A COMPUTATIONAL NEUROSCIENCE MODEL WITH APPLICATION TO ROBOT PERCEPTUAL LEARNING

By

Mert Tugcu

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

August, 2007

Nashville, Tennessee

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Overview

Many biological creatures, human beings included, learn from experience. In order to understand the underlying mechanism of such a magnificent system, engineers and scientists from various disciplines have put a lot of effort in this research area. In this learning process, evidence from the cognitive sciences shows that the working memory plays a crucial role, in part by fixing attention on the most relevant data. In particular, it has been shown that there are adaptive working memory structures in primate brains [1,2,3]. We believe that recently developed computational models of such systems may be useful for engineered systems.

In robotics, one important objective is the ability to teach the robot new skills and have it reason about the current tasks at hand without explicit programming. Indeed, this is central to open-ended development, developmental robotics and autonomous mental development. One way to accomplish this idea is to have the robot learn from its own past experience, which would take the programming of robots away from writing unnatural, large amounts of tedious code. Additionally, this allows the system to adapt itself to changing environments. Obviously, such natural and experience-based learning is widely used by human beings, primates, mammals and many other animals. Another desired characteristic of a robotic system is to facilitate the decision making process by focusing on the information needed by the current task and discarding the irrelevant data [4].

Main Objectives

The main overall objectives of this research are to develop a system, such that:

- The system supports research into developmental robotics (and open-ended developmental robotics), such that it can learn behaviors and concepts which can, in turn, be used to learn more complex behaviors.
- The system possesses a rich, powerful perceptual system, which can reliably discriminate percepts and provides robust performance in complex, unstructured and unmodified environments.

Many of the systems in the developmental robotics literature have only crude perceptual capabilities and as a result, the environments are usually very simplified by modifications, such as by employing artificial percepts. Such systems often fail in complex, uncontrolled environments, especially under different lighting conditions. One of our main goals is to build a perceptual system which can reliably operate in these types of environments to perform various tasks. In our system, we exploit the fact of having a reliable and powerful perceptual to support robust task execution. In fact, one of the concepts we discuss in the next chapter is that the use of overly simple perceptual systems may significantly hamper the performance of system development approaches, both the traditional modular approach and the developmental approach. A powerful and robust perceptual system can provide a much richer description of the environment and thus overcome other limitations. In fact, one possibility we consider is that the need for the deep integrated learning approaches of developmental robotics may be lessened by such a perceptual system to the point that the modular approach still has much utility.

At the very least, the system should be capable of learning the percepts and be able to robustly discriminate them. However, any visual system capable of operating in a complex, unstructured environment will encounter many percepts, and typically most of them are not relevant to the current task, and thus are distractors. This suggests the need for a capability to focus attention on the smaller number of items that are relevant to the task. Furthermore, support for developmental robotics implies that the system must use some kind of learning approach to learn to associate perception and action, and perhaps other concepts as well, in order to perform a task.

These two requirements suggest that prefrontal cortex working memory models may be a good fit for the learning portion of the system. In particular, we choose to use the "Working Memory Toolkit" as the computational working memory model for our system. This model has been developed by Dr. David Noelle and his Ph.D. student Joshua Phillips at Vanderbilt University, and the model will be explained in detail in the subsequent chapters.

Other than these main objectives, we would also like to support the following sub-goals in our system:

- Focus attention on the most relevant features of the current task.
- Support generalization of tasks without explicitly programming the robot.
- Guide perceptual processes by limiting the perceptual search space.
- Provide a focused short term memory to prevent the robot from being confused by the "out of sight, out of mind" problem.
- Provide robust operation in the presence of distracting irrelevant events.

To explore these issues, we propose a realistic system consisting of a perceptual system, actuators, reasoning, and short and long term memory structures. The proposed architecture is shown in Figure 1. The center of this system is the adaptive working memory module. An adaptive working memory does not function in an isolated manner. It has to be a part of a complex system; hence it was integrated into a robotic system in order to provide the embodiment required for exploring the issues of task learning [5]. It can be seen from Figure 1 that the system structure may be viewed as composed of five fully connected modules.

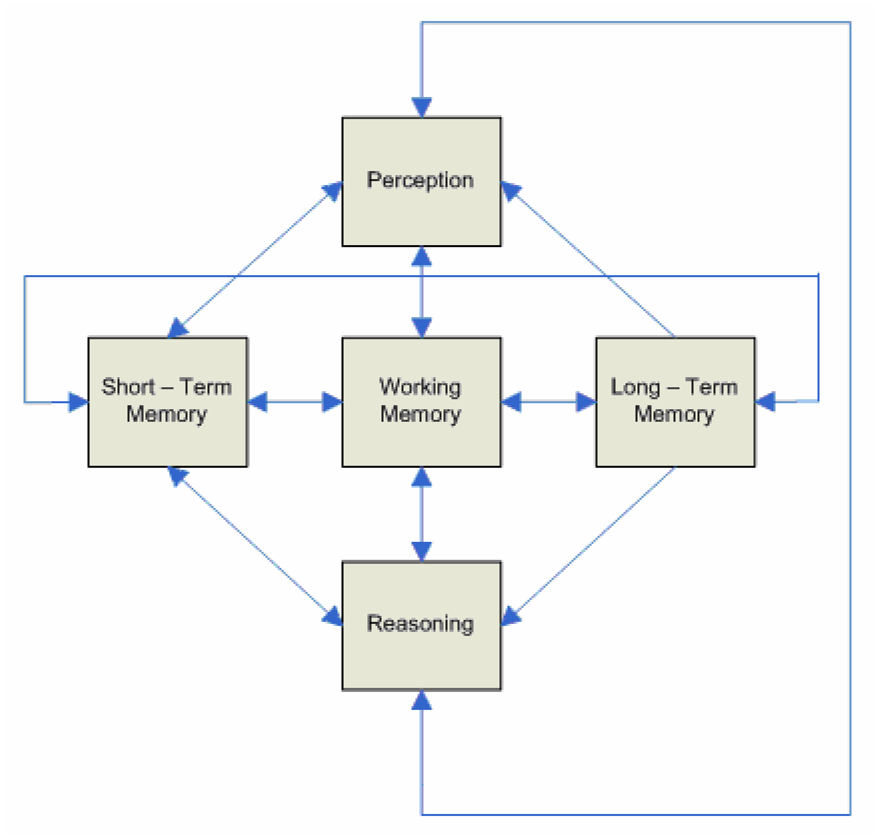


Figure 1. Proposed System Architecture

The short-term memory (STM) holds the data gathered within a recent time frame. The duration of this frame is not necessarily fixed but typically is at the scale of minutes. Furthermore, the STM is decomposed into two parts, a spatial STM called the Sensory EgoSphere (SES) [6] and another for non-spatial information. The long-term memory (LTM) is also divided into two parts, a spatial LTM called the Landmark EgoSphere (LES) [7] and another for non-spatial data, such as the necessary information on how to perform the tasks known by the robot. The LES basically holds data about objects that are permanent in the robot's workspace.

The spatial reasoning module is updated by all the other modules and provides the capability to reason on the spatial relationships among the objects the robot knows about. Essentially, it is responsible for reasoning about the objects currently within the robot's focus of attention, i.e., the contents of the working memory.

The perception system conveys data to the other, and its perception processes may be guided by these modules. The perception system identifies the objects and events within the sensory range of the robot using trainable classifiers. This part of the system is one input for the working memory, and may also provide some guidance to the reinforcement learning process of the working memory. A reinforcement learning algorithm is a machine learning algorithm, where the agent selects its actions based on its past experiences (exploitation) and its new choices (exploration). The goodness of the selected actions is associated with a reward signal, which encodes the success of an action's outcome. Therefore, the goal is to learn to select actions that maximize the accumulated reward over time. A type of reinforcement learning algorithm, Temporal Difference (TD) learning algorithm, will be discussed in Chapter IV. The perception system may detect an interesting object that is present in the environment of the robot and may give a signal to the working memory system in order to "explore" this interesting object. An important point here is that, reinforcement learning algorithms must carefully balance the "exploration" versus "exploitation" of previously learned knowledge in order to ensure the discovery of better choices, when they exist [12]. Therefore, by detecting an interesting object in the environment, the perception system can be an important factor for this balance issue.

Goals of the Perception System

It is a well known fact that the autonomous vision problem is extremely difficult and has not been solved in general, which results in robot vision systems which tend to be designed for specific types of problems and circumstances. Moreover, the results are often fragile, and once trained, it is often cumbersome to train the system to recognize new objects.

The system that is being developed for studying the working memory is expected to place a wide range of demands on the perception system. Moreover, the perception module has been designed as a general purpose module that can be utilized for other robot applications. Hence, we desired to develop a perception system with broad capabilities and the ability to learn new visual tasks. Some of the design criteria and guiding principles for the perception system are:

- The system should be taught simply by choosing and labeling examples from acquired images.
- The system should use categorization and classification methods to determine which features have strong identification capability.
- The system has the ability to learn *attentional* features for narrowing the focus of a visual search space.

- The system should be able to detect novelty, i.e., to detect that is viewing something that has rarely, if ever seen before.
- The system should not be used only in a specific environment but should be able to be used in many environments.
- If the system fails or is confused by an object that was taught to it (misclassification), it should be able to be corrected and trained immediately.

The details of the perception system will be discussed in the following chapters; however, at this point, it is worthwhile to emphasize some of the other flexible vision systems that have been reported in the literature.

In order to recognize various types of activities of mobile objects in a scene, a method, which is mainly an application of Bayesian Networks, is described in [14]. It is a generic automatic video surveillance application which detects and tracks moving regions and recognizes the type of mobile objects and their activities. Initially, several image features, such as width, height, the aspect ratio of the mobile object computed from the width and height, and color histogram, are computed from those regions. Additionally, some of the mobile object's properties are observed, such as angle, speed and distance from a reference object. Next, a single state scenario is defined which is characterized by the observed properties of the moving object. For example, a typical single state scenario, "*object A is slowing down towards object B*", may be confirmed by observing the direction and the speed of the object and the distance between the two objects. Since the input entities are statistically independent, a simple Bayesian classifier can be used to obtain the distribution of the combined result. If there is a hierarchy of scenarios, each layer in that hierarchy can be represented by a Bayesian classifier, and the whole hierarchy can be considered as a Bayesian belief network. The belief propagation is computed from the bottom layer to the top layer.

Another computer vision and machine learning system, which employs a statistical Bayesian approach, is described in [13], for modeling and recognizing human behaviors in a visual surveillance task. In order to detect and classify the behaviors of a single person and common interactions between two people, the system utilizes supervised statistical machine learning techniques. One of the goals of this system is to recognize new behaviors and construct a model with as little training as possible. Initially, the system detects and tracks objects in the scene. At first, the eigenspace model associated with the background (stationary part of the image), is computed. This is a model that provides an accurate estimate of the stationary parts of a scene. Once the model of the background image is computed from a sample set of N images, the segmentation of the moving objects is simply obtained by calculating the Euclidean distance between the input image and the projected image that is obtained from the eigenspace model of the background image [13]. Next, in order to track and model each pedestrian in the scene, 2D blob features, such as color, texture, motion and heading and the spatial relationships relative to all nearby moving objects, are computed. In order to model the behaviors and interactions, HMMs (Hidden Markov Models) and CHMMs (Coupled Hidden Markov Models) were used and the results are compared. Basically, a HMM is a probabilistic, state-based statistical learning approach where the system being modeled is represented with a number of discrete states. The CHMM is an extension of the HMM where the states are coupled via matrices of conditional probabilities which model temporal influences between their hidden state variables [13]. The graphical representations of both an HMM and a CHMM is shown in Figure 2. The behaviors are defined as primitive or simple behaviors and complex behaviors. For instance, a behavior could be "follow, reach and walk together" or "approach, meet and go on separately". The results demonstrated that CHMM modeling performs better than classical HMM in terms of both training efficiency and classification accuracy.

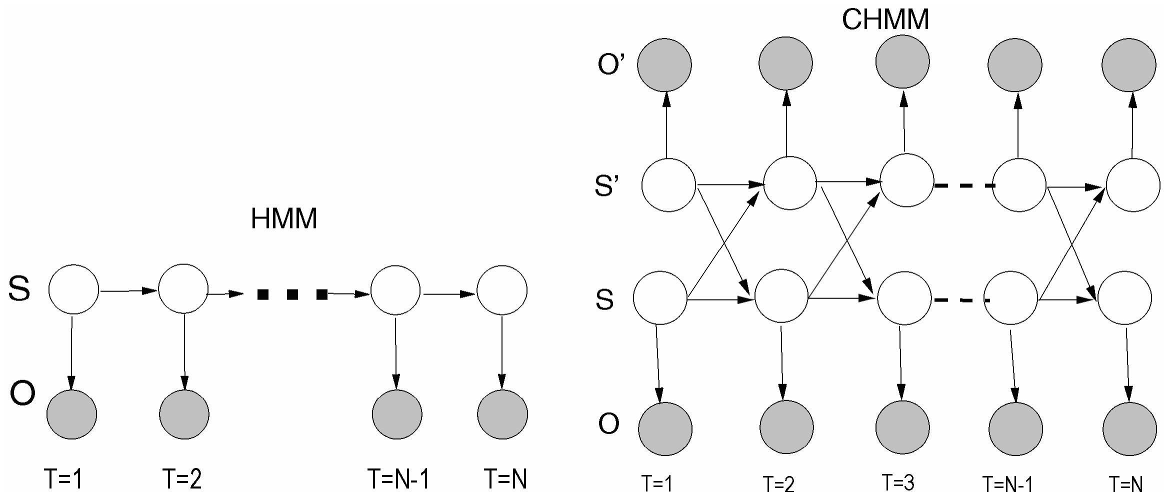


Figure 2. Graphical representations of HMM and CHMM [13].

In [15], the authors point out the necessity of a very large feature space, as well as a method that produces distinctive features from this space, in order to learn the distinctions between objects at various level of detail. For example, they proposed a system which generates primitive features which can be spatially combined into more complex, compound feature sets. Primitive features are referred to as the local appearance descriptors, such as the local intensity gradient and another description of the texture significance. However, primitive features themselves are not very distinctive. The compound features are defined as rigid geometric combinations of primitive features, defined by the angles and distances between them as shown in Figure 3. In order to recognize the objects, these feature sets are used to construct a bayesian network structure. One critical limitation of such system is that it is restricted to high-contrast edge, corner and texture information, as the authors agree on this point [15]. However, a complete model should also integrate color information, as well as blob-type features.

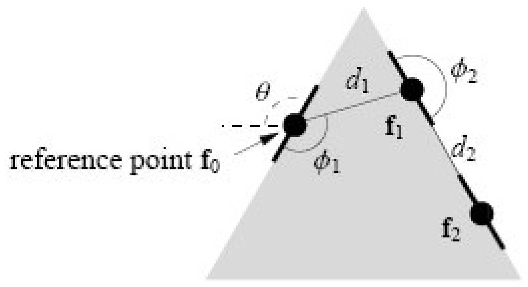


Figure 3. A compound geometric feature of order 3, from three primitive features [15].

Returning to our proposed research, the working memory module in Figure 1 acts as a central component of the system, in that it communicates directly with all other modules, acquiring necessary data from the perception system, passing data to and from the short-term and long-term memory modules, as well as being the primary source of information for the reasoning module. It retains only a limited amount of data (referred to as chunks), most relevant to perform the current task. This provides some means of deciding which information should be retained in the working memory as well as which should be discarded in the current context. Additionally, it also provides the effect of focusing the attention of the robot onto particular data. One way to design this selection mechanism would be to write task-specific procedures that represent the knowledge regarding the task relevance of candidate working memory objects. In this architecture, we have pursued a more general and adaptive approach. The working memory system of this architecture provides a mechanism, in which learning is based on experience and decides which chunks to maintain in memory. This learning mechanism is grounded in computational neuroscience models of the working memory circuits of the prefrontal cortex [4]. It has been discovered by neuroscientists that certain neurons in the midbrain cells communicates via the neurotransmitter substance, dopamine. The studies confirm that this neurotransmitter encodes for changes in expected future reward, suggesting that these cells in the midbrain may be involved in a kind of reinforcement learning based on predicted reward, such as temporal difference (TD) learning [8]. Neural instantiations of such reinforcement algorithms have shown how action sequences might be learned from the occasional delivery of a reward signal [9]. The extensive presence of the midbrain dopamine neurons in the prefrontal cortex has suggested that such a reinforcement learning scheme might guide the updating of the working memory contents, as well as the learning of motor sequences [10]. Based on the success of computational neuroscience models of working memory updating that utilize temporal difference learning [11], the working memory system of this robot architecture uses TD learning to determine which informational chunks to retain in memory. The working memory system will be presented in detail in a later chapter.

Summary and Organization

In this Chapter, a brief overview of the system and the goals that we would like to achieve in this work are presented. Our desire is to build a developmental system that learns everything from scratch incrementally by interacting with the user and the environment. The perceptual system plays an important role in this process; however, in order to focus attention onto the most relevant features of the current task, as well as to learn new skills, a biologically inspired adaptive working memory model is included. As a working memory model, the Working Memory Toolkit (WMtk) library, which was developed in the Electrical Engineering and Computer Science Department at Vanderbilt University by Dr. David Noelle and Joshua Phillips, has been chosen. This toolkit is the core of our implementation and is adapted to a robotic application. A complete robotic system was built in order to address the research issues that were mentioned earlier. The dissertation is organized as follows: In Chapter II, a discussion of the necessity of open-ended developmental robotics is made, followed by a review of the relevant work in the literature. Chapter III starts with a definition of perceptual learning, emphasizes some of the challenges and continues with a brief overview of the early stages of the human visual system. Chapter IV presents the concept of working memory, reviews some of the important models of working memory in the literature and introduces a biologically inspired working memory model and its implementation details. Chapter V addresses the proposed system architecture and its components in detail, starting from the model of the perception system and its capabilities. Subsequently, the chapter concludes with a discussion of the system's cognitive structure. Finally, Chapter VI presents the proposed methodology and experiments that demonstrate the utility and capabilities of this framework in a mobile robotic platform.

CHAPTER II

THE NECESSITY OF OPEN-ENDED DEVELOPMENTAL ROBOTICS

Overview

As mentioned in the first chapter, one of the objectives of this research is to build an open-ended developmental robot system, which can learn simple behaviors and build up more complex behaviors by utilizing the previously learned behaviors. Indeed, the ultimate, long term goal in robotics is to build a robotic system, which can develop new skills incrementally, in an autonomous open-ended approach through interactions with the environment without having a pre-designed, task-specific representation [48]. In general, traditional machine learning algorithms typically employ task specific methods and only the parameters pre-determined by the human programmer are updated. These methods often fail to respond to the dynamically changing states of the uncontrolled environments. Additionally, such methods may not represent a developmental entity, such as a human mind.

A relatively new area of study, called autonomous mental development (AMD) aims to fill the gap of how machines do not benefit from traditional machine learning and AI techniques. In [57], it was pointed out by J. Weng and his colleagues that a mental development process must be able to generate representation and architecture autonomously online directly from raw sensory signals and also must be task non-specific. Table 1 shows the basic differences between the traditional engineering (manual development) paradigm and the proposed mental development paradigm:

Table 1. Main differences between traditional and developmental programs [57].

Properties of Program	Traditional Programs	Developmental Programs
Sensor-specific and effector-specific	Yes	Yes
Program is task-nonspecific	No	Yes
Tasks are unknown at programming time	No	Yes
Generate representation automatically	No	Yes
Animal-like online learning	No	Yes
Open-ended learning of more new tasks	No	Yes

The traditional learning programs in Table 1 also include traditional artificial neural networks. Weng argues that due to the fact that a neural network can only accept a fixed size, offline sensory data vector, the development of something as flexible as a mind is not possible. However, a developmental robot, which runs a developmental program, has the ability to add new data that it has never seen before, online, in the real physical world. Furthermore, it can develop simple, basic skills for many different tasks and these skills may be utilized to learn more complex skills. Another practical advantage of using developmental programs is the ease of the training stage. A human child's brain is not manually trained with data with one training session. Instead, the child is taught by its parents from time to time by approving or rejecting behavior. A developmental robot should be trained in the same way, by interacting with the environment and with the user, just like a student-teacher relationship.

Why is there a necessity for open-ended developmental systems? As mentioned previously, the target domain in which robots should be able to operate is the real unconstrained world, where the environment is highly complex and dynamic. It is obvious that the human designer cannot predict every possible situation in the initial stage of programming. Therefore, the system should have the capability to cope with unexpected changes in the environment. The environment also has a significant effect on the complexity of a task. In psychology, the term “muddy” is often referred to describe concepts that cannot be clearly defined or specified [48]. According to Weng and Hwang [48], the complexity or "muddiness" of a task can be considered as a product of muddiness measures in five individual categories: (1) external environment, (2) input, (3) internal environment, (4) outputs, and (5) goal. Weng and Hwang[48] uses the term “muddy” as a general term to measure how much intelligence is required to execute a task. A task can be considered extremely complex or “muddy”, if all the five categories have a high measure. For instance, a visually-guided navigation task could be considered to have relatively high complexity in (1), (2), (3) and (5), but moderate in (4).

Our approach to the problem

In both the working memory and developmental (AMD) robotics communities, there seems to be a particular interest in learning strategies that support a type of deep integrated learning. This is in contrast to a more modular and hierarchical learning and categorization of behaviors/skills that is often found in more traditional AI approaches. While we certainly do not disagree with the developmental robotics and working memory communities on this, we believe that a somewhat modular approach still has a deal of practical utility, especially for robots.

In particular, we believe that some past limitations of a modular approach may be related to the use of an overly simple, impoverished, perceptual system that is incapable of describing realistic complex environments. Thus we propose to use a modular approach to behavior/skill learning combined with a powerful perceptual system.

The system learns basic, simple behaviors, which are needed to learn more complex behaviors with the working memory model. The perceptual system learns about the percepts and how to discriminate them. However, the system does not know what these percepts mean, so therefore, using the working memory, the robot learns the meaning of the percepts with respect to movement. This behavior, move to open space, is a very simple behavior and in this behavior, the percepts are associated with the task of motion. However, this behavior is crucial for a navigation task for a mobile robot. Once this association is complete, the system encapsulates this behavior. The robot learns and encapsulates other behaviors and skills that are necessary for a complex task. However, the automatic encapsulation problem is not addressed in this work. This problem by itself is an important problem and should be addressed in future work.

Review of Related Work

This section provides a brief overview of some of the efforts that contribute to the area of developmental robotics. In general these methods, as well as the method presented here, are inspired from biology, neuroscience and psychology. First, one of the earliest methods for the field of autonomous mental development will be addressed. Then, other relevant efforts from the literature will be summarized.

It was first pointed out in [48], that many traditional machine learning algorithms, including artificial neural networks, are *computational frameworks* not *developmental frameworks*. According to Weng, two types of paradigm exist: The first one labels the traditional machine learning methods as *manual development*, where the programmer understands the task (not the machine), chooses a representation, and maps the task to the representation. Thus, the programmer conveys his understanding into the task representation. Next, the programmer writes code which controls the machine to execute the task using the task-specific representation. The second paradigm is called the *autonomous developmental paradigm* where the programmer writes a developmental program and the robot learns the task by interacting with the user/teacher and the environment. The result of such a mechanism should be that the robot generates its own internal representation.

Prior to our study of Weng's work, we had developed a tree structure which efficiently carves the pattern space into finite regions. In other words, the tree structure is a classification method, which divides the pattern space into many small subspaces. This hierarchical structure comprises some of the developmental part of the system and achieves logarithmic time complexity. The details of this search tree structure will be explained in detail in the subsequent chapters. Weng and Hwang [59] developed a different tree structure called the "Incremental Hierarchical Discriminant Regression (IHDR) engine". Weng argues that the system must deal with high-dimensional feature space. In order to demonstrate the proposed method, a vision-guided navigation experiment, which uses the IHDR tree system, is presented in [48], [49], [59]. A mobile robot, called SAIL, developed at Michigan State University is trained by pushing it around the corridors of the Engineering Building. The robot has two pressure sensors on each side. The difference of the two pressure readings is translated into heading information and associated with the images. The grayscale intensity of the pixels is used as the components of the feature vectors and the resolution of the images is rather low, 30 x 40. Therefore, the number of dimensions on which the system operates is 30 x 40, i.e., 1200, and only one feature vector is generated from an entire image. Linear discriminant analysis, which is a parametric technique, is applied to the feature vectors to select the most discriminative features. Our system uses high resolution images and we obtain more than two thousand vectors from one image. Additionally, our system does not rely on any parametric methods, and exploits the use of a sparse vector representation which we have found to be a very important quality. This representation will be explained in detail in Chapter V.

The methodology proposed in this dissertation, which focuses on issues of developmental robotics, is applied to a complex vision-guided navigation task. Part of the complexity of the task is that no a priori information is given to the system, such as a cartesian map or coordinates of objects, etc. Instead, a qualitative navigation approach is taken, where the robot proceeds by detecting natural landmarks that are present in the environment. The system learns everything from scratch, including what objects represent a landmark, using both supervised and unsupervised learning. In order to achieve the navigation task, simple behaviors are learned and preserved in the system in order to learn more complex behaviors. This type of learning/teaching approach is also known as "shaping" and is extensively used in the animal experiments in psychology [58]. The desired animal behavior is achieved by initially learning a very simple task, followed by similar, slightly more difficult tasks. Thus, the animal is trained to perform a complex behavior in stages with different reward strategies for each training phase. In [50][51][52], a computational model of such a conditioning process is presented as an extension of reinforcement learning.

In [50][51], the model is applied to a very common task in cognitive neuroscience called the Delayed Match to Sample (DMTS) task to study working memory in rats and monkeys, and simulation results are presented. Figure 4 shows the configuration of the simulation environment called the "Skinner box".

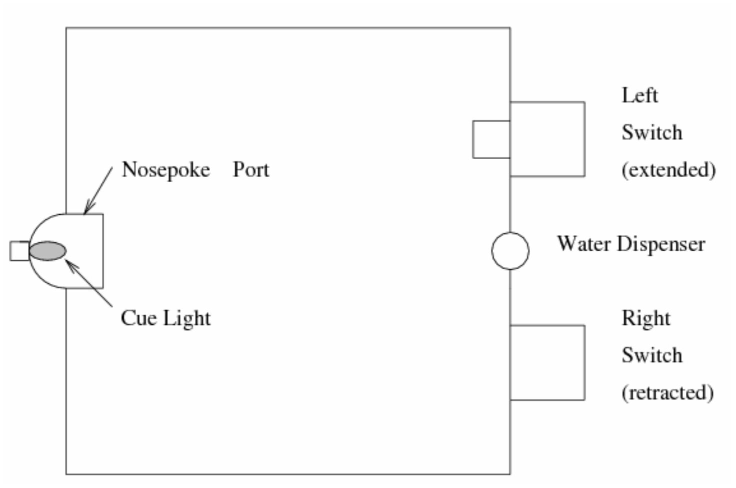


Figure 4. Skinner box configuration for the DMTS task [50][51].

Initially, at the start of a trial, only one switch extends and the rat must press that switch. Next, the cue light turns on and the rat must turn around and go to the opposite wall and make repeated nosepokes. The purpose here is to move the rat away from the switches. The rat stays there for a variable delay period averaging a minute. Once the cue light turns off, both switches are extended and the rat must go back to the switch that was pressed previously. In order to receive a water reward from the dispenser located between the two switches, the rat needs to remember which switch was pressed in the delay period and choose the correct switch. The task is composed of multiple stages; hence, both the rat and the model need to be trained in several stages with different rewards that govern the behavior for each stage. Initial robot implementation is also presented in [50], however it is mentioned that the system uses only a crude vision perceptual system to demonstrate the capabilities of the learning model on the robot. In [51], a simple color detection algorithm is used to teach the robot to sort the objects based on their colors into bins using the proposed model. However, the need to add facilities for refining the perceptual space with experience is also mentioned in [51]. In their robot implementations, a human trainer has to be present all the time and press a button for each correct action that the robot chooses.

Krichmar and Edelman [53], argue that devices, based on the working principles of the nervous systems, may provide the groundwork for the development of intelligent machines which operate on neurobiological principles rather than computational ones. They have designed a series of theoretical models, called brain-based devices (BBDs), and applied them to physical robots in order to investigate the functionality of different regions of the brain. The robots equipped with these models are used for studying perceptual categorization, operant conditioning, episodic and spatial memory formation and motor control. They emphasize that models of brain function should reflect the dynamics of different brain regions, their structure and the connectivity of these regions, rather than implementing them as a single neuron layer. According to them, to construct such models, detailed neuroanatomy and neural dynamics should be taken into consideration. Another argument given in [53], which supports the developmental robotics discussions presented in this chapter, is that a real environment for studying a behavioral task is crucial. Real environments are rich in information, multimodal and noisy, thus simulating (artificial design) every aspect of such an environment would be computationally expensive or even may not be possible.

In order to investigate the functional anatomy of the hippocampus region and its surrounding regions, Krichmar et al. [54] programmed a physical robot to solve a dry variant of the Morris water maze task. In this task, the robot uses its spatial and episodic memory by associating perceptual cues in the environment with a particular location. This task is also used as an evaluation of spatial learning in rats. The experimental environment is shown in Figure 5. The target location, which is the hidden platform shown in Figure 5, can only be detected in close range by the robot's front IR sensor. The robot starts a trial in one of the four locations shown in Figure 5 (a) with numbers, and stops if it detects the hidden platform or until a time limit of 1000 seconds is reached. In the former case, the robot receives a reward. The experimental results show that the robot learned to navigate to the target location from multiple starting positions in about 8 trials.

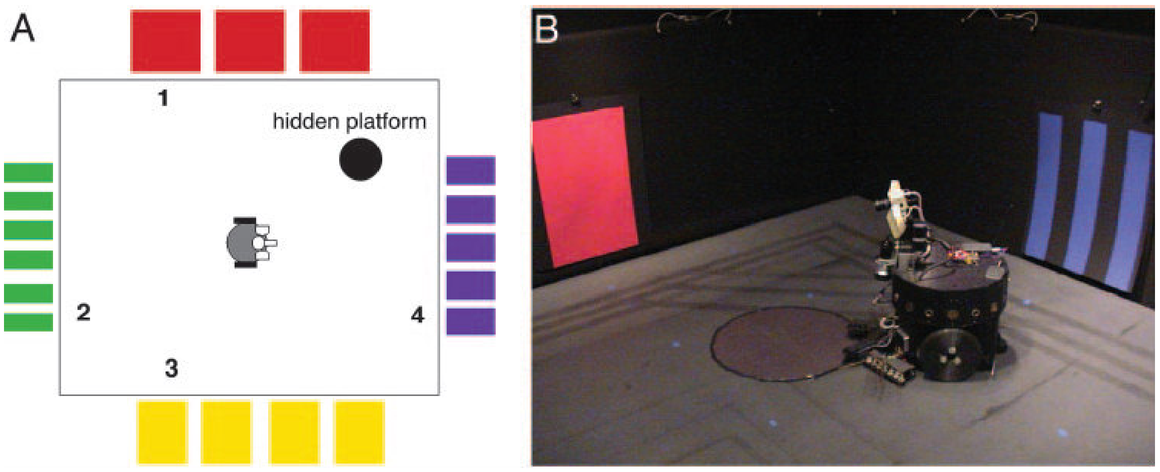


Figure 5. (a) Schematic of the experimental environment. (b) Snapshot of the robot (Darwin X) used in the experiment [54].

A slightly modified version of the task, inspired by Krichmar's work, is presented by Bursch, et al. in [55], in a simulation environment, shown in Figure 6. In contrast to Krichmar's experiment, the robot starts a trial in random locations and does not use odometry information. In order to create a mapping between a perceptual state and an action for that state, a Self-Organizing Feature Map (SOFM) [56] is used and the heights (in pixels) of the colored panels are presented as an input to the SOFM. Two navigational approaches are proposed and compared. The first method is based on creating a probabilistic graph between the nodes of the SOFM and then searching for the optimum path in the graph in order to navigate to the target position. The second approach utilizes the Working Memory Toolkit (WMtk), which we also choose as a working memory implementation for our system. It was reported that the first approach outperforms the WMtk with relatively little experience. However, as the number of training samples increases, the WMtk actually performs much better than the graph search technique.

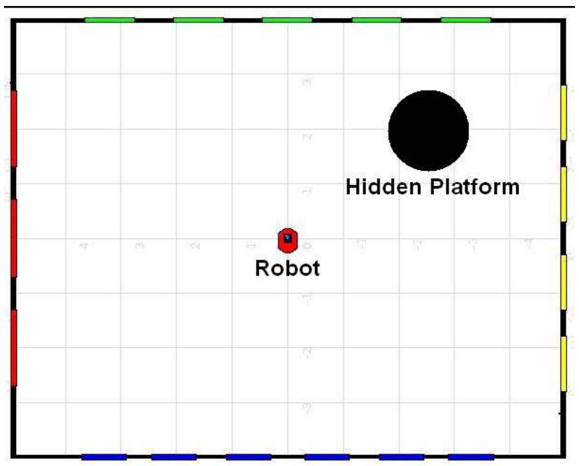


Figure 6. The simulation environment used in [55].

Summary

Traditional machine learning algorithms are usually targeted to small problem domains and use pre-defined representations. These techniques are prone to problems due to the limited representation of the environment. The methodology proposed by this dissertation follows the constraints for a developmental system and aims to contribute in the research field of developmental robotics.

The system is capable of learning new skills and percepts online in uncontrolled environments while performing a task and supports explanation of its knowledge to the user in an interactive way. In order to perceive such a dynamic, noisy and uncontrolled environment, the system uses a large number of very high-dimensional but sparse feature vectors, which can be easily generated on a single training session. However, the sparse vector representation is directly encoded into the system, therefore, the high number of dimensions has somewhat less effect on computational cost. As the training data set grows larger, the system quickly adapts to the dynamic changes in the environment, such as the changes in lighting and perspective.

CHAPTER III

PERCEPTUAL LEARNING

One of the most important functions of the brain is certainly the learning mechanism. The term "learning" literally corresponds to "knowledge or skill acquired by instruction or study" [16]. On the other hand, the definition of perceptual learning by Gibson is "Any relatively permanent and consistent change in the perception of a stimulus array following practice or experience with this array will be considered perceptual learning" [17]. A substantial amount of research has been conducted on perceptual learning since the early 1990's, suggesting that perceptual learning happens in the early stages of the brain, such as the primary visual cortex [18].

Perceptual learning should be considered as a part of abstract or cognitive learning for any real-time or artificial system. If considered from the Artificial Intelligence (AI) point of view, perceptual learning should be considered as an active process that embeds a particular abstraction, reformulation, and approximation within the abstraction framework [19]. That is to say, perceptual learning is a particular abstraction layer which focuses only on the relevant data that is needed by the current learning task.

According to the previous studies in neurobiology and machine learning, perceptual learning has been considered to occur at the cognitive level where identification and categorization is performed. Humans perform learning at the perceptual level for many real world tasks. In order to make the categorization task possible, the very basic features of the objects, such as color or shape information, are obtained in the perceptual level. Additionally, in any learning task, it is well known that there is a strong interaction between perceptual learning and cognitive learning.

One of the most challenging problems for neuroscientists is the object recognition problem. It is a very complex problem domain and has not been solved in general. The key point in this problem domain is prediction or *generalization,* which is the ability to apply knowledge from the past experiences to a new condition. For instance, the perceptual system should be able to generalize in a way that enables it to recognize objects despite large variations in appearance within the environment. In general, perceptual systems are very vulnerable to light or brightness changes, and occlusions in the environment.

There is a question that needs to be addressed before modeling a perceptual learning task: What are the input sources and the form of the information that will be used to guide the learning task? Depending on the answer to this question, supervised models can be used, which require labeled training data to obtain a desired output, or unsupervised models, which have the capability to extract some statistical information from the data, without any explicit guidance [20].

In *supervised learning* techniques, the system is trained by a teacher, who provides a category label for each training set. The input training set is typically vectors and the output may be a continuous value or it may be a class label of the input objects. Supervised learning is often referred as *learning from examples*. The important questions that need to be considered are:

"How can we be sure that a particular learning algorithm is powerful enough to learn the solution to a given problem and that it will be stable to parameter variations?"

"How can we determine if it will converge in finite time or if it will scale reasonably with the number of training patterns, the number of input features or the number of categories?"

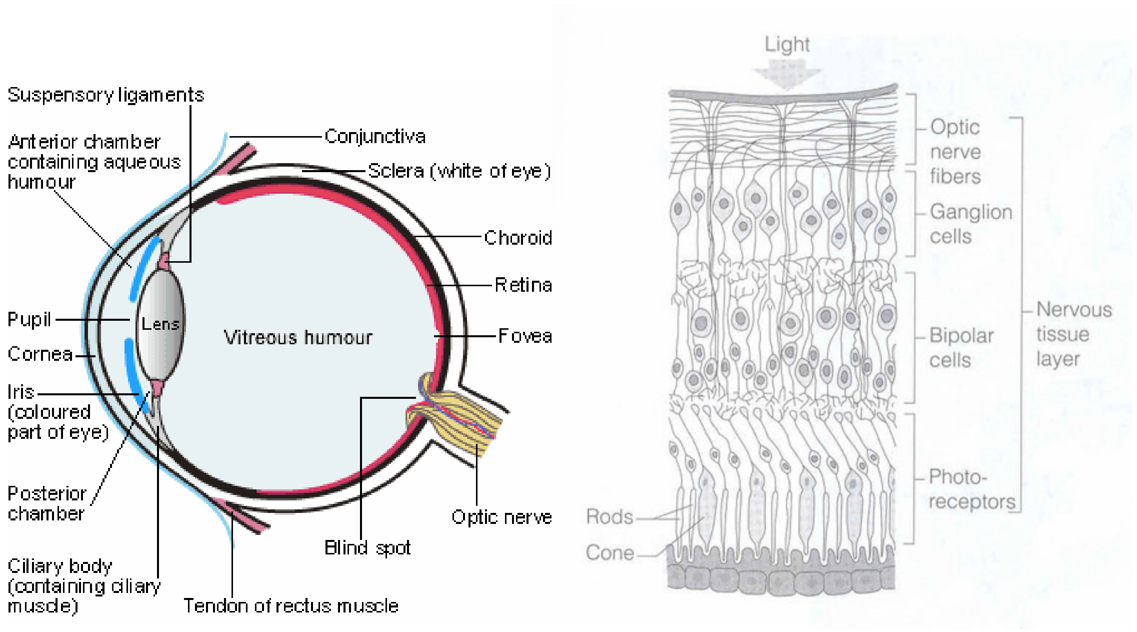
"How can we ensure that the learning algorithm appropriately favors 'simple' solutions rather than complicated ones?" [21]

In the case of *unsupervised* (also called self-supervised) *learning* there is no a priori knowledge of categories and the system groups or clusters the patterns which are expected to have some common features according to a similarity measure. The fact that the number of categories is unknown may result in unsupervised clustering algorithms producing in inappropriate representations, at least from the point of view of a human observer.

Having mentioned the challenges in perceptual learning, it is appropriate to look at the biological aspects of the human eye-brain system. From the moment an image is captured, until an object is recognized, the process performed in the human brain is highly complex.

An Overview of the Early Stages of the Human Visual System

A main sensory modality for human beings is vision. Visual perception begins in the eye, which focuses light onto the retina. The structure of the eye is given in Figure 7 (a). The purpose of the retina is to convert the light stimuli into neural impulses and forms the first stage of visual processing. The retina is composed of three layers, namely, the receptor cell layer, the bipolar cell layer and the ganglion cell layer as shown in Figure 7 (b) [22]. First, the incoming light is converted into an electrical signal within rod and cone photoreceptor cells. The rods are primarily located in the periphery of the retina and used to detect low levels of light.



(a) (b)

Figure 7 (a) The structure of the eye [60]. (b) The neural structure of retina [61].

On the other hand, the cones are concentrated in the center or fovea of the retina. The distribution of rod and cone cells is depicted in Figure 8. The cones primarily function to discriminate the color features of objects at medium to high light intensities. Depending on the wavelengths of the light that is absorbed, the cones are also divided into three types; short middle and long for blue, green and red colors, respectively.

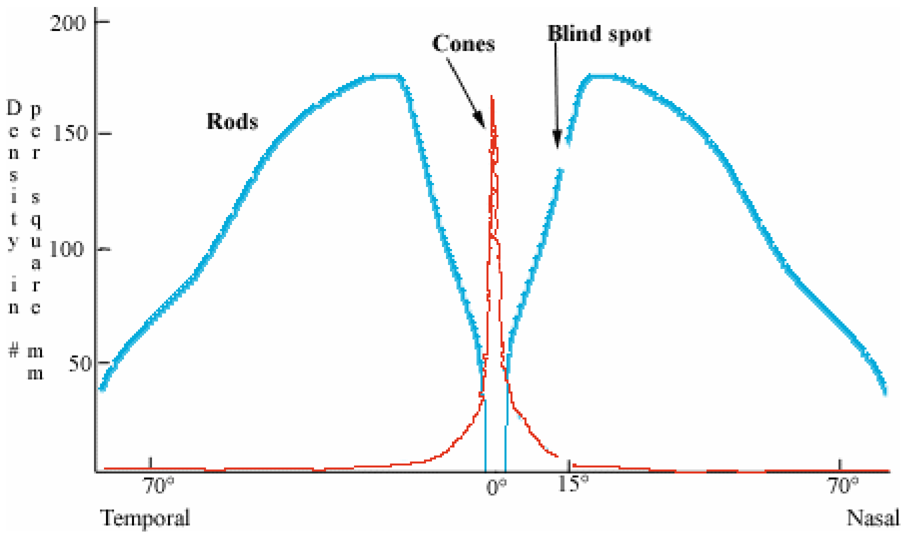


Figure 8. The distribution of the rod and cone cells around fovea. Horizontal axis is the angular separation from the fovea [62].

The neuronal impulses are then passed along to bipolar and horizontal cells, which in turn synapse onto ganglion cells. There are two types of retinal ganglion cells, G_1 and G_2 which have different responses and transmit different sequences of action potentials [23]. If the light is received in the retina, G_2 fires, G_1 turns off and vice versa. This is referred as ON and OFF responses. If the intensity of the light is increased in the center of the receptive field, the firing rate of the signals for the cells that have ON response also increases, but for the cells that have OFF response decreases. The opposite is true when the intensity of the light decreases. These responses are compared in the first cortical visual area, which helps to detect the orientation of the edges. The optic nerve, which is formed by the axons of the retinal ganglion cells, transmits the neural signals to the lateral geniculate nucleus (LGN), which establishes a pathway between the retina and the primary visual cortex Figure 9.

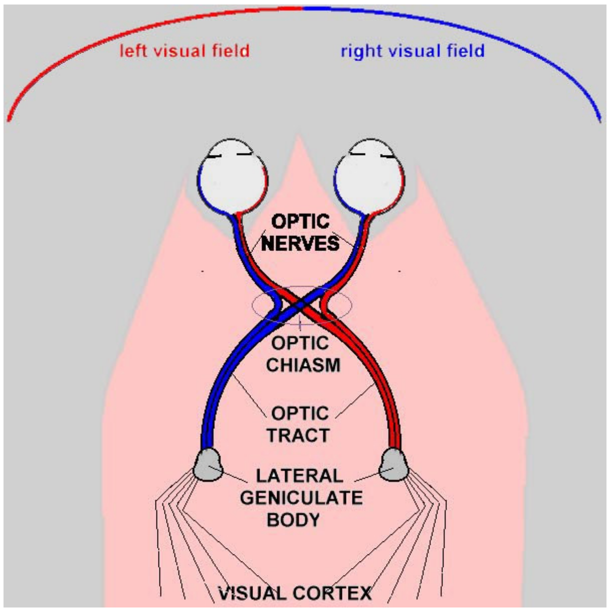


Figure 9. The pathway from retina to primary visual cortex.

The LGN neurons mainly project to the first cortical visual area, the primary visual cortex (also known as the striate cortex or V1), which lies posteriorly in the occipital lobe. This section of the brain is an important area, in that partially processed information from the retina and LGN is separated here and processed for more elaborate analysis in the specialized visual areas of the extrastriate cortex. The neurons in V1 exhibit strong responses to a small set of stimuli. Small changes in visual orientations, spatial frequencies and colors can be discriminated by these neurons. It has been shown that the optimal stimulus is a sine-wave grating which is classified by its spatial frequency [22][25]. Most neurons in the striate cortex respond best when a sine-wave grating of a particular spatial frequency is placed in the appropriate part of the visual field. For orientation-selective neurons, the grating must be aligned at the appropriate angle of orientation.

Additionally, there is a particular class of neurons in V1 which are unresponsive to single lines, bars, or edges but seem to be responsive to texture [22][24]. Responding preferentially to a sine-wave or square-wave grating of a particular spatial frequency and orientation, these cells respond to the ‘texture’ of the pattern. Natural surfaces tend to have a rough texture. Being sensitive to texture and texture orientation, these cells detect the presence of a surface and also its orientation.

It is very well known that there are two parallel sensory processing streams in the visual system, and that they are responsible for object identification (e.g., color and shape) and relative spatial object location (e.g., position and motion). The ventral and the dorsal pathways have popularly become known as the 'what' and 'where' systems [26], and project to different cortical areas, and subsequently to different areas of the prefrontal cortex.

Up to this point, the posterior or sensory portion of the brain has been discussed. Next, we move to the anterior part, which is mainly dedicated to motor operation. This portion is basically responsible for movement, such as skeletal movement of various body parts, ocular movement, expression of emotion, speech, visceral control and so on, together with considerable functional cooperation between them. The cortex of the occipital, parietal, and temporal lobes mainly supports perception and perceptual memory, while the cortex of the frontal lobe supports action and motor memory. The prefrontal cortex provides neural support to three cognitive functions that are crucial for performing temporal sequences: working memory, motor attention, and inhibitory control [27]. Working memory is memory in the active state that is used for the performance of actions toward a goal. Motor attention is the preparation for action or attention directed to prospective action. The inhibitory control of interference is integrative and protects the structure of the behavior, speech, or thought from interfering influences, external or internal, that may conflict with it and lead it astray.

The Role of Attention in Perceptual Learning

The purpose of the long term memory is to retain information that will be selected, by the attention process, for further processing in working memory. The performance of perceptual learning is strongly dependent on the attention process. Attention is a cognitive process which focuses on a particular stimulus. It is one of the most intensely studied topics within psychology and cognitive neuroscience. In spite of the fact that there are many cognitive processes associated with the human mind, attention is considered to be a hot research topic, because it is strongly tied to perception. Attentional mechanisms are needed in perceptual learning to guide further processing of information relevant to the current task, while ignoring other information present in the visual environment that is irrelevant.

The selection and determination of how many stimuli may be processed simultaneously depends on a variety of factors. We hypothesize that, at first, a goal to perform an action is known, which exerts a major influence on the type of stimulus to which the sensory system is attuned. Next, attention is focused on the stimuli needed to continue to process the current task. For example, when you would like to grasp an object that is out of your reach, you start to approach it. The position of the object guides the direction of your movement.

It is acknowledged by many theorists that attention acts as a gate to working memory. Only the information needed to be focused is stored in working memory.

Therefore, this mechanism gives us the ability to filter out irrelevant information, as well as reduces the load on cognitive processing systems.

CHAPTER IV

A WORKING MEMORY FRAMEWORK FOR ROBOTS

In cognitive science, working memory is defined as a theoretical framework which refers to a temporal type of storage that retains elements that are active and being manipulated for a short period of time. The contents of the working memory are not kept in the active state for a long time unless it is repeatedly utilized in some way. Working memory can also be thought as the distinction between short-term and long-term memory. Short-term memory refers to a part of the memory, where a small amount of information is stored for a limited amount of time and highly available to other memory sections of the brain. However, we can see that in the literature, especially in textbooks, there are often contradictions about the distinction between short-term and working memory structures [28]. On the other hand, long-term memory has a very large capacity and the information is considered to be stored indefinitely. In other words, long-term memory is capable of storing many memories throughout our lifetime.

In the literature, there are many working memory models, which have commonalities and differences [28]. However, it seems that there is a consensus in many of the models which agree on one of the most important aspects of a working memory system, which is limited capacity. Clearly, there are attentional limits on what we can process at one time. It was suggested many years ago that this capacity was about seven plus or minus two numbers in a digit-span task, which is an experiment in which one reads a list of numbers and immediately after must repeat the list [29]. However, a recent suggestion was made that the number of memory elements or so-called "chunks" that can be maintained and manipulated by working memory is approximately four [30].

In the late 1960s, many theories had been concentrated on the short-term memory (STM). For example, the Atkinson-Shiffrin model is one of the most significant [31]. According to this model, the human memory system is best represented as series of temporary sensory registers through a limited capacity short-term store (STS). The model flowchart is shown in Figure 10. The input stimuli from the environment are stored in sensory registers, such as a visual register, an auditory register and haptic register. It was also assumed that all the cognitive activities and working memory operations take place in the STM. Moreover, there is a bidirectional flow between the long-term store (LTS) and STS, suggesting that the information is retrieved into STS and can be combined and processed along with the new information that has entered into STS via the sensory registers. Hence, this model assumes that some memories might be produced by the combination of the new information with the permanent information from the LTS.

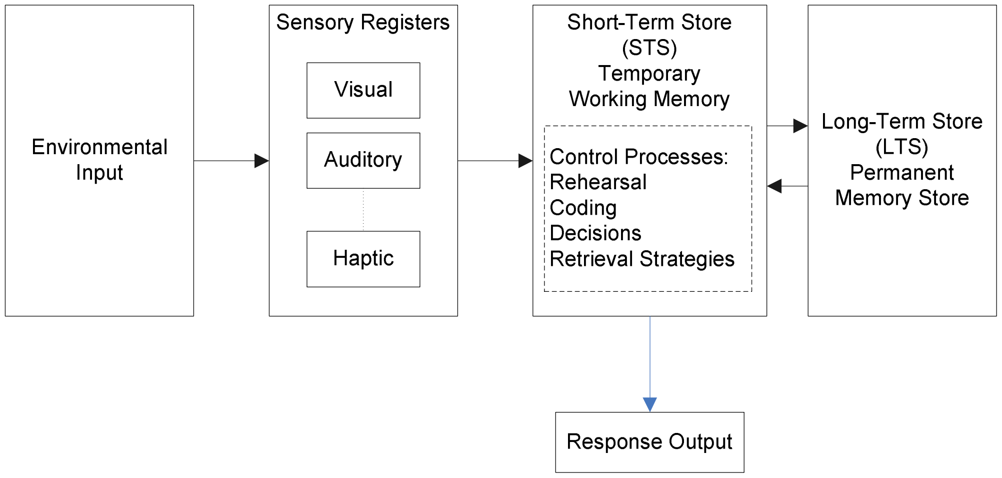

Environmental Input
Sensory Registers
Visual
Auditory
Haptic
Short-Term Store (STS) Temporary Working Memory
Control Processes: Rehearsal Coding Decisions Retrieval Strategies
Long-Term Store (LTS) Permanent Memory Store
Response Output

Figure 10. The Atkinson-Shiffrin Information Processing Model [31].

Neuropsychological studies showed that this model has problems with its assumptions concerning learning as well as the data regarding the impact of damage to STS [32]. Despite such damage, patients with STS impairment had few cognitive problems and apparently have normal LTM. Therefore, the STS by itself should not serve as a unitary working memory structure.

Baddeley and Hitch [33] proposed a three component model of working memory system, which is composed of an attentional control system, the central executive, and two storage systems, the visuospatial sketchpad and the phonological loop as shown in Figure 11.

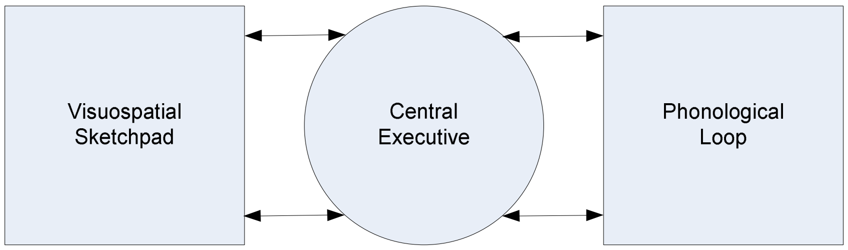


Figure 11. The three-component model of working memory [33].

The phonological loop is a short-term auditory memory subsystem in, which the information is stored in an acoustic format and if not rehearsed, the memory trace decays over time. The visuospatial sketch pad is responsible for visual information. It has a limited capacity, in which the retention of the objects is dependent on the binding together of constituent features, a process that demands attention [32]. The last component, the central executive, is the most important, however the least understood part, where the information from the other components are combined and processed.

In addition to this multi-component working memory model, Baddeley extended the model by adding a fourth component [34]. The revised model is shown in Figure 12.

Evidence from the studies with the patients who have STM deficits, suggested the need to assume there was a further back-up store, namely the *episodic buffer*. The new component, the episodic buffer, holds representations that integrate phonological, visual, and spatial information, and possibly information not covered by the subsystems.

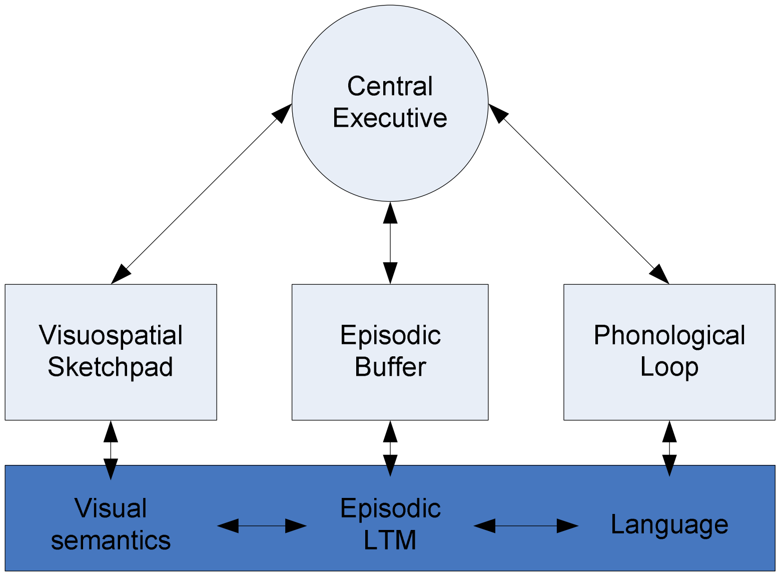


Figure 12. The extended version of the multi-model working memory [34].

The episodic buffer is assumed to have a limited capacity and to be capable of holding information in a multi-dimensional code. It provides a temporary interface between the subsystems and is controlled by the central executive. Its main functionality is to pass and retrieve information from the episodic LTM into the central executive.

A Biologically Inspired Working Memory Framework for a Robot: The Working Memory Toolkit (Wmtk)

The limited capacity property of a working memory system provides focus for the robot to search for appropriate actions in order to accomplish the given tasks. In fact, the key point faced by the working memory system that is utilized in this proposed robot architecture is the determination of which chunks[1] of information should be actively retained in the working memory, and which may be safely discarded, for the critical task success. A straightforward approach for this critical selection mechanism would be to write procedures or methods that could be used to determine which chunks are essential for a particular task. In the proposed architecture, more general and adaptive learning methods are pursued. The working memory model that is used in this architecture facilitates an adaptive mechanism that intelligently updates the contents of the memory by *learning* from *experience.*

According to previous studies, regions of the prefrontal cortex (PFC) in the human brain are involved significantly in working memory [36]. During working memory operations, dopaminergic midbrain neurons physically fire [38] and dopamine levels in the PFC distinctively rise [37].

The alteration of the dopamine levels in certain neurons in midbrain cells suggests a prediction scheme in the *expected future reward.* Learning by associating stimuli with rewards and punishments (the reinforcers) is called reinforcement learning [39]. The neuro-reinforcement learning computational models [40][41] describe the midbrain dopamine neurons in terms of a very popular machine learning algorithm, the *temporal difference (TD) learning* algorithm [42].

The main objective of the agent (i.e., robot) in this type of learning is to maximize the total amount of reward that it receives. Thus, the goal is not maximizing the instantaneous reward, but the cumulative reward in the long run [42]. If the reward is a

[1] In this context, the term "chunks" will be used to refer to the objects that are utilized by the working memory.

scalar and denoted as $r(t)$, where $r \in \Re$ for each time step *t,* then the expected future reward $R(t)$ can be formalized as a weighted summation of the rewards:

$$R(t) = \gamma^0 r(t) + \gamma^1 r(t+1) + \gamma^2 r(t+2) + \ldots = \sum_{k=1}^{N} \gamma^{k-1} r(t+k-1) \qquad (3.1)$$

where γ is a parameter, $0 \le \gamma \le 1$, called the *discount rate.* This parameter influences the present value of the future rewards. In other words, a reward received in *k* time steps in the future contributes to the sum only γ^k times of its value, if it were received in the present. As γ approaches 1, the effect of the future rewards becomes more significant. The above equation can be rearranged as

$$R(t) = \sum_{k=1}^{N} \gamma^{k-1} r(t+k-1) = r(t) + \sum_{k=1}^{N} \gamma^k r(t+k) \qquad (3.2)$$

and can be represented by its recursive form such that:

$$R(t) = r(t) + \gamma R(t+1) \qquad (3.3)$$

If we denote $\delta(t)$ being the difference between the predicted and the actual total future reward, then we obtain

$$\delta(t) = r(t) + \gamma R(t+1) - R(t) \qquad (3.4)$$

Equation 3.4 is known as the *temporal difference error*, since it is the difference between two successive estimates from the term $\gamma R(t+1) - R(t)$. The prediction error $\delta(t)$ plays an important role, in that the activity of dopaminergic neurons in the midbrain can represent this quantity.

Dr. David Noelle, and his Ph.D. student Joshua Phillips implemented an open source software library, written in ANSI C++, called the *Working Memory Toolkit*

(WMtk), which provides an abstraction layer and can be easily integrated into robotic control mechanisms [35]. WMtk essentially implements the computational model of working memory updating, which is grounded in interactions between the PFC and the dopamine system and utilizes a neural network version of the TD learning algorithm. It contains classes and methods for constructing a working memory system to select working memory contents. The toolkit is very flexible and configurable, such that it lets the designer to adjust several parameters, such as the working memory size, learning rate, exploration rate and so on. Moreover, the WMtk requires several user-defined functions in order to extend the flexibility and programmability of the toolkit.

The main object in the toolkit is the Working Memory object. The fact that the WorkingMemory object maintains untyped pointers, means that the chunk information is not restricted to a particular data type, thus there is no limitation on what kind of information may be grouped into a chunk. The candidate chunks that are provided to the system are not necessarily all stored as working memory content; however the WorkingMemory object learns which chunks to maintain or discard using the TD learning algorithm.

In order to build a robotic working memory system, initially a WorkingMemory object is created. This object requires several parameters and some user functions to configure the internal setup of the WMtk. The first parameter is the working memory capacity parameter, which is consistent with the limited capacity property of a biological working memory system. The reason that WMtk does not limit the structure of chunks is it cannot extract the meaningful features from the candidate chunks in order to evaluate the retention of a chunk. Therefore, the designer must provide a helper function, which encodes all of the relevant features within a chunk into a vector of real values. In a similar manner, another user-defined function is required that encodes relevant features of the current system state and/or the environment (e.g., robot's current sensory state) into a vector of real values. The working memory system will then assess the value of these various chunks given the current system state and/or the state of the environment. The designer must also provide a function which handles memory management for a chunk after it is no longer needed by the working memory system. In other words, this function might be used to free a dynamically created object, such as the chunks themselves or to cache extraneous chunk information if needed later. The last parameter is another user function that provides real valued instantaneous reward information. After multiple training episodes, this reward information will be used by the working memory system in order to learn when to update the contents of the working memory store.

Once all the initial setup is complete, the WorkingMemory object performs the following steps: On each time step of the task, a new list of candidate chunks is provided to the WorkingMemory object by the robot control system. Primarily, these chunks are merged with the ones that are currently held in the working memory. Now, the key objective of the WorkingMemory object is to determine which objects to retain or discard. The working memory system considers every possible subset of the total collected chunks in the memory and each subset is encoded into vectors of real valued feature sets. These features are then compared with the robot's current state to produce an input vector for the adaptive critic, which then tries to estimate the expected future reward. As a result, the combination of chunks, which provides the highest estimate of future reward in that subset, will be maintained in the working memory system. The remaining chunks are safely discarded from the system. Finally, the temporal difference error is then obtained from the reward function value of the previous episode's estimate and the current episode's estimated future reward. The adaptive critic's weights are then changed using the calculated temporal difference error [35]. A specific case which briefly demonstrates the design principles of the WMtk is shown in Figure 13. The interpretations of the chunks for our system are also depicted in Figure 13. In this case, WMtk selects chunk3 from the input chunk vector and chunk7, which is already held in the working memory. The center component represents the WMtk itself as well as the chunks that are held in the working memory store.

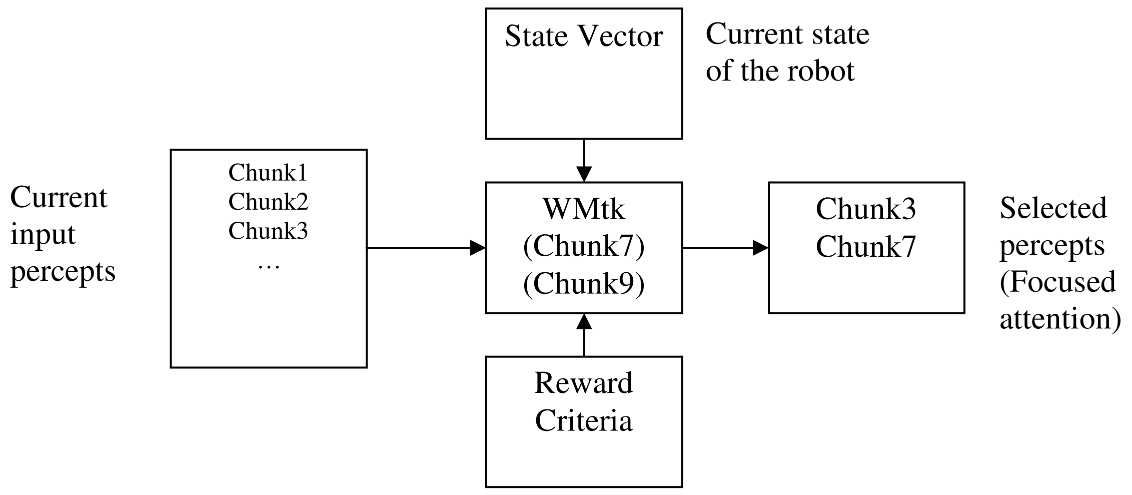


Figure 13. An example which demonstrates the WMtk's design principles. Among the input chunk vector, WMtk decides to choose chunk3 and chunk7 in this current state.

Even though the basis of the WMtk relies on established reinforcement learning techniques, the most distinctive feature of the toolkit is that it is not limited to selecting one action out of a small fixed discrete set of actions, as most of the traditional reinforcement learning systems select. The toolkit explores every possible combination of collection of chunks, which can be fit within the limited capacity of the working memory. The combination of chunks which provides the highest estimate value of the future reward is selected and the chunk in this combination is retained in the working memory. This feature of the WMtk can be very useful in robotic applications. The robot is not limited to only one single item to consider. For instance, in many applications a landmark location is represented by only one percept. However, a landmark location may be represented by a combination of percepts, which may help the efficiency of the robot's localization process. In addition, this would also help to increase the number of unique landmark locations in the navigation path of the robot.

CHAPTER V

THE PROPOSED SYSTEM STRUCTURE

Overview:

A focus of this work is to establish a biologically inspired system that employs perceptual knowledge along with the working memory system in order to provide broad capabilities for visual learning tasks. Our main overall objective is to develop a system, which supports research into developmental robotics and uses a powerful perceptual system for this purpose. Such a requirement necessitates a large set of goals that the system needs to satisfy. In general, robot vision systems tend to be designed for specific types of problems and circumstances. The resulting systems typically operate in restricted environments. Our desire in this work is to have the robot operate in real world environments that are not modified by using artificial landmarks. The perceptual system facilitates a supervised learning method where the robot acquires the information from the user.

First of all, in order to perceive the information from the environment, the robot needs to learn how to perform segmentation from examples and continually refine these capabilities as new data is acquired. For example, the robot could have been trained in an environment where the lighting condition is different than in an environment where it currently needs to perform segmentation. In such a situation, we would like the robot to adapt to the new environment on the fly and refine its previous perceptual knowledge. In many cases, the standard method for modeling involves generating a Gaussian or a Gaussian mixture model for each class or percept in order to analyze future samples.

Such a model is usually constructed by utilizing a relatively few samples, which then can be used for identifying other samples in various proximities of the original samples. One argument in favor for this type of model is that the calculations are computationally efficient, after the model is created. Nevertheless another argument, one that is not in its favor, is that the estimation of the most accurate models, such as Gaussian mixture models, are computationally expensive as the number of samples or the dimension of the samples increases. In this architecture, 10001 different elements are used to construct a very high dimensional feature vector from a small region in the image. Hence, the dimensionality is high, and for the model to be accurate there must be an adequate number of samples to obtain one or more clusters in the feature space. In order to analyze the unknown samples, the model estimation approach in this system is a straightforward non-parametric model that allows simple updates to the model without the need for recalculation. Such flexibility allows the robot to make simple updates on its perceptual knowledge and the model can be quickly updated by adding a few samples on the fly. As the number of samples increases, the computational cost also increases. Therefore, this leads to a need for memory consolidation in order to reduce the computation cost. The details for estimating the model and the memory consolidation process will be explained in this chapter.

Once the model is constructed and applied to the image, the resulting segmented image is presented to the user for evaluation and correction. This is very similar to a relationship between a teacher and student, where a teacher might grade and correct the student's work. If a correction is necessary, the user selects the segmented percepts that are misclassified and the system generates new data points for each of these percepts. At this point, the perceptual model can be either recalculated or can be updated by adding new data samples to the system without recalculating the entire model.

It is important to maintain a strong connection to the features of the percepts in the environment and to their semantic meanings. In other words, the robot should learn what the percepts mean for itself. For example, a percept could mean a target object or might be perceived as an obstacle. In this context, understanding the meanings of the percepts is referred as *perceptual grounding*.

Although it is not within the scope if this dissertation the robot should also be able to detect novelty, such as new objects or events that it has never been exposed to before, and use this to trigger learning. In such cases, the robot should ask the user what the new object is and learn what it means with regard to itself and to its current task. This idea could also be extended detecting a *context* change in the environment. The robot should have the capability to generate a measurement of how much novelty it sees in the environment. For instance, when you see a lot of new objects, you may realize that you are located in a different context or location. Consequently, the robot can therefore infer about the new context and interact with the user for confirmation or additional information. The perceptual part of the vision-based autonomous mobile robot system is shown in Figure 14.

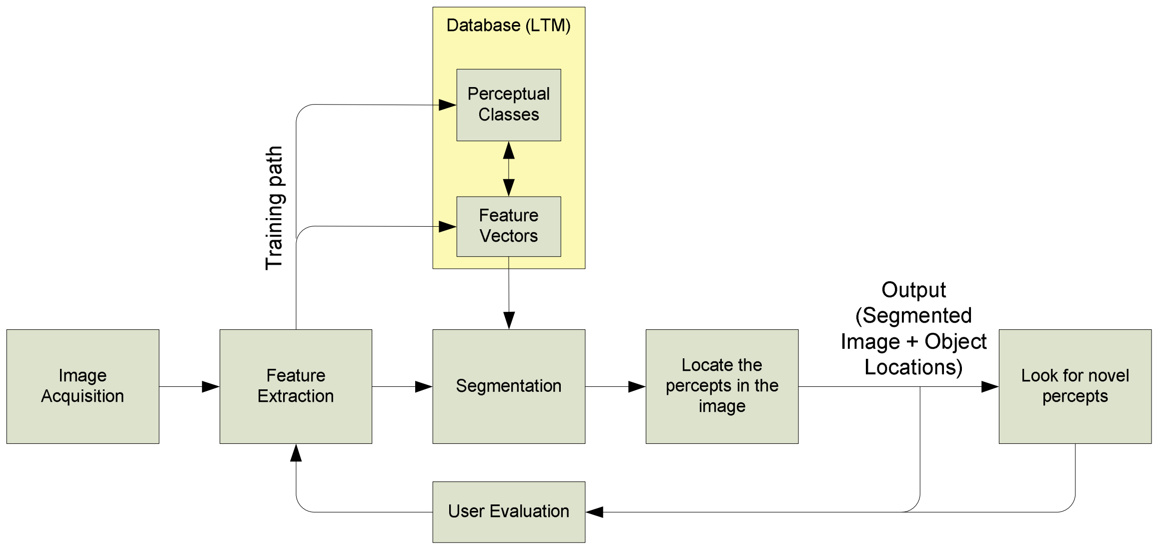


Figure 14. Perceptual part of the vision-based autonomous mobile robot system.

A flowchart that characterizes the perceptual system and the desired human-robot interaction concept of the perception system is shown in Figure 15. Initially, the user teaches percepts by identifying and labeling the percepts. The system extracts the necessary information from the identified percepts in order to form its perceptual knowledge. This process is continued until a sufficient number of samples to represent the percepts is acquired. Next, the system forms its generative and discriminative models. By means of the models generated, the system segments the percepts in the environment and then presents an image with the segmented percepts. The user evaluates the segmentation image and identifies the percepts that are not segmented or not segmented well enough. It is important to note that the system can be trained with any new perceptual data any time, even with the percepts that the system has never been trained before. Once the corrections are made to the system by the user evaluation, the system stores new samples in its memory and updates its perceptual model. If required, a memory consolidation process can be performed at this point.

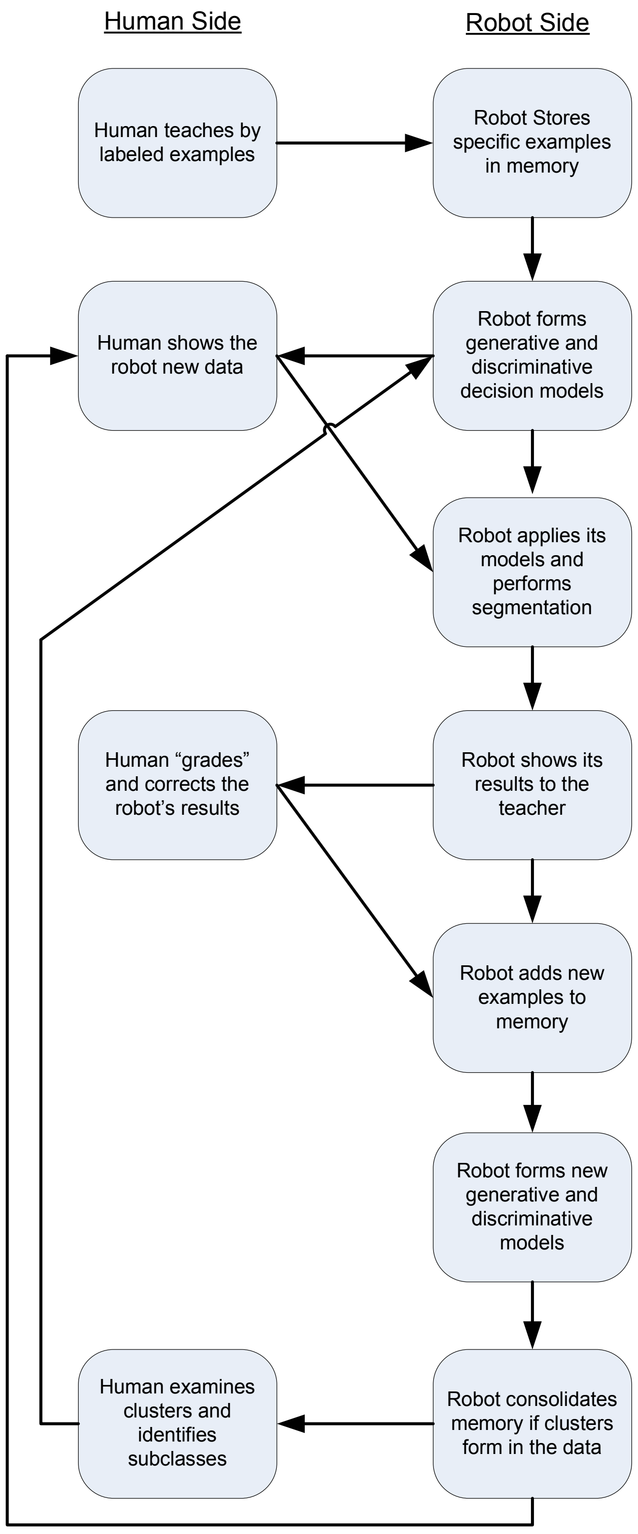

Human Side
Robot Side
Human teaches by labeled examples
Robot Stores specific examples in memory
Human shows the robot new data
Robot forms generative and discriminative decision models
Robot applies its models and performs segmentation
Human "grades" and corrects the robot's results
Robot shows its results to the teacher
Robot adds new examples to memory
Robot forms new generative and discriminative models
Human examines clusters and identifies subclasses
Robot consolidates memory if clusters form in the data

Figure 15. The flowchart of the desired human-robot interaction

Justification of using extremely high dimensional feature space

In a high-dimensional space, often, the input vector for the system cannot exhibit all the features. Thus, the vectors results in a highly sparse format, especially in our vision system. Encoding the input vectors in this fashion provides a significant performance gain, especially for non-parametric techniques such as an exhaustive nearest neighbor search between the vectors in the pattern space.

In the beginning of our work, we wanted to explore a feature space having a very high number of dimensions, and the reason for this is that we believed that such a feature space would have a very large capacity to learn. Therefore, we suggested that one way to obtain a robust performance in complex environments is to use a very high dimensional feature space. However, we are not implying this is the only way to achieve robustness and there may be other approaches, but to our knowledge, at least in higher level biological systems, the eye-brain system exploits a high number of features and not all the features are necessarily present at a single time in the brain.

One of the consequences of using a high dimensional space is that typically, the parametric classifiers become impractical as the number of dimensions increase. These are any of the parametric techniques which involve computing eigen-values and eigen-vectors, covariance matrices, or acquiring covariance matrices in the first place or forming enormous data matrices and computing singular value decompositions, and so on. As a consequence of this we needed a simple classifier, and the fact that we wanted to use a very high dimensional space led us to select the nearest neighbor (NN) method as our classifier. The NN method does not need any kind of 2nd order covariance information. The NN method only needs to compute distances. Furthermore, there are no assumptions about the distributions and it is not restricted in the choice of distance metric.

Another consequence of using a high dimensional space is the need for a large database. A practical rule of thumb is that the amount of data required is approximately five times the dimension of the feature vectors. However, in a visual application, it is not difficult to obtain large amounts of training data.

These requirements, that the system must to operate on such a large database and use NN as a classifier, mean that pure, exact NN would be too slow and computationally very expensive. As a result, an approximate NN implemented by a search tree would be much faster, and since the database is large we expected it to yield acceptable performance. Another aspect of this design is that the non-parametric search tree also allowed us to continually train the system during use, as necessary.

It may seem that in a high dimensional space the distance calculations are expensive. On the contrary, it is not expected that all the features are present at all times, especially in the case of features we have chosen. Therefore, the feature vectors are very likely to be sparse. Hence, if the vectors are encoded in a sparse vector representation in the system, the calculations can be dramatically simplified, and the memory storage significantly reduced. Local high dimensional color histograms of small regions in an image exhibit this property. We note that in our system, the computational cost of distance calculation is virtually unrelated to the dimension of the vectors. This particularly addresses, to some extent, the "curse of dimensionality" problem, at least in terms of computational cost. Consequently, a high dimensional space is not a problem for our system. Furthermore, due to the design of our system, the features can be computed in a highly parallel fashion. Parallel implementation will be addressed as a future work in the conclusions.

Visual Features

At first, an image is captured from a digital video camera and corresponding red, green and blue (RGB) values for each pixel are obtained. Next, the RGB image is switched to the HSV color space. The HSV color space is based on intuitive color characteristics such as hue, purity and the intensity of a color. As shown in Figure 16, the coordinate system is cylindrical and the colors are defined in a hexcone. The hue value, which defines the color family, ranges from 0° to 360°. The saturation parameter S is the degree of purity from 0 to 1. The value parameter V defines the brightness of a color and it is also from 0 to 1. The saturation S and value V can be also viewed as adding or subtracting white and black to a color.

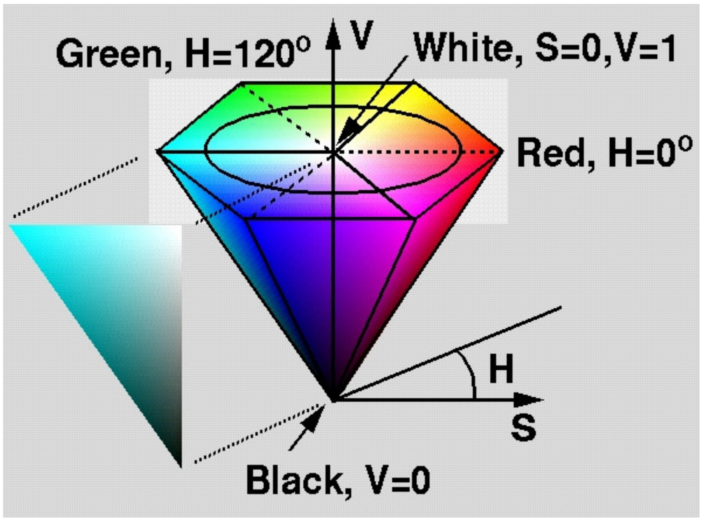


Figure 16. The HSV color domain representation.

The color space conversion from RGB to HSV is computed as follows:

$$
\begin{aligned}
&MAX = \max(R,G,B) \\
&MIN = \min(R,G,B) \\
&V = MAX
\end{aligned}
$$

$$
\begin{aligned}
&if\ (MAX = 0),\ \ then\ \ V = S = 0\ \ and\ \ H\ is\ undefined \\
&S = \frac{MAX - MIN}{MAX} \\
&if\ (R = MAX),\ then\ \ H = \left(0 + \frac{G - B}{MAX - MIN}\right) \times 60 \\
&if\ (G = MAX),\ then\ \ H = \left(2 + \frac{B - R}{MAX - MIN}\right) \times 60 \\
&if\ (B = MAX),\ then\ \ H = \left(4 + \frac{R - G}{MAX - MIN}\right) \times 60 \\
&if\ (H < 0),\qquad then\ \ H = H + 360
\end{aligned}
$$

Once the HSV values are computed, the next step is to construct a probability density function (pdf) of the HSV distribution of colors. The pdf is essentially a histogram of HSV colors and computed using a color quantization method as follows: The hue space is evenly distributed into 100 bins, ranging from 0 to 1. The saturations and values are each evenly distributed into 10 bins that are also ranging from 0 to 1. Therefore, the individual color *features* can be obtained by combining these, resulting in 10,000 different color features. For a small region in the image, an HSV color histogram is obtained by accumulating the HSV values of each pixel in the region. The region referred to here is a moving window of a particular size (currently 15x15 pixels). The center of the window is then moved throughout the entire image and a feature vector (color histogram) is obtained for each displacement. Currently, the displacement is 10 pixels in the horizontal and vertical directions. Since there is only a relatively small number of pixels in each region (225 for 15x15 regions) and the total number of colors that can be represented is 10,000, the feature vector has a highly sparse format. In other words, there are zeros in most of the bins in the histogram. Therefore, each feature vector is encoded using a sparse vector representation. A sparse vector is composed of two vectors, a value vector that represents the non-zero values and an index vector that represents the indices of these non-zero values in the original vector. This representation is useful in several ways in our system. First and foremost, increasing the dimensionality of the feature vectors does not affect the computational cost for computing the distance between two feature vectors. For instance, in our system, the distance between two feature vectors are calculated by using the Euclidean distance measure. The Euclidean distance between vectors $\mathbf{X}$ and $\mathbf{Y}$ is given by:

$$D = \sqrt{(\mathbf{X} - \mathbf{Y})} = \left(\|\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 - 2\,\mathbf{X}\cdot\mathbf{Y}\right)^{1/2}$$

In order to compute the norm of a vector, only the non-zero elements of the vector are sufficient. Furthermore, only the non-zero elements that are in the same dimension are necessary to compute the inner product of two vectors. This can be easily checked by looking to the index vectors of two sparse vectors that are pointing the same non-zero element positions. Thus, as long as the number of non-zero elements is constant, increasing the number of dimensions does not affect the distance calculations. If the feature vectors are constructed from an N X N region, the maximum number of unique element indices to be accessed for computing a distance is $2N^2+1$, where the additional one is due to the texture measure in both vectors. Secondly, the memory is significantly preserved by using sparse vector representation.

One of the reasons that we use such a high dimensional space is to differentiate colors as much as possible. Another secondary reason is the resemblance to the sparse encoding employed in the brain. Theoretical and experimental studies suggest that the primary visual cortex (V1) uses a sparse code to efficiently represent natural scenes [46].

In order to understand the "roughness" intensity of a region, a spatial filtering technique, based on the Laplacian operator, is applied to the image and added as the last element in the feature vector. The Laplacian operator is a 2D isotropic measure of the 2nd spatial derivative of an image. The Laplacian operator is often used for edge detection, where the regions that have rapid intensity changes are highlighted in the image. The Laplacian $L(x, y)$ of an image having pixel intensity values $I(x, y)$, is defined as:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Since an image is composed of a set of discrete pixels, the following kernel, which approximates the 2nd derivatives in the definition of the Laplacian equation above, is applied.

$$K_{LAP}(x, y) = \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & +8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} / 8$$

If the image has *M* rows and *N* columns, and the kernel has *m* rows and *n* columns, then the size of the output image will be *M*-*m*+1 rows, and *N*-*n*+1 columns, after the convolution. The convolution is given by:

$$O(i, j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i + k - 1, j + l - 1) K(k, l)$$

where *O* is the output image and *i* runs from 1 to to *M*-*m*+1and *j* runs from 1 to *N*-*n*+1.

Adopting a Matlab-like notation, the texture feature is computed as follows:

$$Texture\,Feature = Conv(K_{AVE}, Abs(Conv(K_{LAP}, I)/256))$$

where *Conv* and *Abs* are convolution and absolute value operators respectively, and K_{AVE} is a normalized averaging filter which has the size of the moving window (i.e. if the window has 15x15 block size than the filter is normalized by 225) and ones in all of its rows and columns.

Proximity Measure

It is expected that the patterns that are members of a specific perceptual cluster should be closely positioned in the pattern space, while those from different percepts should be positioned further apart from one another. Once the set of visual features are computed, the next step is to define a similarity (or dissimilarity) measure between two feature vectors, which will aid in to revealing the organization of the patterns in the space. Based on this similarity measure, we then design an algorithm which efficiently searches for similarities and dissimilarities among the input patterns and clusters them in the feature space. At this point, we need to define a proximity measure, which will aid the robot to find the natural groupings of data in its perceptual space. There are many proximity measurements in the literature, such as Euclidean, Mahalanobis, Tanimoto distances and their variations are some of the examples.

As pointed out by [47], the best approximation to an invariant relation between data and distances for a unitary or holistic stimuli, such as hue, saturation or brightness of a color, is the L_2 norm, commonly known as the Euclidean distance. In a K-dimensional space, the Euclidean metric distance is given by:

$$d_{ij} = \left(\sum_{k=1}^{K} \left| x_{ik} - x_{jk} \right|^2 \right)^{1/2}$$

where K= 10001 in this system.

The search tree and the Clustering Algorithm

As mentioned earlier, as the dimension of the vectors increases, the number of training samples should also be increased considerably as much as possible in order to obtain meaningful percepts. However, due to the fact that the system operates in such a high dimensional feature space, an efficient, robust and yet an accurate search algorithm is desired. For instance, if a pure nearest neighborhood were performed over such a large data set, it would be very time consuming and computationally expensive. Hence, the data stored in the memory should be consolidated and organized as much as possible.

The training data and the tree structure construct the LTM or the database of the system. In order to obtain such an efficient and yet reliable search algorithm, a tree-structured vector quantization method is applied to generate an approximate 3-way nearest neighbor searching tree. The tree structure is formed as follows: Initially, at the root or first level of the tree, three points, which represent the cluster centroids, are selected randomly and then the whole data set is clustered into three subsets by assigning each feature vector to its closest representative cluster center according to the proximity distance measure. At the second level, three subsets are obtained and the same procedure is applied, which in turn results in 9 subsets or in other words nodes for the tree. This procedure continues until either all the leaf nodes belong to the same object class (a pure node) or the number of leaf nodes is below some limit, e.g., a hundred. Every feature vector in the leaf nodes has a landmark associated with it. This completes the construction of the tree.

Since all the centroids for each node in the tree structure are known, searching the tree is straightforward. Given a new feature vector, the three similarity measures, which are between the new vector and the centroids of the three sub-nodes at the second level of the tree that belong to the root node, are computed. The winning sub-node is the one that is closest to the given feature vector. At the third level of the tree the same procedure is applied and a winner sub-node is selected and the fourth level of the tree has been reached. This procedure is terminated when the search has descended to a leaf node. If the leaf node is pure, that is all the feature sets belong to the same class, then the vector is labeled as the leaf node's class label. If not, that is the data set in the leaf node is mixed and below some threshold limit, then a nearest neighbor search is applied using the vectors of the leaf node, and the vector is labeled with the training vector's label that is closest. This completes the search algorithm.

Immediate simple updates to the search tree

Now assume that the tree structure has a relatively high number of nodes. It would be cumbersome and computationally expensive to re-compute the tree structure, when additional training is required. Earlier, it was mentioned that the non-parametric structure of the method enables simple updates to be performed on the fly. A very straightforward approach may be taken when a new training vector arrives. It is possible to treat the new training data as testing data and perform a search on the tree. Once the search is complete and reaches a leaf node, the training data is assigned to that leaf node, so that the population of that leaf node is incremented by one. This allows a very fast and computationally efficient algorithm to update the perceptual database.

As the number of training vectors increases in time, the reliability of the perceptual system is also expected to be increased as well. It is desired to have a large number of training samples in order to obtain meaningful clusters in a high dimensional feature space. However, this does not necessarily mean that all the obtained training samples will equally contribute to the efficiency of the search tree. For example, suppose that there are ten thousand training points in a cluster that would represent some pattern in the environment. It is possible to represent the same pattern using only a few hundred or fewer data points in the search tree, which would dramatically increase the performance of the system by decreasing search time. In other words, only the points which provide a significant contribution to representing clusters in the pattern space should be maintained. This can be also related to the *forgetting* behavior of human beings. For example, in time, we may forget many of the events that we have experienced before and we may only remember the ones that are most important to us. Explanation of this is left to future work.

Novelty Detection

Furthermore, the structure of the tree also aids the robot in detecting objects that the robot has not seen before, or if the environment itself has changed significantly. In the pattern space, the more the features of objects are further apart from the features which the robot has been trained before, the more novelty exists in the perceptual space. Consequently, substantial change in the novelty of many objects may lead the robot to reason that a context change in its environment has occurred. One approach would be to obtain similarity measurements from the nodes of the search tree and construct a novelty image where novelty is estimated by the distance to the approximately nearest training vector. The distances then, can be mapped into a grayscale image that would indicate novelty.

The Cognitive Structure: The Perceptual System and the Working Memory Structure Combined

After the segmented image has been obtained, the final stage of the perceptual system is to produce inputs for the working memory system, referred to as "chunks", in order for the robot to acquire sensory-motor associations to guide its actions. The input chunks for the working memory system are obtained by applying a connected-component labeling algorithm to find the locations and some other properties that describe regions, called "blobs", in the segmented image. Each blob represents a specific cluster in the feature space, but it also represents a clustering of pixels in the image space. This simultaneous clustering in both the feature space and the image space is usually indicative of a meaningful percept.

Since this architecture is applied to a mobile robot platform, the connected-component labeling algorithm is only applied to the lower half of the entire image. It is assumed that the objects that appear in the upper half contain many distracting percepts from distant objects. Also, distant objects experience greater variability in lightning conditions. This also helps to reduce the number of inputs for the working memory system to consider, as well as to improve the performance of the perceptual system.

The final flowchart of the system is presented in Figure 17. The top half of the diagram represents the perceptual system as shown in Figure 14, previously. The lower portion of Figure 17 represents the working memory system, which is combined with the robot's sensory-motor scheme (i.e., actuators and sensors such as sonars to avoid obstacles). The combination of both the perceptual system and the working memory system yields the cognitive structure of the robot.

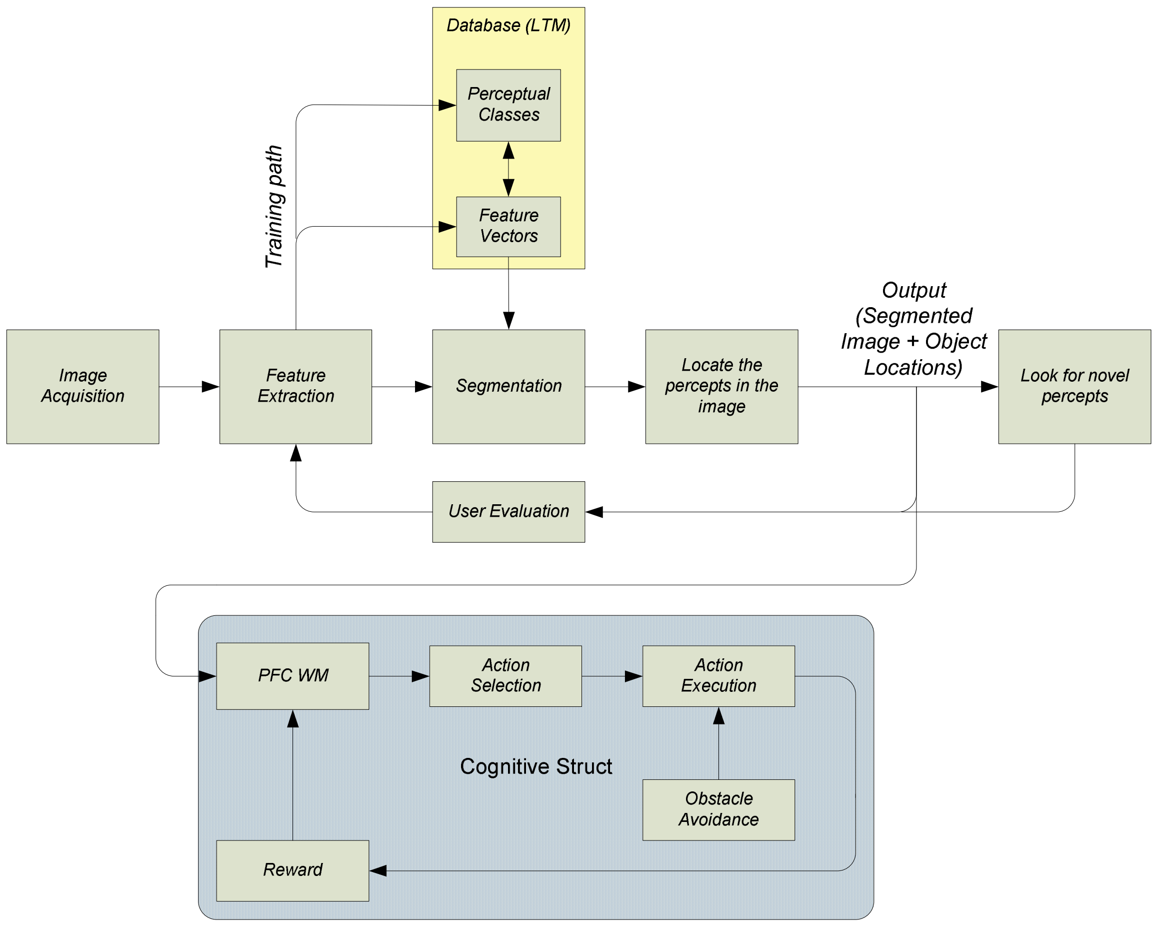


Figure 17. The proposed system architecture, perceptual system and working memory combined.

The WMtk, intelligently decides which input chunks to maintain or discard in the working memory depending on the reward criterion. This criterion is dependent on the current task of the robot and is specified by the user as a requirement for the WMtk. The reward is generated in a discrete-time manner (namely "episodes") and is a real number. Depending on the selected action in the previous episode, as well as the reward signal generated for that particular action, the WMtk updates its internal parameters and outputs the working memory contents. The robot then selects and executes an appropriate action based on the contents of the working memory.

CHAPTER VI

THE PROPOSED METHODOLOGY AND EXPERIMENTS

As mentioned previously, a main objective of this research is to integrate a perceptual system into a biologically inspired working memory system in such a way that would yield a basic framework for visual learning tasks. Our approach in this work is not intended to focus on a specific task, such as navigation, object recognition, etc. In contrast, the aim is to provide a general framework for developmental learning, in order to investigate how well a neuro-computational PFC working memory model performs on a robotic platform in a real world environment with complex tasks. Additionally, we seek to explore the use of PFC working memory in robotic applications. Furthermore, we propose that this general framework may be integrated into specific robotic applications which involve learning.

We propose to conduct several experiments that would exploit the usefulness and capabilities of this framework in a mobile robotic platform. A fundamental property of a mobile robot is the ability to navigate safely in an uncertain environment. In order to achieve this task, the system needs to acquire sensory-motor associations to guide the behavior of the robot. Each experiment is related and builds on each other in order for the robot to navigate successfully.

The experiments will take place in the hall outside of the Intelligent Robotics Laboratory in the third floor of the Featheringill Hall. This hall is composed of several objects and is shown in Figure 18. The floor consists of three tile structures: The yellow floor tiles in the center with white floor tiles, along the edges near the wall, and black stripes of tiles are located periodically across the center of the hall. The walls consist of wood panel sections, light blue painted wall sections and a light blue railing. Additionally, there are also brick columns and blue floor tiles at the ends of the hallway.

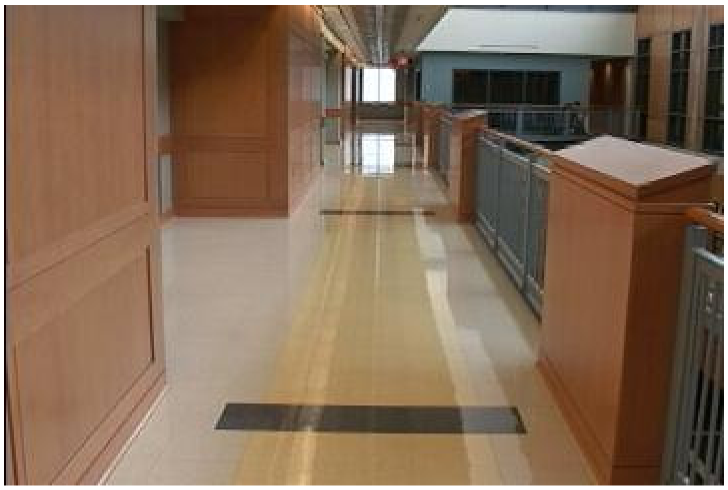


Figure 18. The hallway outside of the Intelligent Robotics Laboratory.

Initially, the robot is trained by capturing an image from the input video stream. The user selects a region in the image and labels it by assigning a category. The features are extracted from the selected region and stored in the memory. Next the search tree structure is computed from the training features. In order to account the different lighting conditions, the features are collected over several days at different times. The feature vectors, the search tree structure and the labels of the feature vectors constitute the training database of the system.

Experiment 1: Learning the Open Space

In order for a robot to navigate safely, it has to move in open space. Therefore, the first step is to learn the percepts that will yield an open path for navigation through the robot's own interaction with the world.

Once meaningful blobs or percepts are obtained from the images acquired from the camera, the robot needs to learn the association of the percepts with the task of motion. In other words, the robot needs to determine whether the percepts represent an obstacle or an open space, since the robot does not have any prior information about these percepts. In order to have such association in this task, the reward criterion for the WM system is selected to be the distance taken by the robot, since percepts that are in the open space allow the robot to move further. The procedure for the experiment 1 as follows:

- The WM size parameter is set to 1 in order to asses the percepts individually. Therefore, the WM can only select one of the available input chunks per episode.
- The robot captures an image from the video stream and segments it into blobs. In order to reduce the number of computations, only the blobs that are located at the bottom half of the image are presented as an input to WM, since the objects that are in front of the robot are required. Initially, since there is no reward information (i.e. no displacement is taken by the robot), the WM randomly chooses one of the available chunks.
- Next, the robot directs itself toward the selected chunk and proceeds forward for five meters or until sonar sensor indicates an obstacle, which forces the robot to stop.
- The distance taken by the robot is returned as a reward for the WM system and associated with the selected chunk previously. If the robot moved for five meters than it will receive the maximum amount of reward, if not, it will receive a partial reward which is the distance it traveled.

- The same procedure is taken, starting from the first step, multiple times, which allows the WM system to intelligently update its internal parameters associated with each available chunk in the environment, depending on the reward received at each episode.

Experiment 2: Learning Landmarks

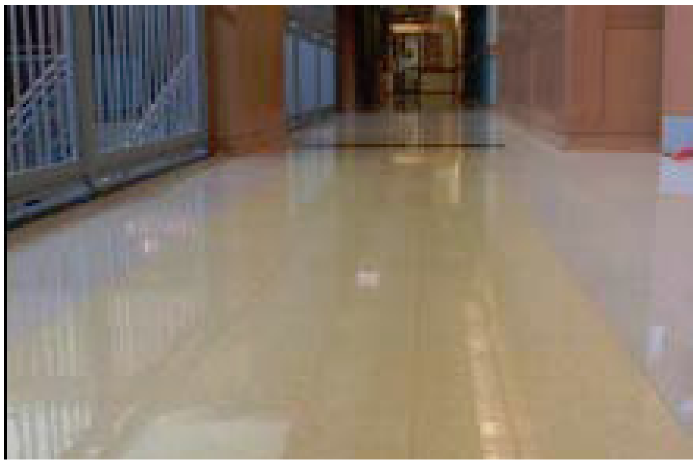
In a navigation task, landmarks are essential, in the sense that the robot might take an action on a specific landmark. In the first experiment, the robot learned which percept will provide the open space to move safely. Now, it must learn about landmark locations in order to localize itself in the environment. In this experiment, the black floor tiles are found to be convenient as landmark locations. For instance, the robot might turn 90° right or left at the 3rd landmark location. The following steps are taken for the experiment 2:

- The WM size parameter is set to values of 1, 2, and 3 to determine which percept or combinations of percepts should be selected for recognizing a landmark location.
- A set of training images are acquired from the environment to represent landmark and non-landmark locations as seen in Figure 19. As mentioned, the presence of a black floor tile in the foreground is an indication of a landmark. The number of landmark and non-landmark images is equal in the collected set of images.
- The training images from the set are selected randomly and processed in order to obtain the chunks for the WM system, using the search tree.
- The percepts that are available are duplicated and presented to the WM system. In other words, the input chunks for the WM system are doubled. One chunk is

considered as a landmark and the other in the same duplicate considered as non-landmark. Once the WM system has selected a set of chunks, a vote is taken to determine whether the majority is a landmark or non-landmark. If there is a tie in the vote, then it is interpreted as a non-landmark by default. Since the training images are labeled, the WM system receives a reward of one if the selected chunk is a landmark, else it receives a zero.

- Each of the WM sizes is trained for both 1000 and 5000 episodes.

(a)



(b)

Figure 19. (a) A landmark image. (b) A non-landmark image.

<u>Experiment 3: Learning the Navigation Task using the learned Landmarks</u>

At this point, the robot knows which percepts yield open space, as well as which percepts are considered to be landmark points. Now the question becomes how can the robot learn to associate a sequence of percepts for open space along with the percepts for landmark points in order to reach a goal position. Once, this is achieved, then the robot can learn a navigation task and accomplish the task successfully with the aid of its knowledge.

In the first experiment, finding the open space to navigate safely could be thought of an action behavior for the robot. However, in the second experiment, finding the landmark percepts could be considered as a perceptual behavior. In the first one, the reward is based on the distance that the robot travels (a continuous value), which is the attentional constraint in order to find the open space. In the second experiment, the reward becomes a boolean type of variable such as true or false. The 2nd experiment is supervised in the sense that the landmark images are labeled, however there is some freedom in the sense that the landmark percepts are not explicitly told to the robot in the images. The robot must discover them on its own.

In this experiment, the robot needs to learn a navigation task by utilizing the previously learned behavior for finding the open space, as well as the landmark percept it has learned. It has to associate actions to the sequence of landmarks seen along the navigation. Additionally, it should also remember (i.e., exhibit a delayed-response) a contingent situation that will change the task completely into a new one. For instance, if it sees a contingent percept (i.e., a green ball) along the navigation path, it has to hold on to it until it reaches the last landmark and then come back to its initial position.

Specifically, the robot will be positioned in front of the Intelligent Robotics Laboratory, looking towards the west side of the hallway. The black stripes along the hallway indicate the presence of a landmark. The objective of the robot is to navigate along the hallway until it reaches the third landmark. However, if the robot recognizes a green ball, as an unexpected, contingent percept along the way, it should remember the green ball until it reaches the last landmark and come back to the starting position. If no green ball is detected along the navigation path, the robot is required to learn to stop at the third landmark. The robot makes a decision, moves with a small distance increment and stops. The time required for this cycle is referred to as one time step. This procedure continues until the navigation is complete. The procedure for the experiment 3 is as follows:

Initially, the robot captures an image and segments the percepts in the environment. A list of possible actions, such as "move", "stop", and "turn around" will be presented as inputs to the working memory. The working memory system will be rewarded or punished by the possible actions chosen by the robot along the navigation path. These three action chunks are always provided to the working memory in the input chunk vector at each time step.

In addition to the action chunks, the working memory is presented with the percepts detected in the environment. However, only “YellowFloor”, “BlackFloor”, and “GreenBall” percepts are presented, if recognized by the perceptual system. Since the task requires only remembering the “GreenBall” chunk, the other percepts are treated as distractors.

Finally, the working memory is also presented with a “Nothing” chunk, meaning that the system does not make any selection. Therefore, the maximum number of chunks that can be presented to the working memory is seven.

The working memory size is set to two. One slot is for an action chunk and the other one is for a percept chunk. Thus, the working memory is required to learn to select an action chunk as well as a percept chunk.

Once the candidate input chunk vector is presented to working memory, a decision is made. The robot either moves, stops or turns around depending on the action selection made by the working memory at each time step. Furthermore, if a “GreenBall” percept is detected, the working memory is expected to remember the “GreenBall” chunk in its second output slot.

The robot learns to move towards the open space for one meter and the cycle is repeated until the third landmark is reached. However, if a green ball is detected, it needs to remember that it has detected an indication for a contingent situation. Therefore, as the task requires, it needs to come back to its initial position by passing again at each landmark, in order to demonstrate the bootstrapping effect.

The objective of this experiment is to have the robot utilize its previously learned knowledge and apply them to learn a complex task. Furthermore, it needs to demonstrate the delayed response effect by holding on to a percept for a period of time. The delayed response task is widely used in cognitive neuroscience in order to evaluate the properties of working memory [4], [35].

CHAPTER VII

RESULTS

Overview

One of the goals that we wanted to achieve while designing the system is to utilize solely visual information. The testbed robot Skeeter, which is explained in detail in Appendix A, is equipped with a single camera to perceive the environment. The front sonar array is only used for obstacle avoidance purposes. In this chapter, the results of the experiments, which are discussed in the previous chapter, are demonstrated. The objective of the experiments is to reveal the capabilities of the system in uncontrolled environments. In the experiments, the system learns new percepts, as well as new skills to learn to ultimately perform a qualitative vision-guided navigation task. Since the task is qualitative navigation, no metric information is involved in the computations. Only the angular positions of the percepts with respect to the robot frame are computed. Before showing the results of the experiments, the features of the perceptual system will be demonstrated. Next, the results of the experiments will be introduced. Finally, a discussion section will conclude the chapter.

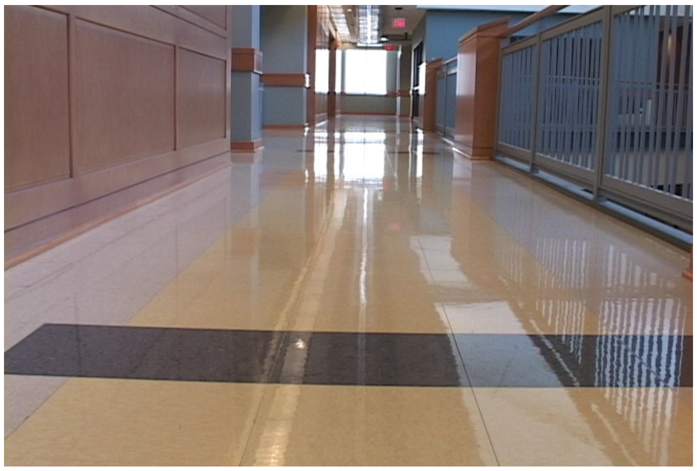
Training and Segmentation

The system is trained simply by choosing and labeling examples from the acquired images. The graphical user interface (GUI) of the system is designed to facilitate the rapid collection of feature vectors. Once a sufficient number of vectors has been collected, a 3-way approximate nearest neighbor search tree is constructed from these vectors in order to consolidate the LTM and reduce the search time. Currently, the system has been trained on 7 percepts and a color value has been assigned to each percept for displaying them as different clusters in segmented images. Table 2 shows the percepts and their respective color value.

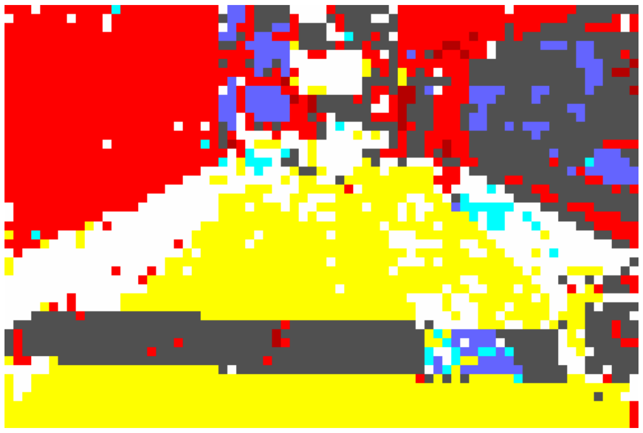
Table 2. The percepts and their respective color value.

Percept Label	Assigned Color Value
WoodPanel	Red
YellowFloor	Yellow
WhiteFloor	White
BlackFloor	Black
LightBlueWall	Light blue
Bricks	Dark red
BlueFloor	Blue

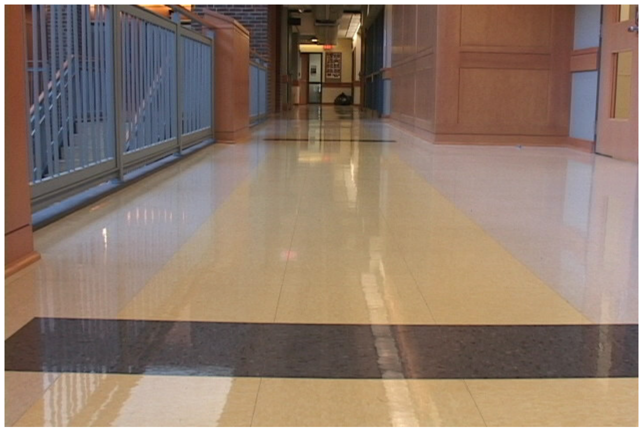
Figure 20 shows typical segmentation results of the system. These segmented images are obtained from a set of 58,198 feature vectors. The vectors were collected over several days at different times of the day to account for percepts in the environment under different lighting conditions. More segmented image results can be found in Appendix B. The results show that the system clearly segmented wood panels, white floor, yellow floor and black floor tiles. However, there are misclassified regions in the segmented image, especially for distant percepts. For instance, the blue railings are in general misclassified due to their thin structure. In order to get better results for this percept, the size of the moving window should be reduced to increase the resolution of the segmentation and capture the thin sections of the railing. However, the trade-off here is the increase in the resolution of the segmented image which will also increase the computational cost. The regions between the railings contain dark percepts which are similar to black floor tiles. Hence, these regions are sometimes classified as black floor tiles. The misclassification of the distant percepts is an expected result, because the percepts are getting smaller and darker on the east side of the hallway. On the west side of the hallway, intense sunlight is coming out of the window, creating white reflections on the surface of the percepts, which are sometimes classified as white floor percept. The wax on the floor tiles causes the surface to be extremely reflective. However, the regions where the reflections are not relatively intense are still segmented reliably. This shows that the perceptual system is very sensitive to hue information. The reflections of some of the percepts such as wood panel and blur railings, are reflected on the floor tiles. Yet the system is still able to identify most of the actual floor percept. Nonetheless, overall, it can be seen that the system produces very good and reliable results on segmentation for the near percepts.



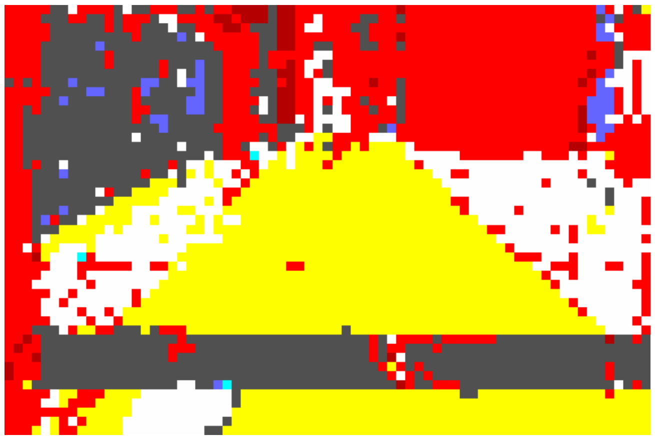
(a)



(b)



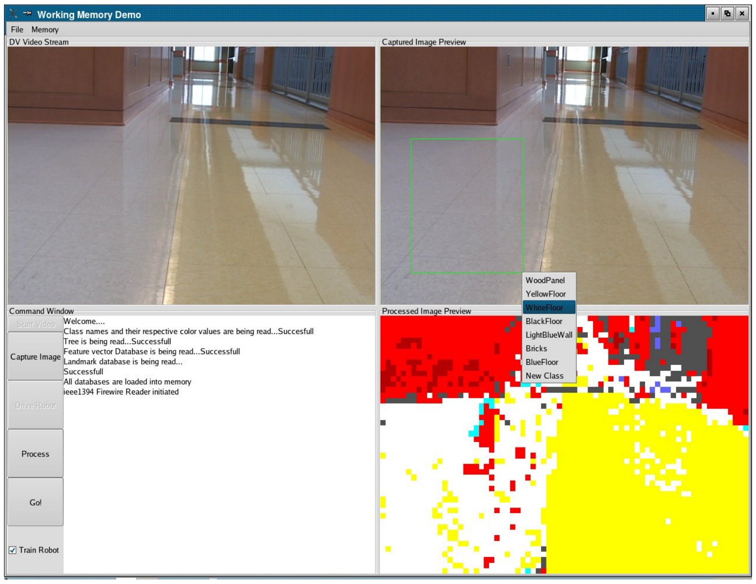
(c)



(d)

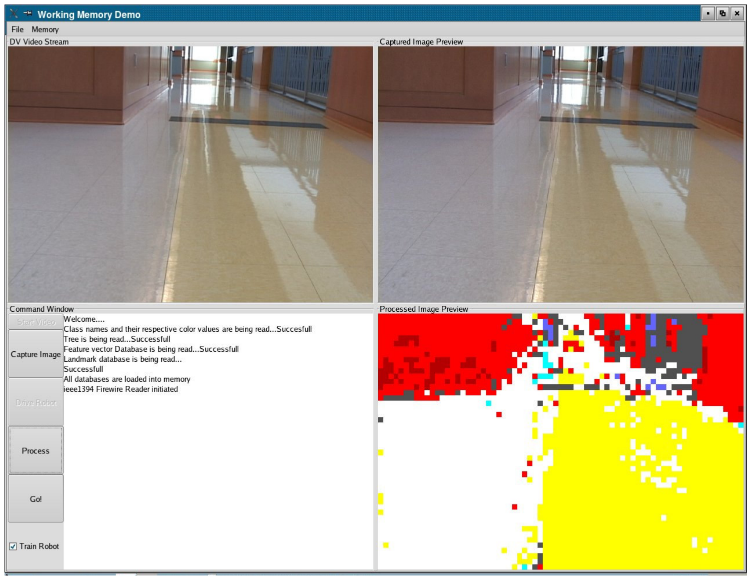
Figure 20. Typical segmentation results of the system. (a) West side of the hallway. (b) Segmented image of (a). (c) East side of the hallway. (d) Segmented image of (c).

As mentioned earlier in Chapter V that the non-parametric structure of the pdf estimation method enables simple updates to be performed on the fly. This enables the user to correct and update the system by adding new examplars to the system's LTM database. This flexibility allows the system to adjust itself online to a particular time of the day when different lighting condition is present. The only drawback of online training is the increase in computational cost, as the leaf nodes of the search tree also increases. However, it is possible to create a new tree branch when the samples in the leaf nodes are exceeding a desired threshold value. Thus, adding new samples to the tree does not effect the computational cost significantly. Moreover, only one or two updates are enough to correct the misclassifications of a percept. As a result, there is no need to recompute the search tree. Thus, with this capability, the system can be trained for new percepts in the environment at anytime. Figure 21 shows the result of such corrections as well as the structure of the GUI. The upper left corner of the GUI is the video stream captured from the digital camera. The upper right corner section of the GUI is the training region, where the training image is captured from the video stream. The user simply selects rectangular regions over the percepts in the environment, labels them and assigns a display color for the segmented image which is shown in the lower right corner of the GUI. In Figure 21 (a), we can see that some parts of the white floor have been misclassified as wood panel and yellow floor. Even though, these small misclassified regions are not presented as a chunk to WMtk, we can still correct the perceptual knowledge of the system in order to get better segmentation quality. The user corrects the system by selecting the misclassified region (green rectangular box shown in Figure 21 (a)) and selecting the correct label from the list of the percepts in the menu which pops up when a selection is made. The result of the correction is shown in Figure 21 (b). The selected region, as well as the other misclassified regions that occur in the white floor percept, are corrected. The same correction is also made in Figure 21 (c) for the wood panel percept. Some regions of the wood panel are classified as brick. Figure 21(d) shows the result of correction made for the wood panel.



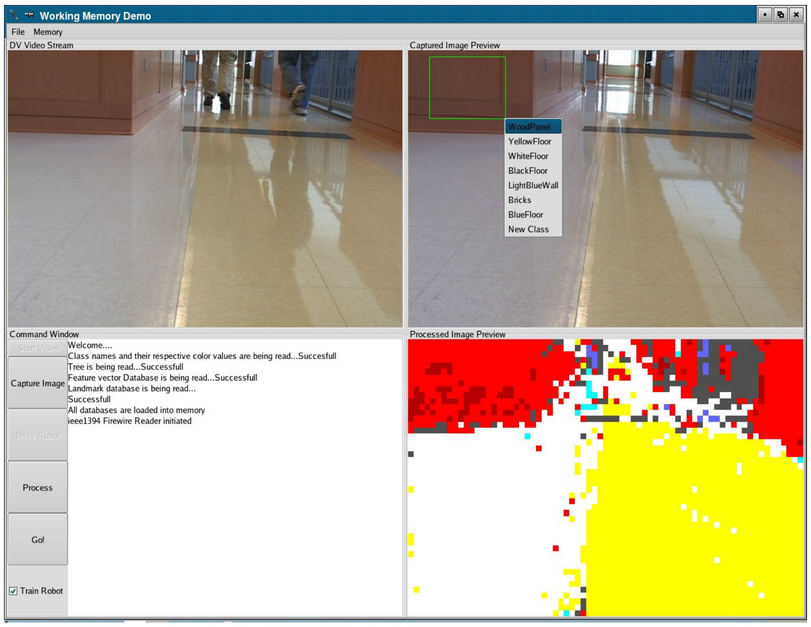
```
Welcome....
Class names and their respective color values are being read...Succesfull
Tree is being read...Successfull
Feature vector Database is being read...Successfull
Landmark database is being read...
Successfull
All databases are loaded into memory
ieee1394 Firewire Reader initiated
```

(a)



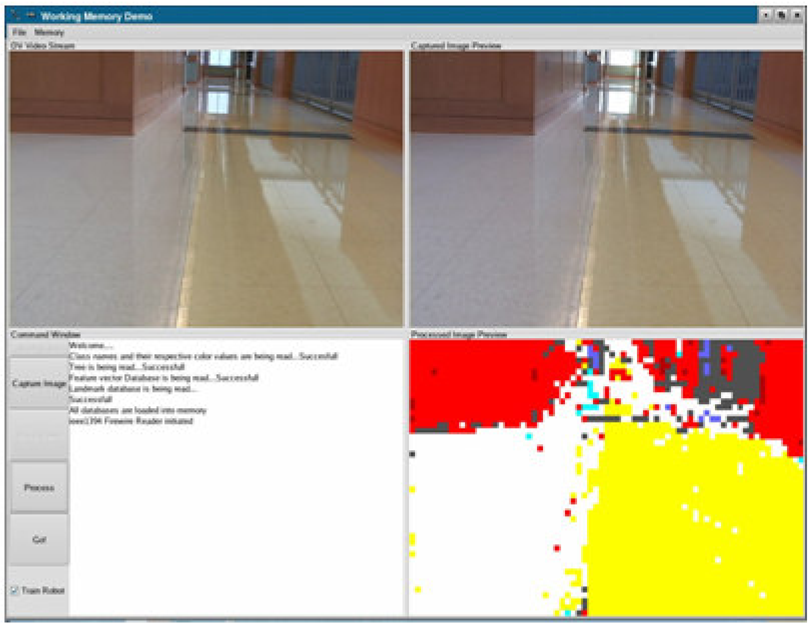
```
Welcome....
Class names and their respective color values are being read...Succesfull
Tree is being read...Successfull
Feature vector Database is being read...Successfull
Landmark database is being read...
Successfull
All databases are loaded into memory
ieee1394 Firewire Reader initiated
```

(b)



```
Welcome....
Class names and their respective color values are being read...Succesfull
Tree is being read...Successfull
Feature vector Database is being read...Successfull
Landmark database is being read...
Successfull
All databases are loaded into memory
ieee1394 Firewire Reader initiated
```

(c)

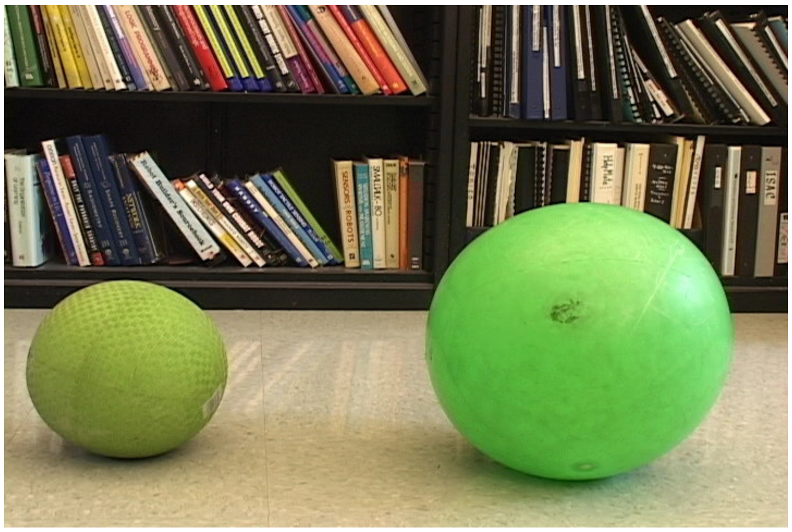


```
Welcome....
Class names and their respective color values are being read...Succesfull
Tree is being read...Successfull
Feature vector Database is being read...Successfull
Landmark database is being read...
Successfull
All databases are loaded into memory
ieee1394 Firewire Reader initiated
```

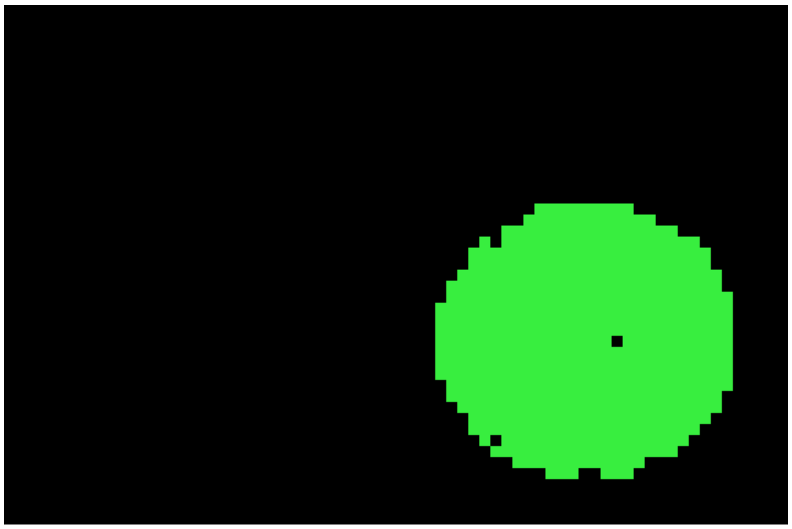
(d)

Figure 21. Simple corrections. (a) Correction made for the white floor. (b) Result of the correction made in (a). (c) Correction made for the wood panel. (d) Result of the correction made in (c)

The perceptual system is very sensitive to color differences. An example case is shown in Figure 22. The system is trained only for the green ball located on the right and shown in green color in the segmentation image. Anything that is different than the green ball object is shown as in black in the background of the segmentation image. We can see that the green ball on the left is not segmented at all, even though the colors are the same, but slightly different. There is also a green book in the book shelf and it is not segmented at all as well. We noticed that our perceptual system shows the same results at different times of the day and at different lighting conditions. This is a result of using a very high dimensional feature space, thus, the colors are finely differentiated.



(a)



(b)

Figure 22. The result of using a very high dimensional space. (a) The system is only trained for green ball located at the right section. (b) The system does not segment the ball on the left at all.

<u>Results of Experiment 1</u>

The task the robot is expected to perform in this experiment is to learn the meaning of the percepts with regard to motion. Once suitable percepts are formed and meaningful blobs are obtained through the perceptual system, the robot learns to associate these percepts whether they represent an obstacle or open space. With a WM size of 1, the weight parameters inside the WMtk are directly associated with the percepts, and are related to the expected reward associated with these percepts. Therefore, it is expected that the WMtk quickly adapts to choose the percept that receives higher reward values with respect to the other percepts in time. In order to evaluate the system's learning performance, two sets of experiments were conducted.

In the first set, the robot was put in the middle of the hallway and let go freely without any user interference. Figure 23 shows the plot of the cumulative reward weight associations for each percept versus the number trials. The WMtk has also a reserved chunk position when none of the chunks that are provided as input are selected. This reserved chunk is indicated in the plot as "No Selection" line. Initially, all the weights for each percept are set to 2.5, which is the average of the minimum and the maximum reward value. Assume that the rewards were set to zero and there were three chunks in the working memory. If two of the chunks represent an obstacle for the robot, most likely the WMtk would make a random choice for the first episode, since no reward signal is yet associated for each percept. If the chosen percept was an obstacle, still the robot would move towards it and receive a less amount of reward. In the next episode, if the working memory had the same obstacle chunk as well as a chunk, representing the open space, the WMtk would select the chunk which represents an obstacle. The reason for this is that the chunks that represent open space have not received any reward and additionally they were all set to zero. Hence the WMtk would select the chunk, representing an obstacle for the robot. However, initializing the weights to 2.5 forces the WMtk to explore chunks that represent open space, when the rewards, which are associated with the obstacle chunks, are lowered.

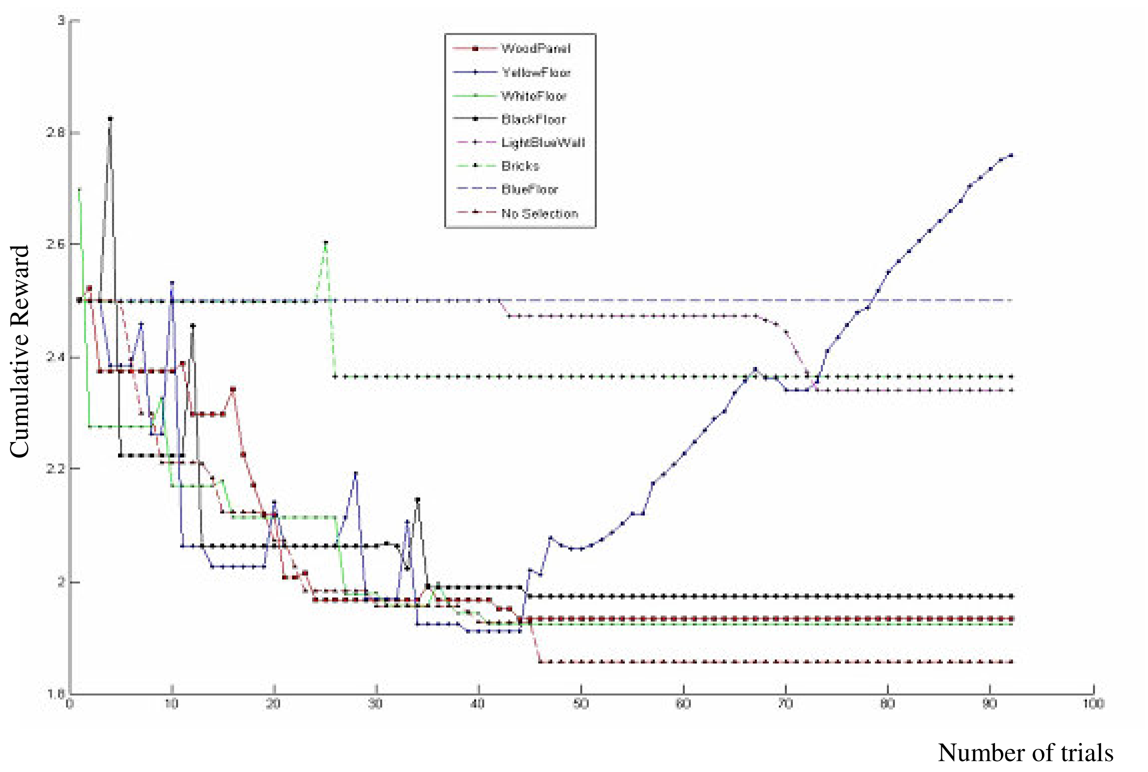


Figure 23.Plot for the reward weights for each percept for each trial.

The percepts that are detected decrease in value until around episodes 50-70 where the dominant percept, labeled as "YellowFloor" begins to grow in a linear fashion. All percepts that are not encountered often do not change much or at all from the initial value. Note that only percepts that are located at the lower half of the image are presented to the working memory. Therefore, some of the percepts are encountered rarely during the motion of the robot, even though they are still in the LTM of the robot.

After "YellowFloor" percept is determined to be the dominant percept associated with open space, it is blocked at the 90th episode to observe how the remaining percepts will be affected in working memory. Figure 24 shows the plot of this situation. The result shows that the percept labeled "BlackFloor" begins to grow and the other percepts that would represent less motion such as "WoodPanel" are further decreased. The blocking of the dominant percept allows for further development of the other percepts clearly showing a second best option for the open path.

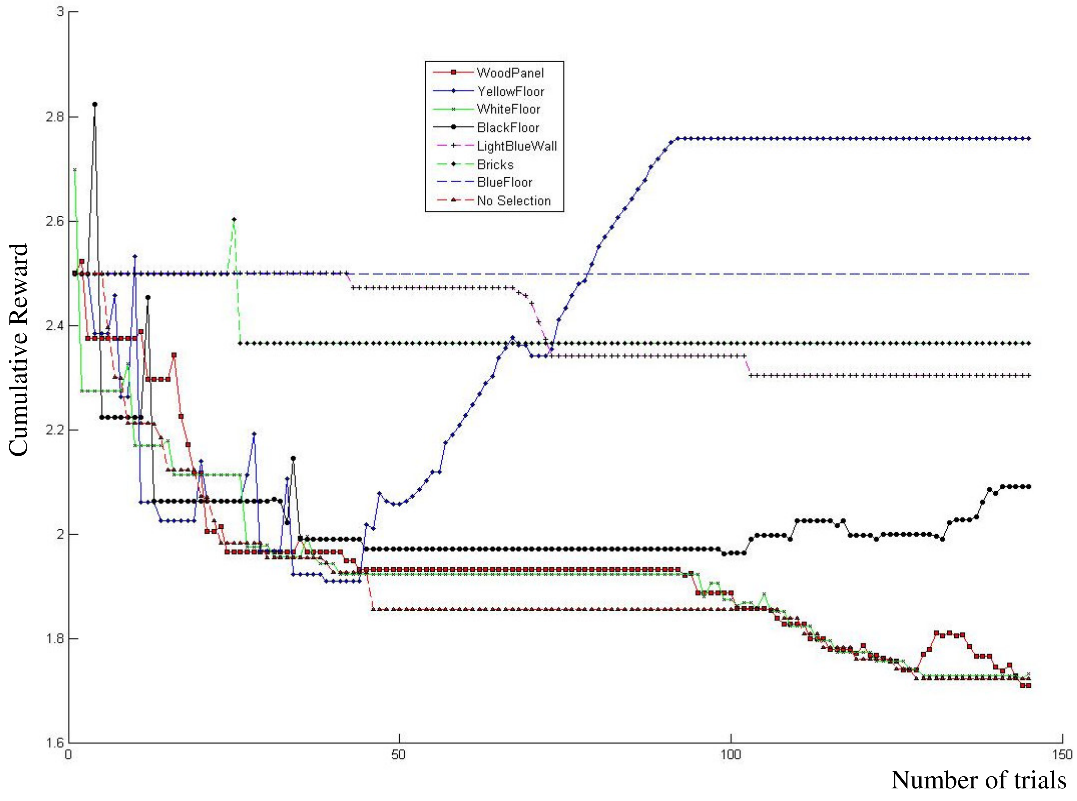


Figure 24. "YellowFloor" blocked at episode 90. "BlackFloor" percept seems to be the second option for the open path.

In the second set, instead of letting the robot go freely, after each trial, the robot was put back to the center of the hallway to begin the next trial. Figure 25 shows the plot for this situation. The purpose here is to see whether this method may improve the learning speed of the robot or not. However, the conditions of free travel versus forced centered travel seem to make no significant difference for the speed with which the robot learned this task.

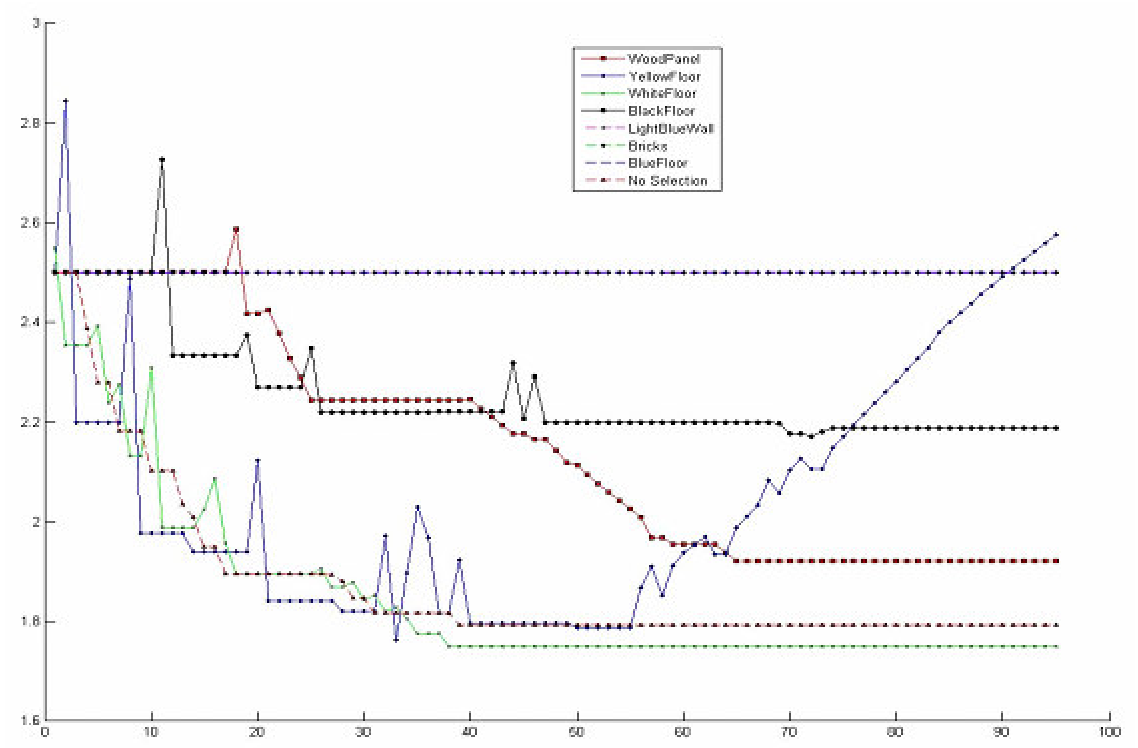


Figure 25 The robot is put to the center of the hallway after each trial.

<u>Results of Experiment 2</u>

In this experiment, the robot utilizes the working memory to learn to recognize landmark locations in the environment. Localization of the robot in the environment is essential for a successful navigation. The black tiles in the hallway are a convenient percept to select as landmark locations. In order to train the system, a set of training images are obtained from the hallway and labeled to indicate the presence of a landmark. The set contains equal number of images that contain a landmark and not contain a landmark. The images are selected randomly and segmented by means of the search tree and the perceptual database.

As mentioned in the experiment description, once the chunks are obtained, they are presented to the working memory as pairs. For example, if the "YellowFloor" percept is recognized in the image, then it is presented to the working memory system as "YellowFloor_L" and "YellowFloor_NL". The reason that the percepts are presented as pairs is to have the working memory system to consider all the possibilities for a percept, which could represent a landmark or not. It is expected that the system should learn "BlackFloor_L" as the landmark percept and if the working memory size is greater than 1, the system should output other percepts with "NL" suffix, denoting that they are considered as non-landmark percepts. In the case of working memory size greater than 1, a voting mechanism is applied to determine the final decision of the working memory system. For example, if the working memory size is three and the selected chunks are "YellowFloor_NL", "WoodPanel_L" and "WhiteFloor_L", the vote result will be in favor of presence of a landmark in the image. However, since in this case the image would be labeled as non-landmark image, the system would not get a positive reward. If the voting result agrees with the image label, the system receives a positive reward of 1, else it receives a zero as a negative reward.

The WMtk has a parameter in order to adjust the exploration rate. A higher value for this parameter increases the chance that the working memory system will explore other options in the choices rather than just pursuing the maximum expected reward. The percentages noted in Table 3, result from the trials with the exploration percentage set at 0.05. Table 4 is the result using only the weights within the working memory to determine landmark choices without exploration, i.e., the exploration parameter is set to 0. The percentage values for both tables are including the early misclassifications that occur in the WM training. The percentages are higher in Table 4 than in 3 and more accurately represent the performance of WM after training. Each of the WM sizes is trained for both 1000 and 5000 episodes.

Table 3. Percentage correct for Landmark classification with WM exploration set at 0.05

WM size	1000 trials	5000 trials
1	93.10%	85.86%
2	90.00%	93.20%
3	95.30%	96.60%

Table 4. Percentage correct for Landmark classification with WM exploration set at 0.00

WM size	1000 trials	5000 trials
1	98.10%	99.52%
2	95.50%	97.42%
3	98.00%	99.38%

Figure 26 shows the learning curve of the system for each working memory size and exploration rate values, as a plot of the average reward values over a sliding 25 point window over 1000 trials. Figure 27 shows the same plots over 5000 trials.

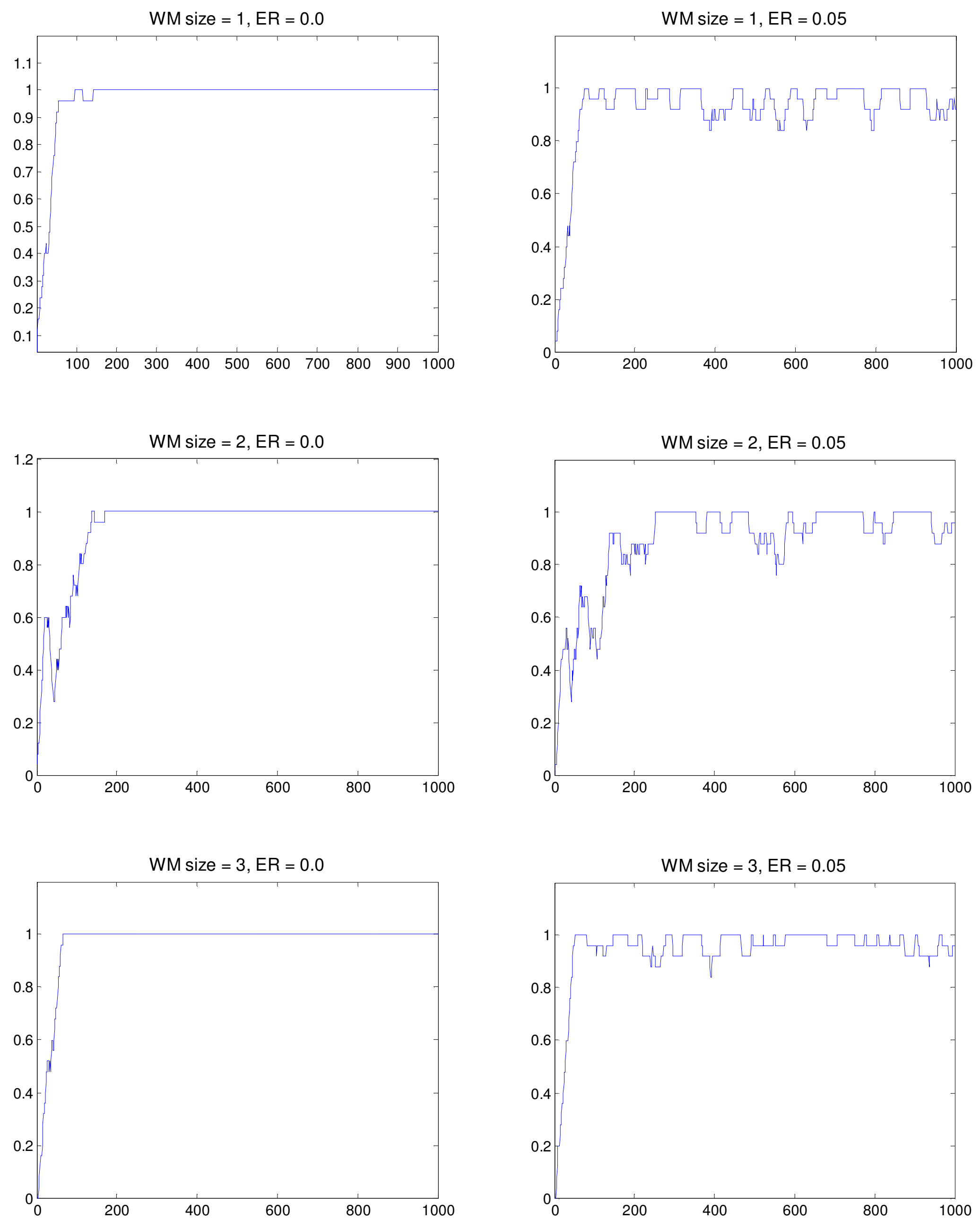


Figure 26. Average rewards for each case (WM size = 1,2,3 and Exploration rate (ER) = 0.0, 0.05) over 1000 trials.

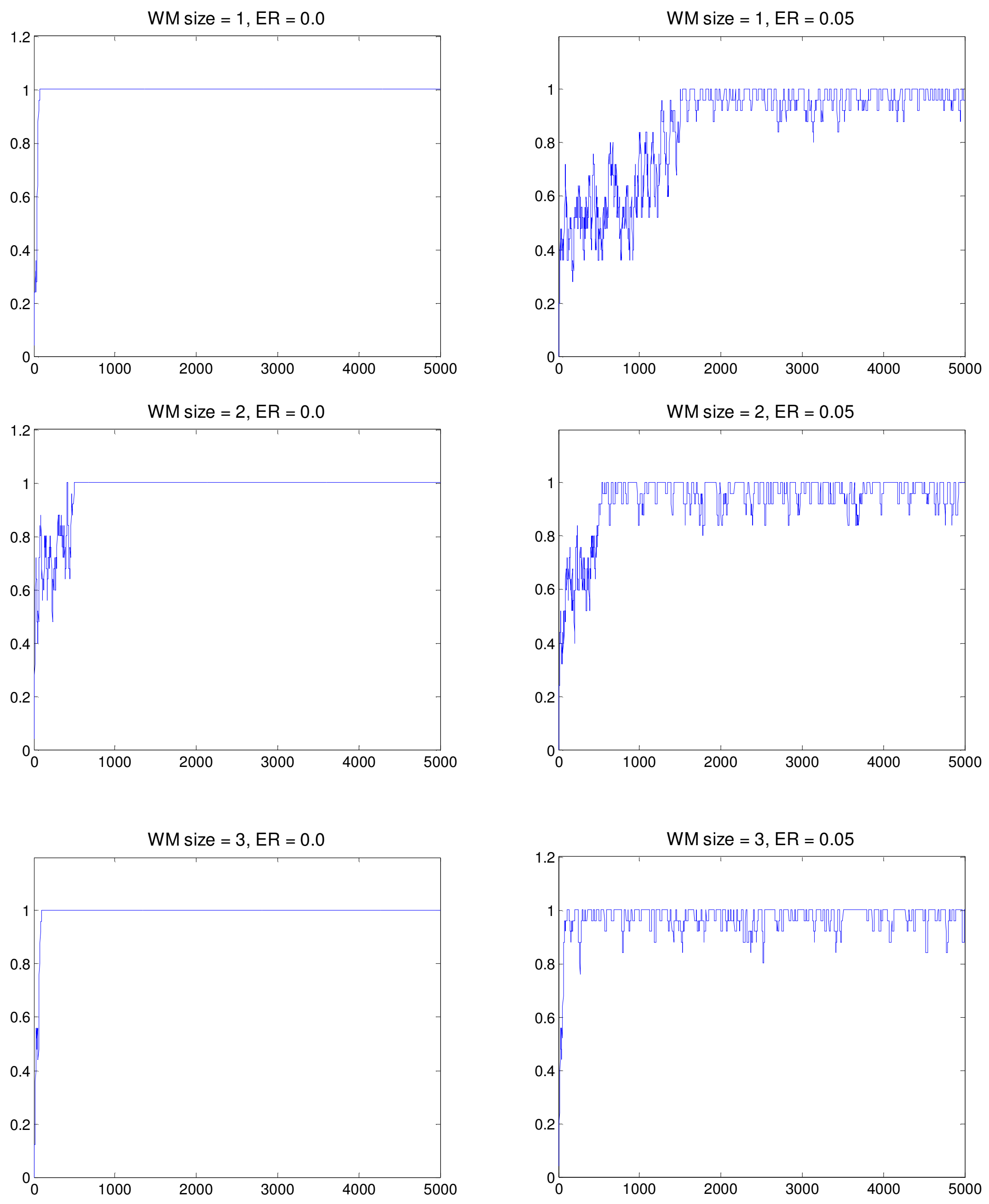


Figure 27. Average rewards for each case (WM size = 1,2,3 and Exploration rate (ER) = 0.0, 0.05) over 5000 trials.

The results are quite good in all cases. The results, which are obtained by setting the exploration rate to 0.05, yields lower accuracy, because the system is exploring new possibilities and thus making more mistakes. However, when this parameter is turned off, the results are much more accurate, as expected.

Results of Experiment 3

In this final experiment, the robot is required to learn a navigation task. In order to perform a navigation task, it should navigate safely in an open path, localize itself in the environment as well as choose the right motor commands. So far, the robot knows which percepts are safe to move, as well as what percept indicates a landmark. Each of these skills has been encapsulated and will be used to perform a vision-guided navigation task.

As mentioned in the description of the experiment earlier, the maximum number of input chunk vectors is seven. The training is performed in a simulation environment and all the possible chunks are presented to the robot at each time step. The robot is trained for each region of the navigation path as shown in Figure 28. Each region may be thought of as the state of the robot (i.e., state 0 is being located at region 1, and state 1 is at region 2, etc.). Furthermore, the detection of the contingent percept is also represented as a state of the robot. However, during the training, this state is randomly set to true once, in one of the regions or not set to true at all. This means that the contingent percept has been detected at a particular region if set to true or not detected if the state stays false in the entire trial. In the real world experiment (not in simulation), this state is set true once the perceptual system detects the contingent percept. The robot is required to choose the correct action chunk at each state. In addition, it must learn to remember the contingent percept if recognized along the navigation path.

The learning criteria faced by the working memory are not trivial, indeed they are challenging. The robot needs to learn a considerable amount of information from a large combinatorial chunk space in five different states. The robot is trained for each region (each of the states from zero to three) of the navigation path and a reward value is given.

However, in order to speed up the learning, the training state is switched from one to another, only if multiple numbers of positive rewards are received in a row at each state. For instance, assume that the number of positive rewards required to switch the state from one to another, is ten. A successful trial is defined as not receiving any negative reward. Therefore, the system must receive 40 positive rewards (10 for each region) in a row in order the trial to be considered as successful. The number of received positive rewards in a row is referred to as the positive reward threshold (PRT). Figure 29 shows the learning curve of the system, as a plot of the average reward values over a sliding 25 point window over 10000 trials for several PRTs. Table 5 shows the number of successful trials for each plots in Figure 29.

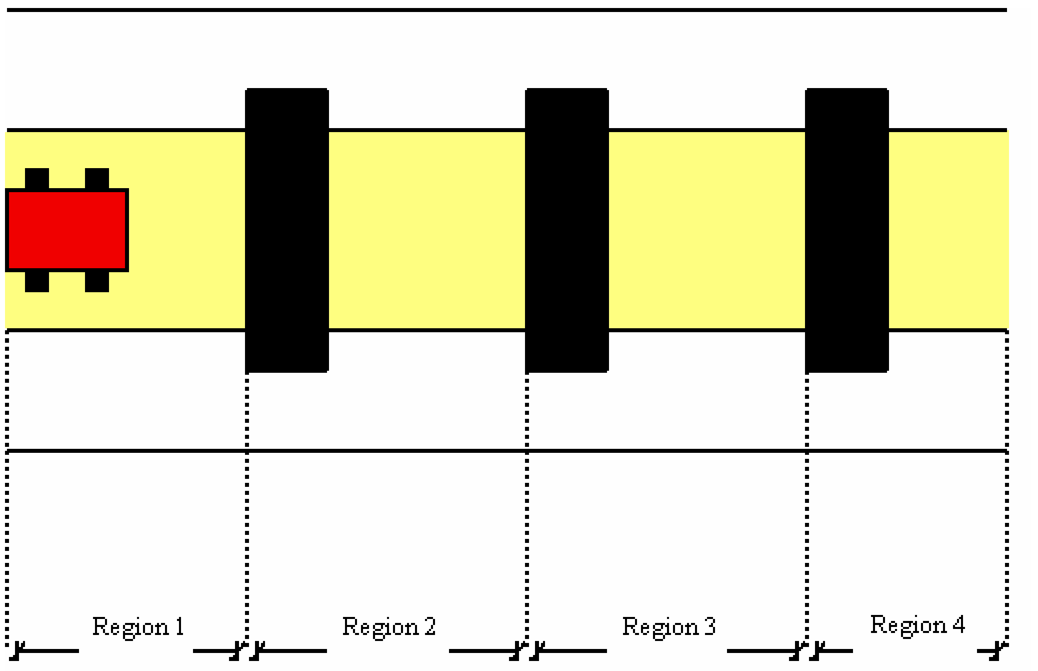
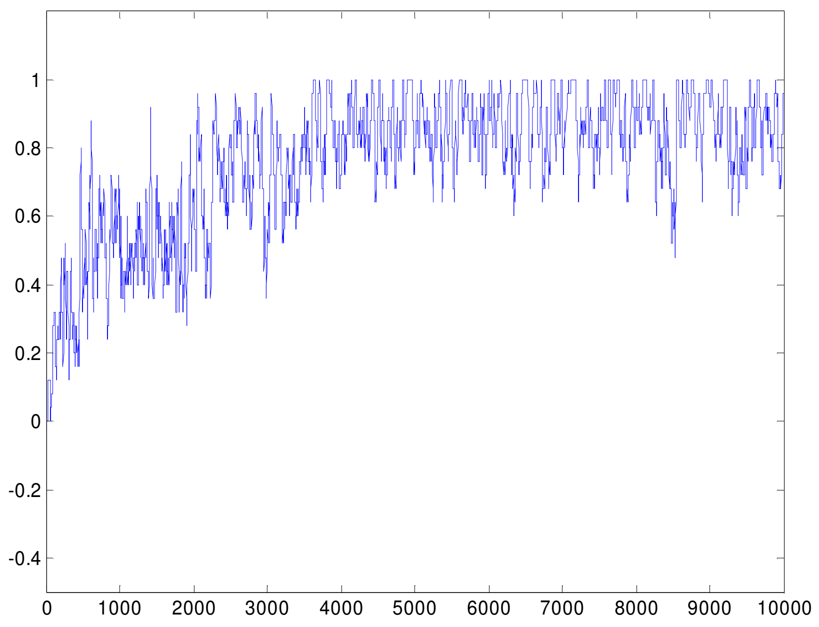
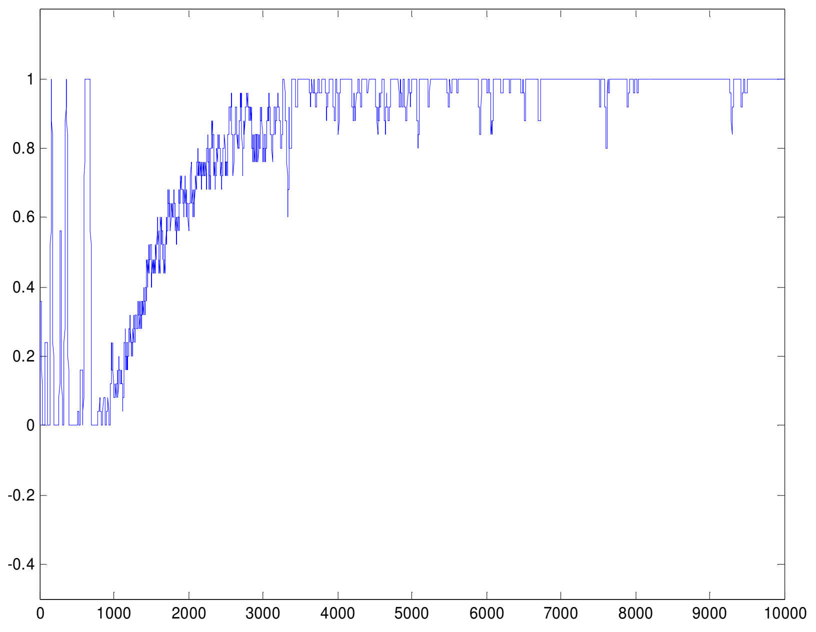


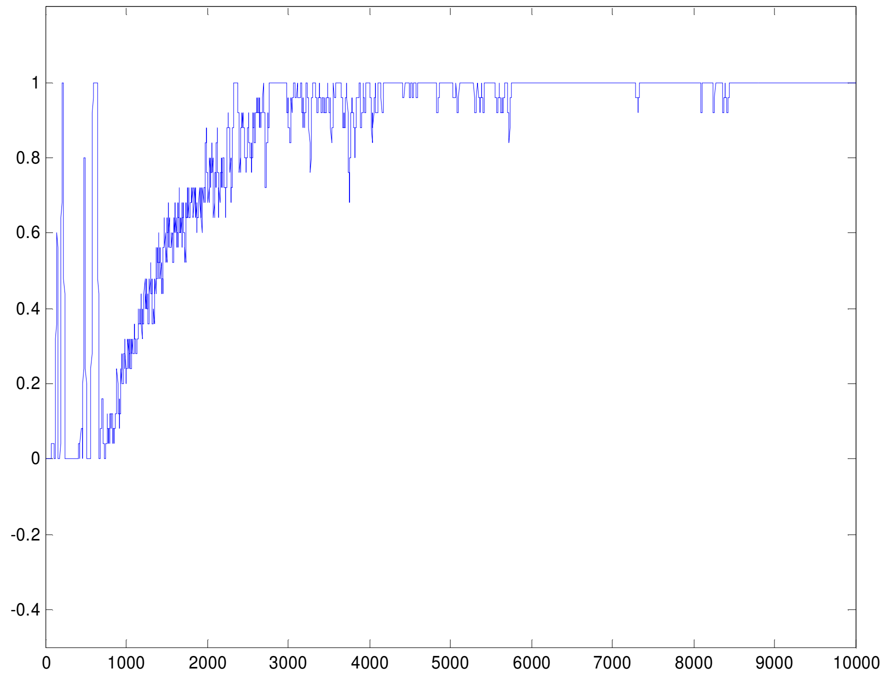
Figure 28. Sketch of the navigation task and the regions that the robot is trained with.



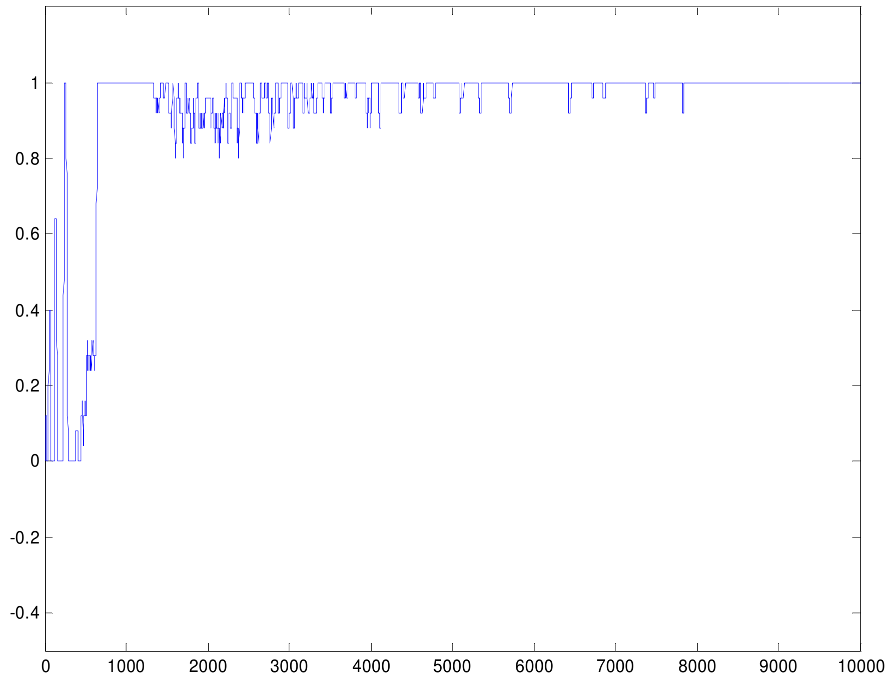
(a) Learning curve for PRT = 1



(b) Learning curve for PRT = 7



(c) Learning curve for PRT = 10



(d) Learning curve for PRT = 20

Figure 29. Learning curve of the system for several PRTs.

Table 5. Number of successful trials for 10000 trials and for different reward thresholds.

Positive reward threshold	Successful trials (Out of 10000)	Percentage of successful trials
1	7591	%75.91
7	8275	%82.75
10	8502	%85.02
20	9330	%93.30

These numbers are not unreasonable, considering the fact that we are imposing a very strict rule, which is receiving multiple positive rewards in a row. Moreover, the initial training stage is also included in these results. If at any time step of the training, the system fails to choose the correct chunk combination, the whole trial is considered to be an unsuccessful trial. In all of the cases, there is a positive learning trend. As the PRT is increases, the system makes fewer mistakes overall and the speed of the training is significantly increased in terms of episodes. With the system settings described in Appendix A, the time required to train the system, when PRT is set to 10, is 12 minutes and for a PRT set to 20, it takes 15 minutes. Thus, the training time is relatively short in a simulation environment.

The robot successfully navigated in the hallway and it came back to its initial position when a green ball percept is presented. The robot chose the correct actions in the action selection slot and chose the "Nothing" percept, meaning that no percept is held in the working memory. However, once the contingent percept is presented along the navigation, the robot remembered the contingent percept in the percept selection slot of the working memory at each time step. Thus, the robot has successfully learned a complex task based on previously learned fundamental behaviors.

The training of the task is highly supervised in the sense that once the perception system detects the contingent percept, it is encoded in the state vector. Thus, it is supervised to detect the contingent percept; however, it is not supervised to remember the contingent percept in the next time steps. Thus, we wanted to relax the system and do not encode the detection of the contingent percept in the state vector. In this case, the system needs to learn to detect the contingent percept and remember it. As a proof of concept, we trained the system for the first region of the navigation, and the system indeed successfully learned to remember to hold on to the contingent percept in the working memory.

CHAPTER VIII

CONCLUSION AND FUTURE WORK

Conclusion

In this work, the robot successfully learned a complex, vision-guided navigation task by means of a powerful and reliable perceptual system combined with a biologically inspired prefrontal cortex working memory model. The task faced by the robot is indeed a challenging task to be learned. The robot initially learned the percepts in the environment and robustly performed segmentation. Next, the robot associated the meanings relative to motion, and landmark detection, to the percepts in the environment and encapsulated them as simple behaviors. These behaviors were then used to build up a more complex behavior. In each stage of the task, only a single camera is used as the main sensory input for the robot.

The working memory model (WMtk) used in this system, has significant distinctive features when compared with the traditional reinforcement learning algorithms. First of all, the robot is not restricted to consider only one item from a small set. The capability of considering multiple items in a situation considerably increases the robot's learning ability. This is a result of utilizing the conjunctive coding implementation exploited in the model. As a consequence of this, the robot can focus on combinations of percepts that are most rewarding and therefore, most relevant to the task. Secondly, as demonstrated in this work, the model can learn complex tasks. The only drawback observed in the task is that the number of trials required to train the system for such a complex task is quite large and it may make it impractical to train the system in some situations. We believe that the reason for this is essentially there is not sufficient communication between the human and the robot. For example, in the working memory experiments that are performed with monkeys, the monkey does not have sufficient communication with the human, except a sip of juice, as a reward for its correct actions. As a result, as the task gets more complex, it is natural to expect more training. Future work should address ways to speed up the training session. One way may be that the human can describe what the robot needs to learn in natural or simplified language terms. This may result in pre-biasing the weights in the conjunctive coding, which may speed up the training significantly. In conclusion, the model still learns the task successfully, especially from a very large combinatorial chunk space. With the system specifications given in Appendix A, the time required to train the system is in the range of minutes and it is relatively straightforward to design a simulator to train the system.

The perceptual system is capable of learning percepts online in uncontrolled/unmodified environments. It is not difficult to obtain thousands of feature vectors from a few images. The system robustly discriminated the percepts under different lighting conditions and different times of the day. This demonstrates that the system learns to generalize the environment under different conditions. We believe the reliable segmentation is a consequence of having an extremely high dimensional space. However, the computational cost is strongly not affected from having such a high dimensional feature space. This is a result of using a sparse vector representation. Additionally, the memory storage is dramatically reduced by using this representation. Therefore, we found that the idea of using this representation is quite useful. Moreover, the high dimensional color histograms have several important advantages. First of all, they are rotational and translation invariant. In addition, they are mildly scale invariant. For instance, the small changes in the scale of a color pattern do not affect the color histogram until the boundaries of the color pattern are outside the region of the histogram. Thus, for a practical range of scales it is reasonably scale invariant. The most important advantage is the sparse representation, because the computational cost only depends on the size of the region in the image.

In conclusion, the system developed in this work, possesses a rich, powerful perceptual system, which can reliably discriminate percepts and provides robust performance in complex, unstructured and unmodified environments and supports research into developmental robotics (and open-ended developmental robotics), such that it can learn behaviors and concepts which can, in turn, be used to learn more complex behaviors.

Future Work

Context Change

Navigation is one of the fundamental tasks in mobile robot applications. There may be times when the environment of the robot can change dramatically. In the following the "context change" will be referred to as a significant change of the robot's environment. For example, moving from indoors to outdoors could be considered as a context change. If the robot has the capability of detecting the context changes, then the robot might be able to deduce some important properties for its navigational tasks. Consider the following situation, depicted in Figure 30. Assume that the robot was trained for Context 1 and Context 3, but not for Context 2. If the robot starts moving from

Context 1 towards Context 3, it has to pass through Context 2. As the robot comes to Context 2, it may detect the new context by comparing its perceptual knowledge of Contexts 1 and 3 deducing that it has not been trained in Context 2. Once it reaches its final destination in Context 3, it would therefore, also deduce that it is not possible to navigate from Context 1 to 3 without passing through Context 2. Thus, context change may also aid the formation of topological maps.

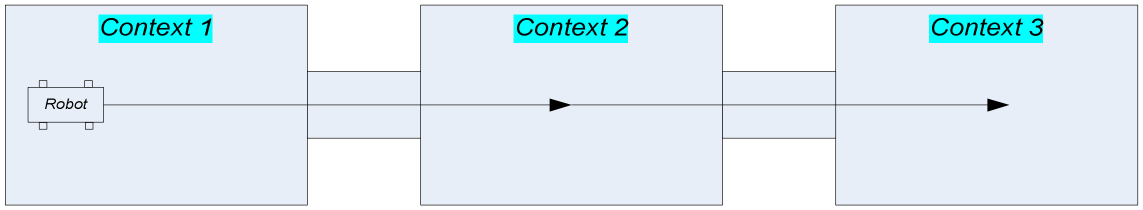


Figure 30. A navigation path. Robot detects that Context 2 is novel and it is a waypoint between Contexts 1 and 3.

Parallel Computatation

The reason that the robot is moved in small distance increments is that the segmentation process is not in real time. Given the specifications in Appendix A, the time required to segment the percepts in a frame is approximately 8 seconds in the current implementation. However, today's graphical processing units (GPUs) offer fast and parallel computing architecture for even non-graphical processing. Figure 31 shows the comparison of floating operations per second for a CPU and several GPU models. In [63], it is stated that the main reason behind these results is that GPUs are specialized for highly parallel computation, which is exactly what graphic rendering is about. Thus, more transistors are dedicated to data-processing than data caching and flow control. Each of the feature vectors from the small regions of the image can be extracted and segmented using a parallel architecture. In this way, we believe that the segmentation process may approach real time, using a parallel implementation.

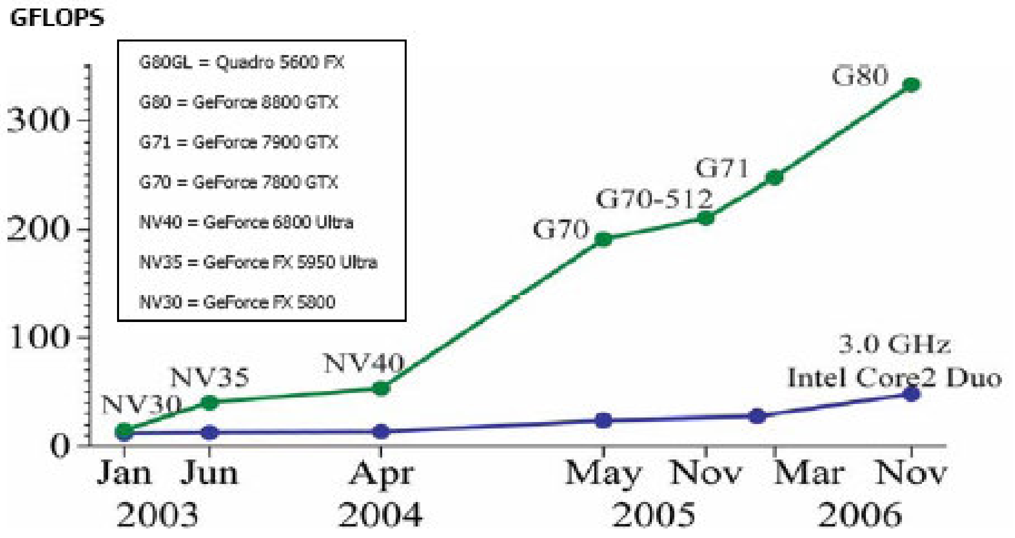


Figure 31. Floating point operations for a CPU and few GPU models [63].

Forgetting Behavior

Even though a fast search tree algorithm is implemented, as the samples in the feature vector database grow, the computation cost of the segmentation process increases. In order to obtain generalization, it is desired to obtain a large number of training feature vectors. However, this does not necessarily mean that all the obtained training samples will equally contribute to the efficiency of the search tree. Thus, only maintaining the feature vectors that are most significant for the segmentation of a particular percept may reduce the computational cost dramatically. This process can be performed in a straightforward manner by only maintaining the vectors that are needed to describea particular cluster in the vector space. All of the vectors in a tightly bunched cluster need not be retained. Discarding redundant feature vectors can be related to *forgetting* in human beings.

APPENDIX A

THE TEST BED AND SYSTEM SETUP

One of the objectives in our research is only to use vision as our main sensory input to the system. The system is relatively a low-cost hardware setup which is not accurately calibrated or highly precise. The versatile Pioneer 2-AT robot has been chosen as our test bed mobile robot platform. It originally comes with a 16 sonar sensor rings, 8 located in the rear and 8 in the front. However, only the front sonar rings are used for obstacle avoidance purposes in our work. The robot is shown in Figure 32.

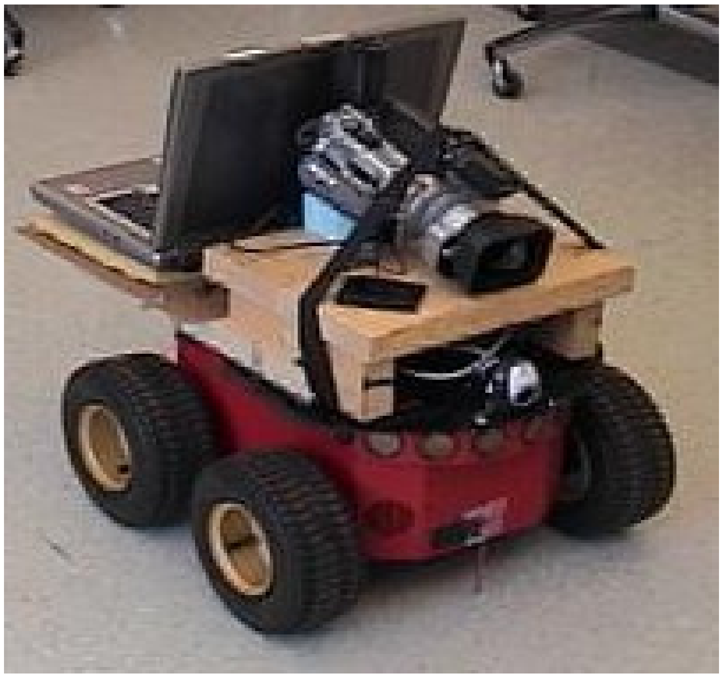


Figure 32. The test bed: Pioneer 2-AT

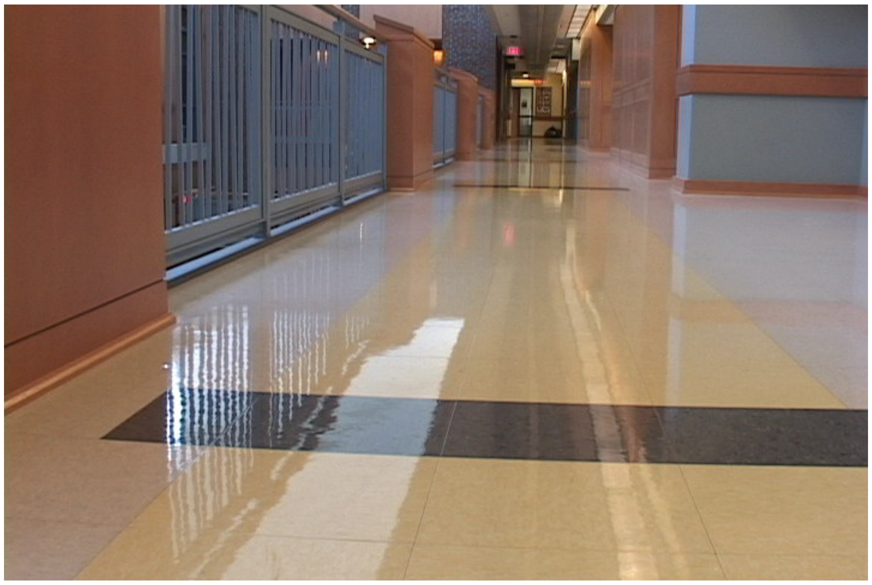
In order to provide the necessary computational power, a laptop computer is strapped on the robot. The laptop has a X86 family, Intel Pentium M 2.13 GHz processor, 1024 MB RAM, and 256 MB nVidia Quadro FX Go1400 video card. The implementation is made on a Linux based operating system, using C++ programming language.

Perception is provided by a Sony DCR – VX2000 digital video camera. The streaming video is captured via an IEEE 1394 firewire interface, connected to the laptop. The resolution of the video stream is 720 x 480 pixels and the same resolution is used for segmenting the percepts (no downsampling is applied). The camera is tilted down about 15°, since the near percepts have higher priority than the distant ones.

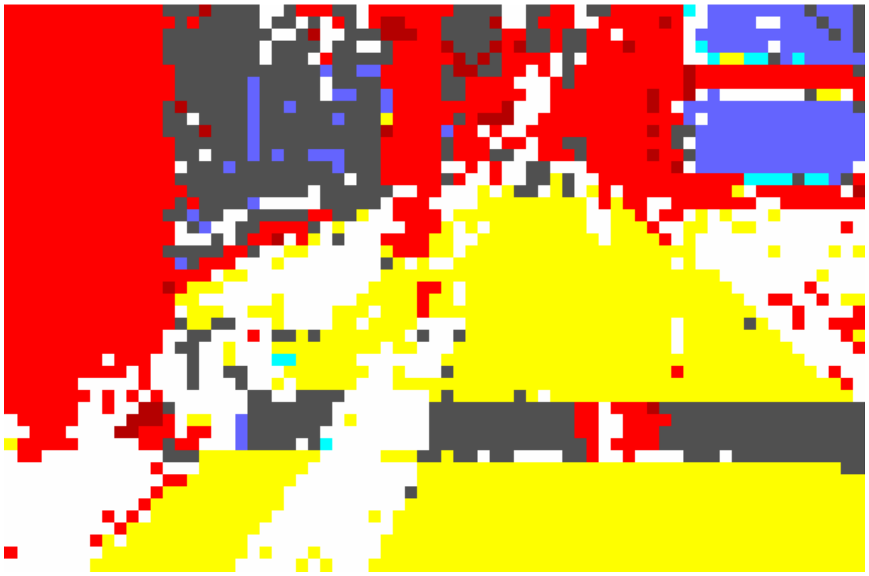
The laptop controls three different processes. The first one is the driver which controls the robot via an RS232 serial port. The second process is dedicated to capturing the video stream from firewire interface. The last one is the GUI processing thread, where the main computations take place.

APPENDIX B

SEGMENTATION EXAMPLES

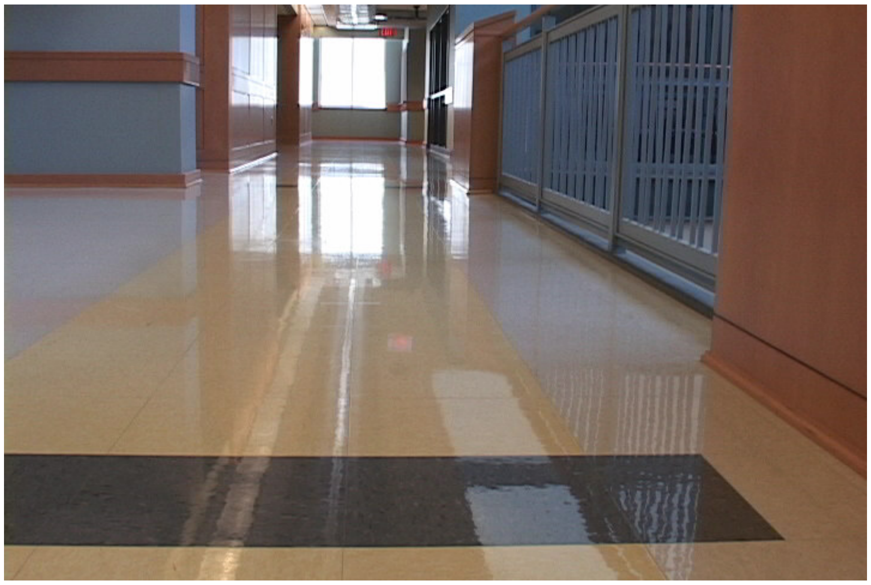


(a)

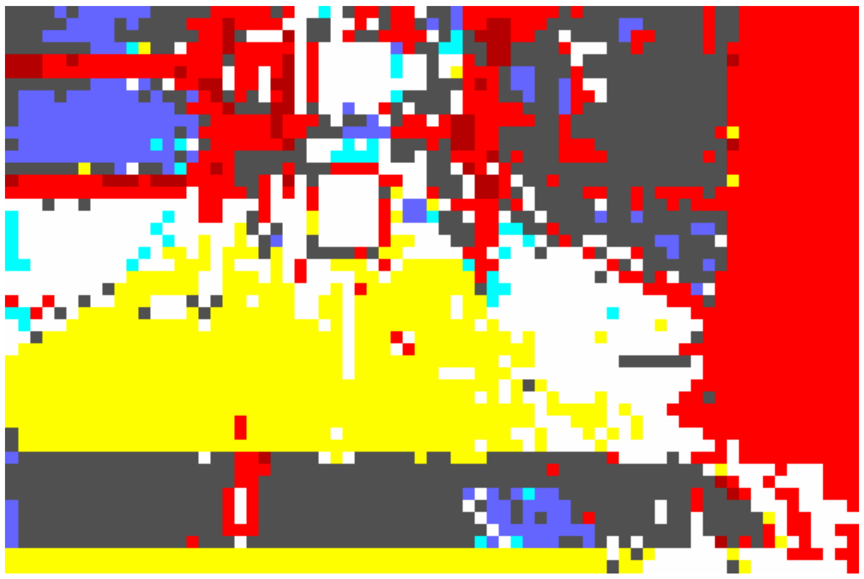


(b)

Figure 33. (a) The hallway with significant specular reflections on the floor tiles. (b) Segmentation of (a). Note that the some of the errors are actually due to reflections in the "mirror-like" floor.

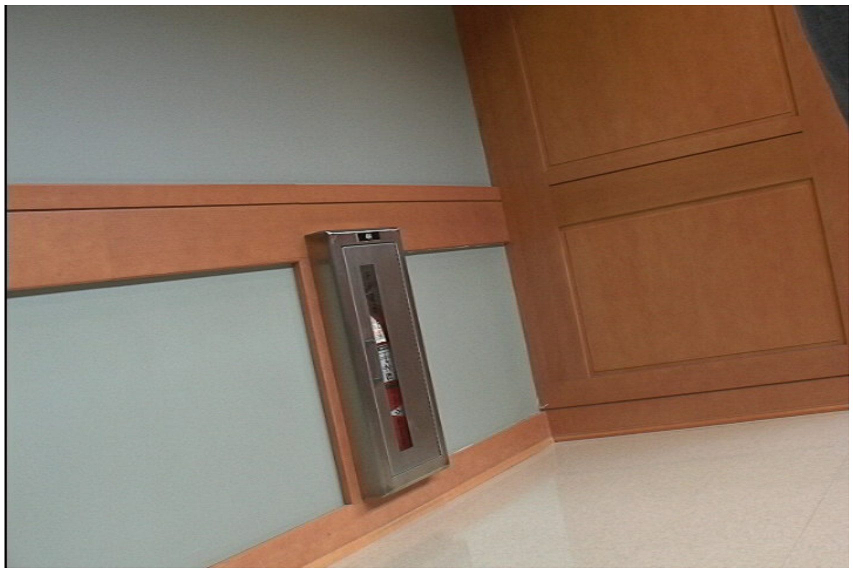


(a)

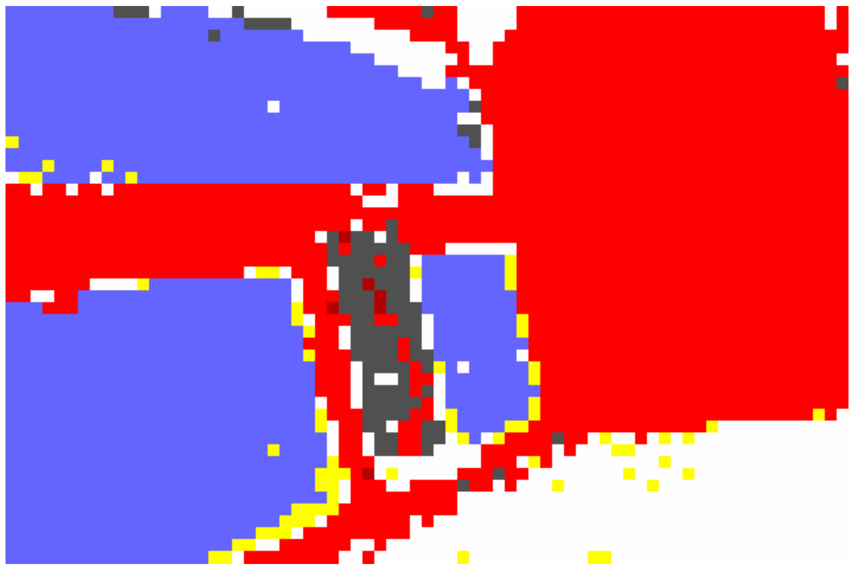


(b)

Figure 34. (a) The hallway with more specular reflections. (b) Segmentation of (a). Most of the percepts, especially the near ones, are segmented properly.



(a)



(b)

Figure 35. (a) The corner of the hallway with less reflection. (b) Segmentation of (a). The percepts are segmented with negligible errors.

REFERENCES

[1] Fuster, J. M. and Alexander, G. E. (1971). Neuron activity related to short-term memory. Science, 173:652–654.

[2] Funahashi, S., Bruce, C. J., and Golman-Rakic, P. S. (1989). Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. Journal of Neurophysiology, 61:331–349.

[3] Miller, E. K. and Desimone, R. (1994). Parallel neuronal mechanisms for short-term memory. Science, 263:520–522.

[4] O'Reilly, R. C., Braver, T. S., and Cohen, J. D. (1999). A biologically based computational model of working memory. In (Miyake and Shah, 1999), chapter 11, pages 375–411.

[5] Brooks, R.A., "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14-23, 1986.

[6] Peters, R.A., Hambuchen, K.E., Kawamura, K., and Wilkes, D.M., "The sensory egosphere as a short-term memory for humanoids," *Proc. IEEERAS Int. Conf. Humanoid Robots,* Tokyo, pp. 451-459, November, 2001.

[7] Kawamura, K., Koku, A.B., Wilkes, D.M., Peters, R.A., and Sekmen, A.,"Toward egocentric navigation," *Int. J. of Robotics and Automation,* vol. 17, no. 4, pp. 135-145, 2002.

[8] Sutton, R. S. (1988). Learning to predict by the method of temporal differences. Machine Learning, 3:9–44.

[9] Montague, P. R., Dayan, P., and Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive hebbian learning. Journal of Neuroscience, 16:1936–1947.

[10] Braver, T. S. and Cohen, J. D. (2000). On the control of control: The role of dopamine in regulating prefrontal function and working memory. In Monsell, S. and Driver, J., editors, Control of Cognitive Processes, volume XVIII of Attention and Performance, chapter 31, pages 713–737. MIT Press.

[11] O'Reilly, R. C., Noelle, D. C., Braver, T. S., and Cohen, J. D. (2002). Prefrontal cortex and dynamic categorization tasks: Representational organization and neuromodulatory control. Cerebral Cortex, 12:246–257.

[12] Sutton, R.S. and Barto, A.G., *Reinforcement learning: an introduction.* MIT Press, Cambridge, Massachusetts, 1998.

[13] Oliver, N.M., Rosario, B., and Pentland, A.P., "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. PAMI*, vol. 22, no. 8, pp. 831-843.

[14] Hongeng, S., Bremond, F., and Nevatia, R., "Bayesian framework for video surveillance application," *Proc. ICPR 2004*, pp. 164-170, August, 2004.

[15] Piater, J.H. and Grupen, R.A., "Feature learning for recognition with Bayesian networks," *Proc. ICPR 2004,* pp. 17-20, August, 2004.

[16] Merriam Webster., www.m-w.com

[17] E. Gibson, "Perceptual learning," *Ann. Rev. Psychol.*, vol. 14, pp. 29–56, 1963.

[18] Fahle, M. and T. Poggio (2002). Perceptual Learning. Cambridge,MA: Bradford.

[19] Bredeche, N., Z. Z. Shi, et al. (2006). "Perceptual learning and abstraction in machine learning: An application to autonomous robotics." Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews **36**(2): 172-181.

[20] Edelman, S. and N. Intrator (2002). Models of Perceptual Learning. Perceptual Learning. M. Fahle and T. Poggio. Cambridge, MA: Bradford.

[21] Duda, R. O., P. E. Hart, et al. (2001). Pattern Classification, Wiley-Interscience.

[22] Tovee, M.J. (1996). An Introduction to the Visual System, Cambridge University Press.

[23] Dayan, P. and Abbott, L.F., 2001. Theoretical Neuroscience, Cambridge, MA: MIT Press.

[24] von der Heydt, R., Peterhans, E. and Durstler, M.R., 1992. Periodic-pattern-selective cells in monkey striate cortex. *Journal of Neuroscience*, vol. 12, pp. 1416-1434.

[25] De Valois, R.L., Albrecht, D.J., and Thorell, L., 1978. Cortical cells: bar detectors or spatial frequency filters? In *Frontiers in Visual Science*, ed. S.J. Cool and E.L. Simith, Berlin: Springer-Verlag

[26] Wilson, F.A.W., O'Scalaidhe, S.P., Goldman-Rakic, P.S., 1993. Disssociation of object and spatial processing domains in primate prefrontal cortex. *Science*, 260, 1955-1958.

[27] Fuster, J.M., 1997. *The Prefrontal Cortex*, Lippincott-Raven Publisher, Philadelphia, New York.

[28] Miyake, A. and Shah, P., editors (1999) *Models of Working Memory: Mechanisms of Active Maintanence and Executive Control.* Cambridge University Press, Cambridge.

[29] Miller, G. A. (1956) The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review,* 63:81-97

[30] Cowan, N. (2001) The magical number 4 in short-term memory: A reconsideration of mental storage capacrity. *Brain and Behavioral Sciences,* 24(1):87-185.

[31] Atkinson, R. C., Shiffren, R. M. (1971) The control of short-term memory. *Scientific American*, 225, 82-90

[32] Baddeley, A. (2003) Working Memory: looking back and looking forward. *Nature Reviews Neuroscience,* 4(10):829-39

[33] Baddeley, A. D. and Hitch, G. J. (1974) Working Memory. In Bower, G. A. editor, *Recent Advances in Learning and Motivation,* Vol. 8: 47-90. Academic Press, New York.

[34] Baddeley, A.D. (2000). The episodic buffer: a new component of working memory? *Trends in Cognitive Sciences*, 4, 417-423.

[35] Phillips J.L. & Noelle, D.C. (2005). A biologically inspired working memory framework for robots. Proc. *of the 27th Annual Meeting of the Cognitive Science Society*, Stresa, Italy, July,

[36] Goldman-Rakic, P. S. (1996). Regional and cellular fractionation of working memory. *Proceedings of the National Academy of Science* USA, 93, 13473–13480.

[37] Watanabe, M., Kodama, T., & Hikosaka, K. (1997). Increase of extracellular dopamine in primate prefrontal cortex during a working memory task. *Journal of Neurophysiology*, 78, 2795–2798.

[38] Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80, 1–27.

[39] Dayan, P. and Abbott, L.F., 2001. *Theoretical Neuroscience*, Cambridge, MA: MIT Press.

[40] P.R. Montague, P. Dayan, T.J. Sejnowski, A framework for mesencephalic dopamine systems based on predictive hebbian learning, *Journal of Neuroscience*, 16 (1996) 1936-1947.

[41] W. Schultz, P. Dayan, P.R. Montague, A neural substrate of prediction and reward, *Science* 275 (1997) 1593-1599.

[42] R.S. Sutton,(1988) Learning to predict by the method of temporal differences, *Machine Learning* 3:9-44.

[43] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[44] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE transactions on Communications*, 1:84–95, 1980.

[45] S. Theodoridis and K. Koutroumbas *Pattern Recognition,* 2nd Edition, Academic Press, 2003.

[46] Vinje, William E. (Program in Neuroscience, University of California at Berkeley); Gallant, Jack L. *Sparse Coding and Decorrelation in Primary Visual Cortex during Natural Vision* Source: *Science*, v 287, n 5456, Feb 18, 2000, p 1273-1276

[47] Shepard, R.N. (1964). Attention and the metric structure of the stimulus space. Journal of Mathematical Psychology, 1:54-87

[48] Juyang Weng; Wey-Shinan Hwang, From neural networks to the brain: autonomous mental development Computational Intelligence Magazine, IEEE, Vol.1, Iss.3, Aug 2006 pp. 15-31

[49] J. Weng, "Developmental robots: Theory and experiments," *International Journal of Humanoid Robotics*, vol. 1, pp. 199–236, 2004.

[50] D.S. Touretzky and L.M. Saksida (1996) Skinnerbots. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson (eds.), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 285-294. Cambridge, MA: MIT Press.

[51] D.S. Touretzky and L.M. Saksida (1997) Operant Conditioning in Skinnerbots *Adaptive Behavior* 5(3/4):219-247.

[52] L.M. Saksida, S.M. Raymond, and D.S. Touretzky (1998) Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems*, 22(3/4):231-249.

[53] J.L. Krichmar and G.M. Edelman (2006). Principles Underlying the Construction of Brain-Based Devices, In Adaptation in Artificial and Biological Systems, T. Kovacs, and J. A. R. Marshall, eds. (Bristol UK: Society for the Study of Artificial Intelligence and the Simulation of Behaviour), pp. 37-42

[54] J.L. Krichmar, D. A. Nitz, J.A. Gally, and G. M. Edelman (2005). Characterizing functional hippocampal pathways in a brain-based device as it solves a spatial memory task. Proc Natl Acad Sci USA, 102: 2111-2116.

[55] M.A. Bursch, M. Skubic, J.M. Keller, and K.E. Stone. A Robot in a Water Maze: Learning a Spatial Memory Task. to be published in IEEE International Conference on Robotics and Automation, 10-14 April, 2007, Rome, Italy.

[56] Kohonen, T. 1990. The self-organizing map. Proc. IEEE 78, 1464-1480.

[57] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, E. Thelen, "Computational Autonomous Mental Development: A White Paper for Suggesting a New Initiative", Technical Report, 2001.

[58] BF Skinner, Science and Human Behavior., Colliler-Macmillian, New York, 1953

[59]J. Weng and W. Hwang, "Incremental Hierarchical Discriminant Regression," IEEE Transactions on Neural Networks, vol. 18, no. 2, pp. 397-415, 2007.

[60] “Anatomy of eye”, 2001, http://www.mydr.com.au/content/images/categories/Anatom/anatomy_of_eye.gif, Website accessed 08/12/2007.

[61] “Internal Anatomy of the Eye”, 2004, http://starklab.slu.edu/PhysioLab/RETINA.jpg, Website accessed 08/12/2007.

[62] “Night Vision, The Red Myth”, http://stlplaces.com/night_vision_red_myth/, Website accessed 08/12/2007.

[63] “Nvidia Cuda Compute Unified Device Architecture”, www.nvdeveloper.nvidia.com, Website accessed 09/12/2007