Learning of Information Gathering in Modular Intelligent Systems

Cem Karaoguz

2012

A dissertation submitted to the Faculty of Technology at Bielefeld University for the degree of Doktor-Ingenieurwissenschaften (Dr.-Ing.) on September, 2012.

Defended and accepted on March 14, 2013.

LEARNING OF INFORMATION GATHERING IN MODULAR INTELLIGENT SYSTEMS

Cem Karaoguz

Dissertation submitted to the
Faculty of Technology
Bielefeld University
in partial fulfillment of the requirements for the degree of

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

September, 2012

Acknowledgements

George Martin says if the bricks aren't well made, the wall falls down. I believe I built up a strong wall thanks to the people helped me to make my bricks. First and foremost, I would like to thank Dr. Tobias Rodemann from Honda Research Institute Europe who supervised me and supported me all the time even before my PhD period. Without you, this thesis would not have been as deep as it is now, you helped me to put strongest bricks. I would like to thank Prof. Britta Wrede as well, my supervisor at Bielefeld University. Even though we have been apart geographically, I always felt your support and made good use of your feedbacks.

Next, I would like to thank Dr. Christian Goerick from whom I learned a lot. Optimism and high spirit, if nothing else. I would like to thank Dr. Ursula Körner, my advisor during the PhD period. I enjoyed our discussions very much. Also, I would like to thank Prof. Edgar Körner, without whom this thesis may not have been initiated.

I am greatly indebted to Dr. Andrew Dankers who collaborated with me, patiently reviewed my papers and made me see things differently. Thank you for everything. I am also indebted to Mark Dunn, my first supervisor at the Honda Research Institute Europe. Thanks to you, I am not doing so many segmentation faults any more.

A big thanks goes to the amor do meu coração, Daniela Pamplona who shared the life with me, a very difficult thing indeed. Muitos beijinhos! I also would like to thank my parents and my brother who never stop sending their love and guidance from thousands of kilometers away.

My colleagues and friends, Dr. Thomas Weisswange, Rujiao Yan, Matthias Platho, Michael Garcia Ortiz, Sarah Bonnin, Tobias Kühnl, colleagues at the HRI-EU, and many other friends in different countries world-wide: thank you for your help, friendship and distraction from the thesis I needed to complete it.

Publications

Elements of this thesis appeared in the following peer-reviewed publications:

- C. Karaoguz, T. Rodemann, B. Wrede: *Optimisation of Gaze Movement for Multitasking Using Rewards*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011, S. 1187–1193
- C. Karaoguz, T. Rodemann, B. Wrede: *A Reward Based Visual Attention Framework for Humanoid Robots* (Abstract only), Proceedings of the European Conference on Eye Movements in Journal of Eye Movement Research, 2011, Vol. 4, Issue 3, S. 259
- C. Karaoguz, A. Dankers, T. Rodemann, M. Dunn: *An analysis of depth estimation within interaction range*, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010, S. 3207–3212

Elements of this thesis will appear in the following peer-reviewed publication:

- C. Karaoguz, T. Rodemann, B. Wrede, C. Goerick: *Learning information acquisition for multi-tasking scenarios in dynamic environments*, IEEE Transactions on Autonomous Mental Development, accepted

Abstract

Research of intelligent systems aims to realize autonomous agents capable of performing various functions to ease every day life of humans. Usually, such occupations can be formalized as a collection of tasks that have to be executed in parallel or in a sequence. Since real world environments are highly dynamic and unpredictable, intelligent systems require cognitive capabilities that can learn how to execute such tasks through interactions. Considering that the system has limited resources for acquiring and processing information, a strategy is required to find and update task-relevant information sources efficiently in time.

This thesis proposes a system level approach for the information gathering process and an implementation that puts this idea into work. The presented framework takes a modular systems approach where modules are defined as elementary processing units for information acquisition and processing. The modular system design helps handling scenario complexity. A module management mechanism learns which modules deliver task relevant information and how the constrained system resources are distributed among these in a reward based framework. This reduces the partial observability caused by the information gathering process and provides better support to other high level cognitive functionalities of the system. Such an adaptive approach also makes it possible to deal with variations in the scenario or environment.

Two different applications in simulation are implemented to test these hypotheses and demonstrate the utility of the proposed framework: the first implements a 'reaching-while-interacting' scenario for a humanoid robot and the second employing an autonomous navigation scenario for a mobile robot. Both scenarios involve dynamic objects, rendering a challenging environment close to the real-world conditions for the system. Results from experiments with these applications provide evidence for hypotheses postulated in the thesis.

Contents

List of Figures

CHAPTER *1*

Introduction

Although it may seem that the concept of intelligent systems was introduced into literature with the boost of Artificial Intelligence in the 1960s, the idea of thinking machines goes back to ancient Greek myths and found itself a place in literature inherited from many civilizations as animated images to be worshiped or humanoid automata (McCorduck [2004]). A modern definition of intelligent systems is done by Russell and Norvig [2002]: intelligent systems are autonomous entities that observe its environment through its sensors, act upon it with its actuators and direct its activity towards achieving goals. This definition is also valid for the notion of intelligent systems subject to this thesis, however it is incomplete. A goal-oriented autonomous system can execute its tasks using designed solutions as long as the state of the world is exposed to the system all the time. However, this is hardly the case due to the fact that the system has limited resources for acquiring and processing information. Moreover, real world environments are so dynamic and unpredictable that a system performing an occupation repeatedly never experiences the same situation even though the task routines are the same. Developmental approaches postulate that a system can accommodate local learning mechanisms that can unfold capabilities to achieve a set of tasks through interactions with the environment. Hence, the definition of intelligent systems in the focus of this dissertation also incorporates learning and development skills such that the system can accumulate knowledge from interactions with its environment and utilize this knowledge to execute its tasks.

At a coarse level of formalization, the basic principle for an intelligent system to accomplish a task comprises acquiring information from the environment, processing the acquired information and acting on the environment. However, the amount of information that can potentially be obtained is vast. Consider the environment where the image in Figure 1.1 was taken. There are various elements in the scene such as musicians

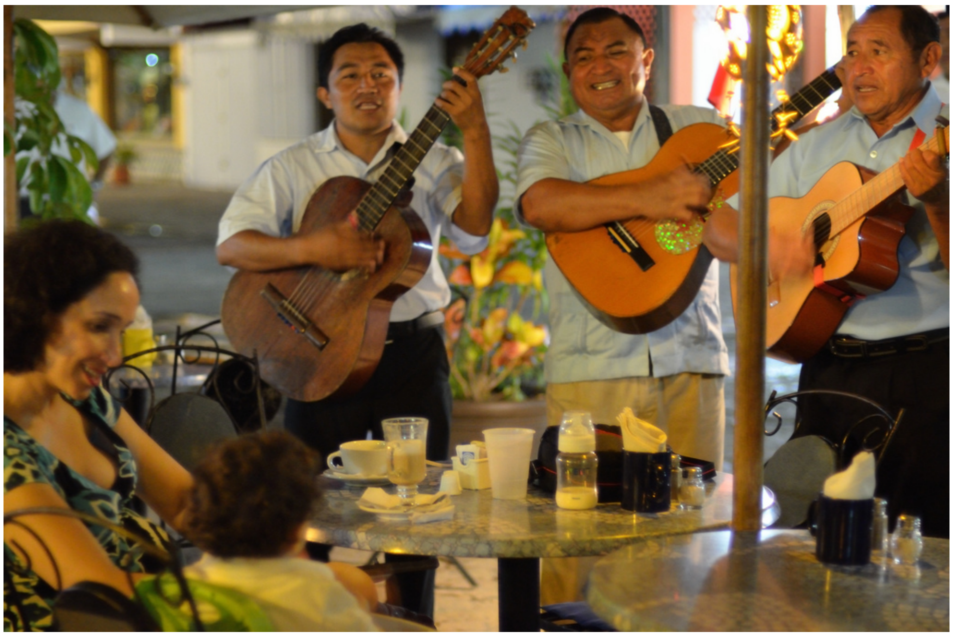


Figure 1.1: A scene with various information sources. The image is taken from the Flickr stock repository (flickr.com/groups/stock).

playing the guitars, the customer and her baby, glasses and other objects on the table. Additionally, it is very likely that there are more elements in the environment that are not captured by the image. A system situated in such an environment has to deal with a great amount of information emerging from these scene elements in order to make inferences about the environment. However, physical, computational and memory constraints in the system make acquiring and processing information about all entities in a scene at the same time impractical and sometimes impossible. Physical constraints are caused by design properties of the system. Sensors that cover a limited area (e.g. cameras with limited field of view) or have variable accuracy within their coverage must be directed towards different regions in the scene in order to collect information from the environment. Computational and memory constraints arise when a system is exposed to more information than its computer and memory could handle. A typical consequence of a shortage in computational power is latencies in data processing that may impair real time interaction of the system with its environment. Lack of memory limits the complexity of the implemented algorithms. Increasing the on-board computational power and memory is limited by the space available in the system as well as considerations of cost and energy consumption. This limitation becomes particularly apparent for small scale robotic systems. Off-board computing may be undesirable due to restrictions like necessity of cable-free communication and limited bandwidth and coverage of wireless connections.

For an intelligent developing system to be able to deal with complex scenes and handle tasks in real time, it is necessary to employ better strategies than dealing with all available environmental information due to the constraints on the information acquisition process. One way biological systems use to solve this problem is attention mechanisms that can refine environmental information in the way that only the information useful to the system is acquired. Implementation of such an attention mechanism to an artificial intelligent system is not straightforward and the following issues have to be considered:

- *Organization*: structuring the process of information acquisition,
- *Value assessment*: finding relevant information sources,
- *Arbitration*: choosing when to gather which piece of information,
- *Partial observability*: minimizing the uncertainties caused by the information gathering process,
- *Robustness*: being able to adapt to different environments and changes.

These are the principle issues that are addressed by this thesis. The following paragraphs in this section aim to build up insights about these problems.

Organization gives a formalized structure to the system. This defines how the information is represented throughout the system describing the interactions between and within the information acquisition and information processing elements (such as local learning mechanisms) of the system. Consequently, organization of a system affects how well the information acquisition process supports learning and executing tasks achieved by various information processing mechanisms in the system.

Value assessment can coarsely be defined as evaluating how useful a piece of information is. A fundamental question lies within this definition: what is useful information? Inspirations from neurophysiological properties of the visual cortex led to development of saliency based methodologies in attention mechanisms where the usefulness of information is defined by the conspicuity of the scene element that emits the information. Conspicuity (or saliency) of a scene element is determined by the contrasts between the feature of an element (e.g. color, shape, texture in vision domain; sound intensity in audio domain) and its neighborhood. In essence, saliency based methods drive the attention based on stimuli and therefore, they are identified as bottom-up attention. Such methods may be useful for simple tasks but it is often argued that a stimulus driven approach alone will be insufficient to support tasks with higher complexity (see Land and Tatler [2009] for a general discussion). One of the reasons is that bottom-up attention does not distinguish between the value of information (i.e. stimuli) from different entities. For example, in the scene shown in Figure 1.1 there are various entities that a bottom-up attention framework would reveal to be important: customer (from face features), musicians (from motion and auditory cues), coffee cup and glasses on the table (from object features such as shape, form, color). However, value of information from each entity depends on intentions or tasks of the system. If the task is to interact with the customer in the scene, the system has to observe her, if the task is to pick up an object from the table then the system has to direct its attention to the object it has to interact with.

Arbitration problem arises when the amount of task-relevant information sources is greater than the resources the system can deploy to acquire them or when the system has to execute several tasks sequentially, where sequential acquisition of information relevant to each task is necessary. A multi-task situation where the system has to execute more than one task in parallel the information from task-relevant entities can be at the same level of importance but can only be acquired one at a time is an example for the first case. A concatenated series of tasks such as grasping a bottle, opening the lid and pouring some water into one of the cups will lead to varying values of information in time. In both cases the choice of how the system employs its perceptual instruments in time is crucial and this is not addressed by a plain bottom-up attention framework.

Partial observability occurs when an attention mechanism is used for information acquisition. When an attention mechanism focuses the perceptual resources of the system on a particular element in the scene to gather information from it, information about every other element in the scene is not accessible until the time they are focused on. Partial observability may have crucial effects on various information processing mechanisms employed by the system especially in dynamic environments since such mechanisms run in parallel and rely on different environmental information. Consequently, the learning and real-time task execution capabilities of the system addressed by such local learning and information processing mechanisms are impaired.

Robustness of the attention mechanism defines how well it can adapt to changes in environmental conditions. For example, a change in illumination conditions may influence the color properties of entities in the environment and an object that is perceived as green before might look like yellow. Furthermore, dynamic entities might appear unexpectedly in the scene. Those entities might be of interest and worth to be attended to, or they might be just distractions. The attention mechanism should be robust enough to be able to cope with such situations.

1.1 Scope of the Thesis

This thesis aims at contributing to the intelligent systems research in two aspects. First, it suggests the idea that the information gathering process can be learned in large-scale complex developing systems. Second, it proposes a mechanism that realizes this idea. A system-level approach is presented to address the problems mentioned above regarding the information gathering process.

A modular system architecture is adopted for the *organization* of the system. Although there are various definitions in the literature for modularity, this dissertation describes it as structuring the system by defining functional elements and interactions between them. Hence, in the modular systems approach elementary functionalities of information gathering and processing (e.g. color tracking, motion detection, object recognition) are encapsulated in modules. Modularity facilitates the system design and operation by formalizing the information flow between functional elements of the system as well as guiding the implementation regarding the external manifestations of the system. Hence, complexities in scenario and system can better be coped with. The link between perceptual modules and information processing modules (e.g. local learning mechanisms, task execution policies) is provided through a scene representation mechanisms that contains and temporally stabilizes environmental information acquired via perceptual modules and keeps it available for information processing modules. The attention problem then, transforms into distributing the constrained system resources (e.g. sensor orienting actuators or computational resources) between module demands (e.g. sensor orienting commands) so that entities in the scene representation are updated in time. The proposed framework for achieving this is the Module Management Learning (MML) mechanism that learns which modules are relevant for parallel-running tasks and how to distribute the constrained system resources among these modules efficiently, addressing the problems of *value assessment* and *arbitration*. Learning is based on reward signals which are coarse indicators of the outcome of interactions of the system with its environment. This is plausible with developmental frameworks since systems using such frameworks utilize internal motivation mechanisms evaluating the interaction experience of the system. A learning based framework can also reduce the *partial observability* caused by the information gathering process by finding efficient strategies to update task-relevant scene entities. Hence, local information processing mechanisms embedded in the system do not have to implement complex algorithms to cope with partial observability. A learning based approach also introduces flexibility to the system and allows it to cope with variations in the environment or scenario. In summary, the hypotheses this dissertation postulates are as follows:

- the modularity approach helps the system to scale with scenario complexity,
- given that a system is equipped with various perceptual modules, MML can find the ones that deliver task-relevant information so that the system can handle multi-tasking scenarios,
- MML can learn an efficient strategy for distributing constrained system resources among task-relevant modules,
- together with the scene representation mechanism, MML can support other local learning mechanisms in the system by reducing the partial observability introduced by the information gathering process,
- the framework increases the capabilities of the system to cope with scenario variations and changes in the environment.

The framework is demonstrated in two different applications with specific instances of a hierarchically organized modular system. The first application considers the problem of learning a gazing strategy for the iCub humanoid platform in a simulation environment (see Beira et al. [2006], Tikhanoff et al. [2008] for more information). The scenario consists of two tasks that have to be executed in parallel: a reaching and an interaction task. For the reaching task the robot has to select, reach and touch one of the three objects that are moving on a table in front of the robot. For the interaction task, the robot has to make eye contact with a partner present in the scene. The system has to execute a set of policies to fulfill the tasks. The policies use the scene representation built by integrating the environmental information gathered via modules. The system learns an efficient strategy to update the scene representation in terms of arbitrating gaze commands from various modules to support the tasks. In the second application the framework is employed for an autonomous navigation scenario where the system has to collect target object without colliding with obstacles in a dynamic environment. The information about the environment is acquired by a number of modules specialized in different types of information (e.g. distance to the nearest obstacle, orientation with respect to the nearest obstacle, position of the target object). Since the system has constrained resources and cannot run all modules in parallel, the proposed framework is applied to learn an efficient use of modules to acquire relevant information. In both applications experiments with numerous different settings each of which defines a different environment and/or unique task constraints are conducted to test the postulated

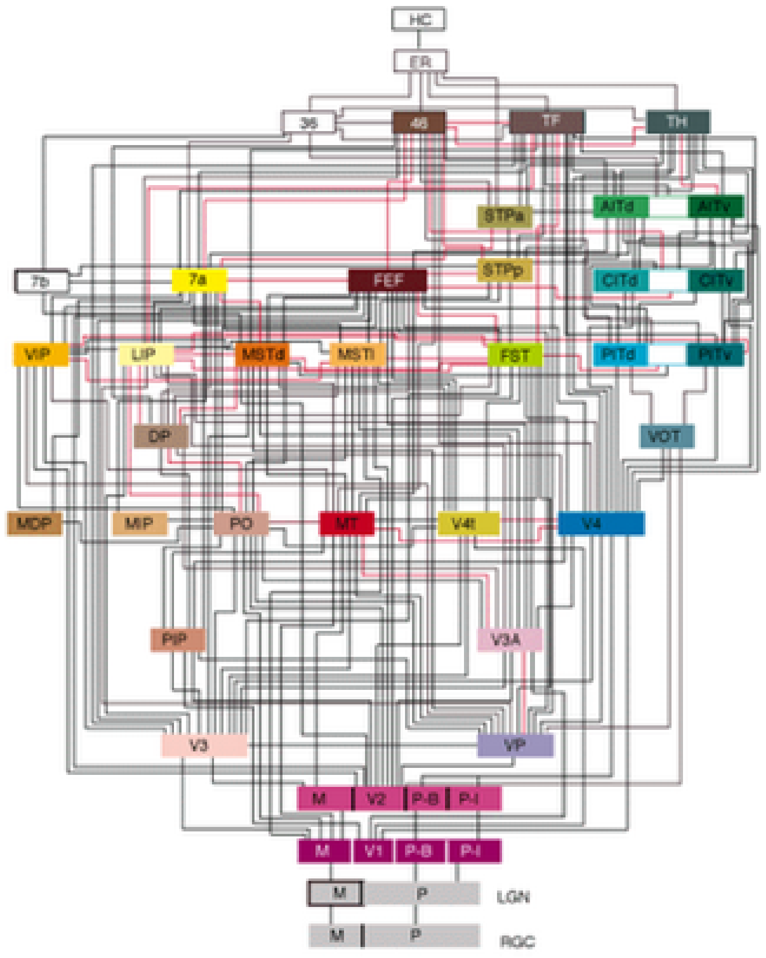


Figure 1.2: Hierarchy of visual areas in the primate brain as shown in Felleman and Essen [1991].

hypotheses and evidences supporting these hypotheses are presented. For comparison additional methods were implemented and tested under the same experimental setups. Results indicate that strategies learned by the proposed framework are better than the strategies that compared methods present.

1.2 Biological Background

Biological systems such as the human brain demonstrate very efficient and powerful information processing mechanisms. Thus, intelligent systems research has always been inspired by biology in principles and methodology. This thesis also follows various principles from biological systems, particularly the human brain. This section explains several related aspects of biological systems: modularity, attention, working memory and reward-based learning.

1.2.1 Modularity in Cognition

The anatomy of the human brain is modular. For example Felleman and Essen [1991] stated that the processing in the primate cerebral cortex is realized via hierarchical organization of structures analogous to modules. Figure 1.2 shows the hierarchy of visual areas in the primate brain that consists of 32 visual cortical areas connected by 187 links. Further separate processing areas can be found for the other sensory modalities such as audio, tactile and olfactory. Moreover, Moeller et al. [2008] showed that in the monkey brain different areas take part in a local functionality, such as face recognition. Another evidence for the modular anatomy of the brain is that lesions of local cortical areas lead to various deficiencies (Newsome and Park [1988]).

There are also ideas explaining the cognitive functionalities of the brain with modularity. Cognitive psychology accommodates numerous studies to explain the functional architecture of the mind leading or following various theories (see Fodor [1983] for a comprehensive survey). Chomsky [1980] postulated the idea that the mind is structured into innately specified organs (e.g. language). According to this theory cognitive skills (i.e. faculties) are discriminated by reference to their propositional content. A different line of study introduces the concept of horizontal faculties where cognitive processes show interactions between faculties such as memory, attention, and perception, which are not domain specific. For example, to store a representation of information, the same faculty of memory is used regardless of the modality of the representation. A third view on the architecture of mind is the idea of vertical faculties (Hollander [1920]). The vertical view differentiates mental faculties based on domain specificity and considers concepts like perception and memory as attributes common to the fundamental psychological qualities. Fodor [1983] revised these ideas and devised a new formal definition for modularity. Some[1] of the requirements this formalism demands are:

- Domain specificity
- Mandatory operation
- Informational encapsulation
- Shallow outputs
- Fixed neural architecture

As stated in the vertical view on faculties, the proposition of modules being domain specific implies that modules are specialized and operate on certain kinds of inputs. Fodor illustrates examples for such modules, in the case of vision, as mechanisms for color perception, for analysis of shape, for the analysis of three dimensional spatial relations, etc. The requirement of mandatory operation suggest that modules should process whenever an input is available. Fodor explains this with examples:

> *You can't help hearing an utterance of a sentence (in a language you know) as an utterance of a sentence and you can't help seeing a visual array as consisting of objects distributed in three dimensional space.* [Fodor, 1983, p. 52-53]

Due to this property, sensory signals (i.e. inputs to the modules) cannot be broken into pieces and processed selectively. Selection is done at module level (i.e. stop hearing a sentence at all) via attention mechanisms. This views the attention process from a modular perspective. Informational encapsulation implies that a module can operate

[1]While Fodor [1983] suggests nine requirements for a system to be modular, only the properties that are relevant to this work are explained here.

without any influences external to the module such as other modules, high level cognitive functionalities or even feed back connections from representations derived from the output of the module. Hence, perceptual modules should be free of top-down processes such as the expectations of the system about the environment to some extend since the purpose of perception is to inform the system about the state of the environment even when the environment is in a state that the system does not expect it to be. Following this, the problem of where perceptual processes interface with top-down processes is addressed by the feature that modules have shallow outputs. Fodor supports the idea that perceptual modules deliver the representations of basic categorizations of objects as output. A basic categorization resides at a level of abstraction where the category can be explained by the properties that modules can detect (e.g. in the case of vision shape, color, local motion, etc.) and maximizes the information per unit of perceptual integration. The final point, fixed neural architecture, refers to the findings that perceptual modules can be associated with neurological structures.

1.2.2 Attention and eye movements

As already stated, biological systems employ attention mechanisms to regulate the information acquisition process. Since vision is the most comprehensive and well understood modality and is mostly used for the applications explained in this work, the emphasis is mostly given to visual attention. Visual attention can be characterized by its status as overt and covert. Overt attention spans out of the sensory space to collect information by directing the eyes towards entities in the environment. Covert attention on the other hand, is the case where one of the entities in the field of sensory space is mentally focused without sensors being directed towards it. One reason why biological systems need to direct their sensory organs towards entities, as in the overt attention case, comes from their physiological properties: sensory organs may have a limited field of perception or variable accuracy across their coverage. For example in vision domain, images seen by the eyes are projected onto retinas and the visual cortex as space variant representations. The main reason of this is the non-uniform distribution of the photoreceptors in the retina. The density of photoreceptors is greater in the center, the region called fovea, and less in the periphery. This results in a high resolution of the foveal region and low resolution of the peripheral region. Keeping target stimuli on the foveae of both eyes, fixational eye movements achieve a high resolution view of visual targets so that the greatest possible amount of information about the target can be obtained. The gaze system has evolved into different subsystems to serve different functions (Goldberg [2000]). Collaboration of different conjugate and disconjugate eye movements achieves fixation on visual targets. These five different eye movements are saccades, smooth pursuit, vergence, Opto-Kinetic Reflex (OKR) and Vestibulo-Ocular Reflex (VOR). Saccades are rapid conjugate movements of eyes that shifts gaze direction between targets in the visual scene. Smooth pursuit is conjugate eye movements purpose of which is to keep a moving visual target on the foveae. Vergence is a disconjugate eye movement that moves the intersection point of the line of sight of the two eyes closer or further away in order to direct the fovea of both eyes at the same visual target. OKR and VOR are reflexive movements that compensate body and head movements so that the motion of the retinal image is minimized. Among these movements saccades are the type that has the strongest relation to visual attention since they are the utility for collecting new information by scanning the environment whereas the other types of eye movements are used to stabilize images on the fovea. Furthermore, saccades and visual attention share several anatomic structures for operation such as the Superior Colliculus and Frontal Eye Fields.

1.2.3 Stimulus-driven and task-driven attention

Attention is mainly driven by two distinct processes: a bottom-up process and a top-down process (Posner and Petersen [1990]). Bottom-up attention is a stimulus driven process where the attention is controlled by the features of the entities in the environment. If a feature of a scene element contrasts with its neighborhood, this feature is called 'salient feature'. Such features can be detected by computations done in low-level structures of the human brain (e.g. retina, superior colliculus) and attention can be directed on them. Saliency can be defined over various features of the information: a yellow lion in a green meadow is salient in terms of color, a loud noise in a quiet environment is salient in terms of sound amplitude, a car coming towards a stationary observer is salient in terms of motion etc. As it can be deduced from these examples, bottom-up attention mechanisms play a major role in survival of biological systems.

A second aspect that controls the attention is top-down influence. The theory of top-down attention states that high level cognitive functionalities influence the attention mechanism to assist goals and intentions of the system. Specific brain areas that are known to have a role in top-down visual attention are frontal eye fields (FEF), supplementary eye fields (SEF) and lateral intraparietal area (LIP) (Goldberg [2000]). Yarbus [1967] published one of the earliest works showing how top-down processing influences visual attention. In his experiments, he showed that observing the same picture, a subject exhibited very different eye movement patterns depending on different tasks that he had been given (e.g. estimating the ages of the people in the picture, memorizing the locations of the objects and people in the picture, etc.).

Today there is an extensive amount of work in the literature analyzing and interpreting the eye movements under various tasks. Land and Tatler [2009] showed that humans follow a consistent pattern for acquisition of visual information (i.e. gazing behavior) while executing domestic tasks. When participants were asked to prepare tea, they first fixated on the kettle, then on the tip of the kettle to open it, then the tap to fill the kettle with water and so on. A very small number of gaze fixations fell on scene elements irrelevant to the task. This is evidence that top-down attention is heavily active during executing tasks. Further observations are recorded for tasks like reading, typing, drawing, walking, driving and playing ball games. In all such experiments eye movement patterns were similar among participants (with a skill-dependent variation) and specific for each tasks. This finding leads to the hypothesis that task specific sequencing of gaze fixations are learned via interactions with the environment.

Foerster et al. [2012] investigated the idea that there may be a long term memory (LTM) based mode of attention selection during execution of automatized sequential high-speed sensorimotor tasks. In their experiments participants performed an automatized speed-stacking task in the light and in the dark. It is shown that participants made systematic eye movements in the dark. Saccadic scan paths and the number of fixations were highly similar across illumination conditions. Finally, neither eye-hand dynamics nor saccade accuracy correlated with hand movement durations in the dark.

It is also interesting to investigate how the human visual system deals with multi-task scenarios where multiple targets are required to be monitored simultaneously. Land and Tatler [2009] state that the human visual system resorts to time sharing in such situations. For instance in a driving scenario the driver has to monitor multiple targets (preceding cars, pedestrians, bicycle riders, etc.) at the same time. This is done by employing sequential gaze-shifts on those targets. This shows that the human visual system achieves an efficient temporal scheduling strategy for gaze-shifts.

1.2.4 Working memory

Baddeley [2003] explains the working memory (WM) concept as a highly dynamic kind of memory that performs in the range of seconds and temporarily stores selected information for detailed analysis. According to observations, many structures in the human brain contribute to the operation of the working memory. It has a central role in understanding highly dynamic and complex scenes and influences the attention mechanism considerably for information acquisition. Chun [2011] states that the visual working memory is visual attention maintained internally over time. When an object is attended by the system, information associated with that object enters into working memory. Contents of working memory are defined by connectivity, keeping representations in LTM associated with the attended object active. Chun [2011] defines working memory as the interface by which relevant perceptual information from the environment is selected and actively maintained by attentional mechanisms as internal representations. In his model, Knudsen [2007] gives working memory a central role in the process of attention. While working memory keeps properties of attended objects, a competitive selection process determines which information gains access to working memory. The strength of competition can be modulated by higher cognitive functions whereas bottom-up saliency filters enhance the response to some stimuli at feature level.

1.2.5 Reward-based learning and attention

The reward system in the brain is a collection of structures that control behavior of the organism by generating pleasurable effects (Schultz [1997]). A reward can be defined as a signal that reinforces recent correct behavior: under certain circumstances (e.g. presence of a stimulus) if an organism acts in a specific way and gets rewarded, the probability of acting in the same way with the next occurrence of the same condition is increased. Thus, the reward system plays an important role in behavior organization and survival of the organism. The oculomotor system of the human and primate brain also uses reward signals to learn making decisions on where to look. Yarbus [1967] postulated the idea that eye movements are driven by a priority or saliency map formed by the brain continuously assigning a value to every part of the visual environment. In relation to this, Gottlieb and Balan [2010] review various findings confirming the existence of attentional decisions that assign value to sources of information. It is shown that activity in LIP is correlated with salient or task-relevant stimuli regardless of whether they are being targeted by saccades or not. Since the activity that is not targeted by saccades can be separated from the metrics, modality, probability and even reward of an action, they propose that such activity encodes the value of information sources. Shadmehr [2010] showed that the programming of motor commands for saccades can be done by minimizing effort while maximizing the value assigned to stimuli as such a model can

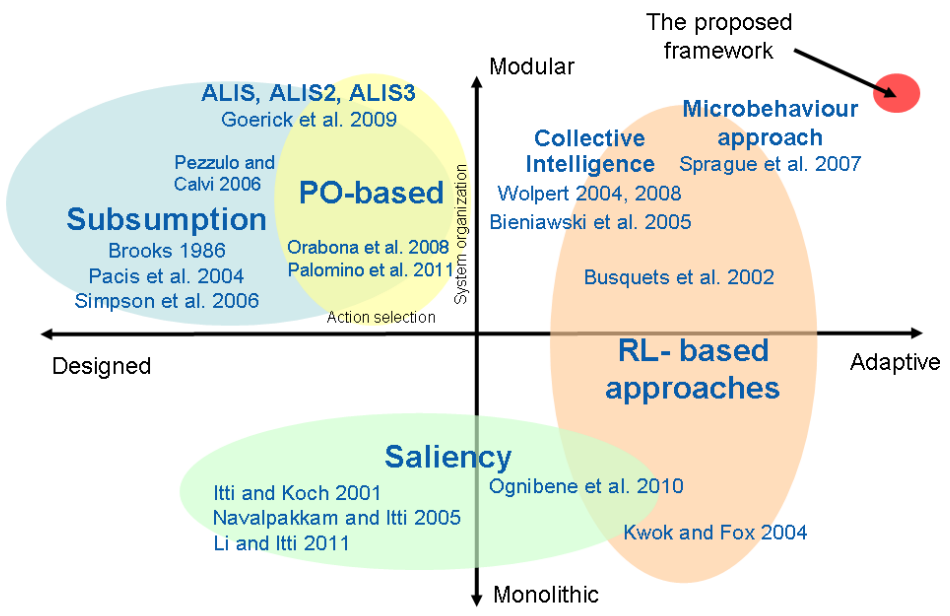


Figure 1.3: Functional and structural taxonomy of related work.

reproduce the relation between saccade metrics such as movement duration, velocity and amplitude. In a parallel study, Trommershäuser et al. [2009] present a framework describing how eye movements can be programmed based on representations of relative expected subjective values and how such representations are updated based on reward prediction errors received as dopaminergic feedback. The authors summarize evidence from various experiments supporting their framework. All of these studies and others which the authors point to indicate that the reward mechanism plays an important role in the information acquisition process of the human and primate visual system.

1.3 Related Work

The problem this thesis addresses is defined in four principle issues. These issues can be grouped into two main domains to facilitate the evaluation of the proposed work and compare it to the related work. These domains are defined as system organization and action selection. Figure 1.3 shows a mapping of the related work in these two domains. The system organization domain inspects the fundamental principles followed in the design of the system in relation to modularity and shown in the vertical axis in the figure. The scale between modular and monolithic along this axis quantitatively illustrates the level of granularity of the functional elements of the system and the potential of controlling these elements for a system to generate diverse behavior. The action selection domain mapped to the horizontal axis illustrates degree of learning involved in the selection of information acquisition actions. This covers the issues of value assessment, arbitration and partial observability. Main concepts that influenced these works (such as RL theory, saliency, subsumption) and their span over the two dimensions are

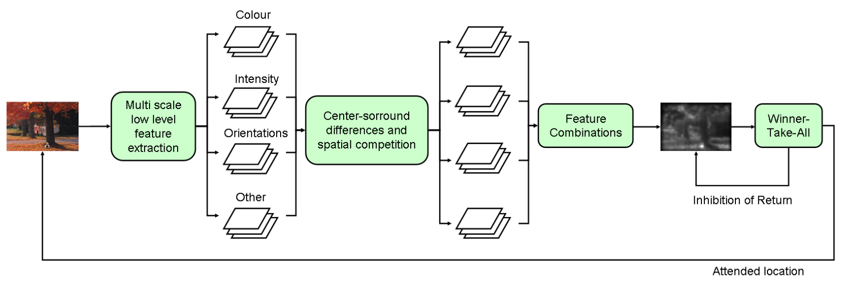


Figure 1.4: Saliency based attention model adapted from Itti and Koch [2001].

also highlighted in the figure. This section gives an overview on related work including solutions they propose, their strengths and shortcomings. Further references (including to some of the related works explained here) related to the specific aspects of the proposed framework will be explained in more detail throughout the dissertation.

A well-established computational model for the saliency based visual attention is proposed by Itti and Koch [2001]. A description of this model is given in Figure 1.4. According to this model, the input image is first decomposed through several feature detection mechanisms (such as color, intensity, orientations). The spatial contrast in each feature channel is encoded by neurons in the resulting feature maps. Feature maps also have local neighborhood relations that enable the spatial competition of the features for salience. The feature maps are then combined into a final saliency map where overall saliency is encoded topographically. The attention shift is done towards the most salient spatial position at any given time. An inhibition of return mechanism suppresses the last attended locations in the saliency map and prevents the attention mechanism from being stuck on a single salient position. This model was used in many applications dealing with simple tasks. For example, Li and Itti [2011] showed that very simple visual search tasks can be achieved by saliency based attention. However, task-based information gathering is not considered in this model. There are various works extending this scheme with top-down information to model the task influence. A comprehensive one is presented in Navalpakkam and Itti [2005] where probabilistic models of object representations are learned from features and these models are used to define weights to bias feature maps so that the target object pops out. The application of this model is limited to simple object search tasks where target objects are already known. However, in developing systems where tasks are learned, task relevant objects should also be learned. The framework proposed in Ognibene et al. [2010] goes more in this direction. This framework presents a model that can learn top-down attention skills incorporated to a bottom-up saliency mechanism. Task-specific knowledge on potential gaze targets was acquired using RL. The bottom-up saliency mechanism served as an exploration map to extract such knowledge. However, the learning is based on spatial relations of objects. This may cause problems with dynamic scenes.

While, saliency based frameworks model the attention process in spatial domain, a new line of research introduces object-based attention. Rather than using the image features independently for the saliency computation, the object-based approach uses these features to group parts of the visual field which are likely to correspond to parts of the same object in the real-world. These entities are referred to as ‘proto-objects’. This concept is close to the modular approach and also compatible with our framework as the information collected from scene entities in the scene representation mechanism can be represented as proto-objects. Alternative examples of proto-object based visual attention models are presented in Orabona et al. [2007], Palomino et al. [2011] and several successful practices of proto-objects in large-scale complex intelligent systems are given in Bolder et al. [2008], Goerick et al. [2007, 2009]. The proto-object based attention models implement an inhibition of return mechanism as in the saliency model. However, the inhibition is done in the object domain rather than spatial domain in this case. This requires an additional mechanism to track proto-objects. In the proposed framework this requirement is eliminated since the system learns temporal sequences of gaze-shift between entities. Moreover, the task influence is realized in the same way as the saliency-model susceptible to the limitations explained earlier and it is not learned unlike in the proposed framework.

An early formalization for modular systems was introduced by Brooks [1986] where a layered control architecture for mobile robots was explained. This approach, called subsumption, builds layers of control systems in a hierarchical manner at increasing levels of complexity. Layers are composed of modules each of which performs a certain computational functionality. Modules that belong to higher-level layers can affect lower level ones by suppressing their outputs. The idea was later used in different applications with various extensions (Brooks [1991], Pacis et al. [2004], Simpson et al. [2006]). Although the subsumption approach facilitated engineering of formalized and successful control architectures for autonomous systems, it poses some limitations (Nakashima and Noda [1998]). The arbitration problem was addressed by prioritisation: modules in higher layers have priority over the ones in lower layers for the access of motor resources. This may cause problems in changing environments. Priorities defined for one situation may not be valid for others. Moreover, as the complexity of the system grows it may be difficult to find optimum assignments of priorities.

Limitations of pre-programmed strategies motivated researchers to investigate adaptive solutions. Reinforcement learning (RL) is a well established approach in this domain that enables systems to learn from coarse feedbacks (i.e. reward signals) indicating the outcome of their actions (Sutton and Barto [1998]). Kwok and Fox [2004] applied RL directly to the active sensing problem in a soccer playing robot without employing a modular architecture. The state space is composed of a representation of objects relevant to the task (i.e. scoring goal), odometry and an explicit representation of uncertainties over information the robot has. The system learns which object to look at depending on the current state. Casting the active sensing problem directly into a RL framework may have some shortcomings. Different kinds of information may be necessary if the approach is desired to be implemented for additional tasks. Simply adding new representations in the model may cause the state space to grow beyond computational limits. This may only be avoided by re-designing the state space.

For modular systems, Wolpert [2004] developed an approach for management of adaptive modules using game theory. In this framework the system acts on the environment through its modules. Each module of the system is an independent RL agent that seeks to maximize its own utility function (i.e. performs a pre-defined task). A global utility function defines a general task (i.e. a goal state) for the whole system and is used to optimize the local utility functions to drive the whole system to the goal state. Several implementations of this approach were presented in Bieniawski et al. [2005], Wolpert and Kulkarni [2008]. This approach assumes that modules are independent from each other. Conversely in the proposed setting, decisions may affect several modules at the same time (e.g. executing a gaze-shift command from one module at a certain time step affects others, which requested gaze-shifts to different areas in the visual field at the same time step).

Sprague et al. [2007] presented a RL based approach for controlling behaviors incorporating visual information. In this framework, small visual routines were handled as complete sensory-motor processes, called microbehaviours. From a functional point of view, microbehaviours can be regarded as modules. A microbehaviour is a RL agent: it has a pre-defined task, it gathers visual information, which predicts the current state of the environment relevant to the task and it takes actions based on this information to fulfill the task and receive rewards. A policy that maximizes the acquired reward is learned during the course of such interactions. The problem of which microbehaviour gets access to the camera control is solved by reducing the cost of uncertainty in the internal scene representation over the microbehaviours. While the model proposed by Sprague et al. [2007] is robust and flexible, the definition of which microbehaviours should be used in which scenarios is still done in a pre-programmed fashion.

Busquets et al. [2002] designed a cooperative multi agent system (MAS) for a robot navigation task. The interaction between different modules yields a complex trade-off mechanism where the resource allocation for different modules has to be managed. RL was used to optimize such a complex trade-off. It was shown that learned strategies performed better than pre-programmed ones. The model uses a mixed repertoire of looking and moving actions. However these can be separated since looking actions do not have a direct effect on the environment, they only change the internal state of the system. Such a separation may result in simpler representations of state and action spaces and a more efficient learning strategy. Hence, the authors reported that an effective strategy is only learned with a careful design of state space.

Pezzulo and Calvi [2006] used the schema concept of Arbib [1992] to construct a behavior coordination architecture based on the model of the praying mantis. The architecture consists of four main parts: actuators realize a given motor command (e.g. looking or moving), perceptual and motor routines define sets of motor commands for primitive actions (e.g. detect color, turn left, etc.), a behavior repertoire defines a number of pre-programmed schemata (e.g. detect prey, chase) and inner states (e.g. hunger, fear, etc.) drive the behaviors in the repertoire. The model selects the appropriate schemata for a given context (i.e. state of the world and its drives) based on the activity levels of schemata. The parameters governing these activity levels are pre-programmed. Also, the coupling between high level behaviors and low level visual routines are fixed. For example a 'detect prey' schema assumes a prey is red and moving. Such assumptions may be relaxed by learning the relations between the visual routines and behaviors (i.e. which visual routines are useful for which task).

Compared to Brooks [1986], Pezzulo and Calvi [2006], Sprague et al. [2007] the proposed framework learns which modules are important for the tasks undertaken. This makes the system more robust against changing environments and may facilitate system design. In contrast to Busquets et al. [2002], Kwok and Fox [2004] in the proposed framework action selection mechanisms for information acquisition (e.g. gazing) and for world manipulations (e.g. reaching, navigation, etc.) are separated. In this way, the information acquisition process can be optimized for parallel running task policies employed by the system. New tasks and policies can be introduced to the system easily without concern about the information acquisition process. As opposed to Ognibene et al. [2010] the proposed method is not restricted to a spatial representation of information sources. Therefore, it may better cope with dynamic scene elements.

Several examples of modular large-scale interacting systems were previously presented in Andreopoulos et al. [2011], Bolder et al. [2008], Goerick et al. [2007, 2009], Jacobsson et al. [2008], Vernon et al. [2007]. The information acquisition processes in these systems follow a manually designed procedure like prioritisation, subsumption or state machines. A closer inspection of some of these systems is given in Section 2.1.

1.4 Overview

Chapter 2 can be decomposed into two main parts. In the first part the modular systems approach, how intelligent developing systems can benefit from a formal architectural definition and modularity both in design and application and several common formal design languages for large-scale modular systems are explored. The second part explains the architectural elements of the proposed system-level solution to the problems addressed by this thesis. This includes the functional description of the systems implemented to test the hypotheses postulated in the dissertation. Additionally, a reference to the experiments exposing the practical benefits of the proposed modular system architecture in handling system complexity is given. Results from these experiments are discussed.

Chapter 3 also consists of two basic parts: in the first part, relevant practices on learning in large-scale systems are explained. Since the thesis focuses on developing systems, the necessity of an internal motivation mechanism is briefly discussed. This is connected to the reinforcement learning theory since it uses reward signals, which are natural products of such an internal motivation mechanism, to learn from the interactions of a system with its environment. Because the large-scale developing systems may require multiple tasks to be learned and executed in parallel, reinforcement learning frameworks that can deal with multi-task scenarios are also explained. Further on, the problem of partial observability is explained. The second part of the chapter introduces the proposed module management learning framework. This is a particular utilization of the idea proposed in this thesis where how to distribute the constrained system resources is learned to achieve an efficient information acquisition strategy for a robust and proper operation of the system. It is discussed how partial observability caused by the information acquisition process is addressed by the proposed framework. Further on, the mechanism that performs the arbitration between modules in time to decide which module should be given access to the constrained system resources is introduced. After that, it is explained how the parameters of this mechanism are learned via rewards. Finally, empirical observations from an abstract simulation of the module management learning framework are given.

Chapter 4 presents the first application of the proposed framework. This involves learning efficient gazing strategies of the humanoid iCub for a multi-task scenario where the robot has to reach one of the three objects moving on a table while keeping interaction with a partner. The chapter starts with discussing the motivations for the active vision approach which prescribes gaze movements for gathering information from the environment. Results from a series of experiments showing the benefits of the frameworks with active vision in depth estimation are presented. More details about the results can be found in Karaoguz et al. [2010, 2011b]. Further on, the system instance implemented for the reaching-while-interacting scenario is described. Finally the experiments conducted with this system are explained and results from those experiments are discussed. Preliminary results from these experiments are published in Karaoguz et al. [2011a]. A more detailed explanation of the work can be found in Karaoguz et al. [2013b].

Chapter 5 explains the second application where the computational resource allocation problem in a mobile agent for a multi-task autonomous navigation scenario is addressed by the proposed framework. The functional description of the system instance implemented for this scenario is given. Further on, experiments conducted with this system are described. This is followed by the presentation and discussion of the results from these experiments. Details about this application can be found in Karaoguz et al. [2013a].

Chapter 6 concludes the dissertation. It re-states the problem this thesis is focused on, postulated hypotheses and how these hypotheses are addressed. It summarizes the results from the experiments supporting the postulated hypotheses. Finally, an overview about open issues for further research is given.

CHAPTER 2

Modular Systems Approach

Although it is difficult to define modularity in cognitive systems (see Fodor [1983] for a discussion), a formal basis can be established for artificial systems to define modularity. In this work, the modularity is described as structuring the system by defining functional elements and interactions between them. Thus, a modular systems approach organizes the system design process with a set of propositions that defines the encapsulation of basic functionalities in the system in elemental computational units at various levels of granularity and communication between these. In the first part of this chapter an analysis of various modular systems approaches proposed in the literature are explained. The analysis considers the formalism these approaches propose, strengths and weaknesses they introduce to the system engineering process and their relation to the theory of cognitive modularity (explained in Section 1.2.1). The second part introduces the architectural elements of the proposed framework that structures the information gathering and processing functionalities in large scale systems. The modular system approach adopted for the engineering of the systems implemented in this thesis is explained both at formal and functional levels. Following this, results from a series of experiments demonstrating how a modular system approach can be beneficial in handling complexity are discussed.

2.1 Modularity in Artificial Systems

There are various studies to formalize modularity for intelligent systems research and engineering in the literature. Dittes and Goerick [2011] present a survey and comparison of some well established and widely used approaches from the perspective of a set of criteria to evaluate how a formalism facilitates intelligent systems research. This section gives a summary on these approaches and presents some representative

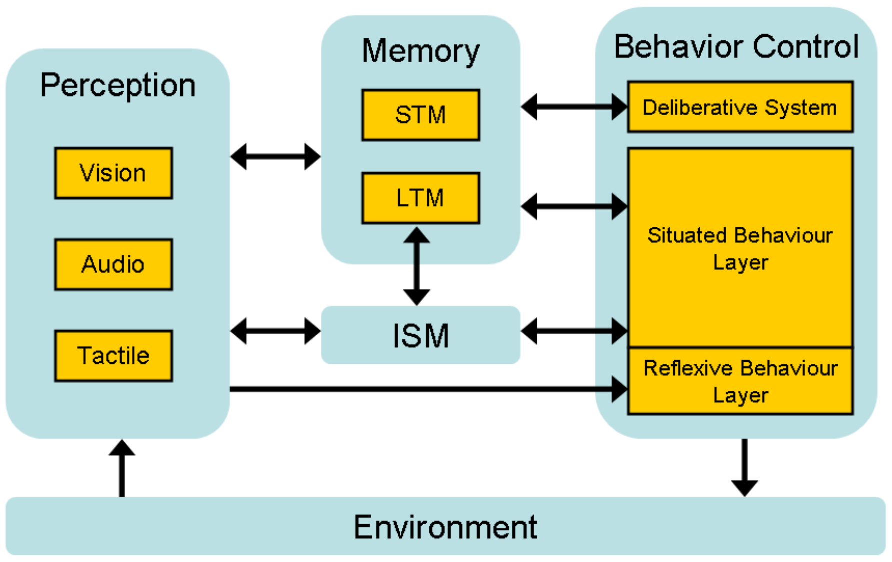


Figure 2.1: An example for the ‘Boxes and Arrows’ approach from Chernova and Arkin [2007].

examples where these frameworks are applied. Further on, the framework used as the basis of systems where hypotheses of this work are tested and its relation to the other approaches and formal definitions of modularity in cognitive systems are explained.

Although efforts on formal definition of modularity for intelligent systems go back to the subsumption framework proposed by Brooks [1986], modular design and implementation has been done for numerous intelligent systems since years before subsumption was introduced. The simplest applications of modularity is done by using flowcharts. Dittes and Goerick [2011] refer to such systems as ‘Boxes and Arrows’ and the same definition is used in this work as well. This is an ad-hoc approach to describe a system with boxes representing specific functionalities (i.e. modules) and arrows representing information flow between them and without addressing interfaces and dependencies between modules. This loose formalism bestows the advantage that it does not limit the range of expression and provides the designer with a great flexibility. However, the lack of formal definitions makes it difficult to compare different system hypotheses constructed with this framework. An example system using ‘Boxes and Arrows’ approach is presented in Chernova and Arkin [2007] and shown in Figure 2.1.

Subsumption architecture proposed by Brooks [1986] achieves the functional decomposition of a system using layers organized in a hierarchical manner (Figure 2.2). Each layer poses a control mechanism for the system and may encapsulate a number of modules performing basic computations. The hierarchy of layers is organized with respect to the complexity of the tasks each layer executes: low level layers perform low-latency stimulus-response type tasks whereas higher level layers conduct high-latency tasks like planning. Higher-level layers can subsume the roles of lower levels by suppressing their outputs. In Brooks [1986], the application of the subsumption framework is done in eight layers that are:

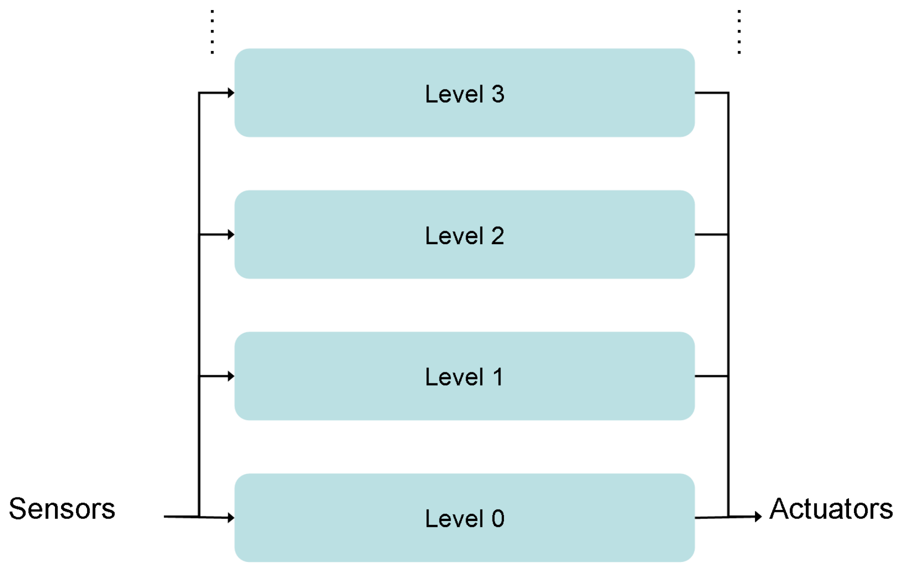


Figure 2.2: Subsumption architecture as proposed by Brooks [1986].

- Reasoning and plan modification
- Planning
- Object identification
- Detection of changes in the environment
- Mapping
- Exploration
- Wandering
- Collision avoidance

from top to bottom. Such vertical decomposition of competencies goes in line with the domain specificity principle of modularity explained in Section 1.2.1. Hence, subsumption facilitated intelligent systems engineering by slicing the problem of behavior organization on the basis of expected realization of the system. Using the hierarchical layer structure intelligent systems can be built incrementally with increasing level of complexity in the behavior. This made subsumption a widely used formal structure for robotic applications. However, the spread of the framework in the scientific community also revealed several shortcomings. One of these shortcomings is the conceptualization of information exchange between layers. Subsumtion does not formally specify how the processing in one layer is accessible to the higher layers: are the modules in a layer informationally encapsulated between layers, or can modules in higher layers access any kind of information in lower layers? Additionally, the information exchange in the other direction (i.e. top-down modulation of low level layers) is restricted in the formalism: higher layers can only act on the output of lower level layers and not individual

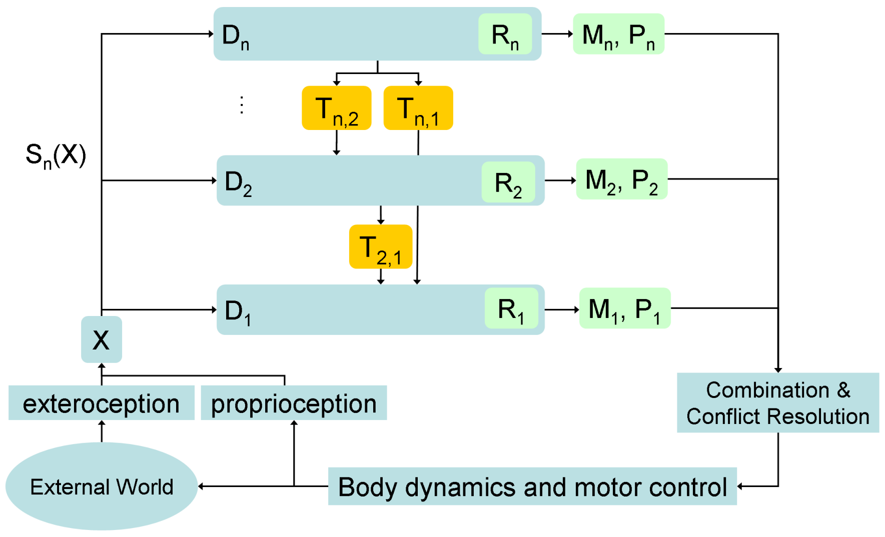
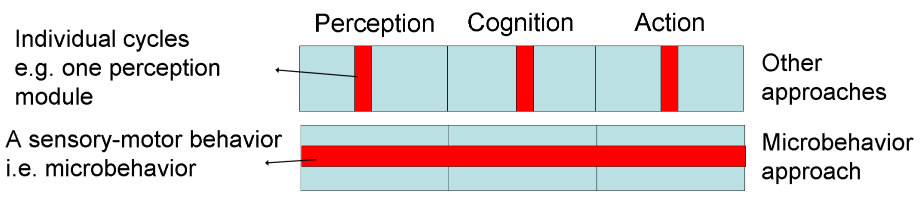


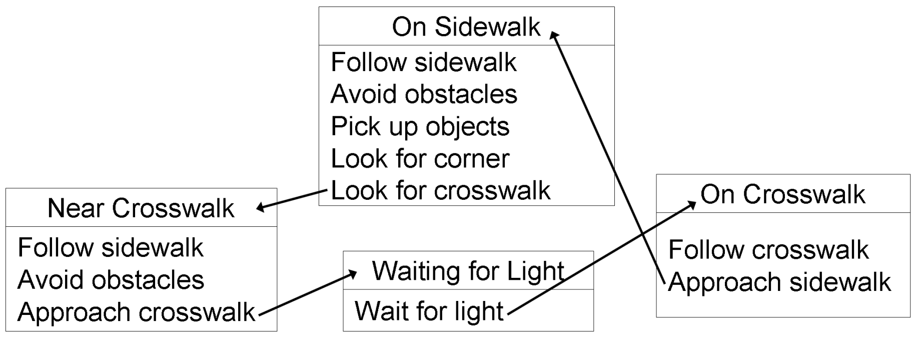
Figure 2.3: Systematica architecture as proposed by Goerick et al. [2007].

modules inside the layer. This may pose a limitation in the range of expression in the design of the system.

The problem of inter-layer information exchange in subsumption is addressed by the Systematica framework proposed by Goerick et al. [2007]. Systematica is an incremental hierarchical control architecture with explicit definition of top-down modulations and bottom-up representations between layers shown in Figure 2.3. In this framework each layer n having internal dynamics D_n may process independently from all other layers. The processing is realized by modules residing in the layer (not shown in the figure). The input space X of a layer is covered by sensory information gained via exteroception and proprioception. A layer can create system-wide accessible representations R_n and can be modulated by top-down information $T_{m,n}$ from another layer m for $m > n$. Layers may generate motor commands M_n with priorities P_n to act in the environment. Any conflict caused by motor commands coming from different layers simultaneously is handled by a conflict resolution mechanism according to priorities. Bolder et al. [2008], Dittes et al. [2009], Goerick et al. [2007, 2009] present successful applications of the Systematica framework as large scale intelligent systems. The requirements for structuring intelligent systems proposed by the Systematica framework are similar to the principals of the theory of cognitive modularity proposed by Fodor [1983]. As in the subsumption framework the breakdown of competencies into layers follows the domain specificity principle. Additionally, explicit definition of top-down influence of higher level layers on lower level ones supports the principle of information encapsulation as modules in a layer can operate without top-down information. Introduction of representations in the formal framework reflects the principle of shallow outputs. In the Systematica framework representations in a layer can be built by contribution of more than one module and may reflect basic categorizations of objects. For example in the systems where Systematica was applied, proto objects generated by perceptual modules were used as low level representations of world entities. Finally, it can be observed that the mandatory operation principal is also valid for the Systemat-


Perception
Cognition
Action
Individual cycles e.g. one perception module
Other approaches
A sensory-motor behavior i.e. microbehavior
Microbehavior approach

(a)



(b)

Figure 2.4: Microbehaviour approach as proposed in Sprague et al. [2007]: (a) functional decomposition of the framework, (b) state machine diagram of microbehaviors for a navigation scenario. Boxes refer to different states in the scenario and every bullet point in boxes refers to a microbehavior.

ica framework as every layer should be able to function independently from all other layers.

Sprague et al. [2007] introduce a different perspective to the system formalization problem. Frameworks like Subsumption or derivatives of subsumption such as Systematica describe hierarchically organized layers that are full sensory motor loops. Layers contain modules that are responsible for elementary information acquisition and processing i.e. perception, cognition and action. Dynamics of these sensory-motor loops are regulated by such modules within the layer. Unlike this approach, Sprague et al. [2007] propose small units of sensory-motor behaviors, called microbehaviours, that are complete cycles of perception, cognition and action as shown in Figure 2.4(a). Although the functionality of a microbehavior is parallel to that of a layer in Subsumption and Systematica frameworks, it is more correct to consider them analogous to modules because, the microbehaviour approach does not impose a hierarchical organization of microbehaviours. Complex system behavior emerges when multiple microbehaviors are executed in parallel to achieve multi-tasking and large scale scenarios are handled by state machines prescribing which collection of microbehavior should be used at which circumstances. An example for a navigation scenario with obstacles to avoid and litters to pick up in the environment is shown in Figure 2.4(b).

While the microbehavior approach presents a novel solution to intelligent system design for behavior organization it comes with several drawbacks. First of these drawbacks is the perceptual redundancy. The fact that microbehaviors are stand alone perception-cognition-action cycles implies that mechanisms which perceptual representations are derived from are re-implemented for different microbehaviors using the same representations. Such a manifestation is easy to picture when two tasks are related to the same type of objects. For example, in the scenario shown in Figure 2.4(b) imagine that both target objects to be picked up and obstacles to be avoided are red in color which is one of the cues microbehaviors ‘pick up objects’ and ‘avoid obstacles’ use to localize corresponding objects. In this case, the perceptual mechanism to detect red color will be implemented for both microbehaviors causing a perceptual redundancy in computation leading to infeasibilities of information acquisition in practice. Furthermore, the microbehavior approach considers interaction between modules only at the action space. This prevents microbehaviors to share whatever knowledge they accumulate through the interaction of the system with its environment. Finally, the question of which set of microbehaviors are relevant under which circumstances is addressed by arbitrary manual programming. This means that application of the system to a new domain may require re-organization of the whole microbehaviour space.

Although this thesis favors modular system design, the question whether an optimum degree of modularity can be determined or even exists can be argued upon. From the engineering point of view, too much granularity may lead to problems in debugging, maintenance, parameter optimization, network overhead etc. whereas in monolithic systems the flexibility to control information flow in the system may be lost, hindering an effective use of constrained resources. Especially for large-scale systems, it is often difficult to make an assessment to determine an optimum level of granularity since this implies design and evaluation of multiple instances of the system with different formal structures and granularity. As the current chapter of the thesis highlights, the level of granularity for the systems implemented in this thesis is chosen with best effort regarding studies in the cognitive systems literature (e.g. Fodor [1983]), successful implementations of similar large-scale intelligent systems and the author’s own experience.

2.2 Adopted Modular System Model

Various instances of a large-scale system are used to test hypotheses on information acquisition in this work. Following a formal structure for the implementation of these systems facilitates design, evaluation and comparison of various system instances. The Systematica framework presented by Goerick et al. [2007] is adopted for the formal structure of these system instances.

2.2.1 Formal Definition of the System

Figure 2.5 illustrates the formal definition of the architecture for system implementation used in this work. The formalization follows the Systematica framework proposed by Goerick et al. [2007]. The architecture consists of three layers. The first layer D_1 is mainly responsible for information acquisition from the environment. Thus, perceptual modules and the module management learning (MML) mechanism reside in this layer. Outputs of modules that are encapsulated by representation R_1 indicate the result of module operations. Since the operation of modules is restricted by the constrained system resources the MML mechanism is integrated in this layer. MML decides which module gets access to the constrained system resources and demand from this module is executed. This is indicated as action M_1. The second layer D_2 builds a scene rep-

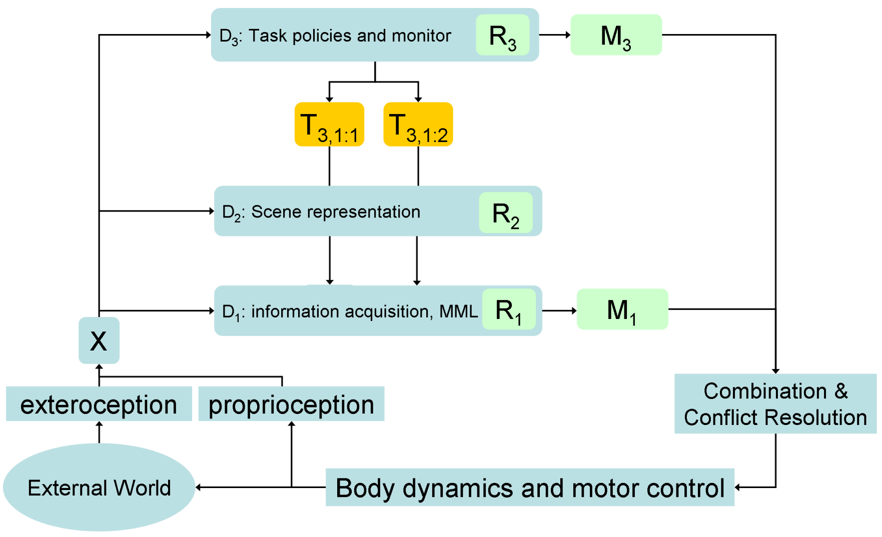

D3: Task policies and monitor
R3
M3
T3,1:1
T3,1:2
D2: Scene representation
R2
D1: information acquisition, MML
R1
M1
X
exteroception
proprioception
Combination & Conflict Resolution
External World
Body dynamics and motor control

Figure 2.5: Adopted modular system architecture formalized as Systematica framework. Representations in the system are: module outputs R_1, scene representation R_2 and rewards R_3. Top-down information channels are: reward information $T_{3,1:1}$ and task influence on information acquisition process $T_{3,1:2}$. Possible actions are: actions for information acquisition M_1 and actions for task execution M_2

resentation by combining the information acquired by the modules in the first layer. Hence, representation R_2 in this layer contains temporally stabilized information about the properties (spatial position, color, motion, etc.) of entities in the environment. The third layer D_3 encapsulates higher level policies for the system to achieve certain tasks. The policies can perform actions M_3 (e.g. steering or breaking actions of a robot car for navigation or reaching actions of a humanoid for reaching an object) to fulfill these tasks. The task monitor in this layer follows the state of the system and the environment to decide if the tasks are fulfilled or failed. Depending on the outcome of tasks rewards that are indicated as representation R_3 are generated. This layer has two top-down modulatory connections to the first layer. The first connection $T_{3,1;1}$ conveys the acquired reward to the MML framework to adapt its parameters. The second connection $T_{3,1;2}$ represents the task influence on the information acquisition process. If task policy demands a specific module to be active this channel can be used to send the request.

2.2.2 Functional Decomposition of the System

Figure 2.6 shows a general functional decomposition and information flow of systems implemented for this work. This section gives detailed information about the aspects of these systems from an operational perspective and definitions that are used throughout the dissertation.

Perceptual Modules

Modules residing in the first layer perform elementary operations for information acquisition and pre-processing. Different modules operating on different domains of modality can be plugged into the designed system. Several examples of such modules can be color tracking modules (red, green, blue, etc.), motion detection modules for the vision

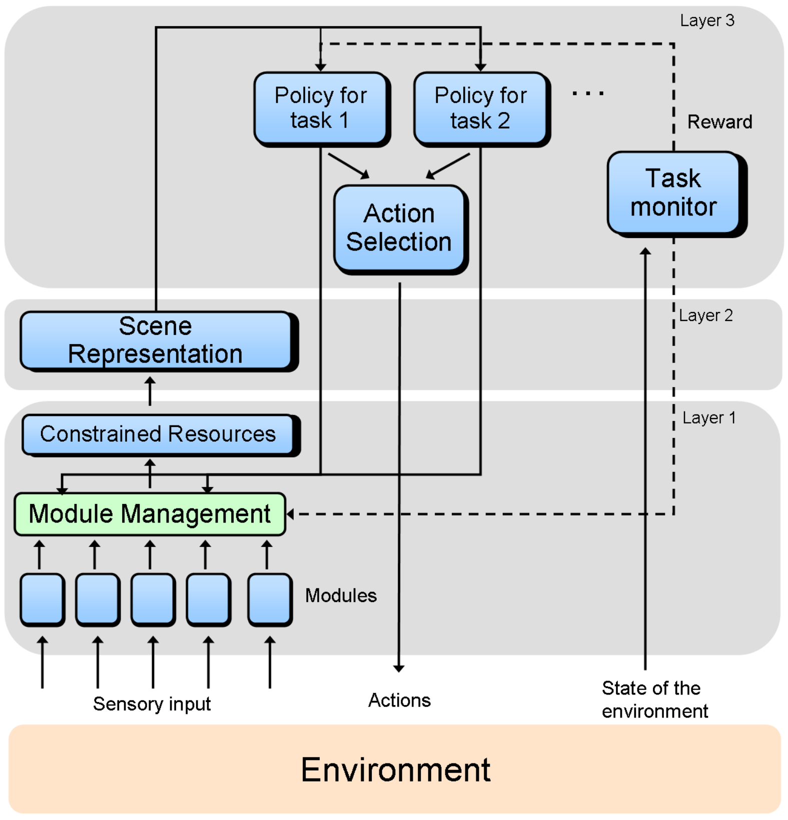


Figure 2.6: Functional decomposition and information flow of the adopted modular system architecture.

domain, a sound localization module for the audio domain, an obstacle detection module for sonar/laser domain. A module may yield representations of whatever feature of scene entities it measures: a red color tracking module may say 'there is a red entity in this position of the scene'. Such representations from various modules are used by the scene representation mechanism in the second layer (dotted arrows from modules to the scene representation mechanism in Figure 2.6) to construct a basic understanding of the scene. Operation of modules is restricted by constrained system resources. At any point in time, modules may generate requests to access these resources (e.g. motor commands for gaze-shifts to access actuators changing camera orientation).

Constrained Resources and Module Management Framework

Systems usually have limited resources: physical, computational, memory, energy, etc. Since multiple modules may demand access to the limited resources of the system at the same time, an arbitration mechanism is necessary to decide which module should be given the access at any point in time. The Module Management framework handles the arbitration between demands from modules.

In this work two kinds of constraints are studied separately: physical and computational constraints. Physical constraints refer to the case where the system has sensors with limited sensory coverage or variable accuracy across its sensory range. In this case the system has to move its sensors to collect information from the environment.

Modules may generate motor commands for sensor orientation at any point in time depending on the result of the processing. For example when a red color tracking module detects red features in the image, it generates motor commands to direct the cameras towards the detected features. This is an attempt to keep them in the center of the field of view so that the maximum amount of information can be obtained. This is because the longer the features stay in the view, the more it is likely to gain information from them. Keeping the features in the center of the camera image makes it less probable that they leave the image due to ego-motion or movement of the object the features belong to. Computational constraints occur when the system has limited computational power and not all modules can be executed in parallel. In this case modules generate demands for access to the computer to process data instead of motor commands.

Scene Representation

A scene representation mechanism implemented in the second layer (see Figure 2.6) can keep and integrate various properties of elements in the scene over time. Yan et al. [2011] showed that even a coarse level of scene understanding can boost the performance of higher level functions in a system. The scene representation mechanism can be seen as a simple working memory (WM). The WM is a list of tuples keeping various properties of the scene elements such as an id for each element indiced by id, the estimated position of the element in the world p_{id} and the age of the entity in the WM a_{id}. The id is simply an integer number and distinguishes scene elements in the WM from each other. The position of the element may be used by high level mechanism to accomplish tasks. The age property makes it possible to delete outdated elements from the WM.

In applications presented in this work the interaction of the system with its environment is divided in temporal sessions called epochs. The WM has no elements in the beginning of an epoch. At every time step an update is done. First, the age of the elements in the WM are updated:

$$\forall id \in WM : a_{id} \leftarrow a_{id} + 1. \tag{2.1}$$

Next, all elements with an age greater than the maximum age a_{max} are removed from the WM. The age constraint reflects the limited capacity of the WM. Since the proposed framework is motivated by constrained resources (computational, physical or memory) it is consistent to employ this constraint even though the memory constraints are trivial for the applications. Subsequent to the age update, new information acquired from the modules at the current time step is integrated into the current state of the WM. To accomplish this the WM has to deal with the correspondence problem i.e. does the new information belong to one of the entities already stored in the WM or does it belong to a new entity? In earlier implementations this problem is resolved by directly using the id of scene elements. Later, a similarity measure is computed between the properties new information is related to (e.g. color, spatial position) and properties of the entities already stored in the WM. If a significant similarity with one of the stored scene elements is detected (e.g. employing a threshold operation) the new information is assumed to belong to that entity and it is updated. A new entry for a new entity is created in the WM otherwise. The age of updated or newly created entities is set to zero.

Tasks and Policies

Tasks are formulated as heuristics defined over various environmental states. For example in the simplest case a reaching task for a humanoid can be defined over states of position of the object to be reached, position of the hand of the robot and a threshold applied to the distance between object and hand position to decide if the reaching task is completed successfully. Formally such interactions are defined in the framework of Markov Decision Process that is explained in Section 3.1.1. Task policies a system employs in the third layer (see Figure 2.6) prescribe control policies mapping environmental states (e.g. object and hand positions for a reaching task), available to the system via the scene representation mechanism, to actions (e.g. arm velocity for a reaching task) to accomplish such tasks. Such control policies can be manually programmed or learned by interaction. Both programmed and learned policies are used in this work to explore varying levels of system complexity. Competency of a system is directly related to the performance of task policies and task policies are influenced by the information the system has about the environment (i.e. the scene representation). Therefore, it is important to employ an efficient strategy for information acquisition. Especially if the system has to execute multiple tasks related to different entities in the scene in parallel, the information acquisition strategy is more crucial.

An additional connection from task policies to the Module Management mechanism reflects top-down influence of task policies on the information acquisition process. If a task policy can infer which scene entity is related to its task it can demand the system to attend to that entity or activate modules, which can deliver information about that entity.

For policies that require common actuators a further arbitration mechanism is necessary. This is denoted as ‘action selection’ functionality in Figure 2.6. Since these mechanisms depend on specific implementations of policies and scenario, these are explained with the applications they are used in.

Task Evaluation and Rewards

A monitoring mechanism in the third layer (see Figure 2.6) observes all ongoing tasks and generates rewards for the system depending on the outcome of the scenario. For successful completion of tasks a positive reward is generated whereas if tasks cannot be achieved a negative reward may be given. Rewards are used by the Module Management framework to adapt its parameters for improving the information acquisition strategy. Rewards may also be given to task policies if those are implemented using reinforcement learning algorithms. In practice such a monitoring mechanism may make mistakes leading to wrong assessments of scenario outcomes. Since this is a broad subject for investigation, such situations are not covered in this dissertation and reward signals used in this framework are noise-free.

2.2.3 Benefits of Modularity in Handling System Complexity

In order to illustrate the benefits of modularity in handling system complexity a case study is given in Section 5.3.1. This study compares two systems designed with a modular and non-modular approaches in performing an autonomous navigation scenario in simulation. The simulation environment consists of obstacles of several types (i.e.

static and moving) that the agent has to avoid and landmark objects that the agent has to visit. The objective of the systems is to learn capabilities (e.g. control policies) of navigating in such environments. The complexity of the scenario can be altered by varying the number of these objects. Hence, experiments conducted in these controlled conditions can report how the performance of both systems scales with increasing difficulty of the scenario. Information about the environment is gathered from a camera with a 90 degrees field of view mounted on the agent facing to the steering direction. There is no resource constraints on gathering information in these experiments. The agent is controlled by velocity and steering angle commands. The non-modular approach implements a reinforcement learning policy that learns a direct mapping of post-processed sensory data to actions of the agent. The modular approach breaks down the scenario into two tasks (i.e. obstacle avoidance and target pick-up) and learns control policies for these tasks. Environmental information for these control policies are supported through perceptual modules and a working memory.

Results show that non-modular approaches for the design of intelligent systems can have difficulties coping with the complexity of the scenario. The system following a modular approach performed better than the non-modular system in more complex situations. Greatest drawback of the non-modular approach from this perspective is that it casts the problem at hand to one specific algorithm. Therefore, the performance of the system facing the problem at various complexities is limited by the capabilities of the algorithm. The modular approach used in this work on the other hand, provides formal definitions for vital elements of the system (such as perceptual modules, scene representation, task policies, etc.) and interactions between them. The scenario can be broken into tasks to be accomplished in parallel and different methods can be applied for solving these tasks. The overall quality of the system emerges from the operations and interactions of the modules in the system.

2.3 Summary

The main purpose of this section was to introduce formal design frameworks for intelligent systems and clarify why such frameworks are important. Intelligent systems research is inspired from biological systems in many aspects. To engineer systems that can show similar characteristics in terms of intelligence a definition of a formal system structure can also be inspired from such biological systems. The theory of modularity in cognitive psychology is a detailed theory about the organization of the human cognition. This section gave an overview on this theory and its formal principals. Further on several influential frameworks for intelligent systems design that support modularity are introduced. Their relation to the cognitive modularity theory as well as their powers and shortcomings are explained. Furthermore, the modular system model adapted in this work is presented. This approach is followed in the implementation of systems used as experimental test beds to evaluate the MML framework. The formal definition of the model and main components from the functional point of view is clarified. Finally, benefits of the modular system design is presented in reference to a case study comparing systems designed with modular and non-modular approaches in an autonomous navigation scenario.

CHAPTER 3

Learning in Large Scale Modular Systems

The intelligent systems subject to this dissertation should have the ability of learning from experience. This implies the development of capabilities that help the system to adapt to its environment through its life-time. This requires a value system for the developing intelligent system so that it can evaluate its actions, perceptions and predictions. The value system is one of the mechanisms that influences what the system can learn and what its behavior will be. At large-scales, a value system uses rewards as learning signal and a metric of how well a system is doing in a given scenario for different local learning mechanisms embedded in the system. The credit assignment problem i.e. distributing a recently acquired reward among these mechanisms is still an open issue (Rothkopf and Ballard [2010], Toutounji et al. [2011]). Moreover, as Mikhailova [2009] states, control of reward generation is also a crucial process as poor implementations may cause the development process to stagnate. Hence, design of such a value system is a broad topic and is not in the scope of this work (see [Mikhailova, 2009, p. 25-40] for a detailed discussion). In this dissertation, the relation between local learning mechanisms is explored. Two main groups of reward-based local learning mechanisms exist in the proposed framework: policy learning mechanisms for task execution residing in the top layer of the system and the module management learning (MML) mechanism for information gathering at the bottom layer of the system. This thesis investigates if the proposed MML framework can support multiple policy learning mechanisms in terms of information acquisition, so that these mechanisms can perform their objectives (i.e. learning policies).

There are two main sections in this chapter. The first section gives a brief overview on reinforcement learning (RL), which is the computational framework for reward-based learning, in order to provide a background knowledge on policy learning mechanisms used in this work. Modular RL frameworks capable of handling multi-task

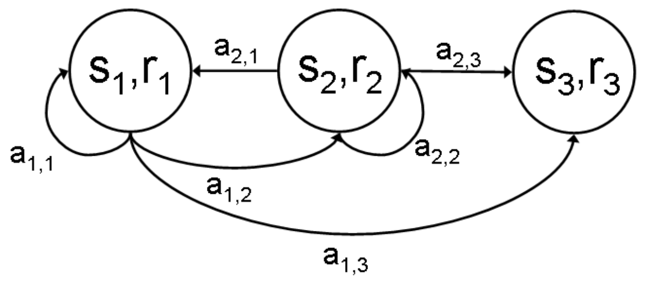


Figure 3.1: A simple Markov Decision Process (MDP) with three states, s_3 being a terminal state. States are shown as s, actions are a and rewards are s.

scenarios are also explained in this section. Further on, it is summarized how partial observability occurs as a result of constrained resources in the information gathering process and how this is addressed in the RL framework. At the end of the section, it is shown that the system level approach proposed in this thesis can address the partial observability in a simpler way. In the second section, the mechanism proposed by this thesis to realize the idea of learning efficient information gathering strategies is explained. This mechanism is referred to as Module Management Learning (MML). The mechanism is explained in two parts: the first part introduces the structure that achieves distributing system resources among modules in time and the second part explains how the parameters of this structure can be learned via rewards to achieve efficient distribution. At the end of this section empirical observations from various experiments are shown. These results show that the proposed MML framework can learn to use a group of modules among a greater set in a desired sequence to maximize the acquired reward and in cases of changes in the desired sequence can adapt its parameters accordingly.

3.1 Reinforcement Learning

RL is learning how to map environmental states to actions throughout interactions with the environment (Sutton and Barto [1998]). Unlike supervised learning, a RL agent is not explicitly told which actions to take nor does it classify states based on the statistical properties of the data as in unsupervised learning. Instead, a RL agent tries to maximize the acquired reward in time. Rewards are a numerical signal representing the utility of environmental states and/or taken actions. Characterizing the interaction of systems with their environment in a simple way and putting the learning problem in this simple framework (see Section 3.1.1), RL found a wide area of applications in the robotics and intelligent systems research. The work described in this dissertation uses several aspects of RL both in theory and implementation. For this reason a brief summary on RL is given in this section.

3.1.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a model of an agent interacting with its environment. Due to its simple formulation, it facilitates describing tasks for intelligent systems. An MDP can be described as a tuple $< S, A, P, R >$ where

- S is a finite set of states of the world;
- A is a finite set of actions an agent can take;
- $P : S \times A \to \Pi(S)$ is the state transition function, producing a probability distribution over all states for each world state and action i.e. $P(s, a, s')$ indicates the probability of ending up in state s' given that the world state is s and the agent takes action a;
- $R : S \to \mathbb{R}$ is the reward function indicating the expected immediate reward of a state as $R(s)$.

For example, a simple MDP formulation of a collision avoidance task in a navigation scenario is illustrated in Figure 3.1. Here, states indicate the distance of the agent to the nearest obstacle. For the sake of simplicity, they are discretized as 'far' s_1, 'close' s_2 and 'crashed' s_3 . The state of crash s_3 indicating a collision is the terminal state and yields a negative reward $r_3 < 0$. Other states yield larger rewards $r_1, r_2 \geq 0$. The purpose of the agent is to maximize some measure of the long-term acquired rewards. Expected sum of reward is one such a measure that can be written as:

$$E[R(s^{(0)}) + \gamma R(s^{(1)}) + \gamma^2 R(s^{(2)}) + ...],$$

where E is the expectation operation, $s^{(t)}$ indicates the state at time step t and $0 < \gamma \leq 1.0$ is the discount factor that ensures the sum is a finite number even though the horizon is infinite (i.e. $t \to \infty$). If the agent follows policy π a value function $V^\pi : S \to \mathbb{R}$ can be written as:

$$V^\pi(s) = E[R(s) + \gamma\big(R(s^{(1)}) + \gamma R(s^{(2)}) + ...\big)|s = s^{(0)}],$$

which can be translated into:

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P(s, \pi(s), s')V^\pi(s'). \tag{3.1}$$

This is the Bellman equation (Bellman [1957]) that expresses the value of a state at a certain point in time in terms of the value of the remaining states that will potentially be visited by following policy π. The optimal value function that maximizes the expected future rewards is:

$$V^*(s) = \max_\pi V^\pi(s). \tag{3.2}$$

A policy that selects the action with the maximum value with respect to the optimal value function in every state becomes the optimum policy $\pi^*(s)$ by definition:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s, a, s')V^*(s'). \tag{3.3}$$

3.1.2 Optimizing Value Functions

Given that the transition function $P(s, a, s')$ is known, the optimal value function can be derived by applying the Bellman equation (Equation 3.1) to all states and taking all actions. This is a dynamic programming approach that was introduced by Bellman

[1957] and imposes an exhaustive search in the state and action spaces. This is generally implausible for intelligent systems that learn by interacting with their environment for several reasons: firstly, visiting every state and action may not be possible for a system in practice as some of the states may be unreachable or some of the actions of the system may be constrained during the learning time. Moreover this process requires extensive amount of resources in terms of computation and time for tasks with broad state and action spaces. As an alternative to dynamic programming, numerous iterative methods were proposed in the literature several of which are compiled in Sutton and Barto [1998]. These methods can learn the optimal value function on-line by immediately using the data gathered from the interactions of the systems with the environment. One of such methods is the value iteration algorithm:

$$V(s) \leftarrow R(s) + \max_a \gamma \sum_{s'} P(s,a,s')V(s'). \tag{3.4}$$

The value function $V(s)$ converges to the optimal value function $V^*(s)$ with a policy that can explore the state-action space. However, the value iteration algorithm requires that a model of the environment $P(s,a,s')$ is known beforehand. Thus, the method can not be applied to cases where such an information cannot be derived. Alternatively, Q-Learning is a model-free approach that can learn the optimal value function iteratively without the knowledge of the transition function. The value function is represented as a Q-function $Q(s,a)$ that predicts a value for a given state s and action a. The update is done as:

$$Q(s^{(t)},a^{(t)}) \leftarrow Q(s^{(t)},a^{(t)}) + \alpha[r^{(t+1)} + \gamma \max_a(Q(s^{(t+1)},a)) - Q(s^{(t)},a^{(t)})], \tag{3.5}$$

where α is the learning rate and γ is the discount factor. After convergence the optimum policy is to choose the action that maximizes the Q-function at any given state:

$$a^t = \operatorname*{argmax}_a(Q(s^t,a)). \tag{3.6}$$

This is analogous to Equation 3.3. Note that the Q-Learning algorithm attempts to learn the optimal policy regardless of the actions being taken i.e. the learning process does not depend on the policy. Such kind of approaches are called off-policy methods and are useful if there can be restrictions on the actions the agent can take. This may occur in multi-tasking scenarios where different control policies for different tasks require different actions to take.

3.1.3 Modular Reinforcement Learning

Although the RL framework is a powerful framework, its applications are normally restricted to rather small systems dealing with simple scenarios with a single task. The main reason for this is the scaling problem: complex multi-tasked scenarios require larger state and action spaces that need to be explored in order to be learned which in turn increases not only the learning time but also the memory and computation needs. Function approximation methods can be used to overcome the memory and computation constraints however, their utilization is limited by non-linearities introduced by the scenario (Parr et al. [2008]). These limitations led researchers to incorporate modularity in the RL framework where multiple tasks in a scenario are addressed explicitly or complex scenarios with a single task are broken into smaller tasks. An efficient way of achieving this can be learning multiple policies for each task which can be interpreted as modules.

One framework that belongs to the family of modular RL approaches is the hierarchical reinforcement learning (HRL). Generally HRL methods address the scaling problem by introducing the concept of abstract actions and primitive actions. The term primitive actions refers to the actions that an agent takes as defined in the RL framework. An abstract action on the other hand, specifies a whole policy that maps states to actions. Various implementations of this approach exist in the literature (see Barto and Mahadevan [2003] for a review). The essence of HRL can be summarized as follows: one policy controls the agent at a time and this policy can select primitive or abstract actions. If the selection is a primitive action than this action is conveyed by the agent directly. If the selection is an abstract action, then the control is handed over to the new policy associated with the selected abstract action. The new policy controls the agent until a terminal state is reached. In that case the control is handed back to the original policy. However, the policy of control can also be descended further down in the hierarchy as the new policy selects another abstract action. Main contribution of abstract actions to solving the scaling problem is that it introduces structure to the exploration process (Botvinick et al. [2009]). HRL can be very effective for scenarios imposing sequential tasks (e.g tasks in a tea making scenario: boiling water in a kettle, putting the boiled water in a glass, putting a tea bag into the glass). However, multi-task scenarios where the tasks have to be executed in parallel cannot be handled as easily with this framework due to its sequential policy employment.

Another modular RL framework is the multiple model-based RL (MMRL) introduced by Doya et al. [2002] where the decomposition of a non-linear scenario is learned through the competition and cooperation of multiple modules. Each module contains a RL controller and a prediction model. MMRL uses prediction errors to weight the outputs of modules and gate the learning of models and RL controllers. Action selection is based on the Softmax method applied to the weighted outputs of the modules. During the course of learning each module specializes to a local solution of the scenario (in terms of prediction and control) in space and time. Hence, the module that performs best in a given state (i.e. the module with the least prediction error) controls the agent.

Modular RL frameworks that use model-free RL methods (such as Q-Learning as explained in Section 3.1.2) also exist. Such a framework is used in various applications such as Rothkopf and Ballard [2010], Sprague et al. [2007], Toutounji et al. [2011]. In the case of Q-Learning, separate Q-functions $Q_i(s, a)$ are learned for each task. If the action space is the same for all Q-functions it can be assumed that the Q-function for the whole scenario is approximately equal to the sum of the individual Q-functions for the separate tasks. Thereby, the best action can be computed as:

$$a' = \underset{a}{\operatorname{argmax}}\Big(\sum_{i=1}^{M} Q_i(s_i, a)\Big), \tag{3.7}$$

where $Q_i(s_i, a)$ is the Q-function for task i, s_i is the current state for task i and M is the number of task policies. Compared to the HRL, both MMRL and model-free modular RL have greater potential to handle multi-task scenarios since actions from individual policies are selected with respect to a measure of importance of the task.

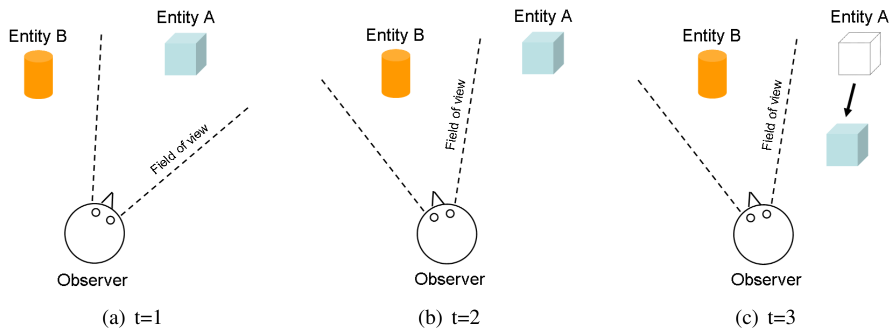


Figure 3.2: A demonstration of partially observability originated from attention shifts.

In the model-free modular RL approach tasks can be defined explicitly in reference to the desired external manifestation of the system. For example, in an autonomous navigation scenario two tasks can be defined as 'avoiding obstacles' and 'picking up target objects'. Q-functions are then, associated directly to these tasks. In the MMRL local aspects of the scenario (analogous to tasks in the related frameworks) emerge in the course of learning the models and RL controllers. From systems design point of view, the model-free approach is closer to the modularity concept since tasks can be defined, added to or removed from the system easily. Moreover, tasks used in the scope of this thesis do not require explicit predictive models to be learned. Hence, a model-free modular RL approach is used in the systems implemented for this thesis.

3.1.4 Partial Observability

Partial observability is the case when a system has no direct access to one or more state variables of the environment at any point in time. Partial observability is a very broad concept and can occur due to uncertainties in the environment (e.g. movement of objects in the scene) or in the system (e.g. movement of the system). This thesis is concerned about the situations where partial observability occurs as a result of the information gathering process of the system. Figure 3.2 illustrates such a situation where the observer attends to Entity A at time $t = 1$ and shifts its attention to Entity B at time $t = 2$. Since the field of view of the observer cannot cover Entity A in this new situation, no information about this entity can be obtained. Hence, if the entity moves in the scene at time $t = 3$ the observer cannot perceive this.

A common approach to handle such situations is using probabilistic modeling. Bayesian filters, Kalman filters and particle filters are several examples of this family (see Simon [2006] for detailed explanations) and widely used in various applications such as tracking objects, handling occlusions and localization. A burden these methods introduce is the necessity of models of the environment (e.g. noise models, movement models).

If partial observability exists in an MDP, it is referred to as Partially Observable Markov Decision Process (POMDP). Handling POMDPs is not trivial and requires restructuring the MDP framework. A POMDP can be described as a tuple $< S, A, P, R, \Omega, O >$ where

- $< S, A, P, R >$ describe an MDP;
- Ω is a finite set of observations an agent can experience of the environment;

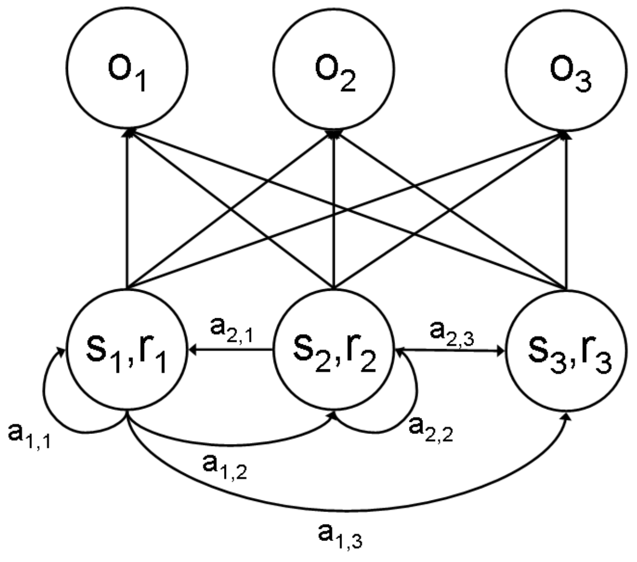


Figure 3.3: A Partially Observable Markov Decision Process (POMDP) with three states, s_3 being a terminal state. States are shown as s, actions are a, rewards are s and observations are o.

- $O : S \times A \rightarrow \Pi(\Omega)$ is the observation function, producing a probability distribution over possible observations for each action and state for each world state and action i.e. $O(o, s', a)$ indicates the probability of making observation o given that the agent took action a and ending up in state s'.

A POMDP formulation of the collision avoidance task example given in Section 3.1.1 is shown in Figure 3.3. States indicate the actual distance of the agent to the nearest obstacle as in the MDP case. However in the POMDP case, the system has no direct access to these states but to a set of observations associated with them. At every time step the system makes a state estimation, referred to as belief states b, based on the last taken action, current observation and previous state estimations. The policy of the agent maps the current belief state into action. For POMDPs using discrete-space belief states the optimal policy turns out to be the solution of a continuous-space MDP, also referred to as 'belief MDP', which is very difficult to derive in the general case (Kaelbling et al. [1998]).

A system level approach is proposed in this thesis to handle cases of partial observability that arise due to the information gathering process. The core elements of the system are perceptual modules, the module management framework and a scene representation mechanism (see Figure 2.6 and Section 2.2.2 for details). The formulation of the idea in the POMDP framework is shown in Figure 3.4. As stated, constraints on resources render the environmental states s hidden and only observations o from these states can be obtained by the system. The perceptual modules are the means of the system to gather such observations. Neglecting noise in sensors, an observation is equivalent to the hidden state it is dependent on as long as the resources of the system are reserved for it. The module management mechanism learns which of the modules can deliver observations relevant to the task or tasks in a multi-task scenario and how to update these observations and consequently the scene representation in time. In this

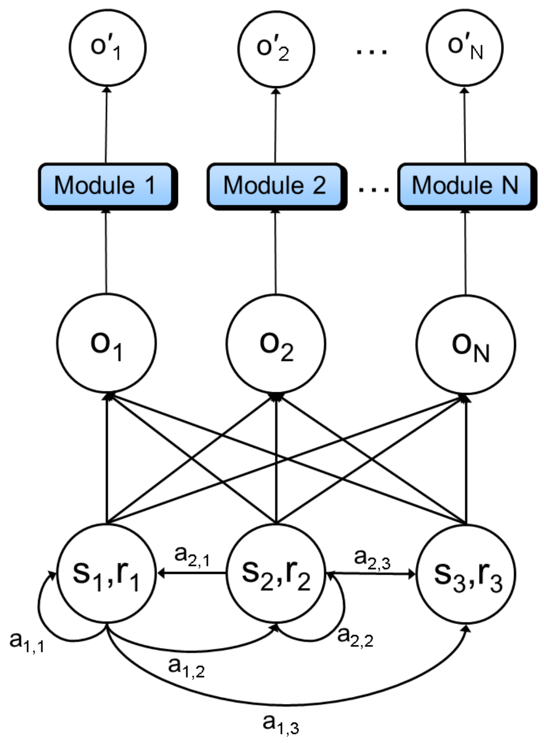


Figure 3.4: The proposed modular systems approach applied to the POMDP framework. Modules can deliver observations o' related to their counterparts o. Observations o depend on the hidden states of the environment s. The system can take actions a based on these observations. Connections between states and observations show hypothetical dependencies.

way the interaction of the system can be modeled by MDPs that can be solved by conventional methods without introducing aspects of the POMDP framework (i.e. belief states, observation model, etc.). Moreover, tasks can be solved by policies defined over mid- or high-level representations of scene entities (e.g. proto objects, objects) rather than low-level features (e.g. edge, color). The use of mid- and high-level representations may lead to an efficient representation of the state space for MDPs since they can be described in a compact way. The proposed system level approach still enables the integration of probabilistic modeling based methods to further reduce the uncertainties in the environment. For example Kalman filters can be used in the scene representation process for tracking task-relevant scene entities which are revealed by the MML.

3.2 Module Management Learning

The concept of the MML framework is explained in two parts: the module management mechanism that achieves the management process (i.e. to decide which module gets access to the constrained resources in time) mediated by its internal parameters and the learning mechanism that uses rewards to adapt those parameters.

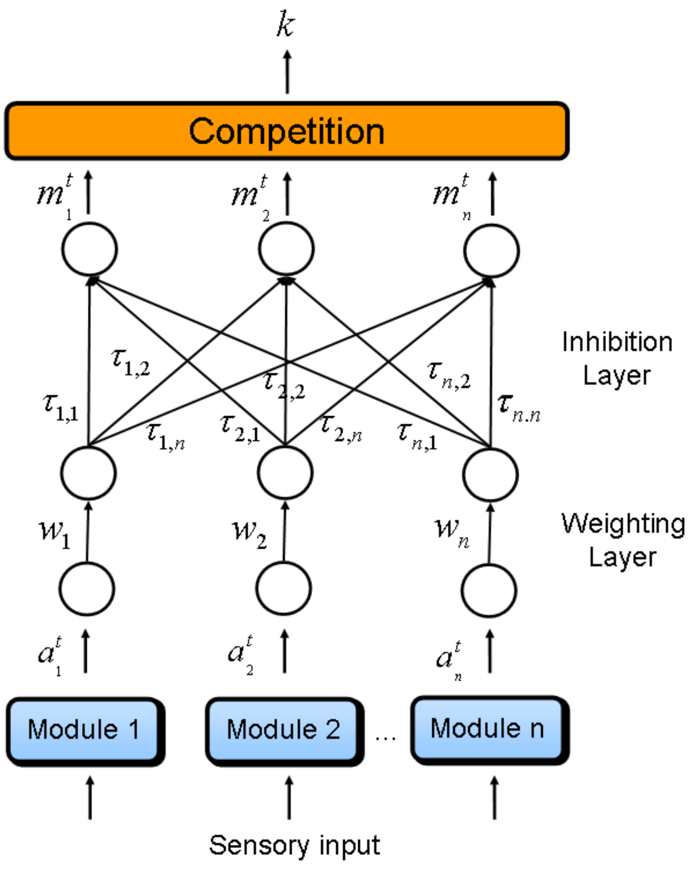


Figure 3.5: The mechanism that performs the management of the module selection process. a_i^t are module activities, w_i are weights, $\tau_{i,j}$ are inhibition parameters and m_i^t are modulated activities with i being module index and t being time step. k is the index of the selected module.

3.2.1 Module Management Framework

The conceptual illustration of the module management mechanism is shown in Figure 3.5. The mechanism approximates a neural network structure composed of two layers: a weighting layer and an inhibition layer. The inputs to the network are individual activity levels a_i^t of modules which indicate the demand of module i for resources at time step t. In most applications a binary value is used for activity level such that $a_i \in \{0, 1\}$ where 1 indicates that a module demands resources and 0 otherwise. However, a continuous value between 0 and 1 can also be used. The weighting process alters activity levels of modules in accordance with their corresponding weight which represents how relevant a module is for the current situation (i.e. scenario). In the inhibition layer, the active module (referred to as module k) in the current time step (i.e. the module using the resources) inhibits the weighted activity levels of all modules. The strength of inhibition changes in time. Dynamics of the inhibition process is regulated by inhibition parameters $\tau_{i,j}$. In practice, the inhibition process controls the temporal sequencing of modules. The final modulated activity levels m_i^t go into a competition process to determine which module will be the active module at the next time step. Different mechanisms can be used for this process such as winner-take-all (WTA) (Hebb [2002]) or Softmax action selection (Sutton and Barto [1998]).

The process of modulating module activity levels is formally defined as:

$$m_i^t = w_i \, y(t; \tau_{k,i}) \, a_i^t, \tag{3.8}$$

where a_i^t and m_i^t are the output activity and modulated output activity of module i at time step t respectively, w_i is the weight of module i, $y(t; \tau_{k,i})$ is the inhibition function parametrized by time constant $\tau_{k,i}$ and k is the index of the active module at time step t. The inhibition function $y(t; \tau_{i,j})$ determines how strongly module i inhibits module j at time t. This is defined as:

$$y(t; \tau_{i,j}) = \begin{cases} 1 - f(t; \tau_{i,j}) & \text{if } i = j, \\ f(t; \tau_{i,j}) & \text{if } i \neq j. \end{cases} \tag{3.9}$$

The case $i = j$ is called self-inhibition and the case $i \neq j$ is called lateral inhibition. The logistic function is used to model the temporal characteristics of inhibition:

$$f(t; \tau_{i,j}) = \frac{1}{1 + \exp\big(-\tau_{i,j}(t - t_c)\big)}, \tag{3.10}$$

where t_c is the time step when the currently active module was selected. In other words, $(t - t_c)$ keeps track of the active module runtime. This is like a clock that is set to zero every time a new module is selected. The logistic function limits the strength of inhibition between zero and one. This prevents instabilities caused by excessive inhibitions. Only the active module inhibits the activity of other modules (lateral inhibition) and itself (self-inhibition). Self-inhibition ensures that the active module releases control at some point (a function similar to the inhibition-of-return), while lateral inhibition has a strong influence on the choice of the next active module. To gain an insight on how the inhibition mechanism works the inhibition functions for self-inhibition and lateral inhibition are shown in Fig. 3.6. When the module is first selected, the lateral inhibition is strong and the self-inhibition weak. The strengths of these inhibitions change during the time the module stays active: lateral inhibitions decrease while self-inhibition increases. Eventually another module is selected and the whole process repeats. The inhibition parameters $\tau_{i,j}$ control the profile of the inhibition function establishing temporal sequencing of modules.

Modulated activity values of modules goes into a competition process to determine which module will be the active module for the next time step. Two mechanisms are used in the applications: winner take all (WTA) and Softmax action selection. WTA simply selects the module the activity of which is the highest:

$$k^* = \operatorname*{argmax}_i m_i, \tag{3.11}$$

where k^* is the index of the active module in the next time step. Softmax action selection samples the winner randomly from a Boltzmann distribution computed from the modulated module activities. This provides a decent exploration/exploitation balance (Sutton and Barto [1998]). The probability of selecting module i for the next time step is computed as:

$$p(k^* = i) = \frac{e^{m_i^t/Z}}{\sum_{j=1}^{N} e^{m_j^t/Z}}, \tag{3.12}$$

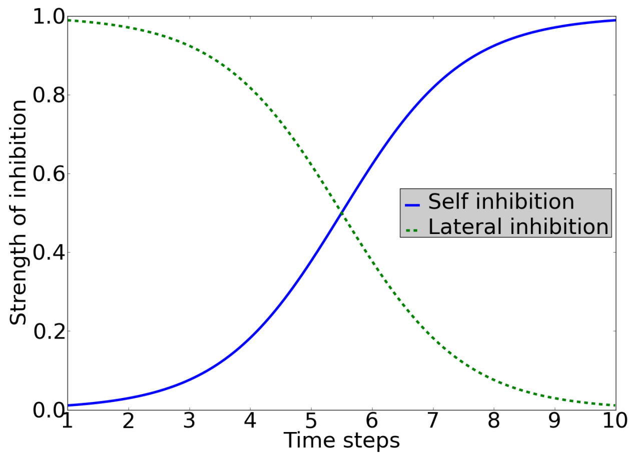


Figure 3.6: Inhibition functions for lateral inhibition $f(t;\tau_{i,j})$ and self inhibition $1 - f(t;\tau_{i,j})$ for $\tau = 0.5$ and $t_c = 0$ illustrating the change of inhibition strength with time for self-inhibition and lateral inhibition.

where N is the total number of modules in the system and Z a positive parameter called temperature. Low temperatures cause a greater difference in selection probability for actions that differ in their value estimates, while the actions become more equi-probable for higher temperatures. Thus, setting the temperature to a very low value approximates a WTA mechanism.

The time course of control dynamics in a system with three modules is shown in Fig. 3.7. Time steps 1, 4, 9, 14, 19 and 24 show the times when the active module changes. The lateral inhibitions for the active module are at the highest and the self inhibition is at the lowest value at these points. As time progresses, the lateral-inhibition decays while the self inhibition increases. The variation in the time course of lateral inhibitions from the active module to the others is determined by the individual lateral inhibition parameters ($\tau_{i,j}, i \neq j$) and influences when and which module is to be selected in the sequence. The time a module stays active is mainly influenced by the self inhibition mechanism. Hence changing self inhibition parameters ($\tau_{i,j}, i = j$) can control the execution time of modules.

3.2.2 On-line Learning of the Parameters

Systems in the scope of this dissertation are engaged in scenarios with one or more tasks that have to be executed in parallel. Control policies employed by the system to achieve these tasks may require information about different states of the environment that are kept as internal representations in the system. The system is also equipped with a number of perceptual modules each of which can acquire information to update representations given that the module has access to the resources it requires. The system has no a priori knowledge on which module can be associated with which task (and related control policy). The decision of which module is selected to acquire information (thus, which representations are updated) in time is done by the Module Management framework that is regulated by weights w and inhibition parameters τ. Changing these parameters results in variations in module selection behavior of the system. It is as-

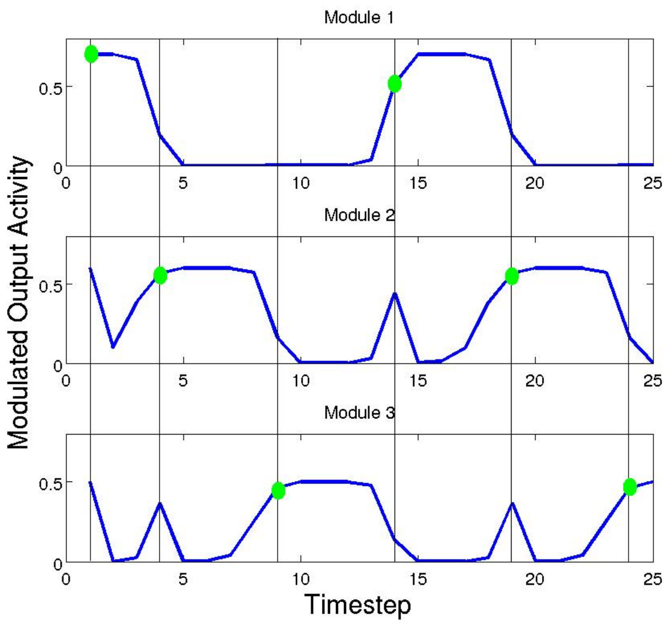


Figure 3.7: Time course of control dynamics in a simplified system with three modules. The vertical axes shows the modulated activity values of modules, the horizontal axis is time. Time steps in which a new module is selected are indicated by common vertical lines in three plots. Selected modules at these time steps are also indicated as green circles in plots. The parameters were set to $w_i = 1.0$ for all modules, $\tau_{i,j} = 0.1$ for $(i, j) \in \{(1, 2), (2, 3)\}$, $\tau_{i,j} = 0.5$ for the rest of the connections.

sumed that there is a value set for parameters w and τ that results in an efficient way of updating relevant states to execute a scenario. The learning algorithm explained here attempts to find such values.

Conventional supervised learning algorithms are not convenient for solving the problem at hand: explicit knowledge on how relevant the information source is and how often it should be updated is required. Instead, a reward based strategy where such knowledge is inferred from the consequences of the actions the system takes is adopted. Basically, the system behaves under the scenario with a set of module management parameters. Depending on the outcome of the scenario a reward is generated. If all tasks were fulfilled successfully a positive reward is given whereas, if one or more failed, the reward is negative. The reward is used to adapt the parameters and a new instance of the scenario is executed. The execution time of a scenario is called an epoch. Although tasks can be formulated as MDPs, common methods to solve MDPs (see Sections 3.1.1 and 3.1.2) may not be feasible for MML since the actions taken within the module management framework (i.e. deciding which module to be active at each time step) do not have direct consequences on state transitions (i.e. they do not change the environment). In the proposed framework, rewards/penalties are used in a Hebbian way to reinforce/weaken connections between modules in the module management framework.

The learning mechanism has to deal with the credit assignment problem: for an epoch yielding a positive (or negative) outcome which internal decisions of the management mechanism were useful (or to blame)? To solve this problem, for every epoch the indices of active modules in the last t_{hist} time steps were stored in a list and the reward acquired at the end of the epoch is propagated over the parameters related to these modules. This is a form of eligibility traces as explained in Sutton and Barto [1998]. After a number of iterations the relevant modules will be revealed statistically in terms of gained positive rewards. The uncertainty of reward delay complicates finding an optimum value for t_{hist}. This is a common problem in RL applications (Sutton and Barto [1998]). Hence, there is no specific rule for selecting t_{hist}. Experiments showed that setting t_{hist} too low results in longer convergence times whereas too high values may lead to instabilities in the learning process. The credit assignment problem can further be addressed by a discount factor $0 < \gamma \leq 1.0$. This sets a heuristics for recency i.e. actions taken closer to the time a reward is acquired are valued more. In essence the value iteration computes the value of a state based on possible actions that can be taken in that state and the values of states that can be reached by taking those actions. Such values can be used as a metric of how relevant modules are to the scenario at hand. As a result, the following update rule is used:

$$\begin{aligned} &\forall n \in \{1, 2, ..., t_{hist}\}: \\ &w^{e+1}_{B(n)} \leftarrow w^{e}_{B(n)} + \alpha(r + \gamma^n w^{e}_{B(n)}), \end{aligned} \tag{3.13}$$

where e denotes the epoch, r is the reward, α is the learning rate, $0 < \gamma \leq 1.0$ is the discount factor, $B(n)$ is a list that stores the indices of active modules in the last t_{hist} time steps of epoch e. Inhibition parameters control transitions between modules. In general it is necessary to reinforce a sequence (e.g. transition from module A to module B) that results in a rewarding outcome and avoid a sequence that yields a negative outcome. Therefore, adaptation of inhibition parameters is done in a Hebbian way (Hebb [2002]) as:

$$\begin{aligned} &\forall n \in \{1, 2, ..., t_{hist} - 1\}: \\ &\tau^{e+1}_{B(n),B(n+1)} \leftarrow \tau^{e}_{B(n),B(n+1)} - \beta r, \end{aligned} \tag{3.14}$$

where r is reward and β is the learning rate. Basically, a positive reward decreases the lateral inhibition parameter between modules $B(n)$ and $B(n+1)$, increasing the probability of selection of module $B(n+1)$ after $B(n)$ was active. If selecting module 1 after module 2 yielded a positive reward this scheme makes it more likely to keep this sequence. In the opposite case, likelihood of selecting module 1 after module 2 is reduced by a negative reward leading to exploration of sequences with other modules. The inhibition mechanism also helps to keep the overall system stable by preventing a strong module dominating the competition process. This is similar to the homeostatic plasticity in the nervous system. Turrigiano and Nelson [2004] showed that synaptic strengths of neurons were adapted dynamically to prevent neural circuits from becoming hyper- or hypo active. Changing the synaptic weights of neurons via a learning process will result in modification of the gain of the network to which the neurons belong to. This will cause too strong (for gains higher than one) or too weak (for gains lower than one) propagation of activities in the network. Local or global inhibitory mechanisms may regulate the gain of the network to provide stability. Similarly in this framework, self- and lateral inhibitions prevent modules with high weights from becoming overly-active and keep modules with low weights in competition.

3.2.3 Regulation of the Learning Process

The learning process can be regulated so that the learned parameters are not changed further after a satisfactory performance is achieved. Performance based regulation also provides robustness to the learning framework. In case of drastic changes (e.g. scenario change) where learned configurations of parameters are no longer valid, the system will start to fail, leading to a decrease in the average reward. In this case learning rates will increase and the system will re-adapt itself to the new conditions. A successful implementation of such an automatic regulation process was previously demonstrated in a visual sensory-motor learning task in Karaoguz et al. [2009].

Average acquired reward can be used as the performance measure for the regulation process. This is computed at every epoch in the following way:

$$\overline{r}^e = \overline{r}^{e-1} + (r^e - \overline{r}^{e-1})/\epsilon_s, \tag{3.15}$$

where r^e is the reward acquired at epoch e and ϵ_s is the smoothing parameter. The initial value is set to zero $\overline{r}^0 = 0$. For the regulation process learning rates α and β are computed on-line at every epoch based on the performance of the system as:

$$\alpha = \alpha_b \exp(-\alpha_s(\overline{r}^e - r_{min})), \tag{3.16}$$

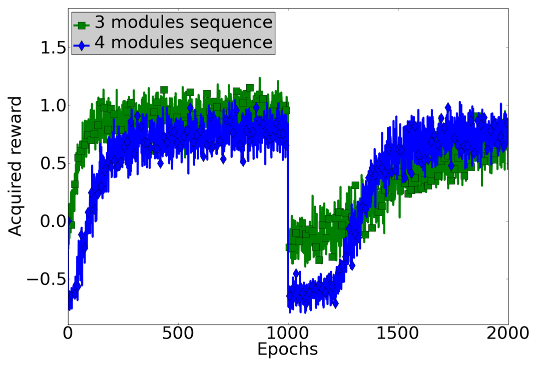
$$\beta = \beta_b \exp(-\beta_s(\overline{r}^e - r_{min})); \tag{3.17}$$

where α_b and β_b are base values, α_s and β_s are the sensitivity of the change of α and β in response to the change in the average reward, r_{min} is the minimum reward that can be obtained and $\overline{r}^e$ is the smoothed average acquired reward at epoch e.

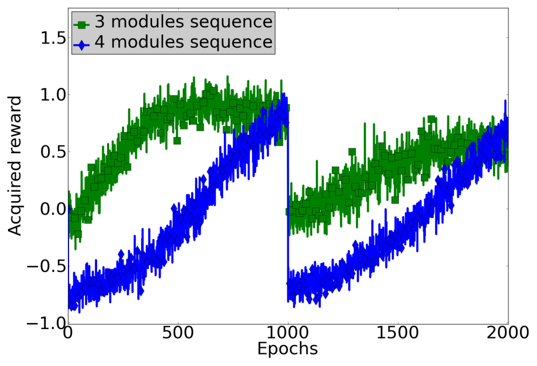
3.2.4 Empirical Observations of MML

A series of experiments are conducted in an abstract simulation to test whether the proposed framework can learn desired temporal sequences of modules using rewards. As a test bed, a system with five modules is simulated. The MML framework is used to learn arbitrary goal sequences of modules. At every time step in an epoch the MML framework selects a module to be active. If the temporal history of the selected modules in the epoch contain the goal sequence then the epoch is ended with one unit of reward plus a time bonus. Time bonus is computed as $(t_{max} - t)/t_{max}$ where t is the time step when the reward is acquired and t_{max} is time out value. If the goal sequence is not found in a pre-defined time period t_{max} the epoch is ended with -1 unit of reward. For each experiment weights in the MML mechanism are set to one and not updated while inhibition parameters are randomly initialized and updated with every reward as the purpose of the experiments is to see the capabilities of MML for learning temporal sequences. Experiments are conducted for two cases of goal sequences: 3-module sequence and 4-module sequence. Goal sequences do not repeat modules (e.g. 3-1-4 is a valid goal sequence while 3-1-3 is not). A goal sequence is determined randomly in the beginning of each experiment. In total 100 experiments are conducted for each case and each experiment is executed for 2000 epochs. Additionally, to test if the framework is robust against changes the goal sequence is switched to another randomly determined goal sequence at epoch 1000.

Figure 3.8(a) shows the acquired rewards in 2000 epochs for 3-module and 4-module cases, averaged over 100 experiments. The framework is able to learn the presented



(a)



(b)

Figure 3.8: Acquired rewards in 2000 epochs for the cases of 3-module and 4-module sequence, averaged over 100 experiments. (a) regulation of the learning process is applied, (b) regulation of the learning process is not applied.

goal sequences. For both cases, learning is accomplished between epochs 0 and 200. Adaptation to the goal sequence change at epoch 1000 takes longer, about 500 epochs. The regulation mechanism for the learning process is also analyzed. Figure 3.8(b) shows results from the experiments with the same settings except for the regulation of the learning process (i.e. a fixed learning rate is used). The plot shows that learning takes much longer without regulation. Learning goal sequences of 3 modules takes about 500 epochs. In the case of 4-module sequence the learning process takes 1000 epochs. Results show that the MML mechanism can learn a set of parameters leading to the selection of modules in a specific sequence that maximizes the acquired reward in time. The mechanism can cope with sudden changes in the goal sequences. Furthermore, the proposed regulation method boosts the learning process reducing the learning time considerably especially for more difficult situations. Further analysis of learning module execution sequences in two different applications of the proposed framework to large-scale developing systems are explained in sections 4.4.7 and 5.3.5.

3.3 Summary

This chapter is started exploring the problem of how local learning functionalities can be integrated into a large-scale system in order to gain the system capabilities of practical intelligence and life-long development. As stated in the literature, establishing such capabilities are facilitated by defining a value system that evaluates the experience of the system through its interactions. Since a natural extension of the value system is reward based learning, an overview to the formal computational basis of reward-based learning, that is referred to as reinforcement learning (RL), is given. Since multi-tasking and partial observability are two general topics covered by this thesis, how a RL framework deals with these problems is surveyed under this overview. Modular RL frameworks and their relation to multi-task scenarios are explained. Given that a system has multiple tasks to accomplish and these tasks are defined over various environmental states, partial observability occurs if the system has no access to one or more of these states at any point in time (a typical case is attention shifts). Further information is given on how the partial observability can be solved in the RL framework. Following this, it is argued that the method proposed in this thesis can handle the partial observability caused by the information gathering process in a much simpler way using a system level approach. Next, the proposed reward-based MML framework is explained. Empirical results demonstrated that the MML framework can learn sequences of modules that maximize the acquired reward and can also adapt to changing conditions.

CHAPTER 4

Application on Gazing Arbitration for Active Vision

An approach where a visual system is able to adjust its visual parameters to aid task oriented behavior is called active vision (Alomoinos et al. [1988]). Such an approach can be beneficial for robotic applications where vision is used as the primary perceptual modality. Systems using the active vision approach require a mechanism that reveals task relevant entities in the scene and plans gaze-shifts on these entities. The Module Management Learning (MML) mechanism proposed in this dissertation can provide a system with such capabilities. Therefore, one of the applications of the MML to test the hypotheses postulated by this dissertation is chosen as such an active vision system implemented on a humanoid in simulation. The iCub humanoid was used as the platform for the system and the experiments were conducted using the iCub Simulator (Beira et al. [2006], Tikhanoff et al. [2008]). The main focus of this application is to analyze various aspects of the system as well as comparing the framework with other methods. Working in simulation facilitates this. A real-world implementation is likely to impose problems in conducting large number of experiments since executing such experiments with a real robot takes significantly longer and controlling the environmental conditions for repetitive experiments is difficult.

The system is composed of various perceptual modules, a scene representation mechanism and high level task policies. Modules generate motor commands for gaze-shifts towards entities in the scene to gather information. Due to physical constraints of the system (i.e. camera system with limited field of view) only one motor command can be executed at a time. Additionally, the scene representation imposes memory constraints by deleting outdated entries (i.e. entries that are not updated for a long time) from the working memory. This is necessary to keep the temporal stability of the environmental knowledge. The MML mechanism is applied to handle these constraints.

The scenario designed for the system consists of two tasks that have to be executed in parallel: a reaching task and a human interaction task. The robot has to reach and touch one of the three differently colored objects moving on a table while keeping interaction with a partner by gazing at him/her from time to time. Despite its simplicity, the scenario reflects various elements of human-robot interaction where a robot's gazing behavior is crucial. Recent studies show that robot's gazing feedback shapes the human-robot interaction in tutoring scenarios (Vollmer [2011], Nagai et al. [2010]). The scenario designed in this work can be considered as a simplified version of a game between a robot and a child. The system employs policies for executing its tasks. Besides MML, the system also accommodates an additional learning mechanism for the object selection policy for reaching. The robot is rewarded upon touching the objects on the table. However, the value of the objects varies and is unknown to the system along with the other environmental parameters such as the movement characteristics of objects, the maximum hand velocity of the robot and manners of the partner towards interaction. These parameters are varied among different experiments. In doing so, a number of instances of different environments are generated and the application is tested on each of these environments. In the beginning of each experiment the system has no idea what it has to do (e.g. which object should be selected, reached and touched? Is it necessary to look at the parter? If yes for how often? etc.). Tasks implemented in the scenario have different characteristics: the human interaction task only involves visual observation whereas reaching is a more complex sensory-motor process. During interactions of the system with its environment the system acquires knowledge about these tasks. Hence, the task performance of the system gradually improves. One of the strengths of the proposed framework is that it can support learning tasks with diverse characteristics and complexity in parallel.

Due to variations in the experimental parameters, it is difficult to derive information acquisition and task execution strategies manually. For example, in one setting it could be more rewarding in the long run to reach for a less valuable object because the speed of the more valuable object is too high to be coped with. However, in another setting the maximum hand velocity of the robot could be high enough to be able to cope with fast objects. A manual design of the gazing process may work in some of these instances but will fail in most. It is shown that the MML framework learns an efficient strategy to update the system's knowledge about environmental states in terms of arbitrating gaze commands from various modules to support learning and executing tasks.

In general, this study investigates the following aspects to test the hypotheses postulated in this dissertation:

- Can the proposed framework find modules, which deliver information relevant to tasks and schedule their utilization in time so that the effects of the partial observability on task policies caused by the gaze-shifting process is minimized?
- Does the framework adapt to variations in the environment and scenario?
- How well does the proposed information gathering mechanism (i.e. MML) support each task in the scenario?
- What are the emerging behaviors in the system after learning?
- What are the individual contributions of different learning mechanisms implemented in the system to the overall performance?

In this section, first the benefits of active vision for humanoid systems are briefly stated and a quantitative analysis of active vision in distance estimation is explained in detail (as presented in Karaoguz et al. [2010]). Next, the system in which the MML is implemented is explained in detail. Finally results from a series of experiments are shown and their interpretations are discussed. Some of the results from experiments with this application are introduced in Karaoguz et al. [2011a, 2013b]. Here a more detailed analysis is presented.

4.1 Why Active Vision?

Active vision is achieved by gaze-shifts that move the focus of attention of a system from one point in a visual scene to another. This is motivated by the human visual system. Studies conducted in neuroanatomy have discovered that images seen by the eyes are projected onto retinas and the visual cortex as space variant representations. The main reason of this is the non-uniform distribution of the photoreceptors in the retina. The density of photoreceptors is greater in the center, the region called fovea, and less in the periphery. This results in a high resolution of the foveal region and low resolution of the peripheral region. Eye movements keep target stimuli on the foveae of both eyes. Hence the highest resolution view of visual targets is achieved. For robotic systems realization of active vision may imply additional effort to implement hardware for the physical movement of the cameras, software for the hardware control and algorithms that can cope with camera rotations and translations. However, even without the use of foveated images (as done in this work) an active vision setup still presents solid benefits that may be worth investing these efforts. One of the most prominent profit is that using active vision leads to accurate depth information at close ranges (Karaoguz et al. [2010]). Moving cameras also introduce new cues for distance estimation such as vergence triangulation (see below) and oculomotor parallax (Santini et al. [2009]). An efficient combination of these cues leads to even more precise estimations (Karaoguz et al. [2011b]). Additionally, the active vision approach allows compensatory camera movements in the case of ego motion (e.g. while walking in humanoids, driving wheeled robots or cars on rough surfaces) for image stabilization (Shibata et al. [2001]). This is analogous to the VOR and OKR in the human visual system. Sub-pixel analysis via small camera movements is yet another potential benefit of active vision. For example, in Khademi et al. [2010] super-resolution is achieved via relative motion of image and pixel arrays of CCDs. Such a motion can be generated with camera movements. Among all potential benefits, this section focuses on how the active vision approach improves the distance (also referred to as depth) estimation.

Complex tasks like reaching, grasping or driving involve depth perception and vision is a sufficient and reliable depth information source for such kinds of tasks. However, depth estimation is an ill-posed problem since the optical process of visual information gathering projects the 3D information onto 2D images (Palmer [1999]). Various methods can be used for visual depth estimation. However, the implementation and performance of such methods are influenced by the stereo camera setup. In computer vision for robotics two setups are used extensively: active stereo vision setup and static parallel-stereo setup. The active stereo vision setup relies upon performing fixations on scene entities (i.e. bringing the projection of a scene entity onto the center of images

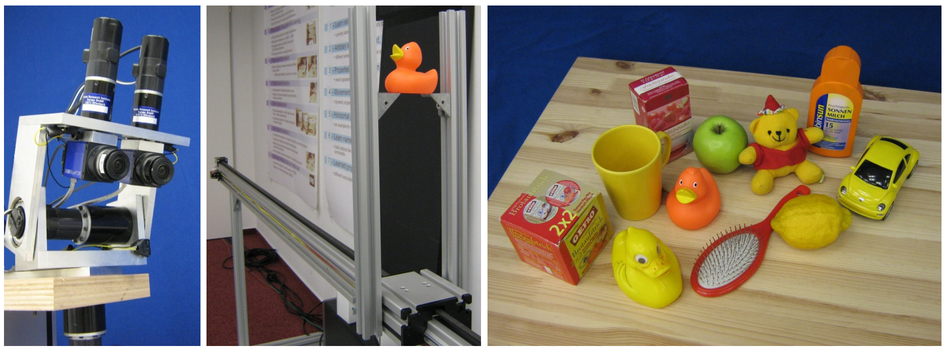

(a) (b) (c)

Figure 4.1: (a) The active vision head used in experiments. (b) The linear unit with a rubber duck object on its carrier platform (upper right corner of the image). (c) The objects used in the experiments.

of both cameras). Whereas, the static parallel-stereo setup positions two cameras parallel to each other with a fixed distance in between (referred to as baseline). A series of experiments was done to analyze the performance of three standard depth estimation methods in an active vision setup. This is done to show that the variation of the camera geometry in the active vision approach does not limit the use of different methods. Moreover, these methods have different characteristics over the range in which the distance is estimated. Analyzing these characteristics may be helpful to decide which method to choose depending on the situation and application. Additionally, the same experiments were conducted for one of these methods implemented in a parallel-stereo setup for comparison. Experiments involve estimating the distance of an object to the camera setup (Figure 4.1(a)) for a number of points on a linear unit (Figure 4.1(b)) for each method used in every setup. For statistical assessment experiments are repeated for 11 different objects shown in Figure 4.1(c) from the HRI150 database (Kirstein et al. [2008]). Results show that the use of active vision yields benefits over traditional static parallel stereo vision especially in the near visual field where accurate depth information supports physical interaction tasks such as grasping.

4.1.1 Depth Estimation Methods

Vergence

Vergence is a disconjugate camera movement where stereo fixation on a point (e.g. an object) in the scene is established. Depth estimation by vergence triangulation using a pinhole camera model is shown in Figure 4.2(a). The distance to the stereo fixation point can be derived from the vergence angle as:

$$z = \frac{b}{2 \cdot tan(\frac{\Theta_v}{2})}, \tag{4.1}$$

where Θ_v is the vergence angle and b is the baseline. The vergence angle is computed from the left and right camera angles as $\Theta_v = \Theta_{left} + \Theta_{right}$. In the experiments symmetric vergence ($|\Theta_{left}| = |\Theta_{right}|$) is applied. Since vergence triangulation implies different camera configurations, it can only be used in the active vision setup.

Stereo Disparity

Stereo disparity is the difference between the projection locations of a single point in the scene on the stereo camera images. Stereo disparity is a common depth cue used in artificial vision systems (Brown et al. [2003], Scharstein et al. [2001]). In active vision estimations, depth computed from stereo disparity is relative to the fixation point (Figure 4.2(c)). In order to obtain absolute depth information (i.e. the distance from the baseline to the stereo fixated object) an active rectification process proposed by Dankers et al. [2004] is used. The computation of depth in this framework is done as:

$$z = \frac{bf}{d} + r + f, \tag{4.2}$$

where d is the disparity (defined as $d = x_{VL} - x_{VR}$, x_{VL} and x_{VR} being the projections of the object on the virtual left and right image planes), f is the focal length of the cameras, r is the distance from the center of rotation of the cameras to the image planes. For estimating depth of an object, first disparity maps over the whole image are computed. Next, a simple color based segmentation process was used to distinguish the disparities corresponding to the object in the disparity maps. Finally, the average of these disparities was taken for depth estimation using Equation 4.2. Two different software packages are used for computing disparity maps: OpenCV version 2.0 (Bradski and Kaehler [2008]) and SRI Small Vision System (SVS) Version 4.4d (2007) (Konolige [1997]). Both software tools use the same block matching algorithm, however disparities are refined via post processing (sub-pixel interpolation and post-filtering) with the SVS. The disparity search range was 32 pixels (-16 to +15) for the active vision case and 96 pixels for the static-parallel stereo case. All other parameters were the same. The method is analyzed both in active vision (Figure 4.2(c)) and static-parallel stereo (Figure 4.2(d)) experiments.

Familiar Size

Familiar size estimates the depth of an object from the size of its projection on the camera images given that the real size of the object is known. Various approaches exist for this operation (e.g. Zhang et al. [2009]). Simple analytical relations derived from a pinhole camera model (Figure 4.2(b)) are used in the experiments. Object depth can be computed as:

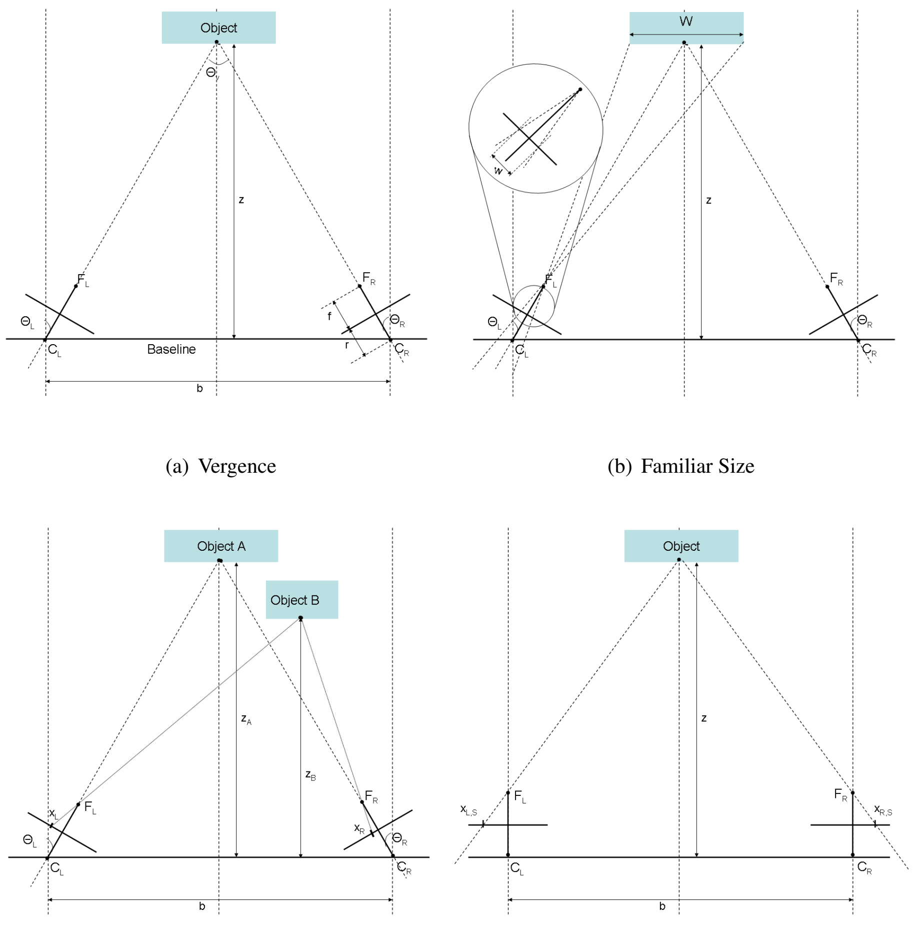
$$z = \left(\frac{fW}{w} + r + f\right) cos(\Theta), \tag{4.3}$$

where Θ is the camera angle and $cos(\Theta) \approx 1$ due to small baseline. The physical size W was measured beforehand for all objects used in the experiments, the size on image w was computed using a simple color based segmentation process (the same used with the stereo disparity method). The width of the objects was used for estimations since it showed better overall accuracy than the height.

4.1.2 Results

The depth estimation results from the three methods used in the active vision setup are shown in Figure 4.3 as mean estimation errors[1] of the methods. The error is defined as

[1]The terms mean estimated depth and mean estimation error imply given information is averaged over all 10 objects.



(a) Vergence

(b) Familiar Size

(c) Stereo disparity in active vision

(d) Stereo disparity in static parallel-stereo

Figure 4.2: Analytical models of the compared depth estimation methods. Depth z is defined by the distance from the baseline to the object. F_L and F_R denote focal points, C_L and C_R denote center of rotations of the left and right cameras, Θ_L and Θ_R are left and right camera angles, w and W are the size of the object on the image and in real, x_L and x_R the projection of the point on the object on the left and right camera images in active vision setup, $x_{L,S}$ and $x_{R,S}$ projection of the point on the object on the left and right camera images in static parallel-stereo setup respectively. The static parameters of the system are as follows: r = 18.75 mm, f = 5.4 mm, b = 65 mm.

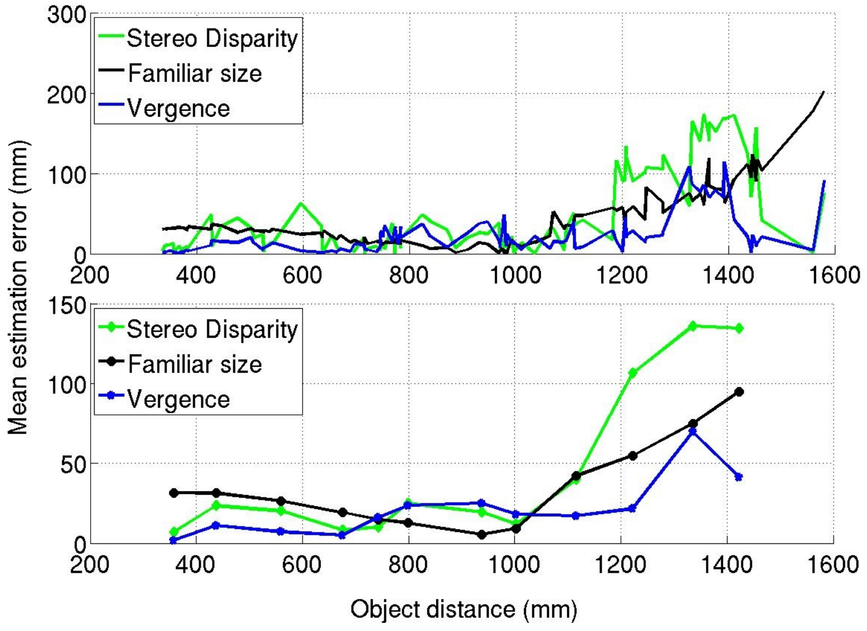


Figure 4.3: Mean estimation errors of the methods. Plots above show pure data; plots below show running average of the data over a window size of eight data points.

the absolute value of the difference between the mean estimated depth and actual depth. The accuracies of the methods were also examined in different ranges. The close, middle and far ranges were defined as 300-700 mm, 700-1100 mm and 1100-1500 mm respectively. These distinctions are also relevant for humanoid applications such as reaching tasks where object detection typically occurs in the far range, approaching takes place in the middle range and grasping is executed in the close range. Table 4.1 shows the mean and standard deviation of the average errors over these ranges. Inaccuracy in the measurement of the vergence angle, which is mainly caused by measurement noise and low resolution of the vergence motors, is crucial for the vergence method, especially in the far ranges. It is expected from Equation 4.1 that even small inaccuracies cause high errors for small vergence angles. This has a high impact on estimation of distant targets. Higher resolution motor encoders may reduce this error. Increasing the baseline may also increase the reliable estimation range. However, a large baseline may result in different views of the object on left and right camera images. This will introduce difficulties in the matching process. Since the active rectification process relies on the vergence angle, the SD method is also affected by inaccurate vergence angles. The accuracy of FS depends primarily on the object segmentation. This may explain the estimation errors in the experiments. Finally, a significant source of error for all methods is due to pixel quantization. Using higher resolution images may reduce this effect. However, trends are expected to stay the same.

Figure 4.4 shows results of the comparison of stereo disparity calculation done for the active vision case and the static-parallel stereo case. Mean estimation errors of each of the two stereo algorithms used in the static-parallel stereo case is shown in Table 4.1. The results showed that all three methods (Vergence, SD and FS) compatible with active vision outperformed the methods using static-parallel stereo setup in the near and

Table 4.1: Mean (and standard deviation) of estimation errors (in mm) for all objects at different ranges.

Methods	Near	Middle	Far
Active vision setup			
Vergence	6.37 (5.51)	20.75 (11.79)	38.85 (32.92)
FS	27.22 (6.12)	11.82 (9.46)	76.68 (38.47)
SD (OpenCV)	14.91 (15.32)	16.78 (12.84)	100.92 (51.56)
Static-parallel stereo setup			
SD (OpenCV)	207.89 (45.73)	38.29 (34.63)	97.14 (75.53)
SD (SVS)	137.35 (81.05)	20.56 (10.42)	25.1 (9.49)

middle ranges. In the experiments for the static-parallel stereo case, a systematic underestimation in disparities (i.e. overestimations in depth) in the near and middle ranges for OpenCV block matching algorithm is noticeable (Figure 4.4). This is not present in the SVS even though both algorithms use the same block matching method. This is likely due to the fact that the SVS applies post-processing on disparity calculation. Lack of this post-processing in the OpenCV algorithm is likely the cause of this systematic underestimation because for a close object situated at a large disparity, the propensity for search algorithms (performing the search starting from zero) to mismatch within a large disparity search range increases. This biases the disparity search towards disparity underestimation when the true object disparity is large. On the other hand, the estimations with the same OpenCV block matching algorithm using the active vision case did not show any systematic overestimations despite no post-processing was applied. The reason for this is likely that the disparity search is done in a smaller range around zero (i.e. the fixation point) in the active vision case, reducing the bias and evenly spreading it in positive and negative search directions.

The stereo disparity search in the static-parallel stereo camera configuration involves large disparities. Setting the disparity search range, which determines the magnitude of disparities that can be detected, too low impairs depth estimation in close ranges. High errors of the SVS in the near range are due to this phenomenon (Table 4.1, Figure 4.4). Removing erroneously labeled disparity values in disparity maps computed via the SVS reduced the mean estimation error to 20.28 mm in the close range. The search range can be increased at the expense of computation time, since increasing the search range means higher number of correlations to compute. This outlines a trade-off between the disparity range and computational resources. Active vision eliminates this trade-off by fixating on the object of interest and constraining the disparity search around zero. A virtual tracking system may solve this trade-off problem for the static-parallel stereo case only if the object is in the field of view of both cameras.

Another benefit of conducting a disparity search around zero for a given search range, as an active vision approach offers, is having better quality of depth information for a specific scene volume than the static-parallel stereo disparity search provides, especially in the near range. This may be especially beneficial in humanoid grasping tasks. A representative example is shown in Figures 4.5, 4.6. Disparities calculated

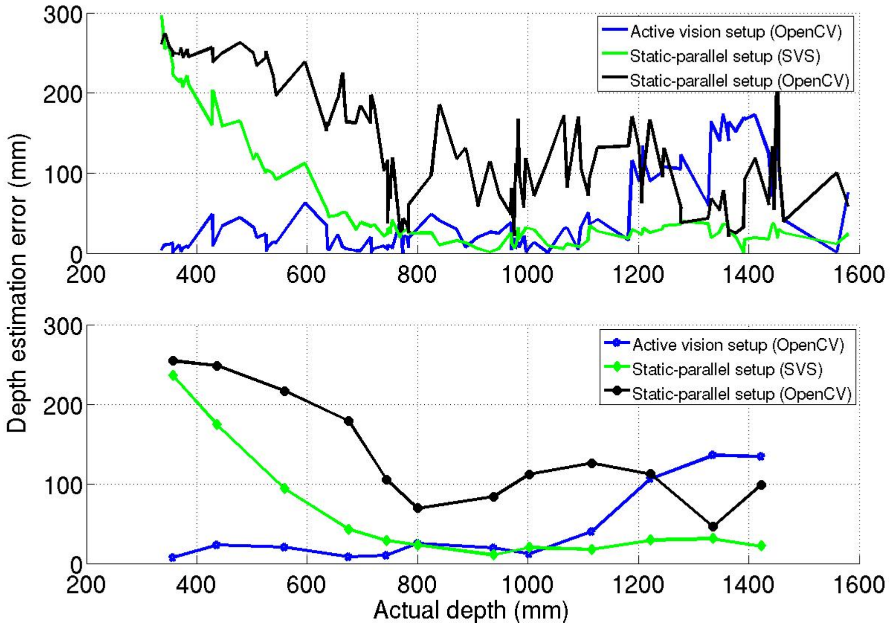


Figure 4.4: (a) Mean estimation results of stereo disparity algorithms with active and static-parallel cases. (b) Mean estimation errors of stereo disparity algorithms with active and static-parallel cases. Plots above show pure data, plots below show running average of the data over a window size of eight data points.

with the active vision setup are distributed around zero more smoothly, implying more continuous and detailed depth transitions across the object (Figure 4.6(a)). Disparity values calculated with the static-parallel stereo setup are more discontinuously clustered and show artefacts in the representation of depth across the object (Figure 4.6(b)).

The results show that the depth estimation cues inspected in this study demonstrate varying characteristics in different ranges. Deciding on using which method under which circumstances may be difficult in complex scenarios. One way to address this problem is computing an efficient integration of the methods that will deliver accurate information in all cases. In Karaoguz et al. [2011b] a reward mediated learning framework is presented where optimal integration of these cues are learned. An alternative approach that also goes in a similar direction with the idea proposed in this thesis is arbitrating methods depending on the circumstances during the execution of the scenario. The following section explores this approach in a multi-task scenario where arbitration of gaze movements for the tasks is necessary.

4.2 System Architecture for Active Vision

In the previous section it was shown that a depth estimation framework may use various mechanisms to measure distance to an object and these mechanisms have varying characteristics in the task space (e.g. close, middle and far ranges). In general, utilizing camera movements among multiple mechanisms of a framework depending on the task may lead to complex trade-offs. For example in a grasping scenario for a humanoid robot, different methods of depth estimation is useful in the detection phase

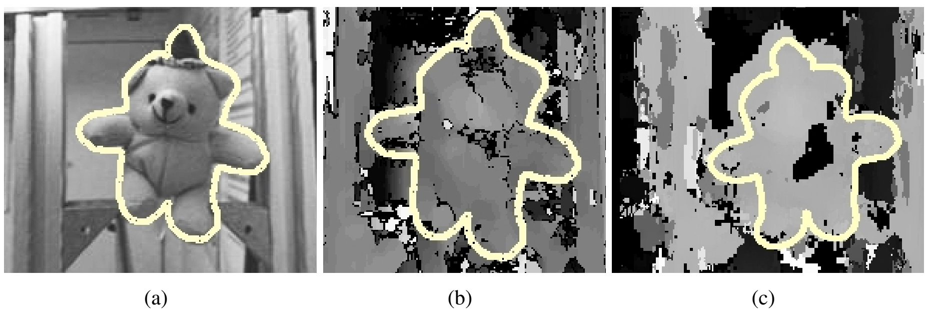


Figure 4.5: Disparity maps computed for an object at 340 mm distance. (a) Left rectified camera image from the static-parallel stereo setup (b) Disparity map computed with active stereo setup (c) Disparity map computed with static-parallel stereo setup. The images are cropped to show only relevant information. The object boundaries were highlighted in all three images. Object appearances differ in images from the two setups due to different viewing angles.

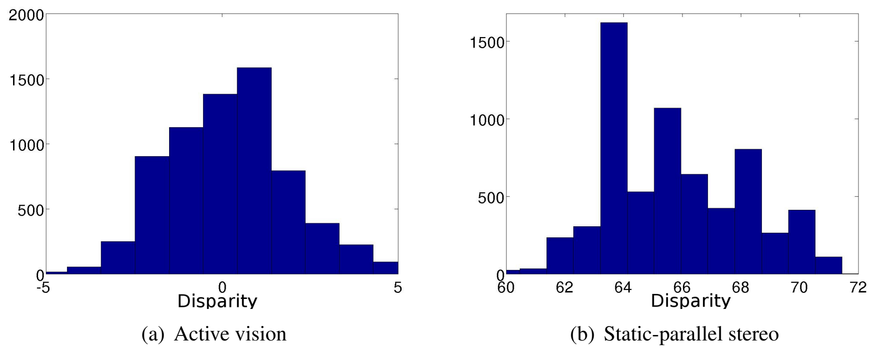


Figure 4.6: Histograms of the disparity values across the object from the disparity maps shown in Figure 4.5. Outliers are left out and only the relevant disparity ranges are displayed.

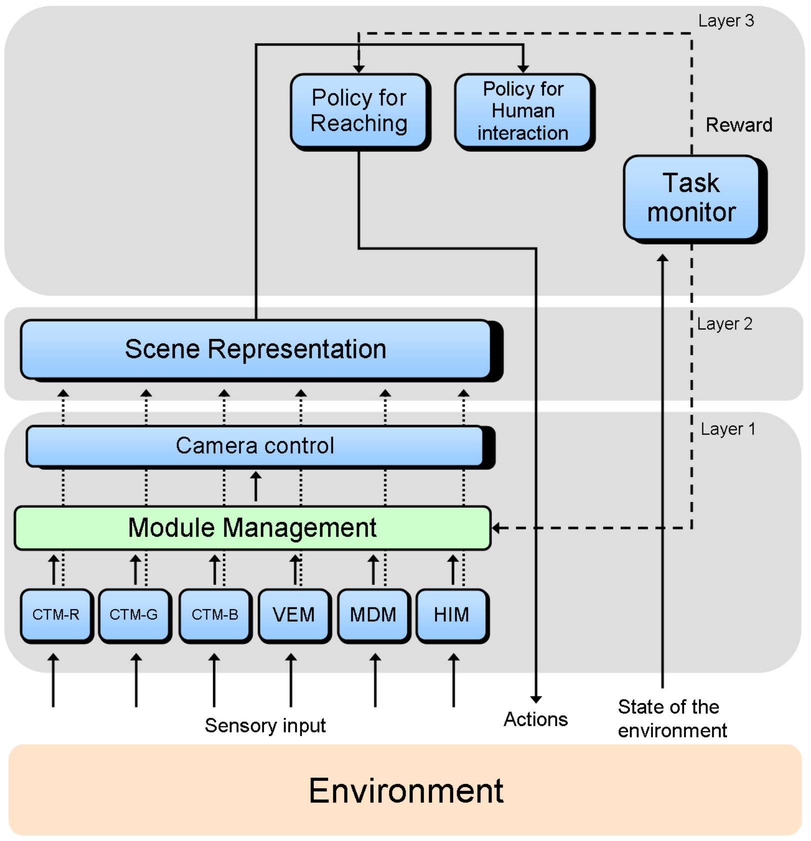


Figure 4.7: The system implemented for the active vision scenario. The abbreviations in the figure are: CTM-R, CTM-G and CTM-B are Color Tracking Module for red, green and blue respectively, VEM is Visual Exploration Module, MDM is Motion Detection Module and HIM is Human Interaction Module.

(where accuracy in the far range is required), reaching phase (where accuracy in the middle range is required) and grasping phase (where accuracy in the close range is required). Hence, an efficient way of arbitration is necessary for mechanisms generating diverse gaze movement commands during the execution of tasks. This section investigates how the proposed framework can be applied to learn this process to support multiple tasks running in parallel. For this purpose, the following multi-tasking scenario is designed: the iCub humanoid should select, reach and touch one of the three objects moving on a table while keeping the interaction with a partner present in the scene. The system implemented for the active vision scenario is a specific instance of the architecture explained in Section 2.2. The general functional description of this system is shown in Figure 4.7. In essence, modules perform elementary operations for information gathering and pre-processing and may generate a motor command for gaze orientation depending on the result of the processing. The mechanism controlling the camera actuators forms the constrained resource of the system: only one motor command can be executed at a time for gaze shifts. The MML makes module selections to arbitrate between motor commands in time. A scene representation mechanism keeps temporally stable information about the environment. Since only the information about entities at which the system is looking can be updated, the strategy employed by the

MML for gaze shifts directly affects scene representation. The information kept in the scene representation is used by higher level policies that map environmental states to actions to execute tasks. The task monitor observes the states of the system and environment and generates rewards depending on the outcomes of actions. Below, detailed information about the fundamental building blocks of the system is given.

4.2.1 Perceptual Modules

Different modules operating on different domains of modality can be plugged into the proposed framework. However, due to difficulties of reproducing stimuli other than visual ones in the simulation environment (e.g. audio signals) only visual processing modules were implemented. Methods that these modules apply are standard. State of the art approaches can be used for better performance. However, the implementation of individual modules is not the focus of this work. A module is allowed to generate a single motor command at a time, any ambiguity in data (e.g. two similar objects in the scene) is assumed to be handled by the module itself. A snapshot of the system with module outputs is shown in Figure 4.8. The modules used in the system are described in the following sections.

Color Tracking Module (CTM)

CTM extracts a specific color feature from images and generates gaze commands to keep this feature in the center of the image. Three CTMs are implemented for red, green and blue and indicated as CTM-R, CTM-G and CTM-B, respectively. At every time step an image $I^t(x, y)$ is acquired where t denotes the time step and (x, y) is the horizontal and vertical pixel positions. Color channels $I^t_c(x, y)$ where $c \in \{R, G, B\}$ are extracted and sent to the corresponding CTM. The module computes conspicuity maps using the color opponency method explained in Itti and Koch [2001]:

$$C^t_c(x, y) = \sum_{c'} I^t_c(x, y) - (\eta I^t_{c'}(x, y) + \vartheta), \tag{4.4}$$

where C^t_c is the resulting conspicuity map for color c, c' denotes the colors other than c, η and ϑ are threshold parameters defining the sensitivity to the difference between the color channels. The resulting conspicuity map indicates the positions of the detected color features the CTM is assigned to. For the image shown in Figure 4.8(b) conspicuity maps computed by CTM-R, CTM-G, CTM-B are shown in Figures 4.8(c), 4.8(d) and 4.8(e), respectively. If the color feature is detected, a motor command for a gaze-shift towards its position is generated. The exact target for the motor command is the center of mass of the positions where the feature is detected. The objects in the environment are unique in color. Thus, no arbitration between multiple objects of the same color is necessary. Conversion from image coordinates to the eye-motor coordinates is done using the kinematics of the robot head (Beira et al. [2006]).

Motion Detection Module (MDM)

MDM detects motion in the current field of view by computing a simple temporal difference between acquired images as:

$$M^t(x, y) = I^t(x, y) - I^{t-1}(x, y), \tag{4.5}$$

where M^t is the motion map in the image coordinates at time step t, I^t and I^{t-1} are the images taken at time steps t and $t-1$, respectively. Since this method is susceptible to ego-motion, the module is implemented in a way that it does not process any information whenever the cameras are moving (i.e. during gaze-shifts). In the case shown in Figure 4.8(b) where the only moving object was the blue cube, the resulting motion map is shown in Figure 4.8(f). If a motion is detected in the image, MDM generates a gaze-shift command to bring the region where motion is detected (i.e. center of mass of the pixels with motion) to the center of the image. As in CTM, conversion from image coordinates to the eye-motor coordinates is done using the kinematics of the robot head. Since all three objects can move (depending on the parameters of the experimental setup), it is possible that motion from multiple objects is detected in one time step. In such cases the resulting motor command will be towards the centroid of all detected motion. Such an implementation aims to keep all moving objects in the field of view.

Visual Exploration Module (VEM)

VEM keeps track of locations, which were not gazed at for a long time and proposes gaze-shift towards these areas. To achieve this an exploration map E^t covering the whole head motor space in two dimensions (i.e. pan and tilt) is updated at every time step t in the following way:

$$E^t(\theta) \leftarrow P^t_{inh}(\theta)(E^{t-1}(\theta) + \xi), \tag{4.6}$$

where $\theta \in \Re^2$ denotes the camera pan and tilt positions in the whole head-centered reference frame. This implies an increment of all values by an amount given by parameter ξ and an inhibition of the values around the gaze position in the current time step θ^t_{cur} done by:

$$P^t_{inh}(\theta) = 1 - G(\theta; \theta^t_{cur}, \Sigma_{inh}), \tag{4.7}$$

where $G(x; x_0, \Sigma)$ is an n dimensional Gaussian function defined in the space $x \in \Re^n$ parametrized by mean $x_0 \in \Re^n$ and covariance matrix $\Sigma \in \Re^{n \times n}$. In the application $n = 2$ (i.e. pan and tilt dimensions) and the covariance matrix is in the form of $\Sigma_{inh} = \left(\begin{smallmatrix} \sigma_{inh} & 0 \\ 0 & \sigma_{inh} \end{smallmatrix}\right)$. The exploration map is normalized as it sums to one to be used as a probability distribution. At every time step, a single motor command is sampled randomly using the resulting distribution.

Human Interaction Module (HIM)

HIM generates motor commands to gaze at humans whenever they are detected. Different methods can be used to detect humans in the scene. For instance face detection and sound localization are two operations that can achieve this with different cues. However, due to difficulties in generating both auditory (e.g. voice) and detailed visual (e.g. face) elements in the simulation environment, such cues could not be used. Instead, the assumed position of the interaction partner is given directly to the module. HIM generates motor commands for gaze-shifts towards this position at every time step.

4.2.2 Scene Representation

A scene representation mechanism can keep and integrate various properties of elements in the scene over time. The scene representation is achieved by implementing

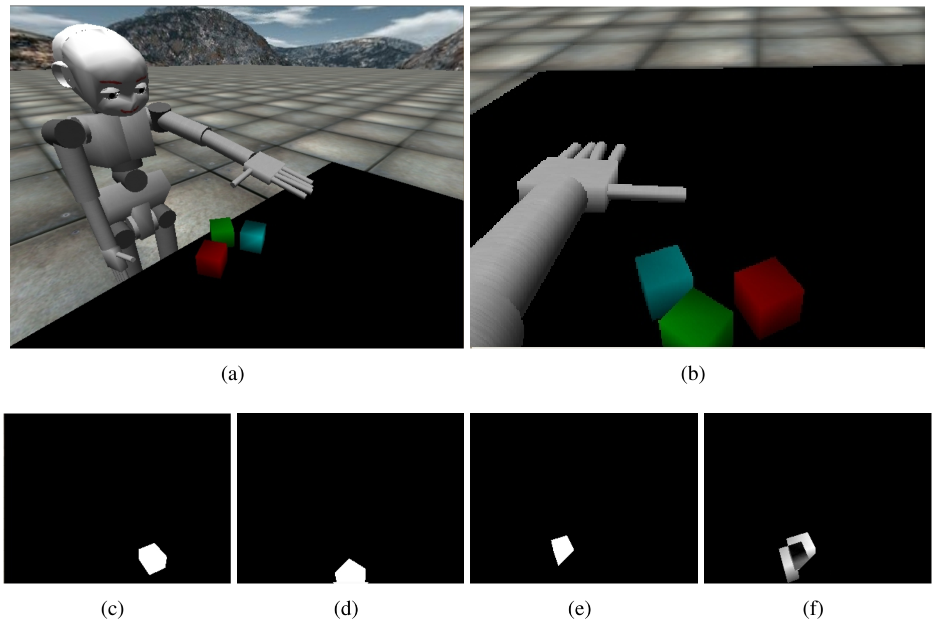


Figure 4.8: A snapshot of the system during a scenario trial. (a) An external view of the robot and the environment from the simulator, (b) view from the left camera, the color of the objects are blue, green and red from left to right respectively, (c)-(e) color conspicuity maps for red, green and blue colors, (f) result of motion detection computation, only the blue object was in motion at this instance.

a simple working memory (WM). Scene elements' properties that are kept in the WM are the objects on the table (red, green and blue) and the interaction partner. The following properties of those elements are stored in the WM: an id for each element id, the estimated position of the element in the world p_{id} and the age of the entity in the WM a_{id}. The storage and update of properties are done as explained in Section 2.2.2. CTMs (R,G and B) and the HIM module are used to distinguish the scene elements for the WM. For example if the CTM-R reports that the red object is at the center of the field of view, the id of that element is used for storing or update.

4.2.3 Tasks and Policies

The scenario designed for the system consists of two tasks that have to be executed in parallel: a reaching task and a human interaction task. The way the system implements a solution for a task is defined as a policy. A policy is roughly a mapping between the information the system has about the state of the environment and actions the system can take. There are no constraints on the implementation of policies. Such mappings can be defined manually or learned through interaction. Both types of policies are implemented and tested for the reaching task in this work. This enables the evaluation on how the proposed methods scale with the complexity of the system. It is assumed that replacing a manually coded policy with a learning mechanism increases the complexity of the system since it has to learn further aspects in the scenario in addition to the MML.

Reaching Task

In this task the system has to reach one of the three objects on the table. The objects are colored in red, green and blue and they yield different amounts of base rewards upon reaching. These are set to different values for every experiment setup. Successfully reaching an object yields the base reward associated to that object. Additionally, the objects are moving on the table. The probability of an object to make a movement in a time step is defined beforehand. This parameter is different for each object, and also set to different values in each experiment setup. This makes the differently colored objects more or less attractive as grasping targets, depending on the relative amount of reward they provide and the difficulty of catching them.

Two different policies are implemented for the reaching task: *Programmed Policy* and *Reinforcement Learning Policy*. In *Programmed Policy* (PP) the object selection is done randomly: At every time step the system samples one of the three objects randomly from a uniform distribution and starts/continues the reaching process with whatever information it has about the object in its working memory. In *Reinforcement Learning Policy* (RLP) case the policy for deciding which object to be selected for reaching is learned via the Q-Learning algorithm (see Section 3.1.2 for details). Here an action-value function $Q(s, a)$ that gives the expected utility of taking a given action a in a given state s is learned. Learning is done on-line using rewards r acquired from the scenario outcomes. The state space is formed by the following five entities: distance of objects (red, green and blue) to the robot, distance of the partner to the robot, position of the hand. Information about these entities is gathered from the WM. Since the distance/position information are real valued variables they have to be discretized to be used as a representation in the state space of the Q-Learning algorithm. This is done in the following way:

$$s = int\Big(s_{min} + \frac{(z - z_{min})(s_{max} - s_{min})}{z_{max} - z_{min}}\Big), \tag{4.8}$$

where s is the encoded value, z is the value to be encoded (e.g. distance of the red object to the robot), $z_{min} = 0.1$ and $z_{max} = 0.5$ are the minimum and maximum distances that can occur in the application respectively and $s_{min} = 1$ and $s_{max} = 5$ are the minimum and maximum values for the state encoding respectively. The state value $s = 0$ is used for the case where no information is available about the entity. The action space of the policy determines which object is to be reached. Therefore, the number of actions M is equal to the number of objects which is three in this work. Softmax action selection is used for this purpose as explained in Equation 3.12. At every time step a new selection is done. Note that any noise that would emerge from uncertainties in the scene representation affects the RL algorithm.

Once the object is selected, the reaching process is realized by minimizing the distance between the position of the left hand and perceived object position (i.e. from the scene representation). If no information is available in the scene representation yet, no action for reaching is taken. The reaching task is fulfilled if this distance is below the minimum reaching error e_{reach}. The hand is moved in three dimensions in space using velocity control. Velocity commands are generated/updated in a ballistic fashion (i.e. no feedback control is used) using the errors between the hand position and the perceived object position. For hand movement, two degrees of freedom from the arm group of joints and one degree of freedom from the waist group of joints are used. Although blocking poses are avoided, collisions with the table may occur if the position of the reached object assumed by the system is different from the actual position of the object. In the case of a collision, the hand is moved back to a safe location and the reaching process continues. If the system fails to update the position information after the collision, the target position could be the same and a further collision may occur. This is not bad for the system as it provides data for the learning process as well, since collisions delay the reaching process which decreases the amount of reward acquired at the end.

Human Interaction Task

For this task the system has to stay in interaction with a human partner present in the environment. The partner is modeled with a single variable called boredom. This variable is updated at every time step: if the partner is gazed at by the robot the boredom variable is set to zero; otherwise, it is increased by a value $0 < \tau_b < 1$ called the boredom rate. If the boredom variable exceeds the maximum value of 1 the partner is bored and the task is unsuccessful. Different values were generated for the boredom rate for each experimental setup.

4.2.4 Task Evaluation and Rewards

A monitoring mechanism observes all ongoing tasks and generates rewards for the system depending on the outcome of the scenario. A scenario is unsuccessful if any of the ongoing tasks failed. For example if the human interaction task fails due to boredom of the partner, the scenario is unsuccessful yielding a negative reward. There is also a time limit t_{max} in which the tasks have to be fulfilled. If this is exceeded the scenario is also unsuccessful. If all tasks of a scenario are achieved at time step t where $t < t_{max}$, the scenario concludes successfully and a positive reward r is generated as:

$$r = r_b + \frac{t_{max} - t}{t_{max}}, \tag{4.9}$$

where r_b is the base reward appointed to the object. This gives a time bonus to solutions yielding shorter task run time. If the object is reached without the partner being bored and time-out not being reached, the scenario outcome is a success.

4.3 Outline of Experiments

For comparison, a number of additional gaze control methods or information acquisition settings were implemented:

1. *Module Management Learning (MML)* is the proposed method.
2. *Saliency* is a simple implementation of the bottom-up attention framework presented by Itti and Koch [2001]. For this method a saliency map covering the whole camera motor space is generated. The same features that are computed by the modules in the MML case are used: color (red, green and blue), motion, visual exploration cue, and human interaction cue. Extraction of these cues are done in

Table 4.2: Parameter values used in the active vision experiments.

General		**MML**	
Image size	160×120	α_b	0.05
τ_b	0.2	γ	1.0
t_{max}	25	t_{hist}	10
e_{reach}	0.12	β_b	0.05
		α_s	3.0
Modules		β_s	3.0
η	1.2	ϵ_s	100
ϑ	10		
ξ	0.01	**Q-Learning**	
σ_{inh}	2	α	0.01
		γ	0.99
WM		Z	0.05
a_{max}	20		

the same way as in the modules. At every time step, motor commands from modules are represented in the map as peaks of activity (set to 1.0) at corresponding positions. An additional map from an inhibition of return procedure is used to keep track of previously visited locations and suppress any activity in these positions of the saliency map. After the inhibition of return the most conspicuous location in the resulting saliency map was selected as the motor command for a gaze-shift if its value exceeds a previously defined threshold. Inhibitions on the map weaken in every time step.

3. *Task based* gazing strategy uses a manual definition of task-module relations to take gazing commands in a top-down manner. Since there are two tasks to support they have to be arbitrated. This is done by randomly choosing one task to support from a uniform distribution (i.e. 50% chance to select each) at every time step. If the reaching task is selected the color of the object with the highest value is determined from the current state of the Q-function and the motor command from the corresponding CTM is executed for gazing. If the human interaction task is selected the motor command from the HIM is executed. This method mimics the top-down task control over a saliency mechanism (for example Navalpakkam and Itti [2005]) without the use of bottom-up features.

4. *Random* is a gazing strategy where at every time step a random module is selected and the gaze motor command from the selected module is executed.

5. *Perfect World Knowledge (PWK)* is the setting where the system is given the precise information about the position of objects at every time step. This means the system can perform the reaching task blindly without employing any gaze on the objects. This method was implemented for RLP to determine how well the policy learning mechanism can do in the absence of sensory uncertainties. Therefore, it can be a reference to asses the performance of the other methods. The gaze is directed to the interaction partner in this setting, so that the second task is always performed successfully.

For statistical assessment 20 different experimental setups were generated and all methods were tested in each of these setups. To achieve these experimental setups a

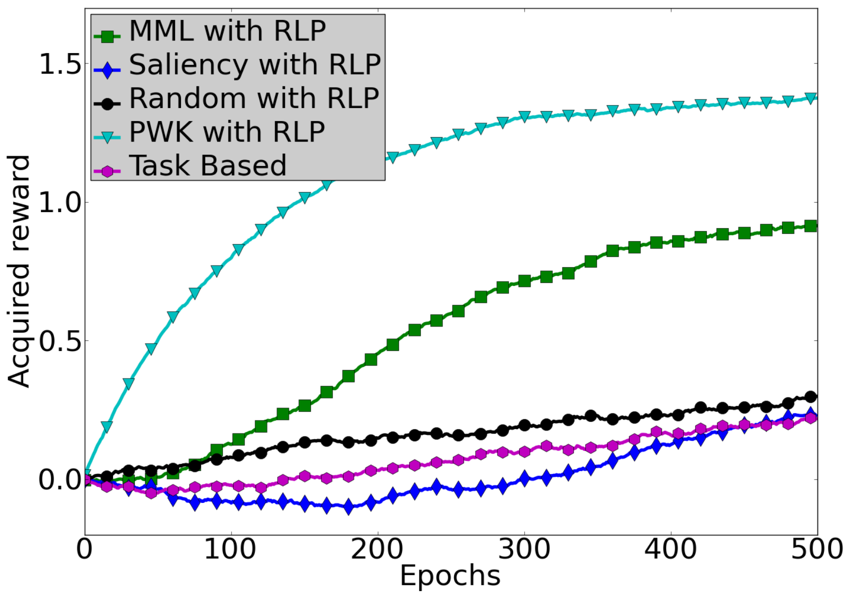


Figure 4.9: Acquired rewards for four different methods used in RLP. The data for the learning method is averaged over the last 100 epochs.

number of environmental variables are defined:

- One object is selected randomly and its base reward r_b was set to 1. Base rewards of the other objects are sampled randomly between 0 and 0.99 from a uniform distribution.

- The probability of an object to make a movement in a time step is sampled randomly between 0 and 1 from a uniform distribution for each object. In case of a movement, the movement vectors of the object in two dimensions (lateral and depth wrt the robot) are sampled randomly between 0 and 0.05 from a uniform distribution.

- The boredom rate τ_b of the interaction partner was set to a random value between 0.1 and 0.3 that is sampled from a Gaussian distribution ($\mu = 0.18$).

- The maximum hand velocities are sampled randomly between 20-40 deg/sec^2 and 10-20 deg/sec^2 from a uniform distribution for X and Y axes respectively.

This allowed us to evaluate and compare the performance of the methods in different environmental conditions. The values of parameters used in the experiments are shown in Table 4.2. Parameters for the MML and Q-Learning are determined empirically with best effort. Every module has different utility and their operations are independent from each other. For this reason, module parameters are selected in the way that each module delivers the maximum utility. Other parameters are selected with regard to computational efficiency (e.g. image size) and scenario execution time.

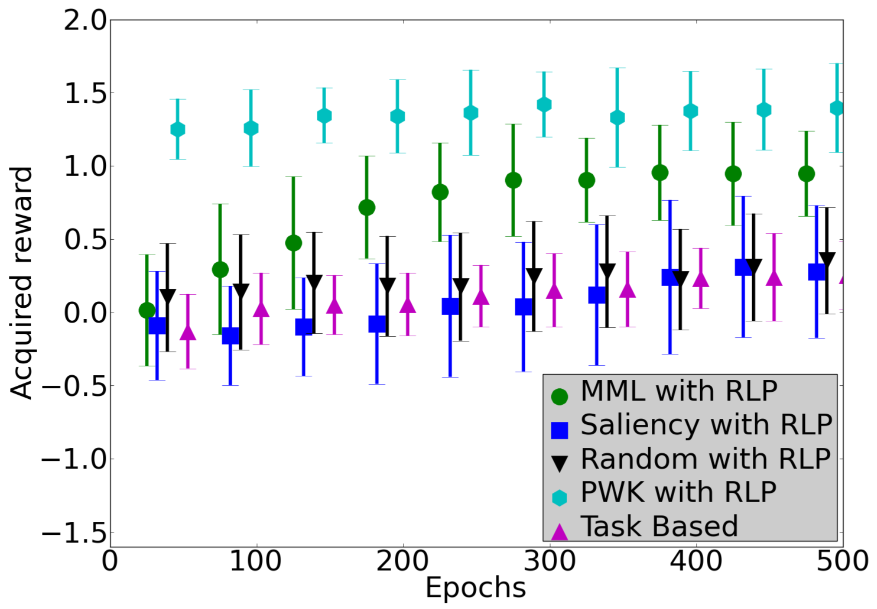


Figure 4.10: Mean and standard deviations of acquired rewards for every 50 epochs over all experiments for four different methods used in RLP. Every data point represents means of acquired rewards in the preceding 50 epochs. Data points of different methods are shifted slightly away from each other in the horizontal axis for better visualization.

4.4 Results and Discussion

4.4.1 Overall System Performance

An assessment of the performance of the overall system can be made by observing the acquired rewards in time. This can give an idea on how different information acquisition methods can support the multi-task scenario. Figure 4.9 shows average acquired rewards for four methods. Rewards are averaged over all experiments for each method and smoothed using Equation 3.15 with $\epsilon_s = 100$. Similarly Figure 4.10 shows means and standard deviations of acquired rewards for every 50 epochs over all experiments. The plots show that the performance of MML approaches the reference PWK in terms of acquired rewards as the learning takes place. Compared to MML, Saliency, Random and Task based gazing strategies were not efficient for acquiring information in the current scenario. The increase in the acquired reward over time with these methods comes from the learning process of the reaching task since no further mechanism (such as MML) exists to learn the human interaction task. However, the support for this learning process was poor.

Another metric that can be used to evaluate the performance of the methods is the task statistics that is shown in Table 4.3. The data in the table indicates how many times on average did the scenario end with the corresponding outcome shown in the first column for the given epoch interval. Two epoch intervals are shown: between epochs 0 and 100 (i.e. before learning or BL) and between epochs 400 and 500 (i.e. after learning or AL). The data is averaged over all experiments. With learning, the proposed method (shown as MML with RLP) increased its success rate from 49% to

Table 4.3: Task statistics averaged over all experiments (in %). Before learning (BL) covers epochs between 0-100, after learning (AL) covers epochs 400-500.

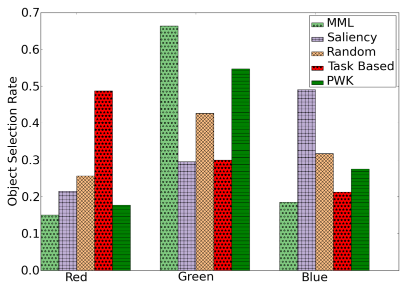
Scenario outcome	Saliency	Random	PWK	MML with RLP	MML with PP	Task Based
Success (BL)	31.3	46.1	91.8	48.6	44.3	41.8
Success (AL)	45.4	53.8	93.4	81.2	67.5	52.1
Time-out (BL)	1.1	11.8	8.1	20.8	20.4	44.2
Time-out (AL)	0.6	9.4	6.5	14.9	25.6	35.5
Bored (BL)	67.6	42.0	0.0	30.4	35.2	13.9
Bored (AL)	53.9	36.6	0.0	3.7	6.7	12.2

81%. Compared to the reference (PWK), MML reached 87% of the performance of the case where all information about the objects was available to the system all the time. The reductions in the negative outcomes (i.e. time-out and bored) are also indicators of the learning process. The significant decrease in the cases where the interaction partner is bored clearly shows that MML learned to cope with the human interaction task as well. Success rates are lower for the methods Saliency (45%), Random (54%) and Task based (56%). Compared to their values before learning, there is slight improvement in the performance of these methods. As stated before, this is due to the Q-Learning algorithm: as the policy for reaching is improved via learning, the system succeeded to grasp objects before the human interaction task fails in some cases. This indicates that those methods can still provide information for the reaching task up to some degree. However, since there is no inherent support for the human interaction task the system could not cope with multi-tasking. A higher performance may be expected from the task based strategy since it imposes to get information from relevant targets, even though the relevance is defined manually. This is due to the lack of a reactive behavior of the system in the information acquisition that introduces exploration in the learning process. In order to explain this, it can be argued that the modules that are not selected in the course of learning, such as VEM or MDM, actually provide the system with explorative behavior. However, it is observed (and also can be seen in two representative experiments shown in Section 4.4.2) that the MML does not rely on these modules as well. This means that the weighting and inhibition processes in the MML mechanism provide the system with the explorative information acquisition behavior.

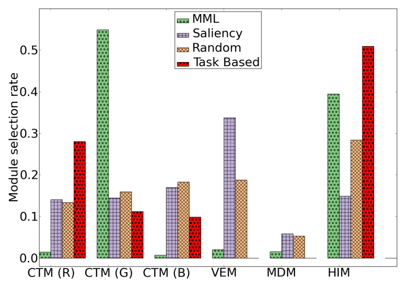
The proposed mechanism modifies a neural network-like structure governing the attention shift process. However, unlike conventional RL algorithms (Sutton and Barto [1998]) this structure is not a direct mapping of states to actions. It rather performs classification of modules and their allocated time. This property makes the MML a quick learner. Hence, the implementation of MML reaches its top performance in about 400 epochs whereas common RL algorithms that learn state-action mappings conclude the learning process in the order of 1000s of epochs.

4.4.2 Support for the Reaching Task

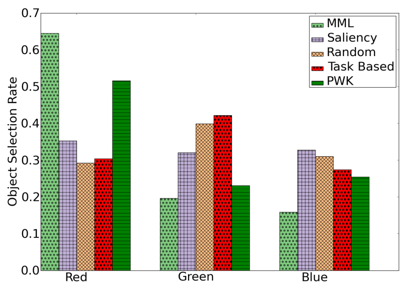
A convenient way of analyzing the support of the compared methods to the reaching task could be to evaluate the learned task policies under the influence of each method. However, due to the influence of varying environmental parameters to the scenario it is difficult to derive a baseline policy and compare the learned ones with that. The optimum policy is usually not as straightforward as ’select the object yielding the maximum reward’ and differs from one experiment setting to another. For example, since objects have different dynamic properties it can be more rewarding in the long term to select a less valuable but also less dynamic object. However, this also depends on the hand velocity of the robot and the boredom rate. Therefore, a quantitative assessment of the system behavior can only be made by evaluating the experiments individually.



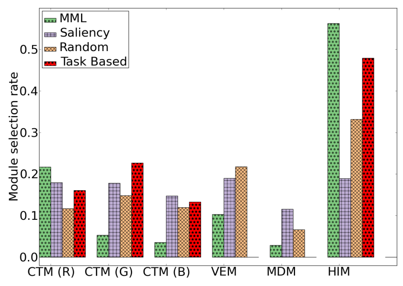
(a) Experiment 13: Object selection rates



(b) Experiment 13: Module selection rates



(c) Experiment 16: Object selection rates



(d) Experiment 16: Module selection rates

Figure 4.11: Module selection rates and object reaching rates, averaged over the last 100 epochs (i.e. after learning) in two experiments. See text for experiment settings.

Figure 4.11 shows results from two representative experiments. The settings of experiment no. 13 are as follows: red, green and blue objects yielded rewards of 0.1, 1 and 0.18 and had movement probabilities of 0.2, 0 and 0.2 respectively, boredom rate is 0.26 and max. hand velocity is 15.4, 39.35 deg/sec. in x and y dimensions respectively. The green object is the best candidate to be selected in this example since it was the most rewarding object and static in these settings. When used with MML the Q-Learning algorithm learned a policy where the green object was the most frequently selected (Figure 4.11(a)). This is also in accordance with the reference PWK case. The gazing strategy learned by MML is also consistent with the learned object selection policy: among the three color tracking modules, the one appointed to green was selected dominantly (Figure 4.11(b)).

The settings of experiment no. 16 are as follows: base rewards are 0.74, 1 and 0.46, movement probabilities are 0.25, 0.44 and 0.84 for red, green and blue objects respectively, boredom rate is 0.15 and max. hand velocity is 17.39, 21.69 deg/sec. in x and y dimensions respectively. In this experiment the system with MML most often selected the red object that has the second highest value after the green object

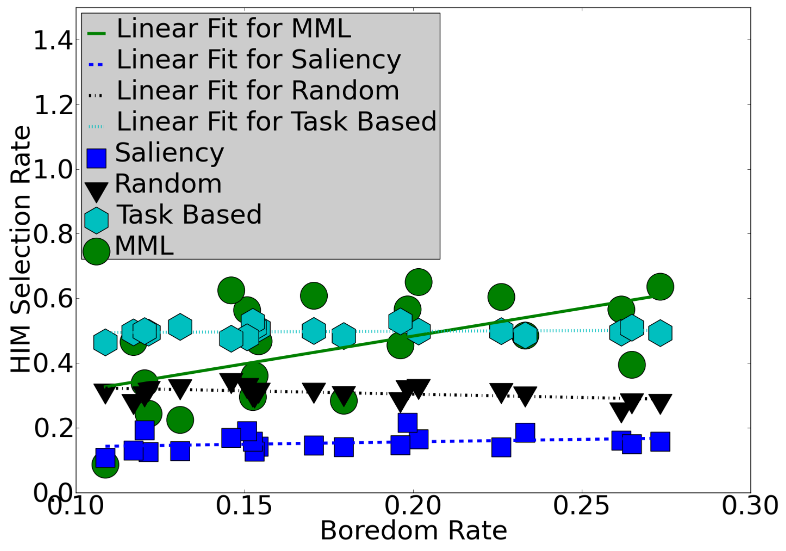


Figure 4.12: Selection rate of HIM with respect to changing boredom rate in 20 experiments.

(Figure 4.11(c)). The movement probability of the red object was lower than the green object. Hence, the system aimed for a less rewarding but more stable object. The gazing strategy learned by MML selected CTM-R more often than any other color tracking module (Figure 4.11(d)). This is because the red object was still dynamic and needed to be updated often for an efficient information acquisition to maximize the acquired reward.

The object selection policies learned by the systems using the methods Saliency, Task based and Random differ from the policies learned in MML case in both experiments. This signifies that the information acquisition policy has an effect on the learning process of the policy. The learned object selection policies supported by these methods can be sub-optimal. On the other hand, MML can adapt its parameters in accordance with the dynamics of the environment and can support the policy learning process efficiently.

4.4.3 Support for the Human Interaction Task

The human interaction task is less constrained by the experimental setup. The major influence on this task comes from the boredom rate. As the boredom rate increases the partner becomes more impatient. This may lead to an expectation that the selection rate of the HIM should be proportional to the boredom rate. Figure 4.12 shows the relation between the HIM selection rate and the boredom rate for all experiments. Linear regression on the data shows the HIM selection rate has tendency to increase with increasing boredom rate for MML. Deviations from the tendency are probably because of the constraints from the reaching task (e.g. in more dynamic scenes the objects need more frequent updates, leaving the HIM less time to stay active).

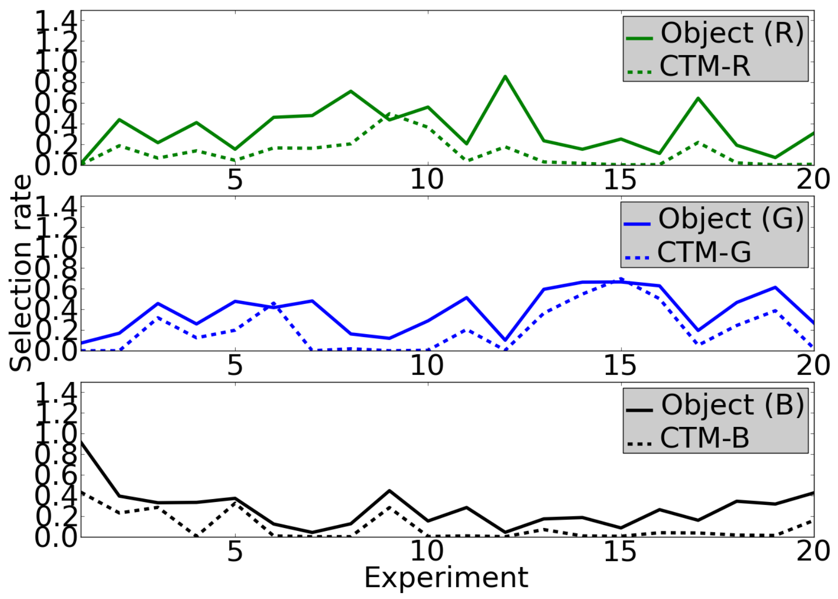


Figure 4.13: Average module selection rates (solid lines) and average object selection rates (dashed lines) for the last 100 epochs (i.e. after learning) in RLP. R, G and B stands for red, green and blue respectively.

4.4.4 Emerging Gazing Behaviors

It was previously shown in Figures 4.11(b), 4.11(d) that MML can develop a gaze strategy where selection of color tracking modules are in accordance with dynamics of the environment. Similarly MML learned to use the Human Interaction Module (HIM) in order to accomplish the interaction task and this was reflected in the module selection rates of the HIM. The motion detection module (MDM) was not favored by MML in both experiments. This is probably because it delivers ambiguous information in cases where multiple objects are in motion. The visual exploration module (VEM) was also not used by MML in both experiments as it may not deliver useful information every time. These examples demonstrate how the MML discards modules delivering ineffective information and selects the task relevant ones for efficiency. It is natural to think that the MDM is a vital information source since the objects are dynamic in the environment. It can be highly prioritized in a manually designed framework. However, better strategies than intuitive ones can be learned with the proposed framework.

An analysis of the relation between the object selection policy for reaching and gazing strategy reveals the co-development of both mechanisms. Figure 4.13 shows average object selection rates and CTM selection rates after learning (i.e. in the last 100 epochs) in each experiment for the system with MML. The results show correlations between the color of the selected object and the CTM for the same color in selection ratings. The correlation coefficients[2] are computed as 0.67, 0.85 and 0.80 for the red, green and blue cases respectively. A similar correlation is not found for the systems using the saliency and random gazing method. The correlation coefficients are 0.25,

[2]Normalized cross correlation is used for the computation of all correlation coefficients in this work.

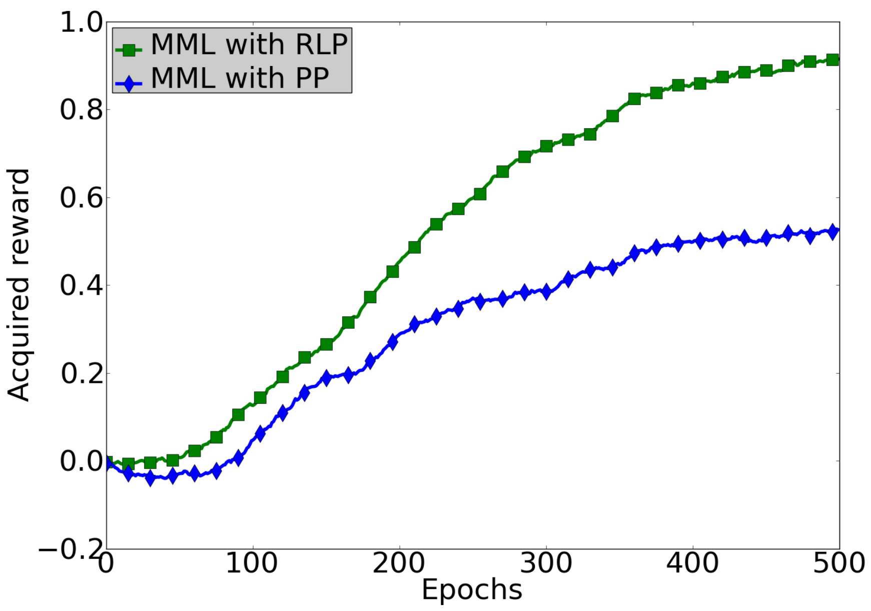


Figure 4.14: Acquired rewards in cases RLP and PP with learning and no-learning settings. The data is averaged over all experiments and smoothed by averaging over the last 100 epochs.

0.34 and 0.11 in saliency and 0.14, 0.44 and 0.20 in random gazing for the red, green and blue colors respectively. On the other hand, the correlation coefficients in the system using Task based gazing are computed as 0.95, 0.97 and 0.93 for the red, green and blue objects respectively. This outcome is the result of the manual coding since the task based gazing imposes 'look where I am reaching' strategy. Correlation coefficients of the MML mechanism are closer to the values of the task based strategy, which demonstrates a system with pure eye-hand coordination for the reaching task. The results indicate that a basic form of eye-hand coordination may have formed by using MML even though the underlying control mechanisms are functionally separated.

4.4.5 Individual Contributions of Learning Mechanisms

In order to evaluate the individual contribution of the MML to the overall performance of the system a second instance of the system is realized. In this instance the reinforcement learning policy (RLP) for object selection is replaced with a programmed policy (PP). In this second case the system chooses a random object to reach at every time step. In summary, in the case of MML with RLP the system employs two learning mechanisms (MML and reaching policy learning) whereas in the case of MML with PP the system employs one learning mechanism (MML). Figure 4.14 shows the acquired rewards in the experiments conducted with each instance of the system. The data is averaged over all experiments and smoothed using Equation 3.15 with $\epsilon_s = 100$. The system using two learning mechanisms achieved a superior performance compared to the system that employs only one learning mechanism in terms of acquired rewards.

Task statistics for the case of the programmed policy are given in Table 4.3 labeled as MML with PP. Compared to the case of the reinforcement learning policy indicated

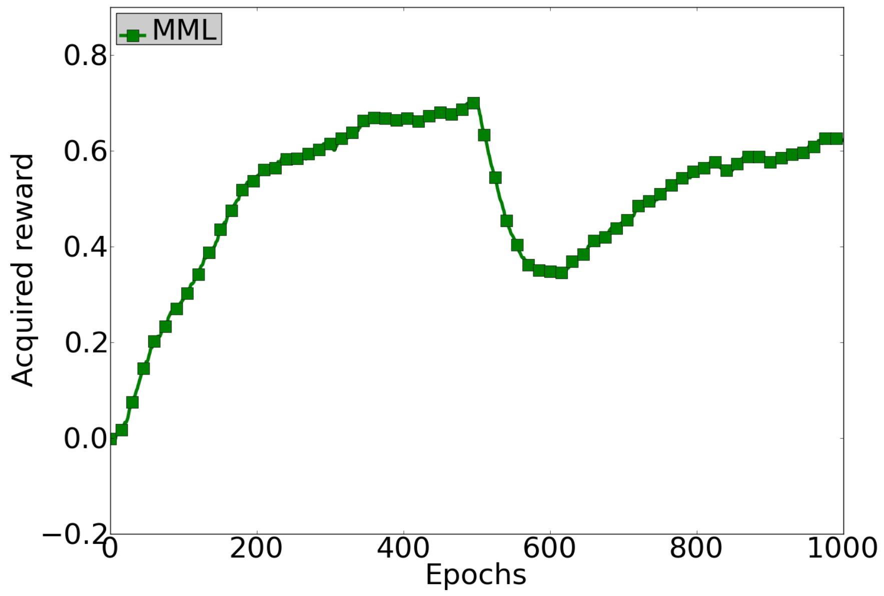


Figure 4.15: Acquired rewards in the adaptation experiments conducted with the system using MML. The data for the learning method is averaged over the last 100 epochs.

as MML with RLP, both systems have similar initial performance statistics (i.e. success rate, boredom and time-out before learning). The success rate increases to 68% for MML with PP and to 81% for MML with RLP. This shows that the reinforcement learning policy is superior to the programmed one for the reaching task. Having that in mind, if the post-learning success rates of the case MML with PP (68%) are compared with Saliency (45%) and Random (54%) strategies, it is visible that the system with the MML still performs better than the other two methods even if it is equipped with an inferior algorithm for one of the task policies (programmed policy). It should be noted that Saliency and Random methods are used in systems with the reinforcement learning policy. This result shows that the MML framework can boost the overall performance of the system even if the system employs an inferior policy.

4.4.6 Adaptation to Changes

It was already shown that the framework can learn efficient gazing strategies in various scenario and environment conditions. To further investigate the flexibility of the proposed framework against perceptual changes in the environment new experiments are conducted (20 in total). In these experiments after 500 epochs, at which point a gazing strategy and a task policy are learned, one randomly selected module amongst the following five is turned off: CTMs for red, green and blue, VEM, MDM. This simulates the case where the stopped module is broken and cannot deliver any information. Fig. 4.15 shows that the performance of the system reduces after the change. However, it can recover adapting its gazing strategy. The final value of average acquired reward after the adaptation (i.e. after 1000 epochs) is slightly lower than the value reached after the initial learning process (i.e. after 500 epochs). This is due to the fact that the adaptation

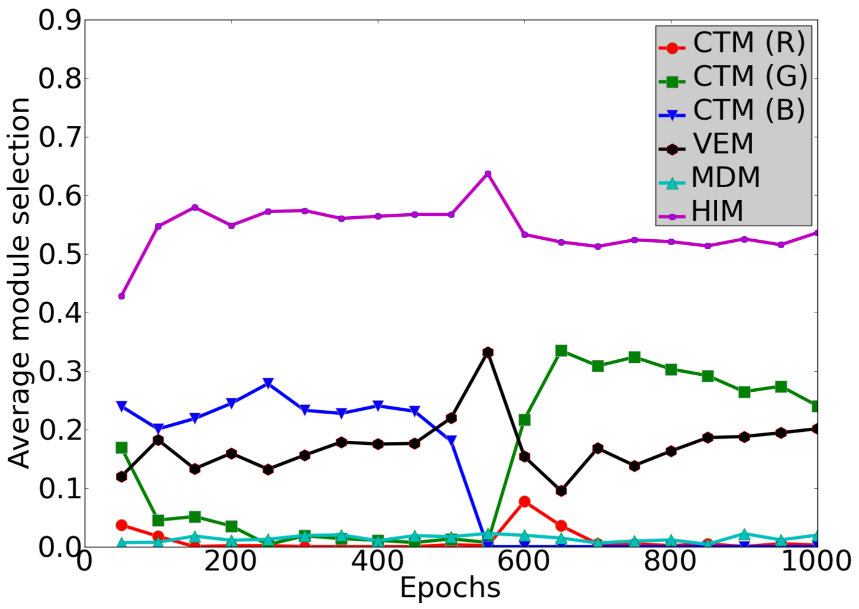


Figure 4.16: Average number of selections for each module averaged over bins of 50 epochs using MML in experiment no. 2. The settings: r_b is 0.26, 0.88, 1.0 and p_m is 0.73, 0.02, 0.22 for the red, green and blue objects respectively, $\tau_b = 0.13$, $v_{max,x} = 19.33$ deg/sec., $v_{max,y} = 39.09$ deg/sec.

of the Q-Learning algorithm used for the object selection policy was not sufficient in some of the experiments. The adaptation process is apparent in Fig. 4.16 where module selection rates during 1000 epochs from a representative experiment (experiment no. 2) is shown. In the beginning, the system learned to attend the most rewarding blue object by selecting CTM-B. The selection rate of HIM was also high due to the human interaction task. After 500 epochs CTM-B was shut down. The attempt of MML to replace this module can be seen in the variations of module selection rates between epochs 500 and 700. The remaining modules were not able to deliver information about the blue object reliably: the information from MDM was redundant since all objects are moving and points suggested by VEM for gaze-shifts may lie in irrelevant areas in the scene. Thus, the system began to select and attend to the green object that was the second most rewarding by selecting CTM-G. The selection rate of HIM goes back to the level it was after the first 500 epochs, after the adaptation happens. These results indicate that the MML mechanism bestows robustness to the system. However, such a flexibility is also limited by the capabilities of the high-level learning mechanisms in the system. The MML framework can address the changes in the perceptual elements of the scenario or environment. The overall flexibility of the system to changes of a larger extent may vary depending on which algorithm is used for other learning problems in the system.

4.4.7 Analysis of the MML Mechanism

Chapter 3 explained the dynamics of the proposed MML mechanism for the module selection process. It has been shown that the mechanism can learn in what sequence modules should be executed to achieve tasks. It can be argued that for multi-tasking scenarios which are in the focus of this thesis, inferring how often the modules should

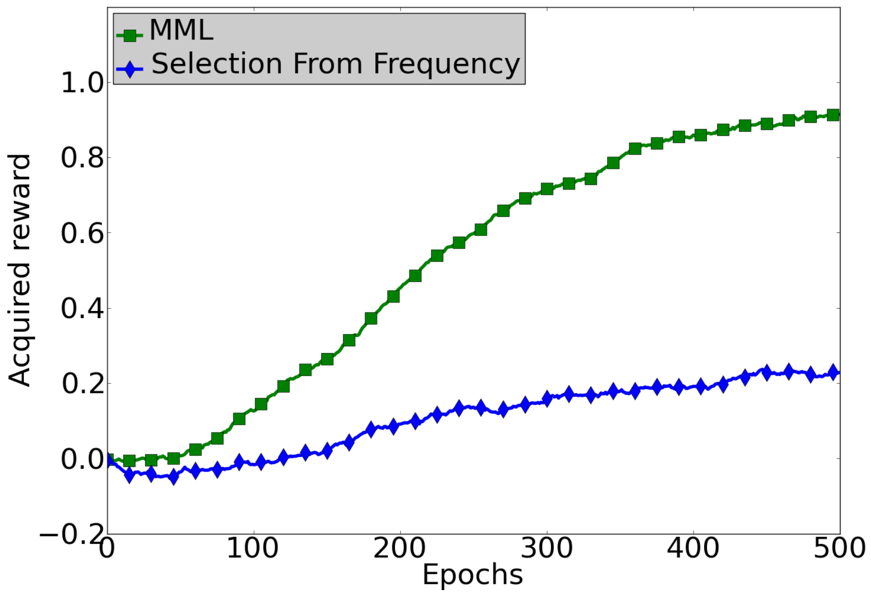


Figure 4.17: Mean acquired rewards. The data is smoothed by averaging over the last 100 epochs.

be executed during a scenario trial is sufficient for an efficient information acquisition strategy. This is because the frequency of information update from a scene entity is directly related to the unpredictability of the entity (e.g. how dynamic the scene entity is) for keeping environmental knowledge of the system up-to-date. To test this an additional experimental setup is designed. Module selection in this new setup is done by reproducing the post-learning module selection rates in the experiments with the system using MML. This is done as follows: module selection rates after the learning is completed (i.e. epochs between 400 and 500) in an experiment with MML are used to generate a probability distribution. With the same experimental settings (e.g. object properties, max. hand velocity, boredom rate) a new experiment is conducted where at every time step the selected module is sampled randomly from the generated distribution. This is repeated for all experimental settings. Figure 4.17 shows acquired rewards from the two experimental setups averaged over all 20 experiments. As the results suggest, a strategy replicating the module execution frequencies is less efficient compared to the strategy learned by the MML mechanism. Hence, temporal sequencing of modules learned by the MML mechanism is also crucial to support the local learning mechanism responsible for the reaching task in the applied scenario.

4.5 Summary

This chapter showed the feasibility of the proposed framework in an active vision application. Firstly, the motivation behind using the active vision approach for robotic systems is explained. In this context, a series of experiments where different depth estimation methods used in both active vision and static parallel stereo vision setups are analyzed. The results showed that active vision setup yields more accurate estimations especially in the near visual field as well as it delivers higher quality surface depth information due to fixated camera geometry.

Next, the application where the proposed framework is used is introduced. In this application, the proposed MML framework is used for managing gaze-shifts for a humanoid to collect information from the environment and support a multi-tasking scenario: selecting, reaching and touching one of the three objects moving on a table and keeping interaction with a human. The scenario seems rather simple, but it is at a level of complexity that an optimum policy is difficult to be derived manually. This is due to the fact that a number of environmental parameters are varied generating a number of instances of different environments. Additionally, the two tasks have diverse characteristics and complexity: the human interaction task requires only passive observation of the partner while the reaching task involves a sensory-motor coordination. It is shown that the proposed framework is robust and can learn efficient gazing strategies for the scenario. This is demonstrated both in terms of acquired rewards in time and the success rate of the system executing the tasks that the scenario imposes.

Learning gazing strategies, as the proposed approach suggests, introduces flexibility to the system, making it possible to cope with changes in the scenario or environment. This is demonstrated in two aspects. First, the system is able to learn efficient information gathering settings for various scenario and environmental conditions. Second, in cases of module malfunctions during the operation MML can find alternative solutions. However, the overall flexibility of the system to such changes is also limited by the other local learning mechanisms present in the system.

It is also shown individually how each of the two tasks are supported by the MML. An important investigated topic is whether the proposed information acquisition framework is able to support other higher level learning mechanisms. To investigate this a reinforcement learning mechanism is implemented in the system to learn an object selection policy for the reaching task. It is shown that the framework can distinguish the modules delivering task relevant information and learn how to schedule them. It is also shown that in general a form of eye-hand coordination is formed even though the control mechanisms for information acquisition (i.e. gazing) and task execution are fairly separated. The relevant module for the human interaction task is also found in every experiment even though no prior information about this task was given to the system. The system eventually managed to utilize this acquired knowledge about the tasks in module scheduling.

An analysis of the MML mechanism evaluating the effects of temporal sequence learning is also conducted. Compared to an information acquisition strategy that only replicates the post-learning selection frequencies of the modules, the MML mechanism achieved better support to the high level learning mechanism for the reaching task. Hence, a higher overall performance in terms of acquired rewards is reached. This shows that learning temporal sequences of execution for modules, as done by the MML mechanism, is crucial for supporting the higher-level local learning mechanisms.

For the applied scenario learning takes about 400 epochs which is a fairly short time for learning approaches using rewards. One epoch takes about 10 seconds on average in the simulations. Hence, the whole learning process takes about an hour to complete. This may be seem like a long time for a human-robot interaction task but it should be noted that through the learning process the system gains re-usable knowledge about the environment shaped by the scenario. From a developmental perspective the learning time is reasonable.

Basically, the framework shapes a bottom-up information acquisition process with the requirements from high-level learning and task execution processes. This introduces an explorative characteristic to the information acquisition process that also benefits other local learning mechanisms in the system. It was shown that a pure task-driven information acquisition strategy hinders the performance of the learning process of the reaching task. It can be argued that the task-oriented character of the framework after learning could constraint the reactive capacities of the system. The information acquisition behavior is governed by the parameters of the module management mechanism. In the absence of tasks these parameters can be set to pre-defined values that would generate the default reactive behavior, so the reactive capacities of the system will be retained.

CHAPTER 5

Application on Resource Allocation in an Autonomous Navigation Vehicle

The second application demonstrating the utility of the proposed framework involves an intelligent developing system with constrained computational resources in an autonomous navigation scenario. Such systems can benefit from abilities of obtaining and processing large volumes of data in various ways such as learning and performing complicated tasks in parallel. This leads to the implementation of large scale systems for robotic applications with improved cognitive capabilities. However, mobile robotic systems usually suffer from limitations in their physical, memory, energy and/or computational resources due to reasons such as restrictions of limited space for on-board computing, limited communication bandwidth for off-board computing, etc. Therefore, it is more practical for intelligent systems to employ better strategies than dealing with all the environmental information available. Rather than acquiring all possible information, a better approach could be attending only to important information sources. This leads to two problems: finding task-relevant information sources and distributing the constrained system resources (e.g. computational power) among these (i.e. when to attend to which source) to cope with multi-tasking scenarios.

There are various works dealing with the problem of handling environmental information for goal-oriented mobile systems. For example, Kwok and Fox [2004] presented a direct application of reinforcement learning (RL) to the active sensing problem in a soccer playing robot. The representation of the environment for the state space consists of task-relevant objects (i.e. scoring goal), odometry and an explicit representation of sensory uncertainties. The system learns which object to look at depending on the current state. However, casting the active sensing problem directly into a RL framework may have crucial drawbacks. Extending the framework or applying it to different scenarios requires adding new representations in the model. This requires re-designing the state space. Moreover, adding new representations may cause the state space to grow beyond computational limits. Busquets et al. [2002] applied a cooperative multi agent system (MAS) for a robot navigation task similar to the one explained in this thesis. The framework learns to manage trade-offs caused by interaction between different agents via RL. The model uses a mixed repertoire of looking and moving actions. However these can be separated since looking actions do not have a direct effect on the environment, they only change the internal state of the system. Such a separation may result in simpler representations of state and action spaces and a more efficient learning strategy. Hence, the authors reported that an effective strategy is only learned with a careful design of state space. Another application of an autonomous navigation scenario is proposed by Sprague et al. [2007] where a RL based approach for controlling behaviors incorporating visual information is presented. The framework introduces microbehaviours which are complete sensory-motor routines for small parallel running tasks. Each microbehaviour employs a RL algorithm to learn an optimal sensory-motor interaction for its task: it gathers visual information, which predicts the current state of the environment relevant to the task and it takes actions based on this information to fulfill the task and receive rewards. A policy that maximizes the acquired reward is learned during the course of such interactions. The problem of which microbehaviour gets access to the camera control is solved by reducing the cost of uncertainty in the internal scene representation over the microbehaviours. While the model proposed by Sprague et al. [2007] is robust and flexible, the definition of which microbehaviours should be used in which scenarios is still done in a pre-programmed fashion.

An application of the proposed framework for such a scenario aims to learn which modules deliver task relevant information and how to distribute computational resources among them. The proposed system level approach helps the system to build an understanding of its environment in a constructive way. In other words, the system is equipped with separate functionalities that can gather various types of information (e.g. color, texture, distance estimation) independently of each other and a mechanism that can build and update a scene representation (e.g. a working memory) from such pieces of information acquired in time. The system has further memory constraints caused by deleting outdated entries (i.e. entries that are not updated for a long time) from the scene representation. The proposed MML framework can handle these constraints by finding task-relevant modules and distributing the computational resources among them in time.

The experiments are conducted in a simulation environment consisting of a vehicle, static and dynamic obstacles and target objects. A snapshot from the simulation environment is shown in Figure 5.1. A simulation engine developed at the Honda Research Institute Europe is used for the implementation. The vehicle is equipped with a camera with 90 degrees field of view and has to develop capabilities of collecting target objects without colliding with the obstacles. The number of target objects, dynamic obstacles and static obstacles are denoted as n_t, n_d and n_s respectively. Objects are distinguished from each other by color: target objects are red, obstacles are green and blue. The vehicle is controlled by velocity and steering angle commands. With this study, it is aimed to test the following hypotheses:

- The modular systems approach provides benefits in handling scenario complexities.

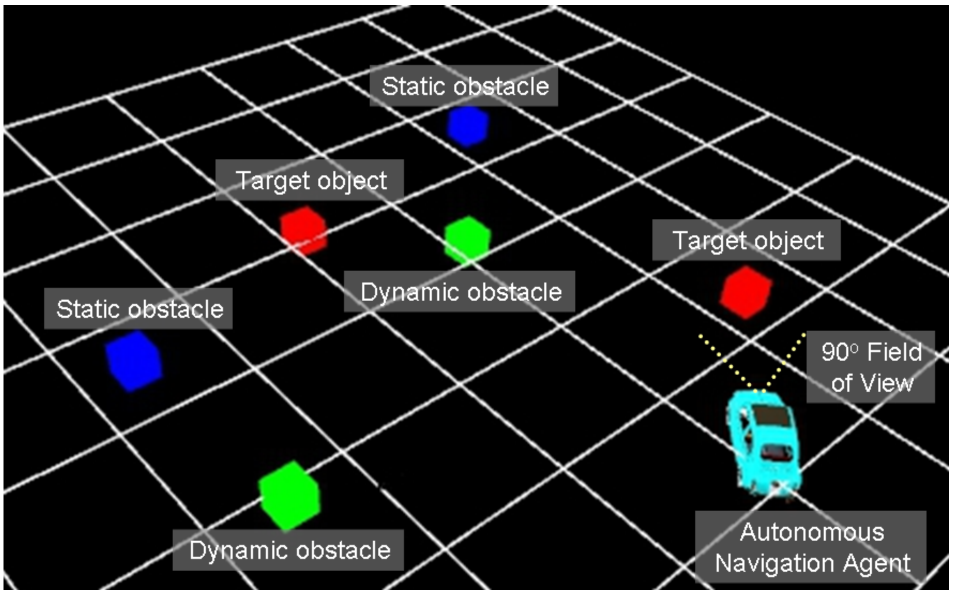


Figure 5.1: A snap shot from the simulation environment where the autonomous navigation scenario is executed.

- The proposed framework can learn how to distribute computational resources among modules in a system efficiently.
- Reducing the problem of partial observability in the information acquisition process, the proposed framework supports multiple parallel running learning mechanisms in the system and facilitates the development process.
- The framework is robust and can adapt to different environments.

The first point does not involve the MML framework but investigates the benefits of the general approach of modularity. In a series of experiments, it is shown that a system following the modular approach can learn to handle tasks better than a non-modular RL method as the complexity of the scenario increases. It is shown that the proposed framework can indeed be applied to various domains as it outperforms the compared manual programming approaches in this application as well. The proposed framework is also compared to the case where all environmental information is exposed to the system at all times to make an evaluation on how well the partial observability is handled. Also, the proposed approach does not aim for specific scenarios and can adapt to varying conditions of the system, environment or scenario. Results from the experiments with this application can also be found in Karaoguz et al. [2013a].

5.1 System Architecture for Autonomous Navigation

The system implemented for the autonomous navigation scenario is another specific instance of the architecture explained in Section 2.2. The general functional description of this system is shown in Figure 5.2. In essence, modules perform elementary operations for information gathering and pre-processing. The assumption for the constrained resources of the system taken in this work imposes that these modules cannot

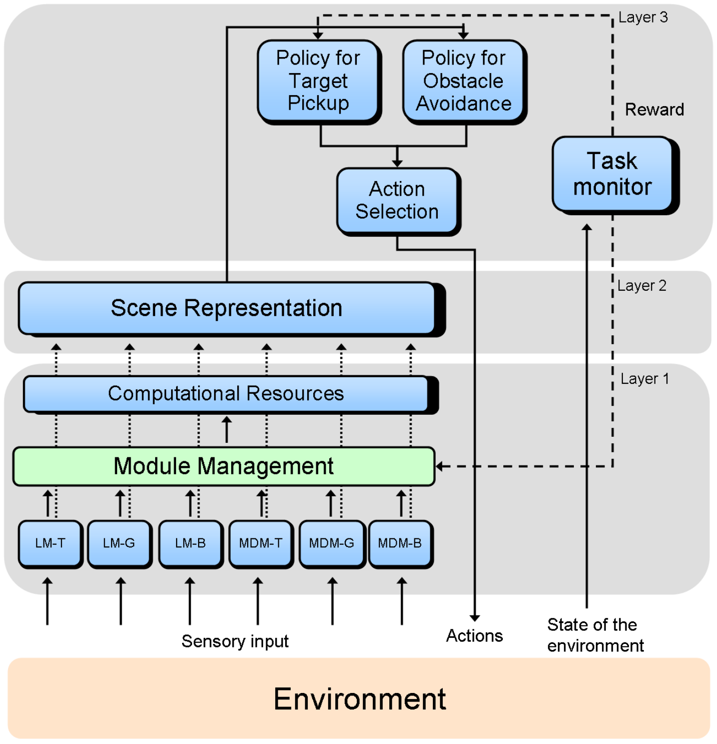


Figure 5.2: The functional decomposition of the modular system implemented for the autonomous navigation scenario. The abbreviations in the figure are: LM-T, LM-G, LM-B are Localization Modules for targets, green obstacles and blue obstacles and MDM-T, MDM-G, MDM-B are Motion Detection Modules for targets, green obstacles and blue obstacles respectively.

run in parallel. Modules have also requirements on time allocated for them to access resources to perform a successful processing cycle. The MML makes module selections to distribute the computational resources between modules in time. A scene representation mechanism keeps temporally stabilized environmental information acquired from modules as a result of their processing operation. The information kept in the scene representation is used by higher level policies that map environmental states to actions to execute tasks. These task policies use whatever knowledge the scene representation has about the environment to decide which actions to take. A modular RL approach (see Section 3.1.3 for a detailed explanation) is followed to learn such policies: the scenario is divided into two tasks as target pick-up and obstacle avoidance and two separate Q-Learning algorithms are used to learn policies for these tasks. Since these RL modules share the same action space (i.e. they both generate velocity and steering commands for the vehicle) an action selection mechanism is implemented to arbitrate actions from the two. The task monitor observes the state of the environment and the agent (e.g. occurrence of collisions, picked up targets) and generates rewards depending on the outcomes of the actions the system takes. The rewards are used by the MML and RL modules for learning. Below, detailed information about the fundamental building blocks of the system is given.

5.1.1 Modules

Modules are basic elements for information acquisition and processing. In practice such modules have different run times: trivial applications like basic filtering algorithms can take little time whereas more computationally intense algorithms such as object recognition take longer to produce results. This fact is incorporated in the simulations as module run-times. Every module needs to be active for a number of time-steps ($t_{run,i}$) to be able to process and deliver the information. The following modules are implemented in the system:

- Target Localization Module (LM-T)
- Green Obstacle Localization Module (LM-G)
- Blue Obstacle Localization Module (LM-B)
- Target Motion Detection Module (MDM-T)
- Green Obstacle Motion Detection Module (MDM-G)
- Blue Obstacle Motion Detection Module (MDM-B)

Object localization modules compute the relative distance z and orientation θ of the objects to the agent via the information from the simulator. Motion detection modules detect movements of corresponding objects and generate a binary variable m as output indicating whether a motion is detected ($m = 1$) or not ($m = 0$). Both LM and MDM apply color filtering using the color opponency method explained in Equation 4.4 to extract relevant objects in images. Motion detection modules compute differences of the color filtered images taken in consecutive time-steps to detect displacements of objects in the image. The field of view for all modules is set to 90 degrees in front of the agent.

5.1.2 Scene Representation

The scene representation is built by a simple implementation of a working memory (WM) to keep and temporally stabilize the information about the environment for various functionalities in the system as explained in Section 2.2.2. The properties of target objects, static and dynamic obstacles are kept in the WM. Every entity has a unique id denoted as id and age a_{id}. Color c_{id}, relative distances (z_{id}, θ_{id}) and motion information $m_{id} \in 0, 1$ (i.e. if they are moving or not) of objects are kept as features of the entities in the memory. The WM is empty in the beginning of an epoch. At every time step an update process of the WM performs the following routine:

- for every id in the WM, all elements with $a_{id} > a_{max}$ are removed from the WM;
- for every id of the remaining elements in the WM, the age is increased by one ($a_{id} \leftarrow a_{id} + 1$);
- if the agent has moved since the last memory update the relative position information of the elements in the WM are updated accordingly;

- if there is an information received from the modules it is compared to the entities in the WM; if a similar entity is found (in terms of type and position) the entity is updated with the newly received information; otherwise a new entity is created in the WM.

5.1.3 Tasks and Policies

A similar navigation scenario is presented by Sprague et al. [2007] where the scenario is decomposed into smaller tasks using a modular RL framework (see Section 3.1.3 for a detailed explanation of the modular RL approach). These tasks can be formalized as individual Q-functions following the assumption that the Q-function for the whole scenario is approximately equal to the sum of the individual Q-functions for the separate tasks. A similar approach is taken in the current application where the scenario is decomposed into two tasks: avoiding obstacles and picking up target objects. The state space consists of the relative position and detected motion of the nearest target and relative position and detected motion of the nearest obstacle for target collection and obstacle avoidance Q-functions, respectively. Since relative position values are continuous variables they are discretized between 1 and 10 using Equation 4.8. In summary, the state vector is in the form of $[\overline{z}_{tar}, \overline{\theta}_{tar}, m_{tar}]$ and $[\overline{z}_{obs}, \overline{\theta}_{obs}, m_{obs}]$ for both Q-functions where tar and obs subscripts denote the nearest target and the nearest obstacle respectively and $(\overline{z}, \overline{\theta})$ are discretized values for relative distance and relative orientation, respectively. In case of no position information the corresponding element of the state vector is set to 0.

The action space is the same for both Q-functions and consists of 9 different actions. These are combinations of three car velocities $(v_{slow}, v_{mid}, v_{fast})$ and three steering commands $(\Delta\theta_0, \Delta\theta_-, \Delta\theta_+)$. These values are set to $(0.02, 0.1, 0.2)$ for velocities and $(0, -\frac{\pi}{4}, \frac{\pi}{4})$ for steering commands respectively. Since both policies share the same action space, the individual Q-functions assumption explained above can be used to compute the final action for the vehicle to take from the combination of the Q-functions as shown in Equation 3.7. The Softmax mechanism (Equation 3.12) is used to sample the action from the distribution obtained from the combined Q-functions. At every time step a new selection is done.

5.1.4 Task Evaluation and Rewards

For easier assessment of performance the scenario execution is divided into epochs. The time line within an epoch is divided into time steps. An epoch starts with initialization of the world by placing objects and the agent randomly in the environment. During an epoch dynamic elements of the scene (i.e. the vehicle and dynamic obstacles) make movements at every time step. Collecting target objects is rewarded by 1 unit of reward whereas colliding with obstacles results in -1 units of reward. An epoch ends in one of three conditions: all target objects are collected, a collision has occurred or the time-out limit t_{max} is reached.

5.2 Outline of Experiments

For comparison four additional system instances using various information acquisition methods are created. These methods can be summarized as:

Table 5.1: Overview of the experimental settings.

	Target object		Static obstacle		Dynamic obstacle	
Setting	quantity (n_t)	color	quantity (n_s)	color	quantity (n_d)	color
A	1	red	1	blue	0	green
B	1	red	0	blue	1	green
C	1	red	1	blue	1	green
D	1	red	1	green	1	blue
E	1	red	2	blue	1	green

Table 5.2: Parameter values used in the autonomous navigation experiments.

MML		**System**		**Q-Learning**	
α	0.001	t_{max}	400	α	0.01
γ	0.99	$t_{run,LM}$	2	γ	0.99
β	0.001	$t_{run,MDM}$	4	Z	0.01
t_{hist}	40	a_{max}	100		

- *Module Management Learning (MML)*: is the proposed method.
- *Perfect World Knowledge (PWK)*: is the hypothetical case where the system is exposed to all information about object positions and motion at every time step. Therefore, information acquisition modules are not necessary in this case.
- *No Resource Constraints (NRC)*: is the hypothetical case where the system has no constraints on computational resources and consequently all modules are executed at every time step and all the acquired information is used.
- *Programmed*: strategy assumes that we have the prior information of necessary execution times of modules and follows a straightforward pre-programmed strategy where every module is executed just enough time steps to get information. Selection between modules is done in the order from the first module to the last, going back to the first after the last.
- *Random*: strategy is the same as the Programmed case except for the module selection process that is done by randomly selecting the next module to execute.

Two setups are designed to investigate two general issues: contribution of modular developing system design in handling scenario complexity and contributions of the proposed framework in resource constrained developing modular systems. Different sets of experiments with varying settings are conducted at each setup. Table 5.1 shows these settings. Each set contains 10 experiments. In all experimental settings, the velocity of dynamic obstacles is set to random values between -0.1 and 0.1 units per time step for X and Y axes separately at each epoch. Parameters used in the experiments are selected with best effort and shown in Table 5.2.

Setup 1: Modular vs. Non-modular Developing Systems

The first series of experiments investigates the benefits of modularity in handling system complexity. Therefore, the information acquisition problem is excluded and the NRC case is applied in these experiments (i.e. MML is not used in this setup). For these experiments an additional method using a non-modular approach to learn autonomous navigation policies is implemented. This approach casts the whole control problem into one RL framework. To be consistent with the modular system, Q-Learning is used as the RL algorithm. The state space is formed directly from the sensor measurements i.e. pixel positions of detected objects in the image. The image size is set to 30×30 pixels. At every time step the state vector is constructed by concatenating the horizontal pixel position of target object, static obstacle and dynamic obstacle detected on the acquired image. If no object of a particular type is detected, 0 is used for the corresponding value. The length of the state vector depends on the number of objects in the scenario. For example, in experiments where only the target object exists dimensions for obstacles are discarded (i.e. state vector has a length of one). The objects are detected in the image by color filtering using the color opponency method as in Equation 4.4. The action space is the same as the one used in the modular system. Modular and non-modular systems are tested in different experiments with increasing scenario complexity and the results are compared. The complexity of the scenario is varied by changing the number of objects in the scene. Hence, settings A, B and C (see Table 5.1) are used to generate three sets of experiments, each set with 10 experiments. Additional 5 experiments per setting are conducted for an additional case where the vehicle takes random actions for navigation without policy learning. This is done as a control case for the learning process of the systems.

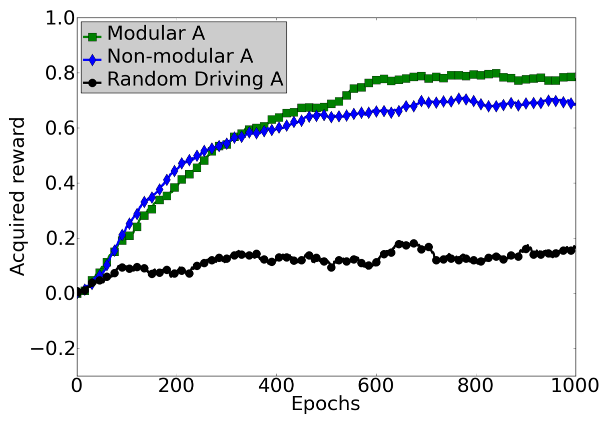
Setup 2: MML in Resource Constrained Developing Modular Systems

The second series of experiments investigates the hypotheses about contributions of the MML on information gathering. Experiments are conducted using setting C (see Table 5.1). In total 10 experiments are done with each of the following methods: MML, PWK, Programmed and Random. Further experiments with settings D and E are conducted to compare the system behavior and the learning processes in different environments. Again, 10 experiments for each setting are conducted.

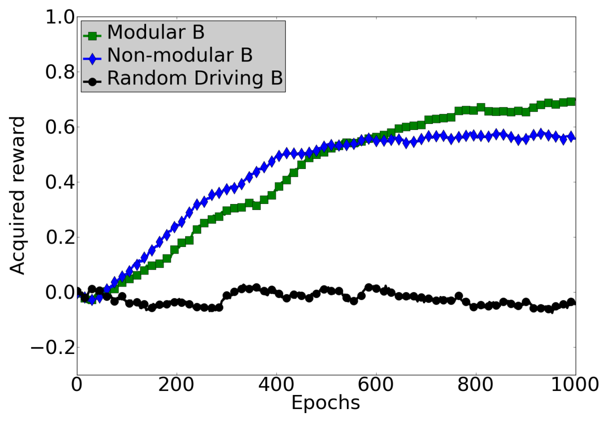
5.3 Results

5.3.1 Benefits of Modular Systems Approach

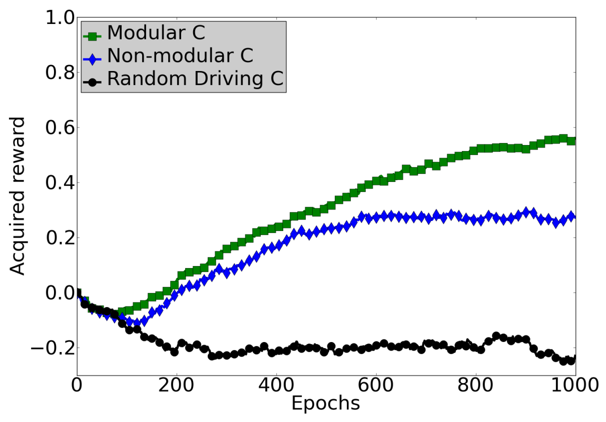
The first setup is designed to analyze the benefits of the modular system approach in handling the scenario complexity. Two system instances designed following a modular and a non-modular approach were tested in three different experiment settings with different levels of complexity of the scenario (settings A, B, and C shown in Table 5.1). Figure 5.3 shows the acquired rewards in experiments with the two systems and Table 5.3 shows statistics of scenario outcomes. Both results are averaged over all experiments run for each experiment setting. Acquired rewards data is further smoothed using Equation 3.15 with $\epsilon_s = 100$. It can be observed that both systems can achieve learning in all experiments. The final values of the acquired reward do not reach the maximum (i.e. 1) in all settings for both systems. This is due to the field of view of the cameras (i.e. dynamic obstacles behind or at the side of the vehicle can collide before being noticed) and under-representation in state/action space. This is expressed in the learning curves as the acquired rewards by the two systems with learning mechanisms



(a) Setting A: $n_t = 1, n_s = 1, n_d = 0$



(b) Setting B: $n_t = 1, n_s = 0, n_d = 1$



(c) Setting C: $n_t = 1, n_s = 1, n_d = 1$

Figure 5.3: Acquired rewards in modularity analysis experiments averaged over all experiments.

Table 5.3: Task statistics averaged over all experiments (in %). Beginning of learning (BL) covers epochs between 0-200, after learning (AL) covers epochs 800-1000.

	Setting A		Setting B		Setting C	
Scenario outcome	Modular	Non-modular	Modular	Non-modular	Modular	Non-Modular
Success (BL)	57	57	46	46	41	36
Success (AL)	81	74	81	69	75	53
Collision (BL)	18	13	33	26	43	44
Collision (AL)	4	6	12	13	19	25
Time-out (BL)	26	30	21	28	15	20
Time-out (AL)	15	20	7	19	5	22

increase with epochs whereas the performance of the random driving stays constant. Task statistics are also in accordance with this results: with the learning progress, success rates (i.e. how many times the target object is picked up) of the two systems increase while collisions and time-outs decrease. Acquired reward from the experiments with the non-modular system reaches the final value (i.e. stabilizes) earlier than in the modular system. This is because even though the dimensions of the state spaces are the same, the modular system has to learn an action space twice as big (i.e. for two Q-functions). Despite this, the modular system achieved the highest performance values in all experiment settings after the learning is done. However, as the complexity grows, the performance of the non-modular system drops significantly. The post-learning success rate of the non-modular system decreases from 74% to 53% whereas the success rate of the modular system stays around 79% for each setting. Even for more complex environments where the total number of objects goes up to 6 the performance of the modular system is retained (see Section 5.3.2, compare the performance of the NRC case in Table 5.4). This is an indication that the modular system scales with scenario complexities.

5.3.2 Overall System Performance with Constrained Resources

The second setup is designed to test some of the hypotheses postulated in this thesis by comparing several instances of a resource constrained modular system with different information acquisition strategies: MML, PWK, Programmed and Random. The overall performance of these systems is analyzed in a series of experiments (10 for each method) conducted in setting C (see Table 5.1). Figure 5.4 shows the acquired rewards averaged over all experiments for each method. Acquired rewards data is further smoothed using Equation 3.15 with $\epsilon_s = 100$. Similar to the results in the previous subsection, the reasons why the final values of the acquired reward do not reach the maximum (i.e. 1) are the field of view of the cameras and under-representation of state/action space. The proposed method performs better than Programmed and Random methods in terms of acquired reward. Another metric to assess the performance of the systems is the analysis of task outcomes. Table 5.4 shows average rates of task outcomes before and after the learning takes place. MML achieves to collect target objects 45% of the time in the beginning of learning. This increases to 74% after learning. Compared to PWK, MML reaches 87% of the performance of the case where all infor-

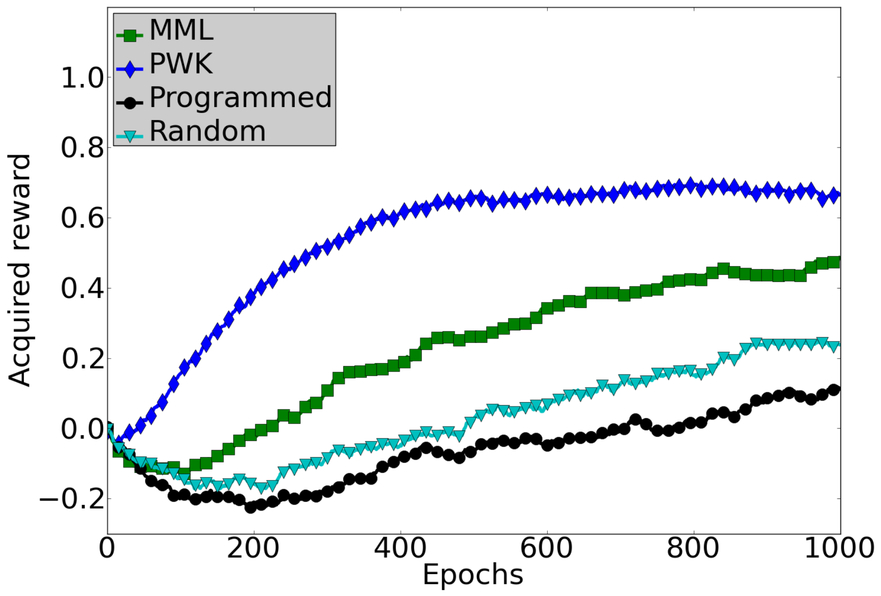


Figure 5.4: Acquired rewards with each compared method averaged over all experiments and smoothed by exponential moving average method with a smoothing factor of 100 epochs.

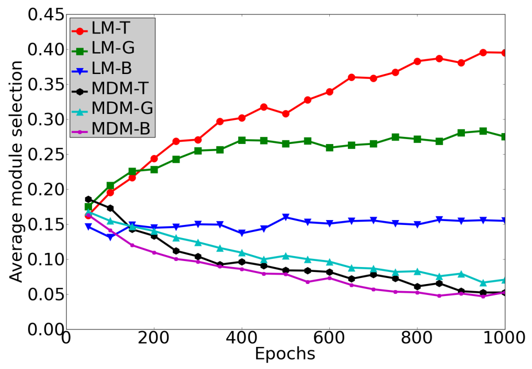
mation is available to the system all the time. This quantifies how well the proposed method can deal with partial observability caused by the constraints on the information acquisition resources. This value is approximately 66% for the programmed and 76% for the random strategies.

Table 5.4: Task statistics averaged over all experiments (in %). Beginning of learning (BL) covers epochs between 0-100, after learning (AL) covers epochs 400-500.

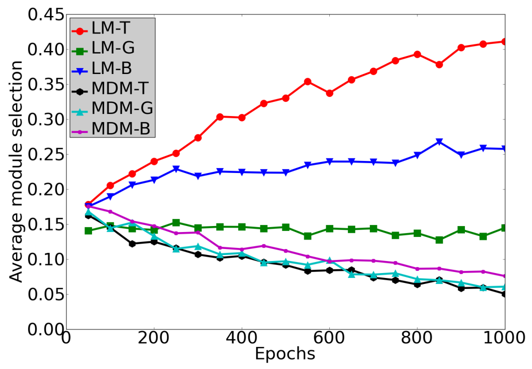
Outcome of tasks	MML	PWK	Prog.	Random
Target picked up (BL)	45	68	36	39
Target picked up (AL)	74	85	56	65
Collision (BL)	55	32	63	60
Collision (AL)	25	15	44	35

5.3.3 Learned Information Acquisition Behavior

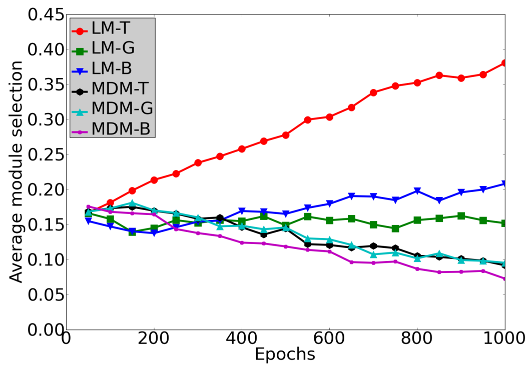
To inspect the information acquisition behavior learned by the MML framework an analysis on how often each module is selected during the learning process can be made. Figure 5.5(a) shows the mean number of selections of each module for 50 epoch bins averaged for all experiments with the system using MML. These experiments used setting C hence, there is 1 instance of each object type (target, static obstacle and dynamic obstacle) in the environment with blue being the static and green being the dynamic obstacle. The graph illustrates the learning process of the MML mechanism. It should be noted that it is not enough to infer which modules deliver crucial information, it is also important to find out how they should be utilized in time. This is not only because the environment is dynamic, but also due to the memory constraints the system posses.



(a) Setting C: $n_t = 1$, $n_d = 1$, $n_s = 1$, dynamic obstacle is green, static obstacle is blue.



(b) Setting D: $n_t = 1$, $n_d = 1$, $n_s = 1$, dynamic obstacle is blue, static obstacle is green.



(c) Setting E: $n_t = 1$, $n_d = 1$, $n_s = 2$, dynamic obstacle is green, two static obstacles are blue.

Figure 5.5: Average number of selections for each module, averaged over bins of 50 epochs using MML for experiments with three different settings (LM: Localization Module, MDM: Motion Detection Module, T: Target Object, G: Green Obstacle, B: Blue Obstacle).

Crucial modules should be executed regularly so that task-relevant environmental information will not be lost and is kept up-to-date. Results show that the selection frequency of localization modules grows relative to the motion detection modules. At the end of the experiments, the selection frequency of the target localization module reaches the highest value, followed by the green obstacle detection module and blue obstacle detection module in that order. This result is in accordance with the experiment settings: the target object is vital for the whole scenario since it is the only reward source. For the collision avoidance task, the green obstacle has to be monitored more often than the blue one because it is dynamic and imposes greater risk for collision. The fact that motion detection modules are favored less shows that this information is not very critical for Q-Learning algorithms i.e. both tasks can be solved without the knowledge of their motion. Eventually the MML mechanism learns to allocate more resources to the localization modules that are relevant to the scenario.

It is also interesting to see what the system learns in different environments. For this purpose two additional sets of experiments (with 10 experiments each) are executed with different settings (setting D and setting E). Setting D exchanges the colors of dynamic and static obstacles i.e. the static obstacle is green and the dynamic obstacle is blue. In setting E, the color coding of the obstacles stayed the same as in setting C (i.e. blue for static and green for dynamic) while the number of static obstacles increased to two. The system is able to learn efficient information gathering strategies under both settings. Success rates for the target pick-up task after learning are 72% and 68% for setting D and setting E respectively. The slight decrease in the performance under setting E is due to the increased complexity introduced by adding an object. Figure 5.5(b) shows mean module selection frequencies averaged over experiments conducted in Setting D. Compared to setting C, in this case the system changes its favored obstacle localization module from green to blue. Figure 5.5(c) shows the average module selection rates in setting E. In this case, the system also learns to favor the blue localization module over the green for detecting obstacles. However, the gap between the selection rates of both types of obstacles is smaller compared to the results from setting C and D. Hence, increasing the number of static object in the environment was enough for the system to render the static obstacles almost as important as the dynamic one. Overall, these three sets of experiments show that the proposed framework is able to learn information acquisition strategies appropriate to the environment the system is in.

5.3.4 Support for Policy Learning

To evaluate how well MML can support policy learning mechanisms the elements of these functionalities can be examined. In the proposed system, the learned Q-Functions can be helpful to make an assessment. Assuming optimal task policies are learned in the case of PWK, similarity between the policies learned in PWK and other methods can be evaluated. Such a comparison may indicate how close the system can get to the perfect world knowledge using the compared information acquisition method. To quantify the similarity between two learned Q-functions a similarity measure is defined as:

$$\frac{1}{S}\sum_{i=1}^{S}(1 - |a_m^*(s_i) - a_{PWK}^*(s_i)|), \tag{5.1}$$

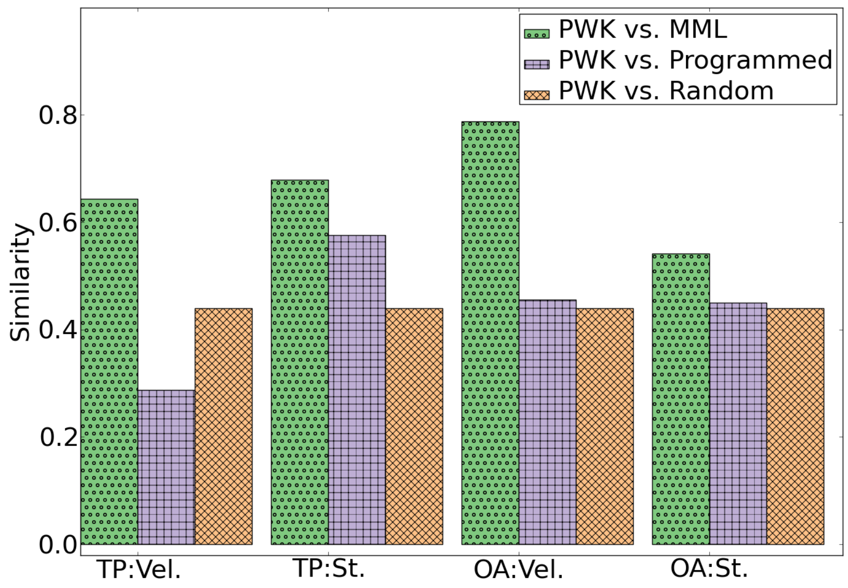
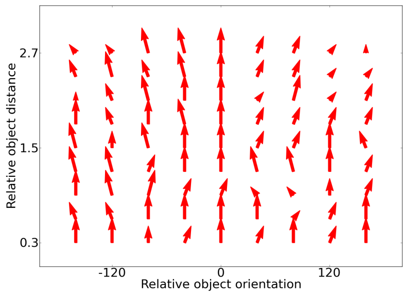


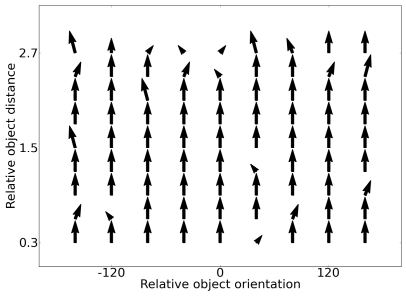
Figure 5.6: Similarities of policies learned in systems using MML, Programmed and Random methods to the policies learned in the case of PWK, averaged over all experiments. The abbreviations are TP: Target Pick up, OA: Obstacle Avoidance, Vel: Velocity and St: Steering Angle.

where S is the total number of states, a^*_m is the best action from the Q-function learned in the system using method m for information acquisition and a^*_{PWK} is the best action from the Q-function learned in the case of PWK. The best actions are determined as in Equation 3.7. Figure 5.6 shows such similarities computed between policies learned with PWK and the other methods averaged over all 10 experiments conducted in setting C. Velocity and steering actions imposed by both task policies learned with the MML are closer to the ones of the optimal policies compared to that of Programmed and Random methods.

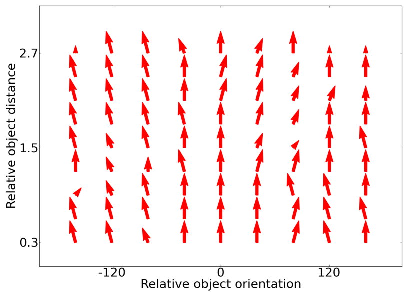
As a representative example, Figure 5.7 shows policies for the two tasks learned in systems using MML, PWK and Programmed methods as information acquisition strategy from one experiment. For the target pick-up task, the learned policy in the PWK case (Figure 5.7(c)) prescribes steering commands towards objects (e.g. when a target object is sensed on the left side, a steering command towards left is imposed). The velocity commands do not vary across the state space and maximum speed is used for most of the cases. This is due to the absence of uncertainties about the world knowledge. A similar policy is learned for the same task with the system using MML as information acquisition method (Figure 5.7(a)). Smaller velocities are favored in this case because of the uncertainties in scene representation. MML manages to reduce these uncertainties enough to learn a near-optimal task policy. The policy learned under the Programmed method (Figure 5.7(e)) is dissimilar to the one learned in PWK case, contains inconsistencies and is not optimal (e.g. when the target is in front of the agent i.e. 0 degrees relative orientation, the policy imposes left and right turns). For the obstacle avoidance task, the policy learned in the case of PWK imposes mainly left



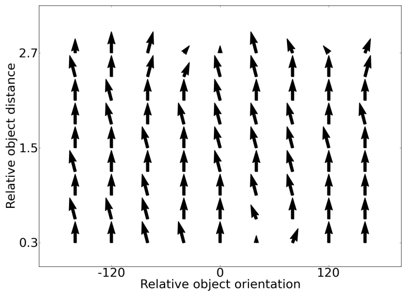
(a) Target pick-up with MML



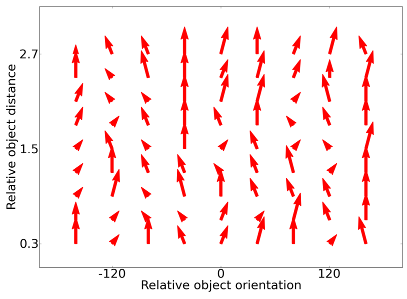
(b) Obstacle avoidance with MML



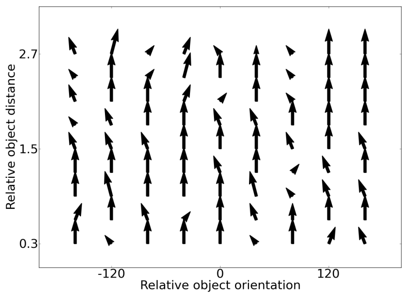
(c) Target pick-up in PWK



(d) Obstacle avoidance in PWK



(e) Target pick-up with Programmed



(f) Obstacle avoidance with Programmed

Figure 5.7: Learned policies for the target pick-up task (a,c,e) and obstacle avoidance task (b,d,f) with different information acquisition methods. Arrows show the best action of the policy when the closest target object in the WM is estimated to be at the corresponding relative distance and relative orientation. The length of the arrow is proportional to the vehicle velocity and the direction of the arrow is the steering direction of the corresponding policy action.

turns, leaving the obstacle on the right side of the agent (Figure 5.7(d)). This strategy is consistent among a number of states (i.e. relative position of the obstacle). The policy learned in the case of MML follows a similar strategy however, right turns are favored in this case and this strategy is taken mainly for the states where the obstacle is far from the agent (Figure 5.7(b)). The policy prefers to take actions early on, as soon as an obstacle is noticed. This can be interpreted as if the policy follows a conservative strategy since the uncertainties in the information acquisition process are not eliminated entirely. However, the contribution of the MML framework is apparent when compared to the policy learned in the case of the programmed information gathering strategy that does not impose consistent actions (Figure 5.7(f)).

5.3.5 Analysis of the MML Mechanism

Similar to the previous application, the effects of learning temporal module execution sequences is evaluated for the autonomous navigation application. For this, an additional setup is designed as done in Section 4.4.7 where module selections are done only by replicating the post-learning (i.e. epochs between 800 and 1000) module selection frequencies in the MML experiments. This is done as follows: post-learning module selection rates in an experiment with MML are used to generate a probability distribution. With the same experimental settings (e.g. object properties, number of objects) a new experiment is conducted where at every time step the selected module is sampled randomly from the generated distribution. This is repeated for all 10 experiments. Figure 5.8 shows the acquired rewards for the two cases averaged over all experiments. Similar to the previous application, if only the frequencies of module selection rates are replicated, the performance of the system is worse than that of the case where MML is used. Hence, temporal sequencing of module executions learned via the MML mechanism plays an important role in supporting the local task learning processes.

5.4 Summary

The proposed MML framework is applied to a resource constrained modular system to distribute computational resources among various information acquisition modules in time. The system is utilized with capabilities of learning how to control a vehicle in a dynamic environment within a scenario where two tasks have to be executed in parallel: picking up target objects while avoiding obstacles. In a series of experiments, it was shown that the modular systems approach is beneficial for handling scenario complexities compared to a non-modular global learning approach. In a further analysis, it was demonstrated that the proposed framework can find efficient strategies to distribute the constrained computational resources among various information acquisition modules in time. As a result, the uncertainties in the information acquisition process are reduced to a level that higher level cognitive mechanisms in the system (the modular RL mechanisms in this work) can perform their tasks. The overall performance of the system in terms of acquired rewards is higher compared to systems using random selection and pre-programmed strategies. Furthermore, task policies learned by the modular RL algorithms are similar for the systems with perfect world knowledge and the proposed framework. The experiments with systems using random or pre-programmed policies end up with inconsistent and sub-optimal task policies. A representative ex-

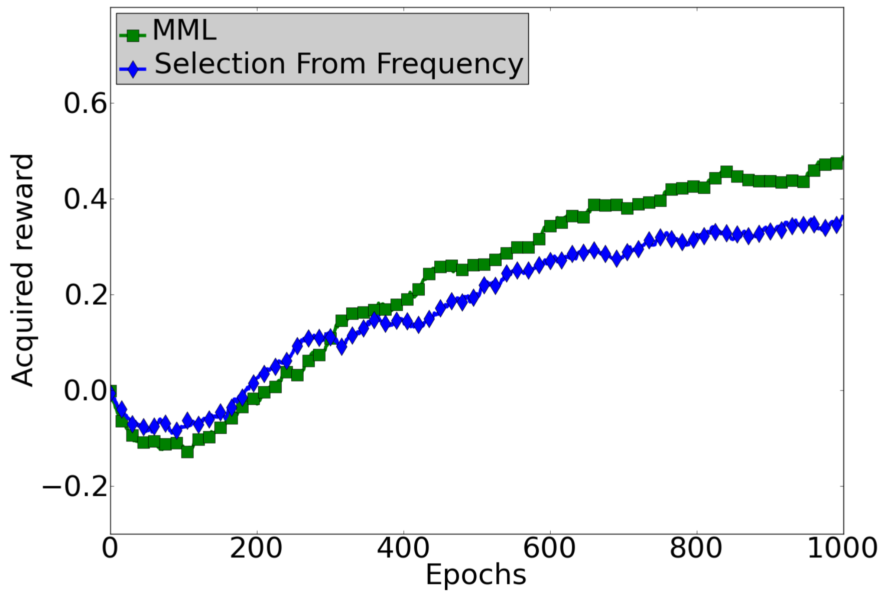


Figure 5.8: Mean acquired rewards. The data is smoothed by averaging over the last 100 epochs.

ample is presented in results. The framework is also robust and introduces flexibility to the system, as it is demonstrated that the framework can adapt to environments with different characteristics. Compared to Sprague et al. [2007] the proposed framework learns which modules delivers important information for the tasks being executed by the system. This makes the system more robust against changing environments and may facilitate system design. Kwok and Fox [2004] applied a RL algorithm directly to learn a sensory-motor policy for their scenario. However, as it was shown in the results, breaking the system into functional elements (this also includes decomposing the scenario into smaller tasks) as the modular systems approach suggests makes the system more capable of handling increasing complexities in the scenario. Additionally, in contrast to Busquets et al. [2002], Kwok and Fox [2004] the proposed framework separates the action mechanisms for information acquisition (e.g. gazing) and for world manipulations (e.g. reaching, navigation, etc.). In this way, the information acquisition process can be optimized for parallel running task policies employed by the system. New tasks and policies can be introduced to the system easily without concern about the information acquisition process. Finally, an analysis of the MML mechanism evaluating the effects of temporal sequence learning is done. An information acquisition strategy that only replicates the post-learning selection frequencies of the modules delivers inferior performance compared to the results from the experiments conducted with MML. This indicates that temporal sequencing of modules learned by the MML mechanism is crucial for supporting the higher-level local learning mechanisms.

CHAPTER 6

Conclusions

6.1 Summary

The ideas and methods presented in this dissertation aim to improve the information gathering process of large-scale systems so that they can better deal with real world scenarios. The scope of the thesis covers developing intelligent systems that can unfold capabilities of using their sensory-motor mechanisms to achieve goals by learning through interactions with the environment. Hence, these systems usually employ multiple local learning mechanisms to acquire such capabilities. Naturally each local learning mechanism may require information about a different set of environmental states. However, the amount of information that can be gathered from the environment is vast and can not be dealt with entirely by the systems due to physical, computational and memory constraints. Instead, the information gathering process can be organized in a way that only the information relevant to the scenario is obtained. This raises several problems to be dealt with and this dissertation addresses several of these:

- *Organization*: structuring the process of information acquisition,
- *Value assessment*: finding relevant information sources,
- *Arbitration*: choosing when to gather which piece of information,
- *Partial observability*: minimizing the uncertainties caused by the information gathering process,
- *Robustness*: being able to adapt to different environments and changes.

Organization is a fundamental issue that influences how the information is gathered and processed by the system. The complexity of real world scenarios can be dealt with by breaking them into several tasks and executing them in parallel. Thus, a formal system definition that supports such a multi-tasking framework can be beneficial both in design and operation of the system. The problems of finding the sources of information relevant to tasks and deciding how to update them in time arise from the fact that the environment consists of numerous task-relevant and task-irrelevant entities some of which are dynamic. Dynamic elements of the scenario also cause partial observability since attention shifts in the information gathering process may leave the system without updated state information of such dynamic elements. Finally, the robustness is an important issue for the information gathering process due to the fact that in real-world scenarios properties of scene entities can vary in different instances of execution of a scenario even though the scenario is the same.

This thesis proposed the idea of learning the information gathering process to address these issues and a system level approach to realize this idea. For the system organization issue the presented framework prescribes a modular systems approach where modules are defined as elementary processing units for information acquisition and processing. It was suggested that such a framework facilitates information acquisition and processing especially for learning multi-tasking scenarios. The main issue addressed by the dissertation is that, which modules deliver task relevant information and how the constrained system resources are distributed among these, can be learned in a reward based framework. This will reduce the partial observability caused by the information gathering process and a better support to other high level cognitive functionalities of the system can be given. An adaptive approach for the information acquisition process also makes it possible to cope with variations in the scenario or environment.

Chapter 2 explained how a modular systems approach facilitates design and utilization of intelligent systems that interact with real-world like dynamic environments to achieve goals. Similar to breaking up a scenario into smaller tasks to handle the complexity, dividing elementary functionalities of the system into modules allows making clear associations between tasks and collections of modules. This requires an architectural formalization that defines aspects like abstractions, representations and information flow in the system. After this, the architectural elements of the proposed system-level approach is presented. This includes the adopted formal system design language and the functional description of the architecture used in this work. Next, Chapter 3 described how a modular system can learn the capabilities of interacting with dynamic environments to achieve goals. Such a developing system requires a value mechanism to monitor and evaluate its own experience with respect to its goals and inform other mechanisms of the system via reward signals. This makes it possible to use a modular reinforcement learning framework to learn policies that map environmental states to actions to execute tasks in multi-tasking scenarios. Reward signals are also used in the proposed information gathering mechanism, referred to as Module Management Learning (MML), to learn task relevant perceptual modules and how to utilize the constrained system resources for them in time. The MML framework learns efficient information acquisition strategies. Consequently the knowledge of the system about the task-relevant scene entities are kept up-to-date, reducing the partial observability of these elements for the modular reinforcement learning mechanisms.

Implementation of the system was done in simulation for two different applications imposing multi-tasking scenarios where hypotheses postulated in the thesis were tested.

The first application, which is presented in Chapter 4, addresses the attention domain for active vision. First off, to motivate this application various potential benefits of an active vision approach to robots are summarized. The advantages gained by the active vision approach in visual depth estimation is further shown with a series of experiments. An observation from these experiments showed that different depth estimation methods have varying characteristics in the task space. This requires a mechanism that arbitrates and coordinates the gaze movements utilized by these methods during the execution of a task. This motivated the implementation of the proposed framework (MML) into such kind of a scenario. This application involves a humanoid robot in a scenario where it has to select, reach and touch one of the three moving objects on a table while keeping interaction with a partner in the scene. The robot can acquire environmental information visually by gazing at the entities in the scene (e.g. objects, partner). Gazing commands are generated by various perceptual modules embedded into the system such as colour tracking modules for red, green and blue, motion detection, human interaction and visual exploration. The fact that two or more modules can generate conflicting gazing commands renders constraints on the physical resources of the system (i.e. camera control). Hence, it is necessary to determine how to distribute the resource among these modules in time. Various aspects of the scenario accommodate challenging problems for the system. For example, dynamic environments (e.g. moving objects or agent) complicate the information acquisition process. The system also has memory constraints, as the scene representation mechanism keeps the scene elements for a limited time unless they are updated. Moreover, the system has no prior knowledge about the crucial elements of the scenarios: the value of the objects (i.e. how much reward an object yields), how dynamic the objects are and how often the partner should be looked at before losing the interaction. The system is also equipped with a high-level local learning mechanism to learn and execute the reaching task (i.e. which object to select for reaching). The proposed MML method is used to learn efficient gazing strategies to support the learning tasks and developmental process of the system. In order to be able to evaluate the performance of the proposed framework, additional methods were implemented. The first of this additional methods is the setting where the system is exposed to all environmental information at every time step without any restriction. The setting is called Perfect World Knowledge (PWK). Since this setting introduces a fully observable process, comparing it with the other settings quantifies their contribution in dealing with partial observability. The second method is a simple implementation of the saliency model presented by Itti and Koch [2001]. This method helps to evaluate how well does an information acquisition method shaped by task constraints through the learning process achieve compared to a purely bottom-up approach. The third implemented method is a random gazing strategy that is a control case to confirm that the performance of MML is not arbitrary.

In the second application, which is presented in Chapter 5, a mobile robot with a limited computational power has to collect target objects while avoiding collisions with obstacles in the environment. Additionally, the system has memory constraints for the same reasons explained in the previous application. The system employs two high-level local learning mechanisms to learn its tasks. The MML method is applied to learn how to distribute computational resources among different perceptual modules efficiently in order to support the high-level task learning and executing mechanisms in the system leading to a smooth development process. In this scenario the classification of objects (i.e. targets and obstacles) is given to the system but dynamic properties (i.e. which types of objects are mobile) are not known. Additionally, the system has memory constraints for the same reasons explained in the previous application. Firstly, a study was done using this application to investigate the benefits of modular system design. For this purpose, two instances of the system are built using a non-modular and modular approaches. The non-modular system learns a policy mapping environmental states to driving actions as a whole. The modular system breaks the scenario into two tasks and learns separate policies for each. The information gathering problem is bypassed in this study (i.e. the environment was fully observable for both systems). Following this, the modular system was tested in the partially observable domain (i.e. with resource constraints). The additional methods implemented in the first application are also used here for comparison (i.e. PWK, Saliency and Random).

The results from the two applications present evidences supporting the hypotheses set in the dissertation. These are summarized below.

The modularity approach helps the system to scale with scenario complexity: this is shown in the autonomous navigation scenario with a series of experiments where the complexity of the scenario is controlled and two systems designed with modular and non-modular approaches are compared. The performance of the modular system was always higher than the non-modular system in terms of acquired rewards and task statistics. Moreover, the decrease of the performance with increasing complexity was smaller in the modular case. These results indicates that a modular approach can better scale with the increasing scenario complexity.

Among various perceptual modules, the MML mechanism can find the ones that deliver task-relevant information so that the system can handle multi-tasking scenarios: in the first application learned gazing strategies were consistent with both reaching and human interaction tasks for all experiments. This was observed as a basic form of an eye-hand coordination for the reaching task and correlation between the boredom parameter and selection rate of the human interaction module for the human interaction task. Also for the second application, an analysis on how often each module is selected during the learning process is made to analyze the information acquisition behaviour learned by the MML framework. This analysis shows that MML can infer task-relevant modules as the learning progresses. Module selection results from similar experiments conducted with different environmental characteristics show that the proposed framework is able to find information gathering strategies specific to the environment.

MML can learn how to distribute constrained system resources among task-relevant modules efficiently in time: in both applications, the overall performance of the system is the best with the MML mechanisms among the methods operating in the partially observable domain (i.e. MML, Saliency and Random). This was shown both in acquired reward in time and task statistics. Since the frequency how often the information from a scene entity should be updated can be related to the unpredictability of the entity (e.g. how dynamic the scene entity is), it could be sufficient to use only such frequencies (learned or derived) for module selection to achieve an efficient information gathering strategy. In that case, learning the temporal sequence of module execution, as done by the proposed MML mechanism, may be unnecessary. To test this additional experiments were conducted. Compared to an information acquisition strategy that only replicates the post-learning selection frequencies of the modules, the system using the MML mechanism showed better performance in both applications. Even though the scenarios in the two applications do not impose any sequence for the information acquisition process, MML achieved better support for the high level learning mechanisms in the system. The reason could be that random sampling of modules creates a higher variance to the information acquisition process for the local learning mechanisms to be able to cope with. The performace gain is higher in the active vision scenario than the autonomous navigation scenario. This shows that the variance changes depending on the scenario.

Together with the scene representation mechanism, MML can support other learning mechanisms in the system by reducing the partial observability introduced by the information acquisition process: comparing the performance of a system with an arbitrary information acquisition strategy to the performance of the system having access to all environmental information (i.e. the PWK case) a metric on how well the utilized information acquisition strategy handles the partial observability can be obtained. In the first application, MML reached 87% of the performance of the fully observable case. Another metric to evaluate the effects of partial observability is to analyze the learned system behaviour. In the first application, a quantitative assessment of the system behaviour is difficult to make due to a large number of variables in the environment. Instead, results from two representative experiments were shown to highlight the fact that MML learns specialized strategies depending on the environment or scenario. The second application was more convenient for examining the elements of local learning mechanisms since the variability in the environment was kept lower compared to the first application. A comparison of the learned Q-Functions under the influence of MML, PWK and Saliency showed that task policies learned in the system using MML are more similar to the ones learned in the fully observable case (i.e. PWK). This is an indication that a partially observable process could be reduced to a fully observable process with an efficient information gathering strategy which can be learned via the MML mechanism.

The framework introduces flexibility to the system, allowing it to find efficient solutions for various scenario settings and adapt to changes in the environment affecting perceptual abilities of the system: in both applications the proposed framework managed to adapt to varying environmental conditions e.g. for the first application, changing scenario variables such as values and dynamic properties of objects, maximum hand velocity and boredom properties of the interaction partner; for the second application, varying number of objects in different environments. Furthermore, online adaptation capability of the proposed framework is also examined. For this purpose, in the first application after the learning process is done (i.e. at epoch 500) one randomly selected module is turned off. This simulates the case where the stopped module is broken down and cannot deliver any information. In these experiments, MML was able to adapt its gazing strategy to the new situation. Simple perceptual changes such as the one explained here can be coped with the MML mechanism. Changes affecting the scenario to a larger extent (e.g. changes in the optimal policy) may be harder to adapt due to limited adaptation capabilities of the other local learning mechanisms. Hence, the overall flexibility of the system may vary depending on which algorithm is used for other learning problems in the system.

6.2 Discussion

Based on the findings summarized in Section 6.1 potential benefits of the proposed framework to intelligent developing systems are discussed in this section. In an overview, the proposed framework:

- facilitates the design process of intelligent developing systems and provides a structured information acquisition and processing functionalities to the system,
- addresses the partial observability problem that occurs due to constrained resources for information acquisition in an elegant way,
- introduces flexibility to the system allowing it to adapt to different environments,
- achieves the learning process online within a relatively short time,
- autonomously modifies exploration/exploitation trade-off of the information acquisition behaviour supporting multiple learning tasks running in parallel,
- can generate reactive behaviour in the absence of tasks,
- can support parallel-running learning processes with different characteristics.

The proposed framework facilitates the design process of large scale systems that needs arbitration and scheduling of motor commands from various information acquisition modules. The modular systems approach adopted in the proposed framework implements individual functionalities of the system as stand alone modules. This structures the information acquisition problem in a formal way by how to distribute constrained system resources among modules to support the system. Thus, modules form a bridge in the information flow from the environmental states to various task execution or learning mechanisms in the system. Using the modular approach also facilitates the implementation, testing and maintenance of elemental functionalities of the system. Reward mediated learning of the module management further eases the system design. Employment of the whole framework requires minimal effort to formalise tasks for the system since it requires only a coarse feedback of the actions the system takes encoded as reward signals. Moreover, implementing task policies (designed or learned) does not have to deal with the information acquisition process since these two mechanisms are separated. Using this feature the system can rapidly be extended with additional task policies and utilised for different purposes.

The proposed framework introduces a novel approach to handle partial observabilities caused by the necessity of sharing constrained resources for the information acquisition process. The usual approaches to take in such cases are: (a) living with partial observability, (b) manually designing a strategy for the information acquisition process, (c) using algorithms that can handle partial observability in the high level local learning mechanisms. Approach (a) degrades the overall performance of the system. This could be tolerable for simple systems but fails at some point as the complexity grows. Approach (b) could also be convenient for simple systems but is impractical for complex systems. Additionally, pre-programmed policies are likely to fail against variations in the environment. Approach (c) imposes an implementation of complex algorithms that require further computational power and memory. The proposed approach learns efficient information acquisition strategies. This is a flexible solution that can adapt to different environments and requires no manual programming. Efficient strategies can reduce the effect of partial observability so that the interaction of the system can be treated as a fully observable case by the higher level local learning and task executing mechanisms. Hence, simple machine learning algorithms can be implemented for such local learning mechanisms. Moreover, the proposed method does not require any model of the environment or uncertainties in the information acquisition process which can be difficult to derive.

A learning based approach introduces flexibility to the system, making it possible to find effective strategies under different environmental conditions. Moreover, the proposed framework is able to adapt to environmental changes, which affect the perceptual properties of the system. However, the overall flexibility of the system to such changes is also limited by the other local learning mechanisms present in the system.

For both applications, learning times are around 400 epochs which is a fairly short time for learning approaches using rewards (i.e. RL, Sutton and Barto [1998]). In the active vision scenario, the whole learning process can be estimated to take about an hour to complete. For a developing system that gains re-usable knowledge about the environment through the learning process, this time is reasonable. In the second application the learning process takes around 1000 epochs. However, the system in this application accommodates further reinforcement learning mechanisms that require additional time for the overall learning process to be completed.

Contributions to Developmental Systems Research

In addition to the results supporting hypotheses of the dissertation, experiments conducted with two applications revealed further findings which may be insightful for developmental systems research. For example, there is accumulated evidence that the information gathering process heavily influences the development process of systems. In both applications presented in this dissertation systems using various methods for the information gathering process result in varying learning performances. The discrepancy is distinguishable at micro-scale if the learned policies are compared to each other. The policies for the object selection task (in the active vision application) or the target collection and collision avoidance tasks (in the autonomous navigation application) learned under different information acquisition methods are different from each other. This signifies that the information acquisition strategy affects local learning mechanisms in developing systems.

In the presented systems using MML for information acquisition, policy learning and module management learning are dissociated mechanisms. The only signal these mechanisms share is rewards received as the consequence of systems actions. Despite of this, these mechanisms are able to support each other. In the active vision application, an analysis of the relation between the object selection policy for reaching and gazing strategy shows the co-development of both mechanisms. The results show correlations between the colour of the selected object and the CTM for the same colour in selection ratings. Hence, a form of eye-hand coordination is achieved by using MML even though the underlying control mechanisms are fairly separated.

In essence, the framework shapes a bottom-up information acquisition process with the requirements from high-level learning and task execution processes. Hence, the system is provided with reactive behaviour for information acquisition that benefits other local learning mechanisms embedded in the system by enhancing the exploration for the learning process. This was demonstrated by the low performance of the system using a purely task-driven information gathering strategy. The framework gains a task-oriented character after learning. It can be argued that this could constraint the reactive capacities of the system. The information acquisition behaviour is governed by the parameters of the module management mechanism. In the absence of tasks these parameters can be set to pre-defined values that would generate the default reactive behaviour, so the reactive capacities of the system will be retained.

The proposed framework can support parallel running learning processes with diverse characteristics. This is shown in the active vision application where the tasks have different attributes. The human interaction task requires passive observation of a partner in the scene whereas the reaching task involves actively manipulating the environment and sensory-motor coordination. Both tasks have different complexity (the human interaction task is probably easier to learn) and temporal requirements for actions (the reaching task requires continuous actions over a time period while the human interaction task requires periodic actions). The system with the MML mechanism was able to support the system to develop into a competent state where both tasks can be achieved in parallel.

6.3 Outlook

The utilization of the proposed framework is demonstrated in two applications for a humanoid and an autonomous navigation agent in simulation. Dynamic scene elements and variabilities in scenarios are incorporated in these simulations in order to represent real world situations as closely as possible. Hence, the framework can be used in applications running in real-world environments. A hardware implementation of the proposed system (e.g. on the iCub humanoid or a mobile robot) can further test this and reveal properties in need of improvement. Additionally, the applications presented in this thesis covers constraints in physical, computational and memory resources. Contributions of the framework in applications to other constraints such as cost, energy etc., or different combinations of such constraints can be investigated.

One feature of the system is that it separates the control mechanisms for information acquisition and task execution. However this separation is not absolute and task policies and task-level action selection can have a top-down influence on the information acquisition process. For example, in the ‘reaching while interacting’ scenario once the rewarding object is inferred by the reinforcement learning, this information can be sent back to the module management system as candidates for gaze targets. This may especially be useful in the case where the objects are not in the field of view as their position information is kept in the scene representation mechanism. However, since the objects are dynamic this information may be imprecise. The resulting gazing action may land inaccurately in the vicinity of the object. From this point on the relevant bottom-up module (e.g. a colour tracking module) may lead the gaze to the object again. In such an approach top-down influence of tasks is not absolute, rather cooperative.

The demonstrations of the proposed MML mechanism showed its use in the information gathering process, i.e. efficient utilization of the perceptual modules are learned in the applications. This can be extended to the other modules responsible for information processing such as an object classification module. Hence, the collections of perceptual and processing modules and how to utilize them in time can be learned for different scenarios. A further promising line of research that can follow this work is the life-time knowledge acquisition for module management. The idea is to store the knowledge of the module management strategies learned for different scenarios. A relevant strategy can be restored whenever the system encounters one of the scenarios again throughout its life time. The proposed framework can already react to changes in a scenario by regulating the learning process based on the overall performance of the system. Additionally, an appropriate representation for tasks and scenarios would be required in order to make associations between learned module management settings and scenarios. The proposed framework can form a basis for such life-long developing systems by providing them with the necessary flexibility in the information flow within the system.

References

J. Alomoinos, I. Weiss, and A. Bandopadhay. Active vision. *International Journal on Computer Vision*, 1988.

A. Andreopoulos, S. Hasler, H. Wersing, H. Janssen, J. Tsotsos, and E. Körner. Active 3D object localization using a humanoid robot. In *IEEE Transactions on Robotics*. IEEE, 2011.

M. A. Arbib. *Schema Theory*, pages 1427–1443. Wiley, second edition, 1992.

A. Baddeley. Working memory: looking back and looking forward. *Nature Reviews Neuroscience*, 4(10):829–839, October 2003. ISSN 1471-003X. doi: 10.1038/nrn1201. URL `http://dx.doi.org/10.1038/nrn1201`.

A. C. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, October 2003. ISSN 0924-6703. doi: 10.1023/A:1025696116075. URL `http://dx.doi.org/10.1023/A:1025696116075`.

R. Beira, M. Lopes, M. Praga, J. Santos-Victor, A. Bernardino, G. Metta, F. Becchi, and R. Saltaren. Design of the robot-cub (iCub) head. In *Proceedings of 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 94–100, June 2006. doi: 10.1109/ROBOT.2006.1641167. URL `http://dx.doi.org/10.1109/ROBOT.2006.1641167`.

R. Bellman. *Dynamic Programming*. Dover Books on Mathematics. Princeton University Press, 2003 edition, 1957. ISBN 9780486428093. URL `http://books.google.de/books?id=fyVtp3EMxasC`.

S. R. Bieniawski, I. M. Kroo, and D. H. Wolpert. Flight control with distributed effectors. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.64.3498`.

B. Bolder, H. Brandl, M. Heracles, H. Janssen, I. Mikhailova, J. Schmüdderich, and C. Goerick. Expectation-driven autonomous learning and interaction system. In *IEEE-RAS International Conference on Humanoid Robots*, 2008.

M. M. Botvinick, Y. Niv, and A. C. Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–280, December 2009. ISSN 1873-7838. doi: 10.1016/j.cognition.2008.08.011. URL http://dx.doi.org/10.1016/j.cognition.2008.08.011.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 1st edition, October 2008. ISBN 0596516134. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0596516134.

R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23, March 1986. ISSN 0882-4967.

R. A. Brooks. Intelligence without representation. *Artif. Intell.*, 47(1-3):139–159, February 1991. ISSN 0004-3702. doi: 10.1016/0004-3702(91)90053-M. URL http://dx.doi.org/10.1016/0004-3702(91)90053-M.

M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008, 2003.

D. Busquets, R. L. de Màntaras, R. L. Opez De, M. Antaras, C. Sierra, and T. G. Dietterich. Reinforcement learning for landmark-based robot navigation. In *Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 841–843, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.4338.

S. Chernova and R. C. Arkin. From deliberative to routine behaviors: A cognitively inspired action-selection mechanism for routine behavior capture. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 15(2):199–216, June 2007. ISSN 1059-7123. doi: 10.1177/1059712306076255. URL http://dx.doi.org/10.1177/1059712306076255.

N. Chomsky. *Rules and representations*. Blackwell, Oxford, 1980.

M. M. Chun. Visual working memory as visual attention sustained internally over time. *Neuropsychologia*, 49(6):1407–1409, 2011. URL http://www.ncbi.nlm.nih.gov/pubmed/21295047.

A. Dankers, N. Barnes, and A. Zelinsky. Active vision - rectification and depth mapping. In *Australasian Conf. on Robotics and Automation*, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.131.8177.

B. Dittes and C. Goerick. A language for formal design of embedded intelligence research systems. *Robotics and Autonomous Systems*, 2011.

B. Dittes, M. Heracles, T. Michalke, R. Kastner, A. Gepperth, J. Fritsch, and C. Goerick. A hierarchical system integration approach with application to visual scene exploration for driver assistance. In *Proceedings of the 7th International Conference on Computer Vision Systems, LiÃ©ge, Belgium, Octobre 13-15, 2009*, pages 255–264. Springer, 2009.

K. Doya, K. Samejima, K. Katagiri, and M. Kawato. Multiple model-based reinforcement learning. *Neural Comput.*, 14(6):1347–1369, June 2002. ISSN 0899-7667. doi: 10.1162/089976602753712972. URL `http://dx.doi.org/10.1162/089976602753712972`.

D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cereb Cortex*, pages 1–47, 1991.

J. A. Fodor. *The Modularity of Mind*. The MIT Press, April 1983. ISBN 0262560259. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262560259`.

R. M. Foerster, E. Carbone, H. Koesling, and W. X. Schneider. Saccadic eye movements in the dark while performing an automatized sequential high-speed sensorimotor task. *Journal of Vision*, 12(2), 2012. doi: 10.1167/12.2.8. URL `http://www.journalofvision.org/content/12/2/8.abstract`.

C. Goerick, B. Bolder, H. Janssen, M. Gienger, H. Sugiura, M. Dunn, I. Mikhailova, T. Rodemann, H. Wersing, and S. Kirstein. Towards incremental hierarchical behavior generation for humanoids. In *IEEE-RAS International Conference on Humanoids 2007*. IEEE, 2007.

C. Goerick, J. Schmüdderich, B. Bolder, H. Janssen, M. Gienger, A. Bendig, M. Heckmann, T. Rodemann, H. Brandl, X. Domont, and I. Mikhailova. Interactive online multimodal association for internal concept building in humanoids. In *IEEE-RAS International Conference on Humanoids 2009*. IEEE, 2009.

M. E. Goldberg. *The Control of Gaze*, pages 782–800. McGraw-Hill, fourth edition, 2000.

J. Gottlieb and P. Balan. Attention as a decision in information space. *Trends in Cognitive Sciences*, 14(6):240–248, 2010. URL `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2908203&tool=pmcentrez&rendertype=abstract`.

D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press, new edition, June 2002. ISBN 0805843000. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0805843000`.

B. Hollander. *In search of the soul and the mechanism of thought, emotion, and conduct*. Number 2. K. Paul, Trench, Trubner & Co., ltd., 1920. URL `http://books.google.de/books?id=71G40or5_IAC`.

L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, March 2001. ISSN 1471-003X. doi: 10.1038/35058500. URL `http://dx.doi.org/10.1038/35058500`.

H. Jacobsson, N. Hawes, G. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. *Proceedings of the 3rd international conference on human robot interaction (HRI)*, page 81, 2008. URL http://dx.doi.org/10.1145/1349822.1349834.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, May 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(98)00023-X. URL http://dx.doi.org/10.1016/S0004-3702(98)00023-X.

C. Karaoguz, M. Dunn, T. Rodemann, and C. Goerick. Online adaptation of gaze fixation for a stereo-vergence system with foveated vision. In *Proceedings of the IEEE International Conference on Advanced Robotics 2009*, pages 1–6, July 2009. ISBN 978-1-4244-4855-5. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5174804.

C. Karaoguz, A. Dankers, T. Rodemann, and M. Dunn. An analysis of depth estimation within interaction range. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3207–3212. IEEE, October 2010. ISBN 978-1-4244-6674-0. doi: 10.1109/IROS.2010.5650054. URL http://dx.doi.org/10.1109/IROS.2010.5650054.

C. Karaoguz, T. Rodemann, and B. Wrede. Optimisation of gaze movement for multitasking using rewards. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1187–1193. IEEE, 2011a. ISBN 978-1-61284-454-1. doi: 10.1109/IROS.2011.6048191. URL http://dx.doi.org/10.1109/IROS.2011.6048191.

C. Karaoguz, T. H. Weisswange, T. Rodemann, B. Wrede, and C. A. Rothkopf. Reward-based learning of optimal cue integration in audio and visual depth estimation. In *Proceedings of the 15th International Conference on Advanced Robotics (ICAR)*, pages 389–395. IEEE, June 2011b. ISBN 978-1-4577-1158-9. doi: 10.1109/ICAR.2011.6088550. URL http://dx.doi.org/10.1109/ICAR.2011.6088550.

C. Karaoguz, T. Rodemann, and B. Wrede. Online learning of information acquisition for developing robots. In *IEEE/RSJ International Conference on Robotics and Automation (ICRA), submitted*. IEEE, 2013a.

C. Karaoguz, T. Rodemann, B. Wrede, and C. Goerick. Learning information acquisition for multi-tasking scenarios in dynamic environments. *IEEE Transactions in Autonomous Mental Development, accepted*, 2013b.

S. Khademi, A. Darudi, and Z. Abbasi. A sub pixel resolution method. In *Proceedings of World Academy of Science, Engineering and Technology*, 2010.

S. Kirstein, H. Wersing, and E. Körner. A biologically motivated visual memory architecture for online learning of objects. *Neural Networks*, 21(1):65–77, 2008.

E. I. Knudsen. Fundamental components of attention. *Annual Review of Neuroscience*, 30(1):57–78, April 2007. ISSN 0147-006X. doi: 10.1146/annurev.neuro.30.051606.094256. URL http://dx.doi.org/10.1146/annurev.neuro.30.051606.094256.

K. Konolige. Small vision systems: hardware and implementation. In *Eighth Int. Symposium on Robotics Research*, page 111–116, 1997.

C. Kwok and D. Fox. Reinforcement learning for sensing strategies. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 3158–3163. IEEE, 2004. ISBN 0-7803-8463-6. doi: 10.1109/IROS.2004.1389903. URL `http://dx.doi.org/10.1109/IROS.2004.1389903`.

M. F. Land and B. W. Tatler. *Looking and Acting*. Oxford University Press, 2009. ISBN 9780198570943.

Z. Li and L. Itti. Saliency and gist features for target detection in satellite images. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 20(7):2017–2029, July 2011. ISSN 1941-0042. doi: 10.1109/TIP.2010.2099128. URL `http://dx.doi.org/10.1109/TIP.2010.2099128`.

P. McCorduck. *Machines Who Think*. Natick, MA: A K Peters, Ltd., 25th anniversary edition, 2004. ISBN 1-56881-205-1. URL `http://www.pamelamc.com/html/machines_who_think.html`.

I. Mikhailova. *Internal control for autonomous open-ended acquisition of new behaviors*. PhD thesis, 2009. URL `http://pub.uni-bielefeld.de/publication/2301987`. Bielefeld University.

S. Moeller, W. A. Freiwald, and D. Y. Tsao. Patches with links: A unified system for processing faces in the macaque temporal lobe. *Science*, 320(5881):1355–1359, June 2008. doi: 10.1126/science.1157436. URL `http://dx.doi.org/10.1126/science.1157436`.

Y. Nagai, A. Nakatani, and M. Asada. How a robot's attention shapes the way people teach. In *Proceedings of the 10th International Conference on Epigenetic Robotics*, pages 81–88, November 2010.

H. Nakashima and I. Noda. Dynamic subsumption architecture for programming intelligent agents. In *Proceedings of 1998 International Conference on Multi Agent Systems*, pages 190–197. IEEE, July 1998. ISBN 0-8186-8500-X. doi: 10.1109/ICMAS.1998.699049. URL `http://dx.doi.org/10.1109/ICMAS.1998.699049`.

V. Navalpakkam and L. Itti. Modeling the influence of task on attention. *Vision Research*, 45(2):205–231, January 2005. ISSN 00426989. doi: 10.1016/j.visres.2004.07.042. URL `http://dx.doi.org/10.1016/j.visres.2004.07.042`.

W. T. Newsome and E. B. Park. A selective impairment of motion perception following lesions of the middle temporal visual area (MT). *Journal of Neuroscience*, 8:2201–2211, 1988.

D. Ognibene, G. Pezzulo, and G. Baldassarre. How can bottom-up information shape learning of top-down attention-control skills? In *Proceedings of the IEEE 9th International Conference on Development and Learning (ICDL)*, pages 231–237. IEEE, August 2010. ISBN 978-1-4244-6900-0. doi: 10.1109/DEVLRN.2010.5578839. URL http://dx.doi.org/10.1109/DEVLRN.2010.5578839.

F. Orabona, G. Metta, and G. Sandini. A proto-object based visual attention model. In Lucas Paletta and Erich Rome, editors, *Attention in Cognitive Systems. Theories and Systems from an Interdisciplinary Viewpoint*, volume 4840 of *Lecture Notes in Computer Science*, pages 198–215. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-77342-9. URL http://dx.doi.org/10.1007/978-3-540-77343-6_13.

E. B. Pacis, Hobart R. Everett, N. Farrington, and D. J. Bruemmer. Enhancing functionality and autonomy in man-portable robots. In *Defense & Security Symposium 2004*, volume 5422, pages 355+. The International Society for Optical Engineering., September 2004. doi: 10.1117/12.553020. URL http://dx.doi.org/10.1117/12.553020.

S. E. Palmer. *Vision Science: Photons to Phenomenology*. The MIT Press, first edition, May 1999. ISBN 0262161834. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262161834.

A. J. Palomino, R. Marfil, J. P. Bandera, and A. Bandera. A novel biologically inspired attention mechanism for a social robot. *EURASIP J. Adv. Signal Process*, 2011:4:1–4:10, January 2011. ISSN 1110-8657. doi: 10.1155/2011/841078. URL http://dx.doi.org/10.1155/2011/841078.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning*, pages 752–759, 2008.

G. Pezzulo and G. Calvi. A schema based model of the praying mantis. In *Proceedings of the Ninth International Conference on Simulation of Adaptive Behaviour, volume LNAI 4095*, pages 211–223. Springer Verlag, 2006.

M. I. Posner and S. E. Petersen. The attention system of the human brain. *Annual Review of Neuroscience*, 13(1):25–42, 1990. URL http://www.ncbi.nlm.nih.gov/pubmed/2183676.

C. A. Rothkopf and D. Ballard. Credit assignment in multiple goal embodied visuomotor behavior. *Frontiers in Psychology*, 1(00173), 2010. ISSN 1664-1078. doi: 10.3389/fpsyg.2010.00173. URL http://www.frontiersin.org/Journal/Abstract.aspx?s=194&name=cognition&ART_DOI=10.3389/fpsyg.2010.00173.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, second edition, December 2002. ISBN 0137903952. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0137903952.

F. Santini, R. Nambisan, and M. Rucci. Active 3D vision through gaze relocation in a humanoid robot. *Int. Journal of Humanoid Robotics*, 6(3):481–503, September 2009. ISSN 0219-8436.

D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, 2001. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.4136`.

W. Schultz. A Neural Substrate of Prediction and Reward. *Science*, 275:1593–1599, 1997. doi: 10.1126/science.275.5306.1593.

R. Shadmehr. Control of movements and temporal discounting of reward. *Current Opinion in Neurobiology*, 20(6):726–730, 2010. URL `http://www.ncbi.nlm.nih.gov/pubmed/20833031`.

T. Shibata, S. Vijayakumar, J. Conradt, and S. Schaal. Humanoid oculomotor control based on concepts of computational neuroscience. *Humanoids2001, Second IEEE-RAS Intl. Conf. on Humanoid Robots, Waseda Univ., Japan*, pages 278–285, 2001.

D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006. ISBN 0471708585.

J. Simpson, C. L. Jacobsen, and M. C. Jadud. Mobile Robot Control: The Subsumption Architecture and Occam-Pi. In Frederick R. M. Barnes, Jon M. Kerridge, and Peter H. Welch, editors, *Communicating Process Architectures 2006*, pages 225–236. IOS Press, sep 2006. ISBN 1-58603-671-8.

N. Sprague, D. Ballard, and A. Robinson. Modeling embodied visual behaviors. *ACM Trans. Appl. Percept.*, 4(2):11+, 2007. ISSN 1544-3558. doi: 10.1145/1265957.1265960. URL `http://dx.doi.org/10.1145/1265957.1265960`.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998. ISBN 0262193981. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262193981`.

V. Tikhanoff, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, and A. Cangelosi. An open source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator. In *Proc. of the Workshop on Performance Metrics for Intelligent Systems, National Institute of Standards and Technology, Washington DC, USA*, August 2008.

H. Toutounji, C. A. Rothkopf, and J. Triesch. Scalable reinforcement learning through hierarchical decompositions for weakly-coupled problems. In *IEEE International Conference on Development and Learning*, volume 2, pages 1–7, 2011. doi: 10.1109/DEVLRN.2011.6037351.

J. Trommershäuser, P. W. Glimcher, and K. R. Gegenfurtner. Visual processing, learning and feedback in the primate eye movement system. *Trends in neurosciences*, 32(11):583–590, November 2009. ISSN 1878-108X. doi: 10.1016/j.tins.2009.07.004. URL `http://dx.doi.org/10.1016/j.tins.2009.07.004`.

G. G. Turrigiano and S. B. Nelson. Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, 5(2):97–107, February 2004. ISSN 1471-003X. doi: 10.1038/nrn1327. URL `http://dx.doi.org/10.1038/nrn1327`.

D. Vernon, G. Metta, and G. Sandini. The iCub cognitive architecture: Interactive development in a humanoid robot. In *Proceedings of the IEEE 6th International Conference on Development and Learning*, pages 122–127. IEEE, July 2007. ISBN 978-1-4244-1116-0. doi: 10.1109/DEVLRN.2007.4354038. URL `http://dx.doi.org/10.1109/DEVLRN.2007.4354038`.

A.-L. Vollmer. *Measurement and analysis of interactive behavior in tutoring action with children and robots*. PhD thesis, 2011. University of Bielefeld.

D. H. Wolpert. *Information Theory - The Bridge Connecting Bounded Rational Game Theory and Statistical Physics*. Perseus books, February 2004. URL `http://arxiv.org/abs/cond-mat/0402508`.

D. H. Wolpert and N. Kulkarni. Game-theoretic management of interacting adaptive systems. In *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems*, 2008. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.4528`.

R. Yan, T. Rodemann, and B. Wrede. Learning of audiovisual integration. In *Proceedings of the IEEE 2011 International Conference on Development and Learning*, Frankfurt, Germany, 2011. Springer.

A. F. Yarbus. *Eye Movements and Vision*. 1967.

C. Zhang, V. Willert, and J. Eggert. Tracking with depth-from-size. In *Neural Information Processing, 15th Int. Conference*, pages 274–283, 2009.