage consistency in the orientation of each object.

Shape	A	B	C	D	E	F	G	H	I
Orientation success	90%	80%	100%	60%	60%	100%	100%	60%	100%

Table 1: Orientation repeatability— 10 trials per object

Figure 7 shows three superimposed internal maps of different approaches to the 'A' shape (typical of the results for each of the shapes). The internal maps are all in roughly the same orientation and the calculated 'initial points' are close together. Figure 8 compares the three associated adjusted-object-vectors for the three different approaches to the 'A' shape. The object vectors for the three different approaches (and hence different physical object orientations) show a high degree of correlation. The orientation process is therefore successful.

5.3 Object recognition

The object recognition process forms two distinct phases: *a learning phase* (where the feature extraction network weights converge) and *a recognition phase* (where the agent has stopped learning and is able to classify encountered objects). The results of the training period are discussed first.

5.3.1 Learning phase

The learning phase of the agent recognition behaviour was implemented by manually placing objects in the agent path; rather than setting the agent free in a large obstacle filled domain for a considerable length of time. The agent orbits the encountered obstacle once, recording the Cartesian position at regular distance intervals, and then (when the agent position is close to its initial position) the turn-towards-object parameter (θ) is multiplied by the time-dependent function displayed in figure 9. This causes the agent to break contact with the obstacle and move off in a straight line. At the same time, the object vector is oriented (using the process described above) and mean-object vector is updated using (19). The synaptic weights of the feature extraction network are presented with the adjusted (ie. zero

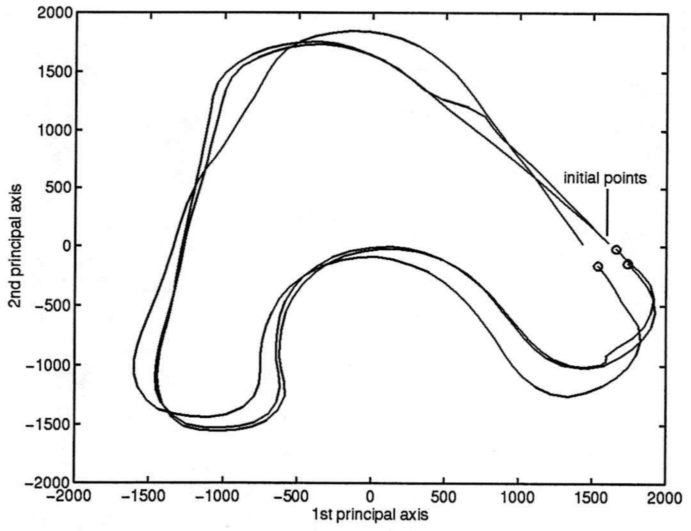

Figure 7: Three agent maps of the 'A' shape

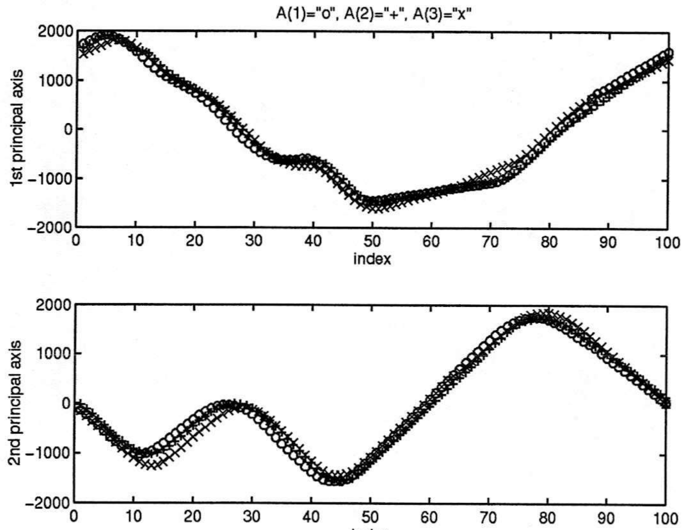

Figure 8: Three object vectors for different approaches to the 'A' shape

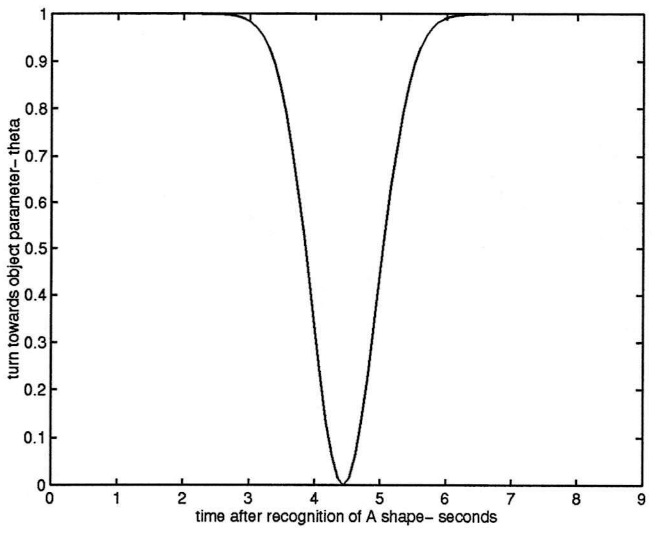

Figure 9: Time dependent theta function

ensemble average) object vector. The mean update and the feature extraction updates are performed quickly and there is no significant interruption to the agent's movement. The synaptic weights are updated using a learning rate of 1×10^{-8}. At any time, the human observer can interrupt (and then re-start) the robot motion by typing a character on the work-station. This enables the experimenter to place the robot in the environment such that a new object is encountered each time.

After some experimentation it was found that each of the nine objects could be distinguished using only three POD modes— the feature recognition network was pruned leaving only three neurons in the layer. At any one time the agent only stores one mean vector, one object vector and the three synaptic weight vectors. Figure 10 shows the eigenvalues of the POD modes spanning the ensemble of object-vectors. Most of the structure of the objects can be described by the three largest modes. After further experimentation it was found that approximately 7–10 cycles through the nine objects (ie. the agent encountered each object shape 7–10 times) were the minimum necessary for the synaptic weights to converge enough for reliable object characterization. Figure 11 shows the spatial structure of the partially converged (ie. after 10 cycles) POD modes. It can be seen that these modes do not necessarily represent any coherent structure or feature obvious to a human experimenter. Reconstructions of the 'E' shape using 9,8,7,6,5,4,3 and 2 neurons are shown in figure 12. Although the agent's 3 mode reconstruction of the object is slightly different from the actual shape, the three associated mode amplitudes are enough to distinguish the object as an 'E'. This is typical for all the other objects.

5.3.2 Recognition phase

The recognition phase of the agent's behaviour begins when the synaptic weights of the feature extraction network are close to convergence (ie. after 7–10 cycles through the training set). The three retained mode amplitudes are then given by the outputs of the three neurons in the feature extraction network. The spatial separations between each of the mode amplitude triples for the nine different objects are depicted in figure 13. Each point is distinct from every other. A traditional minimum distance classification algorithm then takes exemplar values of the mode

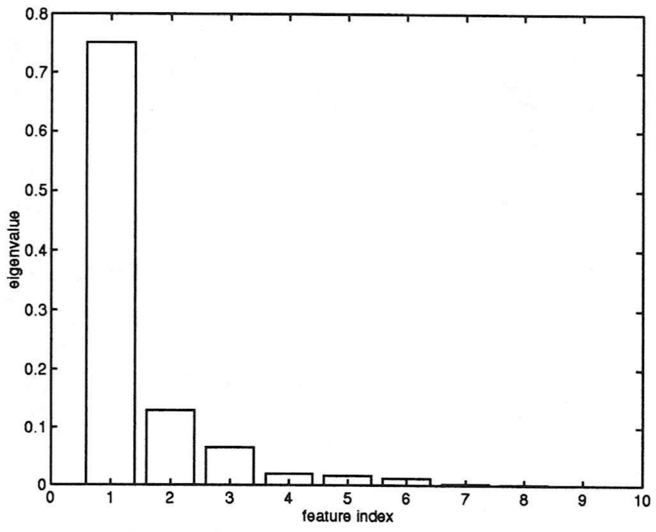

Figure 10: POD mode eigenvalues

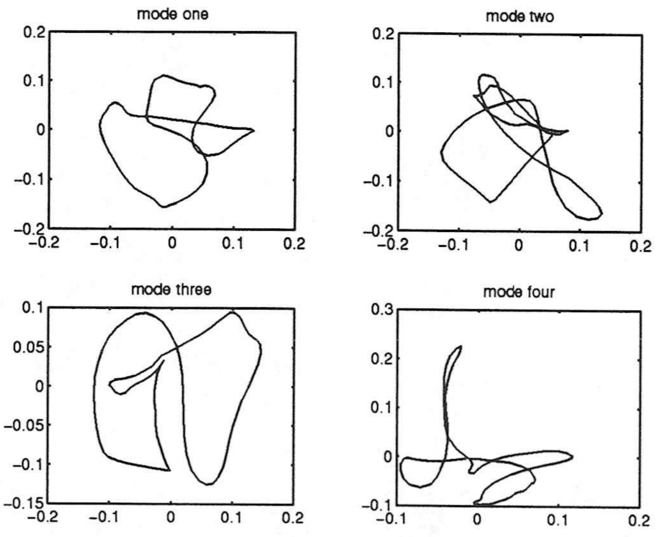

Figure 11: Spatial structure of the first four POD modes

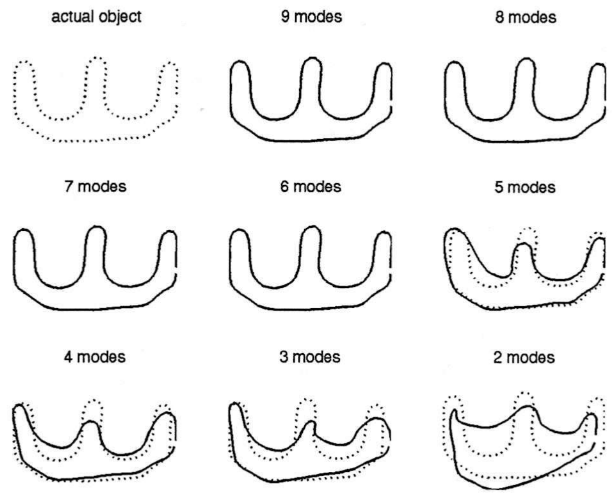

Figure 12: Reconstructions of the 'E' shape

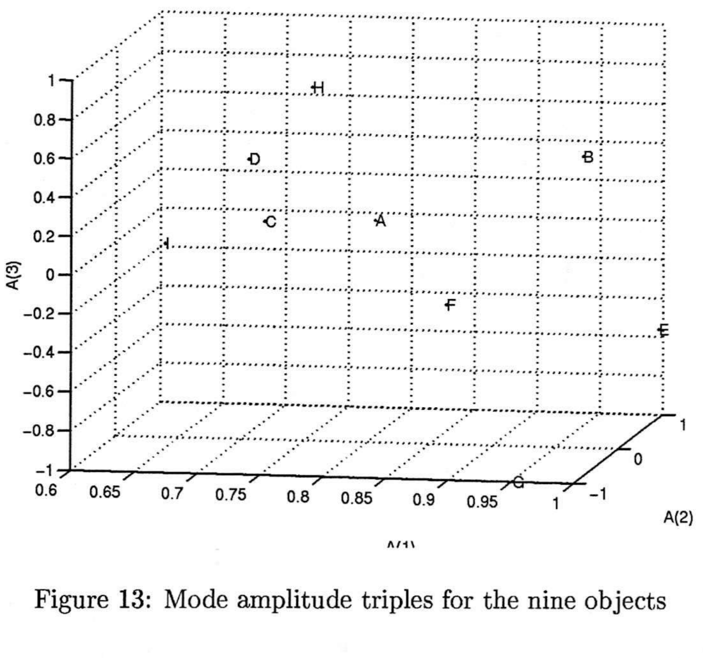

Figure 13: Mode amplitude triples for the nine objects

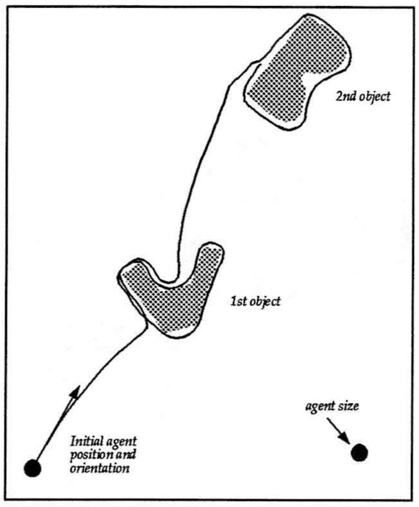

Figure 14: Example of agent behaviour

amplitude triples for each of the nine objects (this is the only non-autonomous step in the behaviour). The classifier computes the distance between an observed triple and each of the exemplars. The best match is found and, if the distance is less than a threshold value, the corresponding object is 'recognized' by the agent. The classifier correctly recognized the shapes with percentage successes displayed in table 2. The table shows the recognition success for five approaches to each object. The average success is high— 77%. All of the failures occur when a non-standard orientation of the object is chosen by the orientation routine or when inaccuracy in the agent odometry routine causes the agent to orbit the obstacle more than once. With more cycles through the training data and more exemplars to match non-standard orientations it is expected that the recognition success will increase. Missclassification is avoided by using a small threshold value.

Shape (5 samples)	A	B	C	D	E	F	G	H	I
Recognition success	80%	60%	100%	60%	60%	80%	100%	50 %	100%

Table 2: Recognition repeatability— 5 trials per object

5.3.3 Example of behaviour

Many different behaviours may be coded into the agent to determine the agents movements after recognition of a particular object. One example of such a behaviour is displayed in figure 14. Here, the agent is told to multiply all movement commands by zero if a 'B', 'D', 'F' or 'H' shape is encountered. Otherwise it behaves as in the learning phase (ie. by multiplying the θ parameter by the function in figure 9. This is observed (in figure 14) when the agent first encounters an 'A' shape: the shape is successfully recognized and the agent moves on. The next shape encountered by the agent is, fortuitously, a 'B' shape so the agent stops.

6 Conclusions and Future Work

An efficient, self-organizing method for characterization of complex objects by an autonomous mobile robot has been presented. The method is simple and robust as a consequence of its biological inspiration. The agent traces out the perimeter of objects it encounters and uses this shape information in an adaptive characterization scheme. A simple method for orienting the objects in memory is principal components analysis. An extension of principal components analysis, namely proper orthogonal decomposition is an optimal method for representation of a data set — the method results in a dimensionality reduction of the object shapes; this reduction allows a large saving in memory. The method performs best on data sets which form a well defined class of objects. Results have been presented which show an agent able to quickly learn to classify complex objects (representations of the first nine letters of the alphabet in this case study) with a 67% data saving and 77% success rate. The recognition failures, however, are all caused by either inaccuracy in the agent odometry routine (giving false position information) or by failure of the agent wall-following process: these inaccuracies cause the agent to orient an object in a non-standard way such that recognition fails. The POD method itself is very robust to noise, however, the orientation process is more sensitive.

Important areas of future work are therefore to increase the accuracy of the path integration process and to investigate the stability of the wall-following behaviour. The incorporation of a compass is likely to increase the path integration accuracy— an ambient light compass could easily be coded into the robot. The turn-towards object process range could be extended by placing lights on the objects— the robot would then detect the object from further away. Training of the agent in a large obstacle-filled domain, without any human intervention, will then be more appropriate.

The method has been tested on a particular set of complex obstacles: other sets of objects both from well defined classes and also from essentially arbitrary groups should also be able to be classified. Future work will be directed at validating this assertion. The characterization ability of the agent could be extended by incorporating more senses (eg. visual or infra-red readings) and the object size could also be used as an aid to characterization. The agent cannot characterize overlapping objects or determine the object shape from a partial orbit: it is therefore likely that future work will also concentrate on extending the method to work with partial knowledge of the object perimeter. Also, it is likely that future case studies may examine applications of the recognition scheme: finding a particular object or making a map with landmarks for example.

References

[1] C. Scheier and D. Lambrinos. Adaptive classification in autonomous agents. *University of Zurich AILab technical report*, 95(12), 1995.

[2] C. Scheier and D. Lambrinos. Categorization in a real-world agent using haptic exploration and active perception. In *From animals to animats: proceedings of the fourth international conference on the simulation of adaptive behaviour, Cape Cod, USA*, 1996.

[3] T. J. Prescott and J. E. W. Mayhew. Building long range cognitive maps using local landmarks. In *From animals to animats 2: proceedings of the second international conference on the simulation of adaptive behaviour*, 1992.

[4] C. R. Gallistel and A. E. Cramer. Computations on metric maps in mammals: getting oriented and choosind a multi-destinational route. *Journal of experimental biology*, 199:211–217, 1996.

[5] R.E. Ellis. Planning tactile recognition paths in two and three dimensions. *International journal of robotics research*, 11(2):87–111, April 1992.

[6] A. Daanje. Ueber die ethologie und blatterolltechnik von Deporus Betulae L. und ein vergleich mit den anderen blattrollenden rhynchitinen und attelabinen. *Verh. Kon. Nederl. Akad. Wetenschappen*, 61:3–215, 1964.

[7] S. Benhamou and V. Seguinot. How to find one's way in a labyrinth of path integration models. *Journal of theoretical biology*, 174:463–466, 1995.

[8] R. Wehner, B. Michel, and P. Antosen. Visual navigation in insects: coupling of ego-centric and geo-centric information. *Journal of experimental biology*, 199:129–140, 1996.

[9] B. Crespi, C. Furlanello, and L. Stringa. A memory based approach to navigation. *Journal of biological cybernetics*, 69:385–393, 1993.

[10] R. Brunelli and T. Poggio. Caricatural effects in automated face perception. *Journal of biological cybernetics*, 69:235–241, 1993.

[11] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, March 1987.

[12] S. Haykin. *Neural networks: a comprehensive foundation.* Macmillan, 1994.

[13] J. Lumley. *Stochastic tools in turbulence.* Academic Press, 1970.

[14] R. Lippmann. An introduction to computing with neural networks. *IEEE ASSP Magazine*, April 1987.

[15] F. Mondada, E. Franzi, and P. Ienne. Mobile robot minituarization: a tool for investigation in control algorithms. In *Third international symposium on experimental robotics, Kyoto, Japan*, 1993.

[16] D.B. Dusenbery. *Sensory ecology.* W.H. Freeman, 1992.

[17] D. B. Reister and F. G. Pin. Time-optimal trajectories for mobile robots with two independently driven wheels. *International journal of robotics research*, 13(1):38–54, 1994.