

Contents lists available at [ScienceDirect](http://www.sciencedirect.com)

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Exploration of configural representation in landmark learning using working memory toolkit

Xiaochun Wang*, Mert Tugcu, Jonathan E. Hunter, D. Mitch Wilkes

Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235, USA

ARTICLE INFO

Article history:

Received 30 September 2007
 Received in revised form 21 June 2008
 Available online xxx

Communicated by H.H.S. Ip

Keywords:

Visual landmarks
 Scene recognition
 Configural representation
 Feature representation

ABSTRACT

The mechanisms by which humans and animals use visually-acquired landmarks to find their way around have proved fascinating. Considerable evidence suggests that animals navigate not only on the basis of the overall geometry of the space but also on the basis of a configural representation of the cues. In contrast to earlier linear models of elemental feature representation, configural representation requires individual stimulus be represented in the context of other stimuli and is typified by non-linear learning tasks such as the transverse patterning problem.

This paper explores the suitability of configural representation for automatic scene recognition in robot navigation by conducting experiments designed to infer semantic prediction of a scene from different configurations of its stimuli. The main contribution of this work is that it provides a methodology for automatic landmark-based scene identification with the aid of a reinforcement learning based software package, called the working memory toolkit (WMtk), which allows reward associations between a target location and the conjunctive representations of its stimuli. Experimental results obtained with two different target locations are presented and compared with those of two other classification mechanisms, a support vector machine approach and a simple linear two-class classifier, perceptron.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Navigation is the ability to perform goal-directed motion. Three systems are fundamental to human and animal navigation, a path integration system for computing and updating the relationship between one's current position and other significant environmental locations, a scene recognition system for guiding navigation through familiar terrains, and a reorientation system for determining one's position and heading when one has become disoriented (Wang and Spelke, 2000).

Dead reckoning, also known as path integration (Mittelstaedt, 1983) and vector navigation (Wehner, 1983) in animal navigation, refers to navigation using motion information to keep a running calculation of the distance moved in a particular direction. The purpose of dead reckoning is to continuously keep track of one's own position relative to the point of departure. However, in the process of updating the representation of its location on the basis of the records of movement, dead reckoning is sensitive to the accumulation of small errors. Once it fails, either partly or completely, the presence of landmark stimuli in the environment is required to

guide the animal's behavior. As a result, the path integration system should be supplemented by a scene recognition system for recognizing significant and familiar places in the landscape to provide an accurate guidance for navigation. Then a series of such discrete external cues can be chained together in order to navigate a more complex path through the environment, which is called route-following and route-learning.

Animals use a strategy called piloting in which external landmarks are used to reach a goal location. The use of landmarks for goal identification is fundamental to path integration (Wehner, 1992), which makes the ability to recall the appropriate memories of contexts a significant part of navigating through familiar terrains. Often an isolated natural landmark on its own may easily be mistaken for another similar isolated natural landmark in a different location. Ambiguity can be reduced by not restricting attention just to the landmark in question but broadening it to include the larger context in which the landmark is set. For example, along a route, a sequence of similarly looking landmarks may be encountered and the different scenes in which they are embedded may also be confusable because they share some elements in common. To lessen ambiguity, the landmark elements that comprise a scene should be bound together into a configural association. Only after the landmarks and the contextual stimuli of a scene have been bound together into an association can they be more resistant to interference from conflicting information acquired in another

* Corresponding author. Tel.: +1 615 385 1194; fax: +1 615 322 6972.

E-mail addresses: xcawang@yahoo.com (X. Wang), mert.tugcu@vanderbilt.edu (M. Tugcu), jonathan.e.hunter@vanderbilt.edu (J.E. Hunter), mitch.wilkes@vanderbilt.edu (D.M. Wilkes).

context. The formation of configural association between a particular set of landmark cues and a particular set of contextual cues is called compound learning, configural learning, or contextual learning, which must take place in every new situation.

For contextual cues to play a role in landmark-based scene recognition, they are assumed to be somewhat invariant to the distance to the observer. It has also been noted that landmarks are likely to be learned from the same vantage point. With increasing distance from that vantage point, the landmark views tend to change gradually and the expectation of encountering the associated local landmark may fall off with distance (Collett et al., 1997). However, landmarks have different effects on navigation depending on their distance: nearby landmarks specify the positions of significant objects, whereas distant landmarks specify the animal's direction (similar to a compass-based system).

When path integration is fully disrupted and people are disoriented, a reorientation system is needed to restore the representation of the spatial relationship between the animal and its environment by analyzing the shape of the current surrounding layout and relating that shape to the remembered one before disorientation.

1.1. Scene recognition

A scene in which events occur can provide a place, space, or context for an experience. The stimulus features that make up this context can be represented in two ways, a feature representation where the scene is represented as a set of independent individual features associated with the scene, and a configural representation (also referred to as unitary representation and configural association) where the scene is represented as a set of features combined into a unitary representation that encodes their conjunction. In other words, the feature representation associates a discrete cue with a response while the configural representation associates a conjunctive representation formed from the combination of multiple cues with a response and different configurations of the stimuli have different responses. There is sufficient evidence that neocortical systems can represent the independent individual features of an environment (Nadel and Willner, 1980) whereas the hippocampus (together with the cortex) uses compound stimuli constructed through conjunctive representation to control performance (Sutherland and Rudy, 1989).

An important source of evidence supporting the view that the hippocampus contributes to memory by associating some features of a scene with their context to form conjunctive representation comes from the studies of the transverse patterning problem, which requires representing the individual stimulus in the context of other stimuli (Dusek and Eichenbaum, 1998). In the transverse patterning problem, two of three stimuli, A, B, and C, are presented on each trial and reward is issued according to the following rule: A^+B^- , B^+C^- , and C^+A^- . The first rule says that A will be rewarded when presented with B. The second rule says that B will be rewarded when presented with C. The third rule says that C will be rewarded when presented with A. When all the patterns are presented equally often, each stimulus will receive the same total rewards. This problem can not be solved by the feature representation. Instead, each of the conjunctive representations, AB, BC, and CA, must be uniquely associated with the appropriate response. It is the unique conjunctive coding of stimuli that configural representations rely on.

To interpret the above theory from another angle, the conjunctive representation can have advantages over the feature representation by solving nonlinear discrimination problems. For example, in a bi-conditional discrimination task of form AB^+ , CD^+ , AC^- , BD^- , a reward is issued for responding in the presence of AB and CD compounds and a punishment is issued for responding in the presence of AC and BD compounds. However, since each element (A, B,

C, D) is equally associated with rewarded and punished trial outcomes, their linear combinations will end up with equal associative strengths across the compounds. To solve this problem, conjunctive representation in the form of the compounds should uniquely be associated with its respective trial outcome. Further, when two events have overlapping features, such as ABCDE and ABCEF, if the single feature representation scheme is used, there will be potential interference. However, if each combination is treated as a unique representation, there will be reduced interference. It is in this sense that conjunctive representation can produce pattern-separation for similar experiences or input patterns (Rudy and O'Reilly, 2001).

1.2. Related work on landmark selection

Three fundamental problems in mobile robot navigation are: self-localization, path planning, and map-building (Siegwart, 2004). Localization is the agent's ability to establish its own position within a frame of reference. As an extension of localization, path planning requires the determination of the agent's current position and the position of a goal location within the same frame of reference. A map, in this context, denotes a mapping of the world onto an internal representation, not necessarily a metric map of the environment but any means of describing locations within the frame of reference.

Given a Cartesian frame of reference, a robot's position is (under ideal, i.e., unrealizable, conditions) always precisely definable, and navigation is simple and perfect. Unfortunately, the frame itself is not anchored in the real world but moves with respect to locations within the world. As one way to overcome this problem, mobile robots use the internal geometrical representations of the robot's environment to perform the navigational task. Though such geometrical representations can either be obtained from sensor data or supplied by the designer, they are subject to alteration in the course of operation. These "classical" approaches are the core of proprioception-based systems. Another way to overcome this problem is to anchor the navigation system within the world itself, rather than within an internal frame of reference, and detect unique features, such as landmarks, to facilitate the robot navigation. By identifying and following landmarks or sequences of landmarks in a specific order, navigation is achieved with respect to the world. This approach is referred to as piloting.

Then the problem becomes how to select consistent landmarks for mobile robot navigation. One technique suggested in the literature is to ask the user to define the landmarks before the robot explores. In this approach, camera images are taken at regular intervals as the robot travels, and objects that human designers believe may be reliably recognized, such as doors or line segments, are extracted as landmarks (Kortenkamp and Weymouth, 1994). However, this approach suffers from the problem of perceptual discrepancy, that is, the robots perceive the world differently from the human. For example, the corners, junctions, or corridors selected by human designers may be good landmarks along the hallway from a human point of view, but may not necessarily be well suited to the perceptual capabilities of the robot because there is no guarantee that the mobile robot can recognize or detect these objects reliably.

To overcome this problem, learning methods have been used to select an optimal set of landmarks for performing self-localization in one specific environment. This is a preferred method, not only because it does not require direct programming and adapts well to the environmental changes, but also that it is often viewed as an essential part of an intelligent system. In Thrun's implementation (Thrun, 1998), Bayesian learning was used and the robot's raw sensory perception was projected onto vectors in a low-dimensional space, which was next used to make up the

landmarks. The projection was followed by an optimization performed by minimizing directly the robot's error in self-localization. Thrun's results yielded better performance over localization using human-determined landmarks including doors, etc. Vlassis et al. developed a similar but computationally cheaper supervised technique and showed that their optimization approach produced better results than the principal component analysis (Vlassis et al., 2000). Fleischer and Marsland used a similar method but with difference in that they did not carry out any analysis of the utility of the landmarks selected, but instead used a self-acquired model of 'typical' sequences of perceptions, which was independent of any particular task or environment (Fleischer and Marsland, 2002). These approaches relied on an acquired model of a previously explored environment and therefore are suitable for off-line selection of landmarks.

Approaches dedicated to on-line landmark selection purposes have also been addressed. In Zimmer's implementation, a topological map was created through a process of 'life-long learning' and can be continuously adapted on line by the robot during exploration. Based on the comparison of accumulated error statistics at each of the nodes, global statistical information was used to decide where to add and delete nodes in the map (Zimmer, 1996). In a related idea, Bourque and Dudek addressed the 'vacation snapshot' problem of deciding in which locations to take camera images in order to obtain a set of images that best represent an entire environment. This was done by keeping running statistics on what is called a 'typical' perception, as well as backtracking to previously visited locations that were subsequently found to be 'atypical' (Bourque and Dudek, 2000). Fleischer and Marsland's method differs from these approaches in that they only used local sensory information to decide when to add landmarks to the map. Related work on the problem of learning symbols to describe a robot's route through the environment was done by segmenting a robot's sensory flow into categories (Linaker and Niklasson, 2000).

In this paper, we provide a methodology for automatic landmark-based scene identification with the aid of a reinforcement learning based software package, called the working memory toolkit (WMtk), which allows reward associations between a target location and the conjunctive representations of its stimuli.

In the following, our computer vision system as a solution to the perceptual discrepancy problem is described in Section 2, which is followed by an introduction in Section 3 to the WMtk, whose two features are utilized for automatic landmark recognition in this work. Experiments are designed in Section 4 to explore the suitability of configural representation for the landmark identification problem in scene recognition and the results were compared with two competing systems, a simple linear two-class classifier, perceptron, and a support vector machine. Finally, conclusions are given in Section 5.

2. Our computer vision system

Natural images contain statistical regularities which can set objects apart from each other and from random noise. Perception is an awareness of things through the physical senses of these statistical properties. A primary goal of human visual system is to recognize and locate objects and to keep track of these objects as new images are captured. This is usually done by segmenting a real world scene into multiple objects each with their own characteristic color, shape, depth, texture and illumination. In this process, low level features such as color and simple texture can be regarded as a set of immutable, universal primitives used to infer high level semantic content of a scene. Similarly, computer vision is the science that develops a theoretical and algorithmic basis by which useful information about the scene can be automatically extracted

in the form of feature vectors and analyzed to infer which features are part of the object and which are not. To mimic human visual system, segmentation is therefore considered as an important but difficult task and provides the starting point for many subsequent computer vision problems. To successfully separate objects from each other, determination of distinguished features is a critical step. In the following, we describe our computer vision system which is inspired loosely from neuro-biological evidences.

2.1. Visual features

Color is an identifying feature that is local and largely independent of view and resolution. However, the color of a single pixel occupies such a tiny portion of an image that it may have little information by itself. In fact, a single pixel may even contain only noise, perhaps from a bad sensor in the imaging array. If we consider the color pixels in a simple $N \times N$ region of the image, then the histogram of these N^2 colors in the region can describe a color pattern which may contain considerably more information than a single pixel, as well as potentially be less sensitive to noise. Also, this color histogram description remains relatively independent of translation and rotation, and is somewhat independent of scale such as the change of distance to the object (Swain, 1991).

The key issue of color histogram based techniques is the selection and then the quantization of an appropriate color space. In color image processing, the most popularly used color models are the RGB (Red, Green, Blue) color space and the HSV (Hue, Saturation, Value) color space. The HSV color space is based on a warped version of the RGB color space and is directly related to intuitive color notions of hue, saturation and brightness. Due to the extensive study on content-based image retrieval, it has been reported that a color quantization scheme based on HSV color space performs much better than one based on RGB color space (Wan and Kuo, 1996; Serrano et al., 2004; Chen et al., 2007). Certain advantages of the HSV color space over the RGB color space are: (1) the HSV color histogram based techniques can provide better correspondence with human visual perception of color and are capable of automatically generating semantic concepts for images from their visual features; (2) the HSV color space offers improved perceptual uniformity and makes it easier to compensate for many artifacts and color distortion by isolating different aspects of color into different channels and thus decoupling the intensity of a color from the color information in the represented image; (3) in a viewpoint of computation time, using HSV color space is faster than using RGB color space.

Due to the facts that HSV-based features have an enhanced distinguishing power as well as correlate well with semantic concepts over RGB-based features, they are used in our approach in the construction of color histogram. It is well known that color histogram, as plain occurrence statistics of colors, has limited discriminative power. To incorporate spatial information, the image is usually divided into possibly overlapping subblocks and the combination of color histograms with other image features such as shape and texture in the same subblock should improve the overall image segmentation performance.

In image processing, the texture refers to the spatial distribution of the grey level that corresponds to the pixels of a particular region. Their characterization is especially relevant in cases where it is necessary to take into account the spatial content of the pixels. Multiresolution texture analysis, particularly the MSAR model (Mao and Jain, 1992), has gained wide acceptance over the years as an effective description of both local and global information. However, MSAR texture features are computationally intensive and replaced by the more efficient wavelet texture representation that can reduce the computational burden significantly (Serrano et al., 2002). However, the problem with some general wavelet

transforms is the shortcoming in their ability to decompose input image into multiple orientations, which limits their application to indoor environments which have clear line structure and large homogenous color surfaces. Due to their similarity to the receptive fields of visual cortical area V1 simple cells and their optimal localization properties in both spatial and frequency domain, Gabor filters (or wavelets) allow images to be decomposed into multiple scales and multiple orientations and are frequently used as texture-based measures in image processing and computer vision (Arivazhagan and Ganesan, 2003; Arivazhagan et al., 2006). As a result, we believe the combination of color histogram and Gabor texture measures is an excellent low level representation to use for segmenting images and identifying objects.

To obtain feature vectors from an image, a moving window of size $N \times N$ is used and shifted by M pixels in the row and column directions (Note that the window should not exceed the border of the image.). The moving windows are overlapping to allow a certain amount of fuzziness to be incorporated in the hope that they may help in obtaining a better segmentation performance. The window size controls the spatial locality of the result and the window shift step controls the resolution of the result. A decrease in the step gives rise to an increased resolution but an increased processing time.

To obtain the color features, we first convert the images (720×480) from the RGB color space to the HSV color space by:

$$H = \begin{cases} \left(0 + \frac{G-B}{\text{MAX}-\text{MIN}}\right) \times 60 & \text{if } R = \text{MAX}, \\ \left(2 + \frac{B-R}{\text{MAX}-\text{MIN}}\right) \times 60 & \text{if } G = \text{MAX}, \\ \left(4 + \frac{R-G}{\text{MAX}-\text{MIN}}\right) \times 60 & \text{if } B = \text{MAX}, \end{cases} \quad \begin{matrix} S = \frac{\text{MAX}-\text{MIN}}{\text{MAX}}, \\ V = \text{MAX} \end{matrix} \quad (1)$$

where MAX is the maximum value of (R, G, B) , and MIN is the minimum. Next, to make the color histogram less sensitive to illumination, we equally divide the normalized hue interval $[0, 1]$ into 100 bins, and the normalized saturation and value intervals $[0, 1]$ into 10 bins each. Each color can be represented by combining the three bins. All possibilities of the combinations equal 10,000 different color bins for the histogram. Then for each $N \times N$ window, the histogram can be constructed by looking at each color bin and finding the number of pixels that are defined in that bin. The total number of pixels in the selection region divides these 10,000 numbers, resulting in a highly sparse feature vector. The reason why we use such a high dimensional feature space is to differentiate colors as finely as possible.

First introduced by Gabor (1946), complex Gabor filters are complex exponentials with a Gaussian envelope and provide a mathematical approximation to the spatial receptive field of a simple cell in visual area V1 as

$$G(x, y, \theta, \lambda, \varphi, \sigma, \gamma) = e^{-\frac{\tilde{x}^2 + \tilde{y}^2}{2\sigma^2}} e^{j(2\pi\tilde{x} + \varphi)} \quad (2)$$

where $\tilde{x} = x \cos \theta + y \sin \theta$, $\tilde{y} = y \cos \theta - x \sin \theta$, θ specifies the orientation of the filter, λ specifies the wavelength of the sine and cosine wave and determines the spacing of light and dark bars that produce the maximum response, φ specifies the phase of the sine and cosine wave, σ specifies the radius of the Gaussian, and finally, γ specifies the aspect ratio of the Gaussian. Given each image, the convolutions of itself with each pair of the Gabor filters are calculated, followed by the computation of the magnitude of each complex Gabor filter. Next the results within each $N \times N$ moving window, as used in color histogram generation, are averaged to obtain a number of texture measures, which are then normalized by their sum for each image and appended to the color histogram to complete the formation of the feature vector.

In this work, the following set of parameters are used, $N = 15$, $M = 10$, $\theta \in \{0, \pi/8, 2\pi/8, 3\pi/8, 4\pi/8, 5\pi/8, 6\pi/8, 7\pi/8\}$, $\lambda \in \{\sqrt{2}, 2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2}, 5\sqrt{2}\}$, $\varphi \in \{0, \pi/2\}$, $\sigma = \lambda$, and, finally, $\gamma = 1$, resulting in 3337 10040-dimensional feature vectors for each im-

age. As will be talked about in the experimental part, we base our choices of these parameters on the preliminary results that give the best balance between the run time and the image segmentation performance.

2.2. Similarity measure

With the set of features being defined, the next step is to construct a multidimensional space of perceptual similarities and to ascribe to each physical stimulus a particular position within this feature space. To do so, a similarity measure should be defined to reveal the organization of the feature vectors and an algorithm should be designed to partition the input patterns into “sensible” clusters based on the adopted similarity measure.

In the multidimensional scaling literature, two metrics, called the city-block distance and the Euclidean distance and obtained by setting $r = 1$ or 2 in the following Minkowski power metric formula, respectively

$$d_{ij} = \left(\sum_{k=1}^K |x_{ik} - x_{jk}|^r \right)^{1/r} \quad (3)$$

are believed to represent two types of processing. As indicated by (Shepard, 1964; Garner, 1974; Shepard, 1987), for unitary or holistic stimuli, such as the hue, saturation and brightness of colors, the closest approximation to an invariant relation between data and distances has uniformly been achieved in a space endowed with the familiar Euclidean metric; while, for analyzable or separable stimuli, such as size and orientation, the closest approach to invariance between data and distances has generally been achieved with the city-block metric. To implement these suggestions, the Euclidean distance is used for the HSV color histogram and the city-block distance is used for the texture measure. Then our similarity measure is defined to be their sum. That is

$$d_{ij} = \sqrt{\sum_{k=1}^{10000} (x_{ik} - x_{jk})^2 + \sum_{k=10001}^{10040} |x_{ik} - x_{jk}|} \quad (4)$$

2.3. Percept formation

Each feature vector appears as a point in the feature space and patterns pertaining to different classes will fall into different regions of the feature space. The learning process is to partition the feature space into sensible clusters based on some properties in common. As a graph analysis of the feature vectors, the minimum spanning tree-based clustering algorithms are well known for their ability to detect clusters with irregular boundaries. A minimum spanning tree (MST) (Jain and Dubes, 1988) is a connected and weighted graph with no closed loops and has the property that any edge in it is the shortest edge across the two partitions which are connected by that edge. Therefore, the removal of the longest edges in order will theoretically produce separate clusters.

To begin this method, in our current implementation, an MST is constructed. Then the edges of the MST are sorted and removed in a decreasing manner until sensible clusters are formed. This results in 11 perceptual groups consisting of 65,798 labeled feature vectors in our database. Each group is assigned a different color for displaying purposes. Some important percepts obtained are listed in Table 1.

2.4. Fast database search tree

Though a pure nearest neighbor classifier is simple and accurate, for our large database, this process is quite time consuming and fast search facility is in need. Common structures used to

Table 1
Major percepts and their semantic meaning

Percept name	Percept symbol	Denoting color
BlackStripe	Object1	Black
LightWindow	Object3	Light Grey
WoodPanel	Object6	Dark Red
WhiteFloor	Object9	White
YellowFloor	Object10	Yellow
GreenBall	Object12	Green

organize data for efficient nearest neighbor search have been developed such as the K -dimensional tree (Bentley, 1975), R-tree (Guttman, 1984), Quad-trees (Finkel and Bentley, 1974), and Binary space partitioning (BSP) trees (De Berg et al., 1997). However, they work well only for dimensions less than 5 and the index structures break down for higher dimensions (Breuning et al., 2000).

For our large database consisting of very high dimensional highly sparse feature vectors, a three-way approximate nearest neighbor search tree is constructed as illustrated in Fig. 1.

Given the obtained database, at the first level, there is only one node, the tree root. At the second level, a set of 3 representative patterns are randomly selected from the whole database as the cluster centers. Then the whole database is clustered into three subsets by assigning each feature vector to its closest cluster center. At the third level, for each of the 3 clusters obtained at the second level, three feature vectors are selected randomly from the pool of feature vectors under it as its new cluster centers, resulting in totally 9 tree nodes at this level. This procedure continues until either all the feature vectors in a leaf node (the tree node that has no child nodes) belong to the same object class (a pure node) or the number of the feature vectors in a leaf node is below some limit, e.g., 50. Every feature vector has a class label associated with it.

Given a new feature vector, to search through the tree, its distances to the cluster centers at the second level are calculated and the winner is the nearest center. At the third level, the feature vector's distances to the three centers of the winner at the second level are computed and the corresponding new winner is selected. This process continues until a leaf node is reached. When a leaf node is reached, if it is pure, assign the associated label and stop. Otherwise, do a nearest-neighbor search over the vectors in that leaf node, assign the label that is associated with the winner and stop. For our database of size 65,798, the tree has so far at most 20 levels according to our experience. As a result, we have had at most $3 \times 20 + 50 = 110$ distance calculations instead of 65,798. It takes no more than 2 s to find approximate nearest neighbors for the 3337 feature vectors obtained from a newly grabbed image. This is a considerable reduction in computation, and similar quality results are obtained (this running time can be further reduced with faster computers).

The segmented images based on our database should yield a set of connected regions in the image. A connected-component labeling algorithm is then applied to find these regions, called blobs, which in turn can be used to produce working memory chunks. As shown in Figs. 2–5, two categories of images along the hallway,

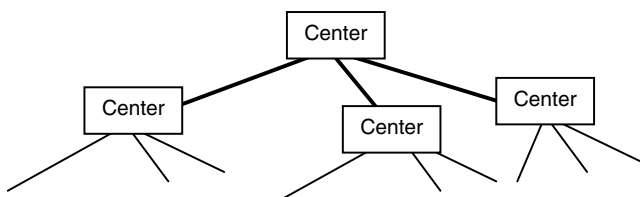


Fig. 1. Approximate nearest neighbor search tree.

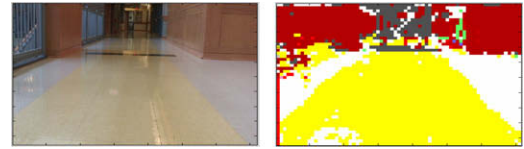


Fig. 2. (Left) image of non-target location (right) segmented version.

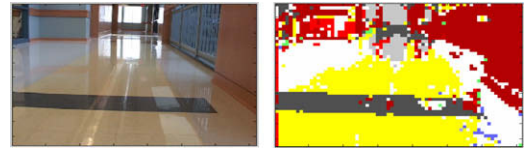


Fig. 3. (Left) image of target location (right) segmented version.

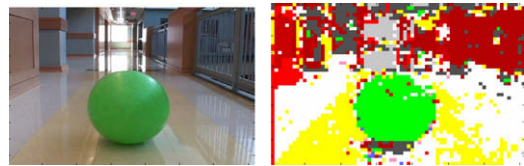


Fig. 4. Image with a distractor: (left) non-target location (right) segmented version.

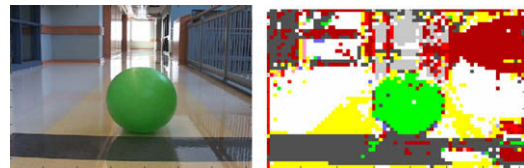


Fig. 5. Image with a distractor: (left) target location (right) segmented version.

denoted as non-target images and target images, were taken for our experiments. Their segmented versions based on the approximate nearest neighbor search tree are shown on the right sides of these figures. From the figures, we can see the approximate nearest neighbor search tree works well.

3. WMtk and its conjunctive coding

For higher animals such as humans, evidence suggests that the occurrence of a response depends on its predicted outcome and behaviors are planned on the basis of future possibilities rather than present contingencies alone (Butler and Hodos, 1996; Striedter, 2004). If the predicted outcome is a reward, the intended response in the form of a behavior is facilitated. However, if an inverse outcome is predicted, the response is strongly inhibited. Based on the same principle, the robot learning problem addresses the question of making a robot perform certain tasks in the world successfully. Suppose G is an identifiable goal state, X and Y are two different perceived states of the world, the learning problem can be formally described by

$$G : X \rightarrow Y : R, \quad (5)$$

which can be interpreted as, suppose the robot finds itself in a state satisfying condition X , if, in reaching the goal G , a state satisfying condition Y becomes active, a reward R is received (Nehmzow, 2003). If a task can be defined in terms of a set of such reward-receiving goals, a qualitative measure of robot performance can be the sum of the rewards it receives over time and the robot learning problem is then to improve robot performance through experience.

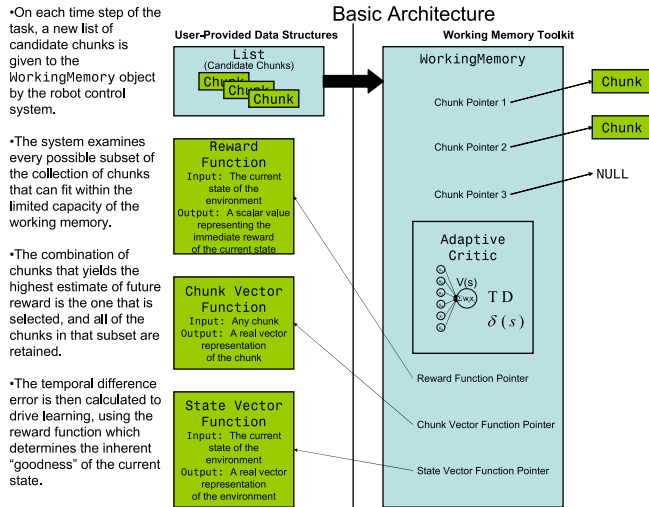


Fig. 6. A schematic illustration of the WMTk (Phillips and Noelle, 2005).

This kind of learning is called reinforcement learning (RL) in machine learning community and is based on the idea that relatively global reinforcement signals (i.e., reward and punishment) can drive learning that seeks to enhance reward and avoid punishment. As a particular type of RL developed by Sutton, temporal difference (TD) learning uses a way to predict the future time within a trial at which reinforcement will be delivered and then uses their predictions to optimize behavior when rewards are delayed (Sutton, 1998).

Based on TD learning, Dr. David Noelle and his PhD student, Joshua Phillips, at Vanderbilt University, created a set of open source software tools, called the NSF ITR Robot-PFC Working Memory Toolkit (WMTk), for developing working memory systems that can be easily and tightly integrated into robotic control system to perform goal-directed delayed-response tasks (Phillips and Noelle, 2005). Working memory system is a psychological model of the human short-term memory that accounts for a limited capacity system for temporarily storing and manipulating information used to control an ongoing behavior. There is biological evidence that the adaptive working memory (WM) structures, existing in primate brains, are important to the learning and performing of tasks by providing the embodiment necessary for accumulating rewards for those features most relevant to the current task.

The WMTk is a software library written in ANSI C++, which consists of a set of classes and methods for constructing a working memory system that uses TD learning to select its contents. Its three most important classes are the Chunk class, the FeatureVector class and the WorkingMemory class. More specifically, as a wrapper for the input information, the Chunk class is basically a simple data structure that takes a pointer to some memory location and a string as arguments and stores some arbitrary piece of information that is believed by the user to be useful for processing by the working memory. When presented to the working memory, the chunks are organized in the form of a feature vector. This vector is wrapped in the FeatureVector class, which provides the interface between the user and the working memory system. On the one side, the user provides Chunk and State information to the neural network. On the other side, the working memory system makes intelligent decisions about memory management, i.e., what to keep in the working memory and what to delete. Finally, as the most complex class in the toolkit, the WorkingMemory class is the workhorse of the WMTk. It is a limited store and supplies all the functionality to make intelligent decisions such as when its contents should be updated and/or protected (erased and/or retained).

In order for it to operate correctly, the required inputs to the WorkingMemory class include the size of the working memory, the size of the feature vector, the size of a state vector which defines the set of possible states the system will be in, the current state the system is actually in, the user-defined reward function, the user-defined state function, the user-defined chunk assignment function, and the user-defined chunk delete function (used to release the memory when no longer needed). A schematic illustration of the WMTk, which is taken from (Phillips and Noelle, 2005), is shown in Fig. 6.

When the working memory size is larger than 1, the toolkit creates a conjunctive code of all combinations of the chunks input into the working memory to mimic the conjunctive representation. For example, if there are three different kinds of input chunks, chunk type #1, chunk type #2 and chunk type #3, and if the working memory is of size 2, a conjunctive coding of totally 16 combinations of chunks is shown in Fig. 7.

After training, each network weight contains a value that corresponds to a particular combination of the WM content and the number of weights can be expressed as

$$(\text{Chunk_Vector_Size} + 1) \wedge \text{WM_size} \quad (6)$$

where Chunk_Vector_Size is the size of the input feature vector to the neural network, WM_size is the size of the working memory, and the extra "1" that gets added to the Chunk_Vector_Size denotes the reservation when nothing is chosen and the working memory is empty. When the working memory size is 1, Eq. (6) simplifies to the linear case, i.e., the input feature vector to the working memory has a single entry. It is these two features of WMTk, TD learning and conjunctive coding, that we use in this work.

4. Experiments and results

4.1. Experiments

Experiment 1: This experiment is to generate a model of the environment. The expected outcome is that the robot can learn the major percepts in this indoor environment and can subsequently segment the images reasonably well and fast. In this process, we particularly want to verify that the use of HSV color space and Gabor filters are the best features to consider when we deal with such a kind of indoor scenarios and to justify the values of some of the parameters we are using. To do so, we rely on a fixed image-partitioning scheme due to its simplicity over several proposals in the literature (Smith and Chang, 1996).

For clustering and unsupervised learning, the features and the similarity search algorithms should be selected in such a way that the intra-cluster distances among the feature vectors can be minimized while their inter-cluster distances can be maximized. However, since there is no way to visualize the spatial structures of the high-dimensional data, every solution to the unsupervised learning has its advantages and disadvantages. For example, though the MST-based clustering algorithms can detect clusters with irregular boundaries better than the well known K -means algorithm, they suffer from the chaining problem, as illustrated in Fig. 8.

Color features are used throughout the literature as an initial low-level feature. However, two completely different spatial orga-

0123	×	0123	
00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33

Fig. 7. An illustration of working memory conjunctive coding.

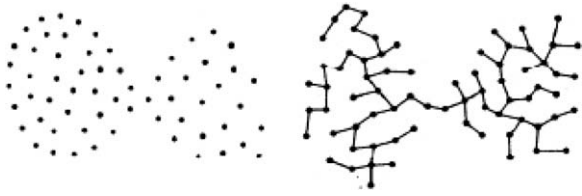


Fig. 8. The chaining problem of MST-based clustering.

nizations of color can give the same color histogram. For these cases, texture measures can be used to enlarge the inter-cluster distance between these two percept groups. For another example, the fixed image-partitioning scheme is simple and good for interior regions. However, they can introduce noise on the boundaries, which can produce the chaining problem for our use of the MST-based clustering algorithm. In other words, the goals of the use of the HSV color histogram and the set of Gabor texture measures as the low-level features and the selection of the window size and the shifting step in our implementation are for each percept group to be well separated from other percept groups in the feature space and for each percept to appear in its segmented image reliably and efficiently. Therefore, N and M should not be too large. Otherwise, some important percepts such as the black stripe tile percept and the window percept at one end of the hallway will not be reliably detected when needed. They can not be too small, either. Otherwise, the running time can not meet the requirement for on-line operation. So the choice of all the settings should be based on the efficiency and the reliability.

To find the best setting for these parameters, we tested four different window sizes and the corresponding shifting steps and three different sets of Gabor filters. The window sizes and shifting steps are summarized in Table 2. The parameters for three different sets of Gabor filters are summarized in Tables 3–5.

The first set of Gabor wavelets uses eight orientations (since indoor environments have more regular lines than outdoor environments) and five frequencies, resulting 40 complex (80 masks) wavelets. The second set of Gabor wavelets uses four orientations and four frequencies, resulting in 16 complex wavelet pairs. This set of Gabor wavelets have larger Gaussians and are much faster than the first set because the masks are smaller and there is less than half as many convolutions as the first set. The last set of Gabor wavelets is a modification to the first set intended to improve localization performance.

To generate a normal model of our indoor environment, 20 training pictures and additional 9 testing pictures are taken along the hallway. For each parameter setting, two MSTs are constructed

Table 2
Some testing window sizes and shifting steps

N	15×15	18×18	21×21	24×24
M	10	12	14	16

Table 3
Gabor filters set I

Symbol	Values
θ	$\{0, \pi/8, 2\pi/8, 3\pi/8, 4\pi/8, 5\pi/8, 6\pi/8, 7\pi/8\}$
λ	$\{\sqrt{2}, 2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2}, 5\sqrt{2}\}$
ϕ	$\{0, \pi/2\}$
σ	λ
γ	1

Table 4
Gabor filters set II

Symbol	Values
θ	$\{0, 2\pi/8, 4\pi/8, 6\pi/8\}$
λ	$\{4, 8, 16, 32\}$
ϕ	$\{0, \pi/2\}$
σ	2λ
γ	1

Table 5
Gabor filters set III

Symbol	Values
θ	$\{0, \pi/8, 2\pi/8, 3\pi/8, 4\pi/8, 5\pi/8, 6\pi/8, 7\pi/8\}$
λ	$\{\sqrt{2}, 2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2}, 5\sqrt{2}\}$
ϕ	$\{-\pi/4, \pi/4\}$
σ	$3\lambda/4$
γ	1

based on the feature vectors extracted from the training images for unsupervised learning, one for the color histograms and the other one for their combination with the texture measures. The core of each scene analysis technique is the segmentation process followed by object identification. The goal of the performance evaluation is to minimize the number of the subblocks with ambiguous labeling while maintaining computational efficiency, i.e., a reliably search tree. As illustrated in Fig. 9, the best classification results were generally obtained by combining the color features with the texture features using the parameter setting mentioned in Section 2.1. We may be stuck in a local minimum, but leave more extensive search to future work.

To infer high-level scene properties from the classification of the low-level image features and to show that configural representations rely on a unique conjunctive code of landmark stimuli and non-landmark contextual stimuli for landmark-based automatic scene recognition, the following experiments are designed.

Experiment 2: This experiment is to demonstrate that a robot can learn one or more percepts to identify a goal location in its environment without a priori knowledge of these representations from a human teacher. This basically is a classification problem that learns the correct correlation between the percepts and the target location. A specific location in a hallway is chosen as a target location to be taught to the robot. This location will have at least one clearly distinguishing characteristic, such as a stripe of black floor tiles. A set of 40 images, as typified by Fig. 3, are taken at this location. Each of these images is labeled as “target”. A second set of 40 images, as illustrated in Fig. 2, is obtained which have no view of the black stripe. Each of these images is labeled as “non-target”. These two sets of images are combined to form a labeled training set. Each image will be segmented into blobs from which candidate working memory chunks are obtained. This collection of blobs/chunks is duplicated so that we have two identical sets. All of the blobs in the first set are given the additional voting label of

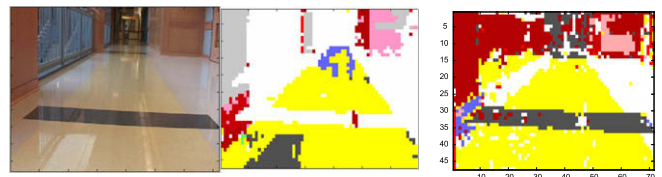


Fig. 9. (Left) original image; segmentation using only color histogram (middle); segmentation using both color and textures (right).

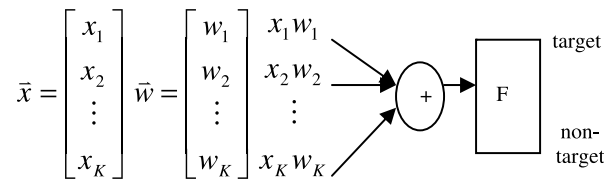
“landmark”, for example, Object1_L. All those in the second set are labeled “non-landmark”, for example, Object1_NL. Each training trial consists of two steps, a voting step followed by a reward-generating step. In the first step, the collection of blobs from the two sets is presented to the WM which selects up to L of them, where L is the size of the WM. As shown in Fig. 10, the robot will use the labels of the selected chunks to vote for a target or non-target label for this image. For example, if there are three percepts in the target image, Object1, Object9 and Object10, after the duplication operation, there are six chunks that are going to be presented to the WM, namely, Object1_L, Object1_NL, Object9_L, Object9_NL, Object10_L, Object10_NL. If, for example, Object1_L, Object9_L and Object 10_NL are chosen by the working memory, the final vote will be that it is a target location. In the second step, if the vote agrees with the label of the image, a reward of 1 is returned. Otherwise, a reward of 0 is issued. For another example, if Object1_NL, Object9_L and Object 10_NL are chosen, the final vote will be that it is a non-target location. Since it disagrees with the label of the image, the reward of 0 is returned. In the case of a tie (including 0 vote), the decision defaults to non-target.

Experiment 3: To further test the reliability of the WMtk, a set of 40 images is collected in which the black stripe is clearly visible in the foreground of the image. Also, in a majority (e.g., 80%) of these images, an irrelevant distractor, e.g., a green ball, is also present. These images are all labeled as “target”. A second set of 40 images is obtained having no view of the black stripe. The irrelevant distractor appears in a minority (e.g., 20%) of these images. The percentage may possibly be varied in multiple tests. Each of these images is labeled as “non-target”. These two sets of images are combined to form a labeled training set. Each image will be segmented into blobs from which duplicated candidate WM chunks are obtained. Then the same voting procedure as described in Experiment 2 will be followed to train and test the robot.

Experiment 4: In this experiment, we want to demonstrate that the conjunctive coding strategy can identify multiple key percepts with a target location. As mentioned in the introduction, landmarks have different effects on navigation depending on their distance to the animal: nearby landmarks specify the position of significant objects, whereas distant landmarks specify the animal’s direction (similar to a compass-based system). For example, in our environment, one end of the hallway has a window while the other does not. This can be seen in Fig. 3 and Fig. 2, respectively. Further, suppose our goal is for the robot to start from the dark end and move to the window end. In this process, the stripes of the black color tiles can appear repeatedly along the hallway. In other words, while moving toward the window end, multiple encounters of the black stripe and window combination should be chained to reach the goal location. So for this experiment, Fig. 3 indicates a target location for the robot to move toward (because it has both the black stripe percept and the window percept) while Fig. 2 should indicate a non-target location. Therefore, 40 pictures with clearly recognized black stripe percept and window percept (de-

noted by the light gray color in the segmented image) are collected as the target location and 40 pictures without such percept combination are collected as the non-target location for the WM system to learn. Based on this experiment, in future work, we will expect the robot to count the number of such combination while moving so that after multiple such encounters the robot can realize it reaches the goal location. These two sets of images are combined to form a labeled training set. Each image will be segmented into blobs from which duplicated candidate working memory chunks are obtained. Then the same voting procedure as described in Experiment 2 is followed to train and test the robot.

Experiment 5: A competing system, based on a single perceptron, is used as the first performance comparison for our Experiments 2–4. The feature vectors, \bar{x} , used as inputs to the perceptron are binary vectors denoting whether one or more blobs of a particular cluster category exists in the image and, therefore, have K -dimension.



$$o(t) = F(\text{net}(t)) = \frac{1}{1 + \exp(-\text{net}(t))} = \frac{1}{1 + \exp\left(-\sum_{i=1}^k x_i w_i - \varphi\right)}, \quad (7)$$

$$w_i(k+1) = w_i(k) + \alpha(\bar{o} - o)x_i, \quad (8)$$

$$\varphi(k+1) = \varphi(k) + \alpha(\bar{o} - o), \quad (9)$$

where $x_i = 1$, if there is one or more blobs of category i in the image, and $x_i = 0$, otherwise, $o(t)$ is the output of the perceptron, \bar{o} is the label of the image and φ is the bias. The perceptron will be trained on the same test image sets as used in Experiments 2–4. The results will be used as a comparison with those of the proposed system.

To further evaluate the learning capability of the WMtk, a support vector machine approach is used to do the same classification task as described for the perceptron. Being a set of related supervised learning methods belonging to a family of generalized linear classifiers for classification and regression, Support vector machines (SVMs) simultaneously minimize the empirical classification error and maximize the geometric margin. For two-class classification problems in an n - dimensional space, the SVMs will construct a separating hyperplane which maximizes the “margin” between two data sets, as illustrated in Fig. 11. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one “target value” (class labels) and several “attributes” (features). The goal of SVM is to produce a model which predicts a target value for each testing data instance when only the attributes are given.

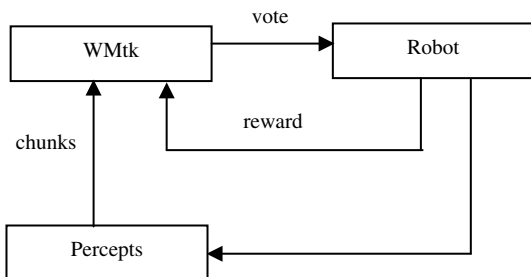


Fig. 10. The voting mechanism.

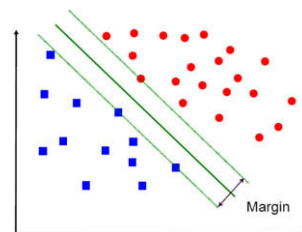


Fig. 11. Separating hyperplanes and the maximal “margin”.

Given a training set of instance-label pairs $(x_i; y_i)$; $i = 1, \dots, l$ where $x_i \in R^n$ and $y \in \{-1, 1\}^l$, the support vector machines require the solution of the following optimization problem:

$$\min_{w, b, \xi} \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i \quad \text{subject to } y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i \quad \xi_i \geq 0, \quad (10)$$

where the training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function $\phi(x)$. Then the SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function, four basic forms of which are

- linear: $K(x_i, x_j) = x_i^T x_j$
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r), \gamma > 0$
- radial basis function: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$.
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

where, γ , r , and d are kernel parameters.

Detailed description of the SVM approach can be found in (Burges, 1998) and will not be repeated here. In this experiment, we used the Matlab implementation of the SVM approach.

4.2. Experimental results

Each experiment consists of an 80-trial training period followed by a 5000-trial testing period. In the training period, target images and non-target images in each combined set are presented alternatively. In other words, all the odd-numbered training trials are from target images while all the even-numbered training trials are from non-target images, or vice versa. However, during the testing period, the images are selected randomly and presented to the WM. Table 6 shows the total accuracy of the WMtk after 5000 testing trails with 3 different working memory sizes and two exploration parameters (0.00 on the left and 0.05 on the right of each experiment) for Experiments 2, 3 and 4, respectively. From the table, it can be seen the percentage values include some misclassifications for the cases when the WM size is one and the exploration rate is set to 0.05. As will be talked about in the discussion subsection, the exploration parameter reflects the percentage of the noise introduced intentionally into the neural network to give other possible solutions some chances so that the final solution will not be stuck in a local minimum.

For working memory size of one, the neural network implementation inside the WMtk degenerates to the linear case, i.e., elemental percept coding. For working memory sizes of more than one, the conjunctive coding becomes possible and it can be seen that the performances increase largely after proper training. The poor performance of the WMtk with size 1 is because using the vote from a single percept to predict the label of a scene is not as reliable as using the vote from multiple percepts. In order to see the learning performance of the system, the plots of the average correct classifications over a sliding window of 25 points for 80 training episodes are shown on the lower part in Figs. 12–14. It can be

seen that the working memory system starts with random guesses, but, with training, it learns the correct choices in about 50 to 60 episodes (on average). Thus, the system is capable of learning both elemental and configural representations with the performances of the latter increased dramatically.

To see the performance of the WMtk improves with learning, its behaviors in the weight space are analyzed in the following. Though all the weights were initialized to be 0.25, they change after learning. Typical plots of the weight distributions for WM of size up to 3 are shown on the upper plots in Figs. 12–14 for Experiments 2, 3 and 4, respectively. It can be seen that, when the WM size is one, Object1 has the highest weight. This is intuitive because, when an image of the target location is presented after training, if Object1 is chosen by the WM, its associated largest weight makes it dominate the output calculation. For WMs of size larger than 1, two distinctive weight peaks appear for Experiments 2 and 3, and three distinctive weight peaks appear for Experiment 4. From the classification results, it can tell that one peak for each case corresponds to a conjunctive code for the images of a target location, and the other peak(s) corresponds to the non-target locations. The combinations of objects that correspond to the peak weights are listed in Tables 7–9 and most of them agree with the design intentions of these experiments. From these tables, we can see that the WMtk can identify the correct landmark(s) and realize the configural representation successfully when the size of the WM is larger than 1. From the results of Experiment 3, the presence of the distractor seems to have little impact on the learning system.

Next, the classification results of the perceptron and the SVM approach (using a linear kernel) over 5000 testing trials are summarized in Table 10. For the perceptron, the outputs are thresholded by 0.5. That is, if the output is larger than 0.5, the image is believed to be the target image. Otherwise, it is believed to be the non-target image.

It can be seen that the perceptron and the linear kernel SVM have done a better job than the WMtk when the WM size is set to 1. The weight distributions and the learning curves of the perceptron for Experiments 2–4 in the training period are shown in Figs. 15 and 16. They have similar structure to those of the WMtk with a WM of size 1. From the overall performance of the perceptron shown in Fig. 17, it undergoes a much longer (about 10 times longer than the WMtk) learning periods. The learning curves of the SVM for Experiments 2–4 in the training period are shown in Fig. 18. They are faster than the WMtk.

4.3. The running time analysis

Though currently we focus on research aspects, the ultimate goal of the research presented here is to implement the proposed system in a real robot for navigation in a real indoor scenario. In the current framework, we implemented all the algorithms in C++ (except the SVM approach) and use a laptop and a workstation computer that both run the Fedora core 4 Linux operating systems. The laptop computer has 2.5GHz CPU and 256MB RAM. The workstation computer is Intel Core 2 Duo Processor E6550 with 2.33GHz CPU and 2GB RAM. We use the timer utilities defined in the C standard library to report the execution time.

The first phase of our landmark-based scene identification is for the video camera (SONY Digital Handycam DCR VX2000 digital video camera recorder) to grab an image and send the data to the on-board computer. This phase can be done instantly. The next phase consists of image segmentation and chunks' extraction. Due to its limited resource, the laptop computer sends the raw image data to the workstation computer for fast processing. Then it takes the workstation computer no more than 1 second to compute the HSV color histogram. The extraction of the Gabor texture measures

Table 6

Percentage of correct location classification with two WM exploration rates for Experiments 2–4

WM size	Experiment 2		Experiment 3		Experiment 4	
	0.00	0.05	0.00	0.05	0.00	0.05
1	2461	2452	2512	2407	2477	2480
2	5000	4808	5000	4613	5000	4860
3	5000	4831	5000	4699	5000	4836

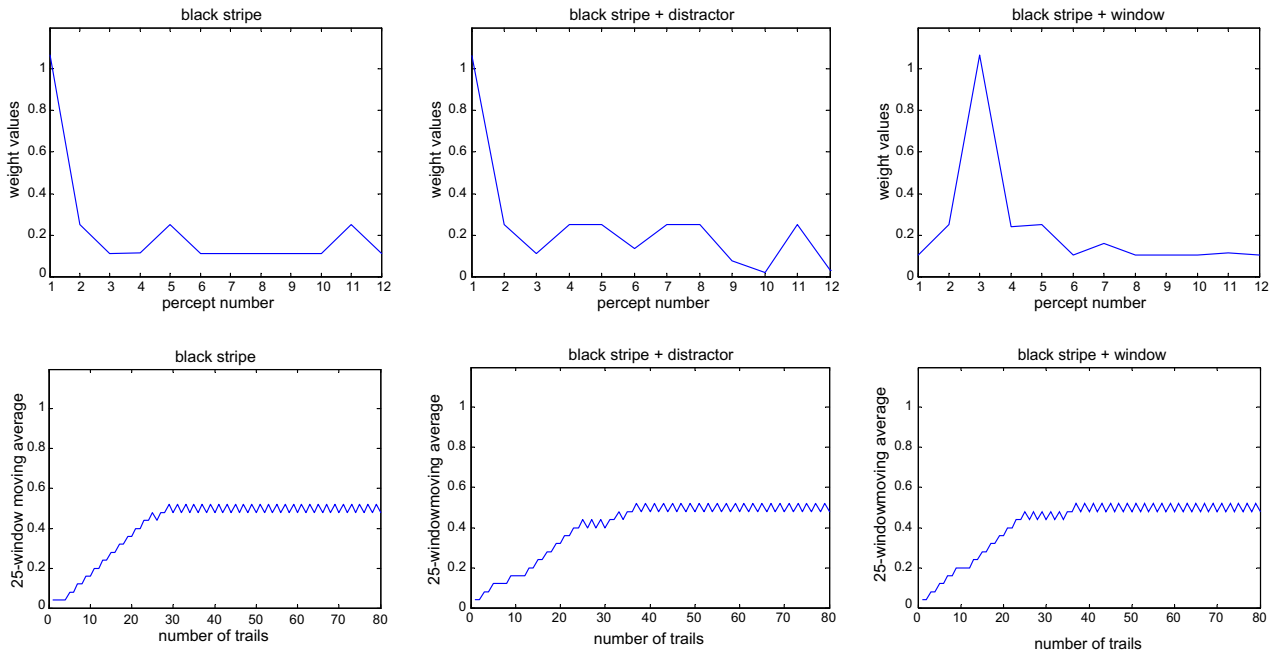


Fig. 12. The typical weight distribution (upper) and learning curve during the training (lower) for working memory of size 1.

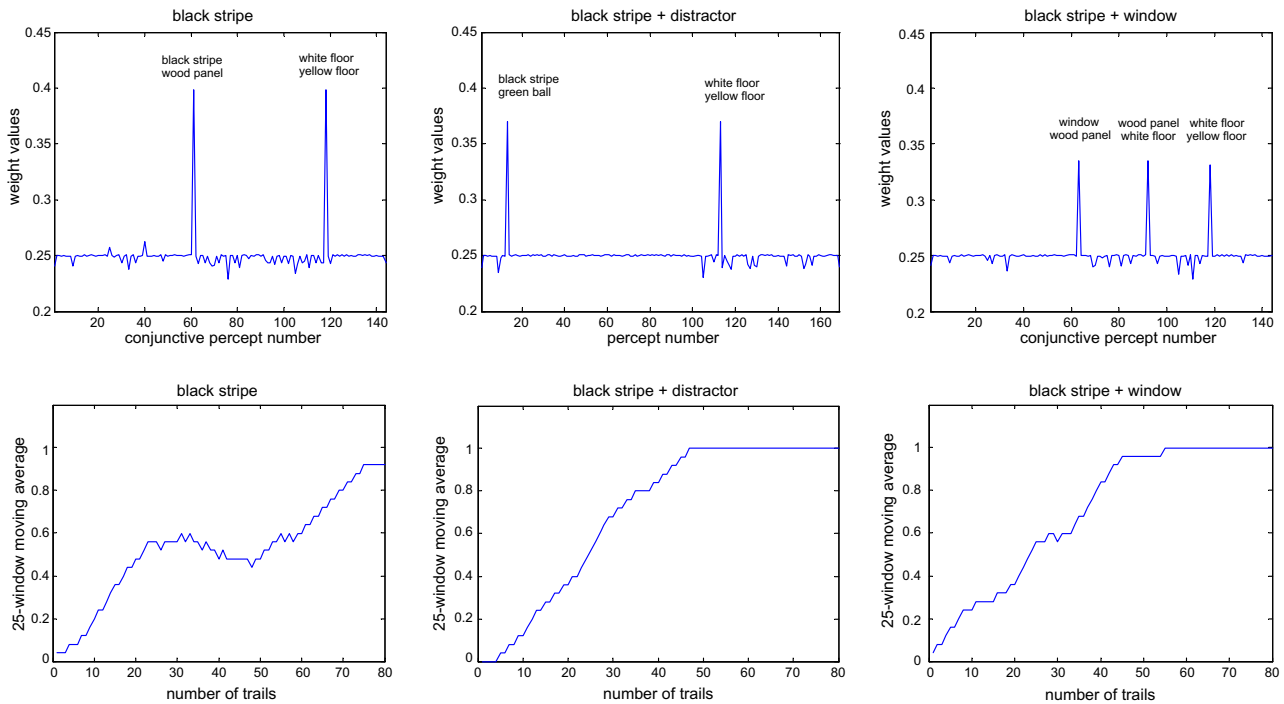


Fig. 13. The typical weight distribution (upper) and learning curve during the training (lower) for working memory of size 2.

takes about 10 s and thus dominates the total running time. This phase requires approximately about 14 s in our current implementation. Finally, the information about the extracted chunks is transmitted almost instantly back to the laptop computer, on which the WMtk is installed and used to identify a scene so as to control the movement of the mobile robot. The execution time of this last phase for each image depends on the size of the WM and, in our current setting, is about 0.0006 s for WM of size 1, 0.008 s for WM of size 2, 1.7375 s for WM of size 3 and more than 10 min for WM of size 4. Overall the execution time of the whole system

before any motion action is taken is about 18 s but can be reduced if a much faster workstation computer is available.

4.4. Discussion

In this subsection, we would like to discuss the potential features of the WMtk which can be used to address some of the issues existing in the experimental results which can be raised and outline its advantages over the perceptron approach and the SVM approach.

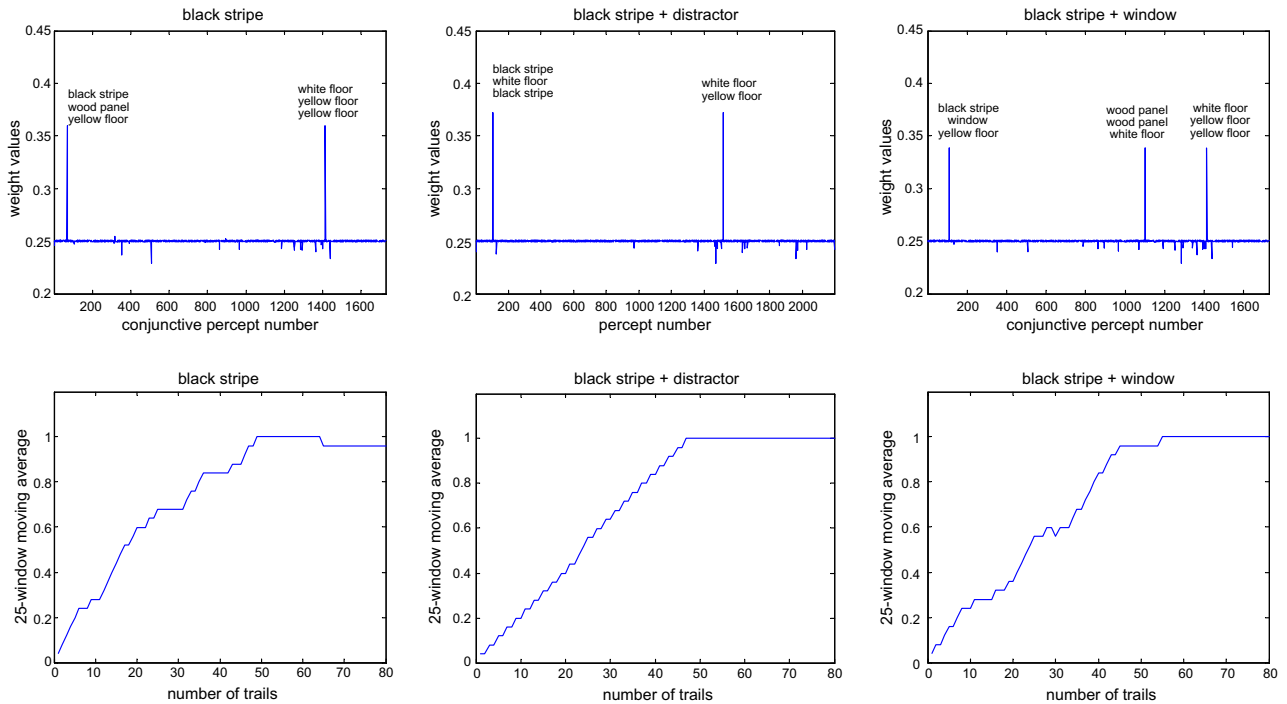


Fig. 14. The typical weight distribution (upper) and learning curve during training (lower) for working memory of size 3.

Table 7

WM conjunctive codes for peak weights in Experiment 2

	Exploration rate set at 0.00		Exploration rate set at 0.05	
	Peak 1	Peak 2	Peak 1	Peak 2
1	Object1	–	Object1	–
2	Object1	Object9	Object1	Object9
	Object6	Object10	Object10	Object9
3	Object1	Object9	Object1	Object9
	Object6	Object10	Object9	Object9
	Object10	Object10	Object10	Object10

Table 8

WM conjunctive codes for peak weights in Experiment 3

	Exploration rate set at 0.00		Exploration rate set at 0.05	
	Peak 1	Peak 2	Peak 1	Peak 2
1	Object1	–	Object1	–
2	Object1	Object9	Object1	Object9
	Empty	Object9	Object12	Object9
3	Object1	Object10	Object1	Object9
	Object1	Object10	Object10	Object9
	Object10	Empty	Empty	Object10

Table 9

WM conjunctive codes for peak weights in Experiment 4

	Exploration rate set at 0.00			Exploration rate set at 0.05		
	Peak 1	Peak 2	Peak 3	Peak 1	Peak 2	Peak 3
1	Object3	–	–	Object3	–	–
2	Object3	Object6	Object9	Object3	Object6	Object9
	Object6	Object9	Object10	Object6	Object9	Object10
3	Object1	Object6	Object9	Object1	–	Object9
	Object3	Object6	Object10	Object3	–	Object10
	Object10	Object9	Object10	Object9	–	Object10

Basically, the WMtk provides the basic algorithms required to learn, from experience, what needs to be remembered in a given situation and what can be safely forgotten. While the adaptive nat-

ure of the WMtk stems from the use of established reinforcement learning techniques, its use of these techniques is unusual. For example, the reinforcement learning mechanisms in the WMtk are not used to select an action to be taken by the system, but are used to select items to be remembered. Thus, the decisions made internally by the WMtk are separated from overt actions, and therefore from resulting reward, by the control systems that make use of the WM contents. These control systems can be arbitrarily complex, using WM contents in different ways in different contexts, which makes the learning task faced by the WMtk quite challenging, since a relatively complex relationship might hold between the WMtk’s decisions concerning memory updating and the whole system’s actions and eventual rewards. In other words, the WM does not learning what to do, but what to remember and where to focus attention. On the other hand, whereas most reinforcement learning methods use one way or the other to minimize the cost of actions taken, the WMtk sees the storage of an item as free of cost, up to the capacity of the working memory. Thus, the WMtk is biased to retain as many items as it can, up to the working memory capacity, until it determines that such retention obstructs the attainment of greater reward. This bias is implemented through a particular approach to the “exploration versus exploitation” problem – an approach which tries to retain novel combinations of the items in the working memory until experience proves otherwise. By mapping candidate collections of items to the reward expected if those items are remembered, the WMtk can opt to retain collections of items that it has never experienced before, selecting that particular collection from the combinatoric space of

Table 10

Percentage of correct location classification of the perceptron and the SVM over 5000 test trials for Experiments 2–4

# of correctness	Experiment 2		Experiment 3		Experiment 4	
	PCT	SVM	PCT	SVM	PCT	SVM
	4907	5000	4919	5000	4943	5000

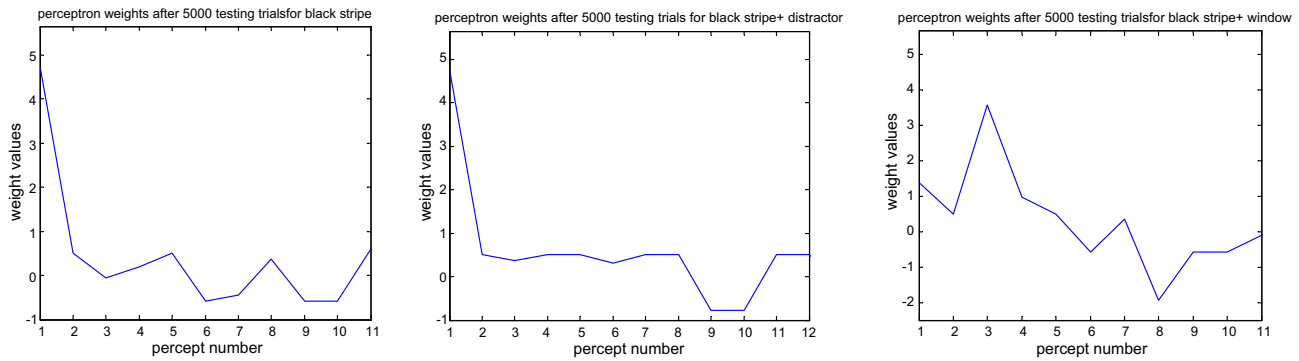


Fig. 15. The weight distributions of the perceptron for Experiment 2 (left), 3 (middle) and 4 (left).

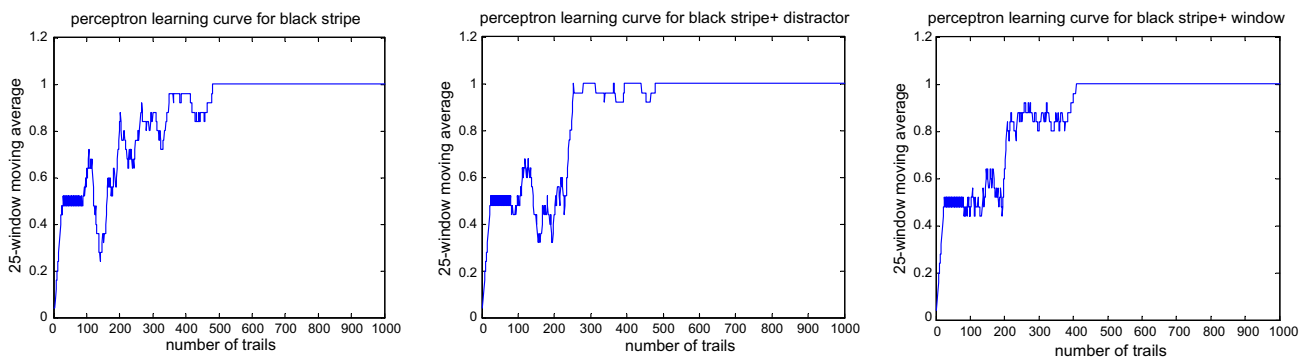


Fig. 16. The learning curves of the perceptron for Experiment 2 (left), 3 (middle) and 4 (left).

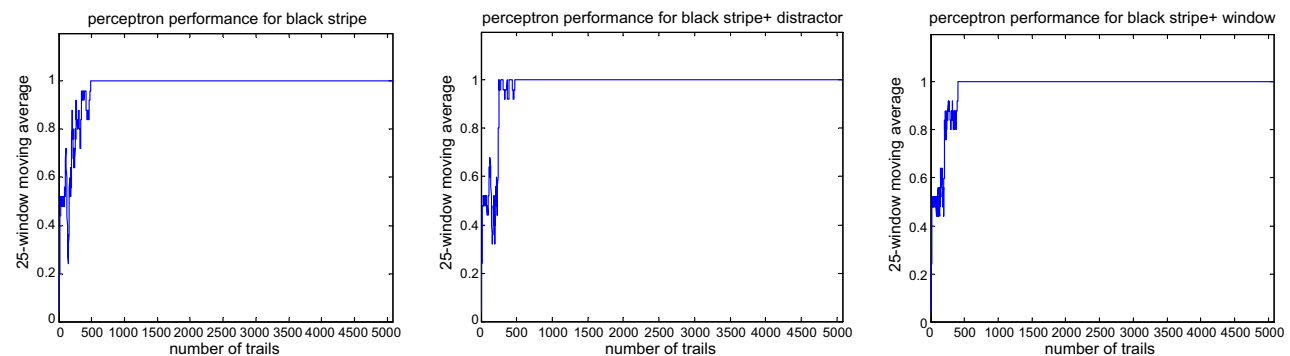


Fig. 17. The overall performance of the perceptron for Experiment 2 (left), 3 (middle) and 4 (left).

possible item collections to retain. In summary, the WMtk is biased to do more work rather than less, filling working memory to capacity unless it learns that this is non-rewarding and then to consider other collections of items to remember from a large combinatoric space. This directly explains why some classification errors happen when the exploration rate is set to 0.05.

The above discussion also explains why there are three weight peaks for Experiment 4 after training, one corresponding to the target image while the other two corresponding to the non-target image. This is because that wood panel, yellow floor and white floor percepts appear in almost all the training images and, with a limited size of the WM, it ends up with multiple patterns corresponding to the same outcome (i.e., non-target location in this experiment). This also can explain why Object6, rather than Object 1, appears in the final conjunctive code corresponding to the target location in Experiment 4 when the WM size is 2 and why there exist

duplicate percepts in the survived conjunctive codes of the final results. More specifically, the WMtk is biased to retain as many items as it can, up to the working memory capacity, until it determines that such retention obstructs the attainment of greater reward. As a result, as long as an item gets into the working memory, it will stay there until the punishments are accumulated to a point it has to be discarded. Since Object6 (wood panel) appears in majority of the images, it is really hard to get rid of it once it gets into the WM. A duplicated item has similar situations until that the corresponding percept is determined to be deleted from the WM.

Though the WMtk has better classification and running time performances than the simple perceptron, those of the SVM approach turn out to be a competitor. However, the WMtk not only learn what to remember but also learns where to focus attention. By surrounding target landmarks with a relatively small set of other contextual stimuli, the WMtk is able to reliably extract

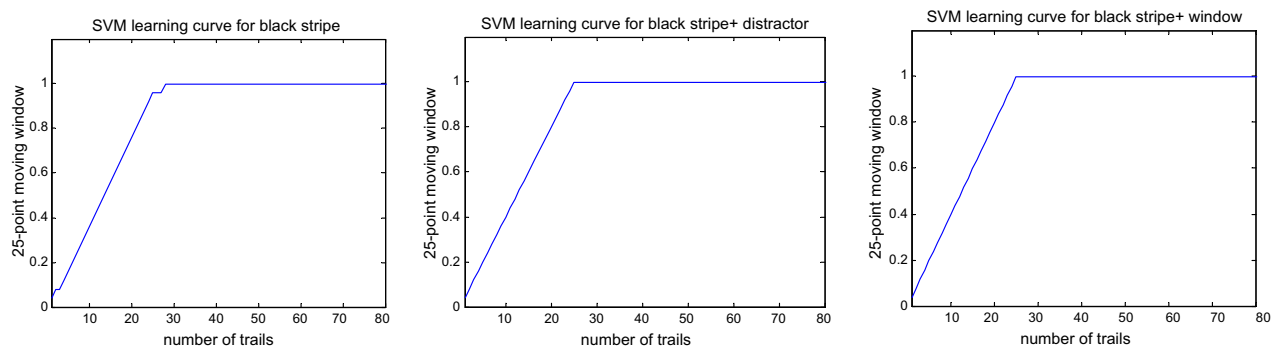


Fig. 18. The learning curves of the linear kernel SVM for Experiment 2 (left), 3 (middle) and 4 (left).

high-level semantic information from a scene that agrees with human intention, which is not always an easy task for the perceptron and the SVM.

5. Conclusion

In this paper, a non-linear model for automatic landmark-based scene identification is explored using the WMtk. The specific scenario that we are studying assumes that there is a target location the robot is able to recognize visually. Based on the experimental results, it can be concluded that the conjunctive coding using the WMtk can do a better job to focus attention even with the appearance of a distract percept. This makes things easier for the tasks that require visually-stable landmarks be identified and tracked so as to build a semantic map of the environment. Further, when compared with a simple neural network, perceptron, and a more involved SVM approach, the WMtk has a better performance than the perceptron and the SVM in the sense of its being capable of high level semantic information prediction in addition to being a good classifier. However, questions exist with regard to how to find the target location automatically and how to find a reward mechanism so that the peak weights correspond to combinations of non-empty distinctive percepts.

Acknowledgements

We would like to thank the National Science Foundation for its valuable support of this work under award 0325641 and Dr. David Noelle and his PhD student Joshua Phillips for the support of WMtk.

References

- Arivazhagan, S., Ganesan, L., 2003. Texture classification using wavelet transform. *Pattern Recognition Lett.* 24, 1513–1521.
- Arivazhagan, S., Ganesan, L., Padam Priyal, S., 2006. Texture classification using Gabor wavelets based rotation invariant features. *Pattern Recognition Lett.* 27, 1976–1982.
- Bentley, J.L., 1975. Multidimensional binary search tree used for associative searching. *Comm. ACM* 18, 509–517.
- Bourque, E., Dudek, G., 2000. On-line construction of iconic maps. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)*, pp. 2310–2315.
- Breuning, M.M., Kriegel, H., Ng, R.T., Sander, J., 2000. LOF: Identifying density-based local outliers. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*.
- Burges, C., 1998. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Disc.* 2, 121–167.
- Butler, A.B., Hodos, W., 1996. *Comparative Vertebrate Neuroanatomy: Evolution and Adaptation*. John Wiley & Sons.
- Chen, W., Shi, Y.Q., Xuan, G., 2007. Identifying computer graphics using HSV color model and statistical moments of characteristic functions. *IEEE Int. Conf. Multimedia Exp.*, 1123–1126.

- Collett, T.S., Fauria, K., Dale, K., Baron, J., 1997. Places and patterns—a study of context learning in honeybees. *J. Comput. Phys. A* 181, 343–353.
- De Berg, M., De Groot, M., Overmars, M., 1997. New results on binary space partitions in the plane. *Comput. Geom. Theory Appl.* 8, 317–333.
- Dusek, J., Eichenbaum, H., 1998. The hippocampus and transverse patterning guided by olfactory cues. *Behav. Neurosci.* 112 (4), 762–771.
- Fleischer, J., Marsland, S., 2002. Learning to Autonomously Select Landmarks for Navigation and Communication. In: *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*. MIT Press, pp. 151–160.
- Finkel, R., Bentley, J., 1974. Quad trees: A data structure for retrieval on composite keys. *Acta Inform.* 4, 1–9.
- Gabor, D., 1946. Theory of communications. *J. Inst. Elect. Eng.* 93, 429–457.
- Garner, W.R., 1974. *The Processing of Information and Structure*. Wiley, New York.
- Guttman, A., 1984. R-tree: A dynamic index structure for spatial searching. In: *SIGMOD'84, Proceedings of the ACM SIGMOD Conference*. ACM Press.
- Jain, A.K., Dubes, R.C., 1988. *Algorithms for Clustering Data*. Prentice Hall.
- Kortenkamp, D., Weymouth, T., 1994. Topological mapping for mobile robots using a combination of sonar and vision sensing. In: *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, Seattle, Washington, pp. 979–984.
- Linaker, F., Niklasson, L., 2000. Extraction and inversion of abstract sensory flow representations. In: *From Animals to Animates: The Sixth International Conference on Simulation of Adaptive Behaviour (SAB'00)*. MIT Press, pp. 199–208.
- Mao, J., Jain, A.K., 1992. Texture classification and segmentation using multi-resolution simultaneous autoregressive models. *Pattern Recognition* 25, 173–188.
- Mittelstaedt, H., 1983. The role of multimodal convergence in homing by path integration. *Fortschr. Zool.* 28, 197–212.
- Nadel, L., Willner, J., 1980. Context and conditioning: A place for space. *Phys. Behav.* 8, 218–228.
- Nehmzow, U., 2003. *Mobile Robotics: A Practical Introduction*. Springer, London; New York.
- Phillips, J.L., Noelle, D.C., 2005. A biologically inspired working memory framework for robots. In: *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*, Stresa, Italy.
- Rudy, J.W., O'Reilly, R.C., 2001. Conjunctive representations, the hippocampus, and contextual fear conditioning. *Cogn. Affect. Behav. Neurosci.* 1 (1), 66–82.
- Serrano, N., Savakis, A., Luo, J., 2002. A computationally efficient approach to indoor/outdoor scene classification. In: *International Conference on Pattern Recognition 2002*, Quebec City, Canada.
- Serrano, N., Savakis, A.E., Luo, J., 2004. Improved scene classification using efficient low-level features and semantic cues. *Pattern Recognition* 37, 1773–1784.
- Shepard, R.N., 1964. Attention and the metric structure of the stimulus space. *J. Math. Psychol.* 1, 54–87.
- Shepard, R.N., 1987. Toward a universal law of generalization for psychological science. *Science* 237, 1317–1323.
- Siegmart, R., 2004. *Introduction to autonomous mobile robots*. MIT Press, Cambridge, Mass.
- Smith, J.R., Chang, S.F., 1996. Tools and techniques for color image retrieval. In: Sethi, I.K., Jain, R.C. (Eds.), *Storage & Retrieval for Image and Video Database IV, IS&T/SPIE Proceedings*, vol. 2670, pp. 426–437.
- Striedter, G., 2004. *Principles of Brain Evolution*. Sinauer Associates.
- Sutherland, R.J., Rudy, J.W., 1989. Configural association theory: The role of the hippocampal formation in learning, memory, and amnesia. *Psychology* 17, 129–144.
- Sutton, R.S., 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Mass.
- Swain, M.J., 1991. Color Indexing. *Int. J. Comput. Vision* 7 (1), 11–32.
- Thrun, S., 1998. Bayesian landmark learning for mobile robot localization. *Machine Learn.* 33 (1), 41–76.

- Vlassis, N., Motomura, Y., Krose, B. 2000. Supervised linear feature extraction for mobile robot localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00), pp. 2979–2984.
- Wan, X., Kuo, C.-C.J. 1996. Color distribution analysis and quantization for image retrieval. In: SPIE Proceedings, vol. 2670.
- Wang, R.F., Spelke, E.S., 2000. Updating egocentric representations in human navigation. *Cognition* 77, 215–250.
- Wehner, R., 1983. Celestial and terrestrial navigation: Human strategies-insect strategies. In: Huber, F., Markl, H. (Eds.), *Neuroethology and Behavioral Physiology*. Springer-Verlag, Berlin and Heidelberg, New York, pp. 366–381.
- Wehner, R., 1992. Arthropods. In: Papi, F. (Ed.), *Animal Homing*. Chapman and Hall, London, pp. 45–144.
- Zimmer, U.R., 1996. Robust world-modeling and navigation in a real world. *Neurocomputing* 13 (2–4), 247–260 (Special issue).