Integrated Robot Planning and Control with Extended Kohonen Maps

Low Kian Hsiang

National University of Singapore

2002

Name: Low Kian Hsiang
Degree: Master of Science
Dept: Computer Science
Thesis Title: Integrated Robot Planning and Control with Extended Kohonen Maps.

Abstract

The problem of goal-directed, collision-free motion in a complex, unpredictable environment can be solved by tightly integrating high-level deliberative planning with low-level reactive control. This thesis presents two such architectures for a nonholonomic mobile robot. To achieve real-time performance, reactive control capabilities have to be fully realized so that the deliberative planner can be simplified. These architectures are enriched with reactive target reaching and obstacle avoidance modules. Their target reaching modules use indirect-mapping Extended Kohonen Map to provide finer and smoother motion control than direct-mapping methods. While one architecture fuses these modules indirectly via command fusion, the other one couples them directly using cooperative Extended Kohonen Maps, enabling the robot to negotiate unforeseen concave obstacles. The planner for both architectures use a slippery cells technique to decompose the free workspace into fewer cells, thus reducing search time. Any two points in the cell can still be traversed by reactive motion.

Keywords: Integrated Planning and Control
Indirect-Mapping Extended Kohonen Map
Cooperative Extended Kohonen Maps
Goal-Directed, Collision-Free Motion in Complex, Unpredictable Environment
Slippery Cells
Nonholonomic Mobile Robot

Integrated Robot Planning and Control with Extended Kohonen Maps

Low Kian Hsiang
(B.Sc. (Hon.) Computer and Information Sciences, NUS)

A Thesis submitted
for the Degree of Master of Science
Department of Computer Science
School of Computing
National University of Singapore
2002

Acknowledgments

I would like to express my gratitude to my project supervisors, A/P Leow Wee Kheng and A/P Marcelo H. Ang Jr., for providing timely advice and guidance during the course of my honours and masters years.

I would like to thank Ms Yeo Hwee Kiang Cocoa for her love, care and concern.

Lastly, I will always be grateful to my family for their constant support when I needed them most.

Publications

This thesis is based on research that was previously reported in the following publications.

1 Kian Hsiang Low, Wee Kheng Leow, and Marcelo H. Ang Jr. Integrated Planning and Control of Mobile Robot with Self-Organizing Neural Network. In *Proceedings of 18th IEEE International Conference on Robotics and Automation*, volume 4, pages 3870–3875, 2002.

2 Kian Hsiang Low, Wee Kheng Leow, and Marcelo H. Ang Jr. A Hybrid Mobile Robot Architecture with Integrated Planning and Control. In *Proceedings of 6th International Conference on Autonomous Agents*, 2002.

Contents

Acknowledgments **i**

Publications **ii**

Table of Contents **iii**

List of Figures **vii**

Summary **ix**

1 Introduction **1**

1.1 Background 1

1.1.1 Deliberation vs. Reactivity 2

1.2 Motivation 5

1.2.1 Why Hybridize? 5

1.2.2 The Art of Hybridization 5

1.2.3 Rethinking Hybridization: Towards Real-Time Performance 6

1.2.4 Empowering Reactivity in a Hybrid System 6

1.3 Thesis Objectives 8

1.4 Overview of Integrated Planning and Control Architectures 9

2 Related Work **15**

2.1 Hybrid Deliberative/Reactive Architectures 15

2.1.1 Top-Down Approach: Integrated Cognitive Architectures 16

2.1.2 Bottom-Up Approach: Integrated Planning and Control Architectures 19

2.1.3 Continuous vs. Discrete Response Encoding 21

2.2 High-Level Deliberative Motion Planning 22

2.2.1 Roadmap 23

2.2.2 Cell Decomposition 23

2.2.3 Potential Field 24

2.3 Low-Level Reactive Motion Control 25

2.3.1 Numerical Techniques 26

2.3.2 Fuzzy Logic 28

2.3.3 Multilayer Perceptrons 29

2.3.4 Reinforcement Learning 29

2.3.5 Feature Maps 30

3 High-Level Deliberative Motion Planning **31**

3.1 Motion Planning: Workspace vs. Configuration Space 31

3.2 Approximate Cell Decomposition: Slippery Cells 32

3.2.1 What is a Slippery Cell? 33

3.2.2 Cell Representation of Free Space 35

3.2.3 Checkpoints Generation 37

3.3 Summary 40

4 Low-Level Reactive Motion Control I **41**

4.1 Target Reaching: Direct-Mapping EKM 41

4.2 Target Reaching: Indirect-Mapping EKM 43

4.3 Self-Organized Training of Indirect-Mapping EKM 45

4.4 Obstacle Avoidance: Braitenberg Type-3C Vehicle 47

4.5 Behavioral Coordination: Command Fusion 49

4.6 Summary 49

5 Low-Level Reactive Motion Control II **50**

5.1 Cooperative Extended Kohonen Maps 50

5.2 Target Reaching EKMs 51

5.3 Obstacle Avoidance EKMs 53

5.4 Summary 55

6 Experiments and Discussion **56**

6.1 Quantitative Evaluation 56

6.2 Qualitative Evaluation 60

6.2.1 Command Fusion: Target Reaching with Short-Range Obstacle Avoidance 61

6.2.2 Behavioral Coordination: Command Fusion vs. Dynamic Neural Fields 64

7 Future Work **70**

7.1 Improvements 70

7.1.1 Reactive Cognizant Failure Capabilities 70

7.1.2 Graph Connectivity and Search Algorithms 71

7.2 Extensions 71

7.3 Applications 71

8 Conclusion **72**

Bibliography **76**

List of Figures

1.1 Robot search and rescue at WTC disaster site 2

1.2 Deliberative planning vs. reactive control framework 3

1.3 Integrated planning and control framework 9

1.4 Integrated planning and control architectures 11

3.1 Slippery vs. non-slippery cell 33

3.2 Robot workspace 34

3.3 Approximate cell decomposition methods 36

3.4 Procedure to add a free bitmap cell to a slippery cell 38

3.5 Algorithm to label slippery cells 39

3.6 Checkpoints generation 39

4.1 Target reaching with direct-mapping EKM 42

4.2 Target reaching with indirect-mapping EKM 44

4.3 Obstacle avoidance with Braitenberg Type-3C vehicle 48

5.1 Cooperative EKMs with indirect mapping 51

6.1 Performance comparison with mean positioning error, target reaching probability, normalized time-to-target and mean deviation from straight line trajectory 58

6.2 Self-organization results of EKM using direct and indirect mapping 60

6.3 Motion of robot in an environment with an unforeseen static obstacle 62

6.4 Motion of robot in an environment with an unforeseen moving obstacle . . . 63

6.5 Motion of robot in an environment that had changed unexpectedly. 63

6.6 Comparison of robot motions in an environment with an unforeseen stationary obstacle between "command fusion" and "dynamic neural fields" architectures 65

6.7 Negotiating unforeseen extended obstacle 67

6.8 Negotiating unforeseen concave obstacle 68

Summary

The problem of goal-directed, collision-free motion in a complex, unpredictable environment can be solved by tightly integrating high-level deliberative planning with low-level reactive control. This thesis presents two such architectures for a nonholonomic mobile robot. To achieve real-time performance, reactive control capabilities have to be fully realized so that the deliberative planner can be simplified. These architectures are enriched with reactive target reaching and obstacle avoidance modules. Their target reaching modules use indirect-mapping Extended Kohonen Map to provide finer and smoother motion control than direct-mapping methods. While one architecture fuses these modules indirectly via command fusion, the other one couples them directly using cooperative Extended Kohonen Maps, enabling the robot to negotiate unforeseen concave obstacles. The planner for both architectures use a slippery cells technique to decompose the free workspace into fewer cells, thus reducing search time. Any two points in the cell can still be traversed by reactive motion.

Chapter 1

Introduction

1.1 Background

There have been many advances in mobile robotics over the past two decades, yet it is not pervasive in our daily lives. This is mainly due to the difficulties of achieving real-time performance and precise, smooth control while executing a sophisticated task in our complex and unpredictable world. Above all, the fundamental problem of autonomous goal-directed, collision-free motion in an unstructured environment has been the crux of much recent research efforts but still, no sufficing solution has emerged to endorse the ubiquitous use of mobile robots in our real world. Many mobile robotic tasks in the area of service and field robotics (Schraft and Schraft, 1999; Shastri, 1999; Zelinsky, 1997), in particular, face this similar problem. They include sewer inspection (Hertzberg *et al.*, 1998; Rome *et al.*, 1999), cleaning and housekeeping (Fiorini *et al.*, 2000), surveillance (Everett and Gage, 1999; Massios *et al.*, 2001), search and rescue (Fig. 1.1) (Davids, 2002; Murphy *et al.*, 2002), construction (Gambao and Balaguer, 2002), and tour guides (Burgard *et al.*, 1998, 1999). All these tasks require the autonomous mobile robot to perform target or goal reaching

Figure 1.1: Robot search and rescue at WTC disaster site. The hazardous surroundings of WTC may risk the lives of the men in the rescue team if they search this danger zone, as the ramshackle structures may collapse anytime. Hence, robots are employed for this reconnaissance task. Photo courtesy of Center for Robot-Assisted Search and Rescue, University of South Florida.

movements (also known as beacon aiming (Collett, 1996) or homing (Rao and Fuentes, 1995)) while avoiding undesirable and potentially dangerous impact with obstacles in dynamic and populated environments.

1.1.1 Deliberation vs. Reactivity

Traditionally, two dichotomous faculties of robot motion research have separately attempted to solve the problem of goal-directed, collision-free robot motion: *high-level deliberative motion planning* and *low-level reactive motion control.* Robot architectures (Fig. 1.2(a)) (Barraquand and Latombe, 1991; Borenstein and Koren, 1991b; Brooks, 1983; Chatila and Laumond, 1985; Fikes and Nilsson, 1971; Giralt *et al.*, 1979; Elfes, 1986; Giralt *et al.*, 1984; Latombe, 1991; Lozano-Pérez, 1987; Moravec and Cho, 1989; Nilsson, 1980; Zelinsky *et al.*, 1993; Zelinsky and Yuta, 1993b) that employ high-level deliberative motion planning display the following characteristics:

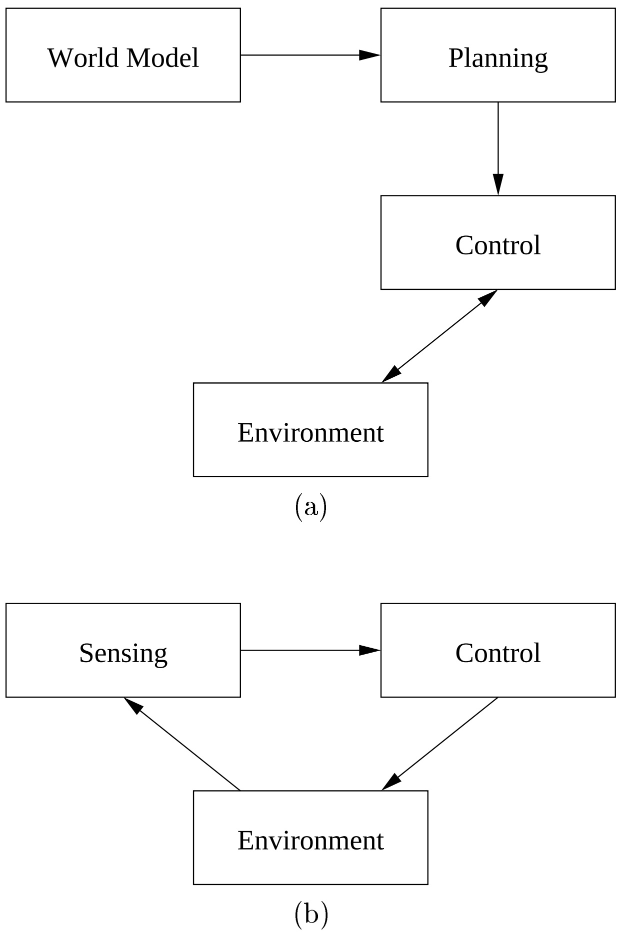


Figure 1.2: Deliberative planning vs. reactive control framework. (a) Deliberative motion planning framework: A world model is used to determine an optimal sequence of motion commands to achieve the desired goal state prior to motion execution. Once a plan has been determined, it is executed by the control. (b) Reactive motion control framework: The locally sensed information about the environment is directly converted into a motion control command, which can be performed very efficiently. This approach is, however, susceptible to local minima.

With an *a priori* global world model (i.e., representation-dependent),

1. a globally optimal sequence of collision-free actions can be determined prior to execution,
2. a globally specified goal that is situated in a complex environment can be achieved only if the world model is accurate at the time of motion execution.

By planning the entire complete path prior to motion execution,

3. unforeseen obstacles in a dynamic environment cannot be negotiated during runtime,
4. a heavy computational burden is incurred, resulting in slow response.

On the other hand, robot architectures (Fig. 1.2(b)) (Agre and Chapman, 1987; Arkin, 1987, 1989b, 1992b, 1995; Braitenberg, 1984; Brooks, 1986, 1989, 1990; Connell, 1987, 1990; Decugis and Ferber, 1998; Gat *et al.*, 1994; Maes, 1989, 1990; Matarić, 1990, 1992; Miller *et al.*, 1992; Payton, 1986; Rosenblatt and Payton, 1989; Rosenblatt, 1995; Rosenblatt and Thorpe, 1995; Rosenblatt, 1997; Rosenschein and Kaelbling, 1986; Schoppers, 1987, 1989) that utilize low-level reactive motion control exhibit the following properties:

With only locally sensed data (i.e., limited or free of representation),

1. the sequence of actions produced may be suboptimal due to lack of global knowledge,
2. the globally specified goal may not be achieved under complex environmental conditions such as local minima.

By interleaving the computation of the next motion command and the immediate execution of this motion,

3. unexpected obstacles can be successfully negotiated during runtime,
4. real-time response and robust execution can be achieved.

The proverb "One man's meat is another man's poison" signifies the relationship between the characteristics of deliberative planning and reactive control. The motivation of the work in this thesis stems from the realization of their complementary characteristics.

1.2 Motivation

1.2.1 Why Hybridize?

The individual drawbacks of deliberative planning and reactive control (Section 1.1.1) handicap their capacity for real-world applications in complex, unpredictable environments, thus weeding out any potential benefits posed by their strengths. Yet surprisingly, developments in high-level deliberative planning have largely ignored the details of low-level reactive control and vice versa. The amalgamation of the two paradigms into one coherent integrated hybrid framework can potentially aggregate their fortes and diminish their respective drawbacks to yield the best of both approaches. Nevertheless, developing such a unification methodology is non-trivial. If done poorly, it can yield the worst of both worlds.

1.2.2 The Art of Hybridization

Hybridization is more of an art than of science. Many hybrid robotic architects acknowledge that the central issue is to concoct an effective division of functionalities between deliberation and reactivity as well as an appropriate interface strategy to coordinate these two components (Arkin, 1998; Hexmoor and Kortenkamp, 1995a,b; Hexmoor *et al.*, 1997; Matarić, 1997). This open-ended guiding principle entails arbitrary architectural decisions between various roboticists, thus conceiving many different hybrid solutions (Section 2.1).

Notwithstanding the vast variety of hybrid perspectives that is accumulated over these years, a definitive resolution to this central issue has still not surfaced. As a result, myopic architectural decisions arise to fabricate hybrid systems that are at times too computationally complex for real world use (Albus *et al.*, 1989; Albus, 1991a,b, 1995, 1997). Often, this complexity is inadvertently mistaken to be inherent in the robot motion problem, but it is actually an artefact of the hybrid solution.

1.2.3 Rethinking Hybridization: Towards Real-Time Performance

With this in mind, it is therefore prudent to focus on real-time performance while moulding the hybridization process. Real-time motion control capabilities chiefly determine a mobile robot's ability to complete its navigation tasks. It is after all a vital ingredient in gaining its acquiescence into our society (Section 1.1).

To achieve real-time performance, my contention is that reactive control capabilities should be fully exploited to satisfy those criteria listed in Section 1.1.1 as much as possible, so that deliberative planning can be consequently simplified. The more competent the reactive control capabilities, the less complex need be the plan. This key notion indicates clearly the division of labour between the two paradigms. It also hints at how the interface/coordination strategy should be devised: the capabilities of the reactive competences determine the terms in which the plans must be expressed. In other words, the reactive competences give semantic significance to the deliberative planner.

1.2.4 Empowering Reactivity in a Hybrid System

Until today, the real-time performance of existing hybrid architectures is still not optimal because the capability of the reactive competences has not been fully realized. Often, the workload of high-level deliberative planning far exceeds that of low-level reactive control (Brock and Khatib, 1999a; Brock and Kavraki, 2000, 2001; Fox *et al.*, 1998; Hu and Brady, 1994; Pyeatt and Howe, 1999; Simmons *et al.*, 1997a; Thrun *et al.*, 1998a); hybrid architects seldom consider using the robust reactive controller to solve the tougher problems in robot motion and rely heavily on the computationally expensive planner to do so. This proves to be fatal in terms of real-time performance. The following three constituents of robot motion have been consequently victimized.

In goal-directed motion, most hybrid architectures employ the deliberative planner to plot the exact motion path with detailed sequence of actions for execution by the actuators. The reactive controller performs only a single task, i.e., simple obstacle avoidance, by making minor modifications to an otherwise good course of action. Can the reactive controller relieve the planner in producing equivalent target reaching movements? This problem demonstrates the under-nourished repertoire of reactive behaviors that is employed in robot motion.

A high degree of smoothness, flexibility and precision in motion control is essential for the efficient execution of sophisticated tasks as well as physical interaction with humans. A high-resolution plan is normally used to achieve this. Can reactive control mechanisms be utilized to perform fine, smooth motion control instead?

Unforeseen, complex obstacles in the environment are the ultimate menace to a robot in motion. Most hybrid architects would insist that replanning is the only way out. This may not be feasible in certain situations where an updated world map is not available or rendered obsolete easily by a rapidly changing environment. Can these obstacles be handled by reactive means then? It is my aim in this thesis to investigate these practical aspects in robot motion.

1.3 Thesis Objectives

This thesis proposes novel architectures for goal-directed, collision-free motion of a nonholonomic mobile robot in an unstructured environment. A nonholonomic robot has restrictions in the way it can move due to kinematic or dynamic constraints on the robot, such as limited turning abilities like an automobile or momentum at high velocities (Arkin, 1998). Hence, it is much harder to control and achieve smoothness in motion (Russell and Norvig, 1995), which makes the problem even more challenging.

The integrated planning and control architectures address the following objectives, which differ considerably from existing software architectures for mobile robots:

1. **Enhancing the capabilities of low-level reactive motion control qualitatively and quantitatively**
   (a) Enriching the repertoire of reactive behaviors (quantitative)
   (b) Performing fine, smooth and efficient motion control (qualitative)
   (c) Negotiating unforeseen, complex obstacles (qualitative)
2. **Reducing the workload of high-level deliberative motion planner**

This thesis emphasizes the first objective on reinforcing the reactive components of the integrated planning and control architectures to achieve real-time motion control capabilities. It de-emphasizes the second objective on motion planning such that issues of graph connectivity and search will not be addressed.

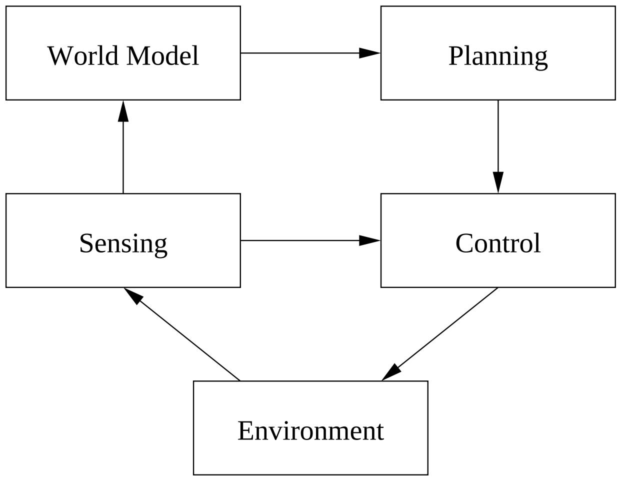


Figure 1.3: Integrated planning and control framework. Whenever the current world model is significantly changed by the sensing data, a new plan is generated by the planning module and supplied to the reactive control module to be executed. The control module also accommodates to unpredictable changes in the environment that occur after planning and during the execution of the plan.

1.4 Overview of Integrated Planning and Control Architectures

This section gives an overview of the integrated planning and control architectures that are implemented to meet the objectives specified in the previous section. Fig. 1.3 shows the general framework of integrated planning and control, which combines the previous two schemes of deliberative planning and reactive control (Fig. 1.2). Notice that deliberative planning and reactive control operate in different servo loops. Short time-scale operations are handled by the fast reflex loops of the reactive components, thus freeing up the deliberative component to plan without worrying about the dynamism in the environment. This asynchronous execution of the deliberative and reactive mechanisms is the key to preserving the reactive capabilities so that real-time performance can be achieved.

Whenever the current world model is significantly changed by the sensing data, a new plan is generated. On this aspect, I wish to point out that since our emphasis in this thesis is on integrated planning and control rather than mapbuilding, planning is conducted on a static world model that is not updated by sensing. Interested readers are referred to *Simultaneous Localization And Mapbuilding* (SLAM) (Durrant-Whyte *et al.*, 2001) and *Concurrent Mapping and Localization* (CML) (Thrun *et al.*, 1998b).

The amount of motion information encoded in the plan is directly proportional to the computational complexity of the planner and inversely related to the capability of the reactive components. On one extreme, a *fine* or *specific* plan comprises of a detailed sequence of motion commands that avoids obstacles marked on the static world model. As such, only reactive obstacle avoidance is required to negotiate unforeseen obstacles by making minor modifications to an otherwise good course of action. On the other extreme, a *gross* or *abstract* plan supplies a series of checkpoints to the reactive control mechanisms, which have to be further supplemented with target reaching capabilities. In general, the reactive components have to accommodate to unpredictable changes in the environment that occur after planning and during the execution of the plan.

Two improvements have been formulated to refine the integrated planning and control framework such that a tight integration of high-level deliberative planning with reactive control at the lowest level can be achieved.

At the highest level of the first integrated planning and control architecture (also known as *"command fusion" architecture*) (Fig. 1.4(a)), the deliberative motion planner, which will be elaborated in Chapter 3, produces a sequence of checkpoints from the start point to the goal. This is unlike conventional planning algorithms that plot detailed paths (Jarvis and

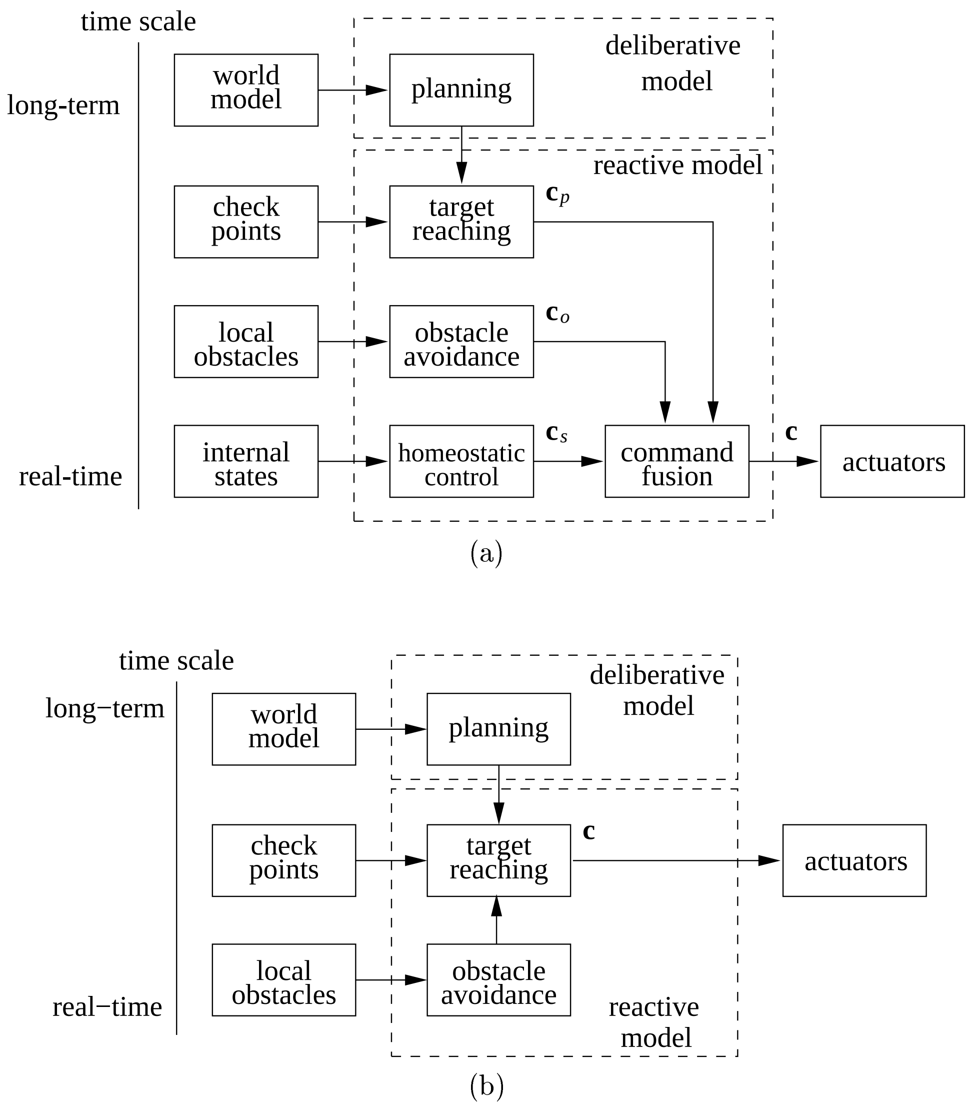


Figure 1.4: Integrated planning and control architectures. (a) "Command fusion" architecture: This architecture uses a command fusion module to coordinate the three reactive behavioral modules by integrating their control commands into one final command that is sent to the actuators. Since fusion is done at the action level, the reactive modules do not interact directly. (b) "Dynamic neural fields" architecture: This architecture integrates the reactive behavioral modules directly by fusing their neural representations. This permits the closely coupled modules to interact more intimately, resulting in enhanced reactive capabilities.

Byrne, 1986; Koditschek, 1987; Barraquand and Latombe, 1991; Rimon and Koditschek, 1992; Zelinsky, 1991, 1992, 1994; Zelinsky *et al.*, 1993; Zelinsky and Yuta, 1993a,b). Thus, the constraint of adhering strictly to a generated path no longer exists. The current implementation focuses on achieving a single goal at a time. To achieve multiple, possibly conflicting goals, other high-level planning algorithms may be used to order the goals.

The reactive model consists of three levels. At the first level, the *target reaching* module determines the motion path between checkpoints. It senses the checkpoint state relative to the current state and outputs appropriate motor control signals. It contains the *indirect-mapping Extended Kohonen Map* (EKM), which is trained to perform fine and smooth target reaching movements. It produces a sequence of very low-level (motor velocity) control commands to move the robot from one checkpoint to the next.

The next lower-level *obstacle avoidance* module senses the presence of local unforeseen static or moving obstacles and produces additional motor control commands using the Braitenberg Type-3C Vehicle (Braitenberg, 1984) method to repel the robot away from the obstacles.

The lowest-level *homeostatic control* module senses the internal state of the robot to maintain internal stability by coordinated responses that automatically compensate for environmental changes (Arkin, 1992a). An example of this low-level control is operational space control (Jamisola *et al.*, 2002; Khatib, 1987), which senses the internal joint angles, velocities, and forces to regulate the dynamic behavior of a robot manipulator during task execution. This module is not strictly required for mobile robot navigation, but is crucial for mobile manipulation tasks (Khatib, 1999) where the robot is manipulating an object or the environment while its base is in motion.

The *command fusion* module coordinates this repertoire of reactive behaviors by integrating their control commands into a final command that is sent to the actuators; these reactive modules are fused *indirectly at the action level.* All these modules, which will be extensively discussed in Chapter 4, constitute the reactive model of the "command fusion" architecture in Fig. 1.4(a).

In the second integrated planning and control architecture (also known as "dynamic neural fields" architecture) (Fig. 1.4(b)), the high-level planning module is the same as that in the "command fusion" architecture. The reactive model, which is examined in greater depth in Chapter 5, is renovated to fuse the modules *directly with their neural representations* by using the concept of multiple cooperative EKMs. This tighter integration permits the closely coupled modules to interact more intimately, resulting in further augmented reactive capabilities of negotiating unforeseen, complex obstacles.

The *target reaching* module operates in a similar fashion as that in the "command fusion" architecture. The *obstacle avoidance* module contains similar EKMs, which provide inhibitory inputs to the target-reaching EKMs at and around the locations where obstacles are detected. The tight integration of the two modules enables the robot while reaching targets to negotiate unforeseen concave and extended obstacles during run-time, which can trap robots controlled by other local obstacle avoidance methods (Arkin, 1989b; Braitenberg, 1984; Khatib, 1985, 1986).

All the modules in these two architectures operate asynchronously at different rates, as indicated by the time scale in Fig. 1.4. The planning module typically operates at the time scale of several seconds or minutes depending on task complexity. The target reaching module operates on the order of 100 ms to 1 second between servo ticks while the obstacle avoidance module operates at intervals of 100 ms. The homeostatic control module regulates at intervals of 1 ms. The command fusion module is activated as and when control commands are generated. The asynchronous execution of modules is the key to preserving the reactive capabilities (i.e., robust and timely response to dynamic environmental changes) as the deliberative component is free to plan on the static, stable properties of the environment without considering the dynamism in the same environment. In this manner, the deliberative planner can be seamlessly integrated with the reactive components or vice versa. In fact, the planner can fail or be removed and the resulting decapitated architecture degrades to a purely reactive system capable of less complex motion tasks. This coincides with one of the aims of Brooks (1986) subsumption architecture that the system should degenerate gracefully with the failure of higher-level components.

Chapter 2

Related Work

In this chapter, a comprehensive literature review is reported for the areas of hybrid deliberative/reactive architectures, high-level deliberative motion planning, and low-level reactive motion control. At the same time, my own work in this thesis will be qualitatively compared with these related works.

2.1 Hybrid Deliberative/Reactive Architectures

Numerous hybrid architectures for robotic agents have been conceived over these years due to different unifying perspectives (Section 1.2.2). Nevertheless, these hybrid systems can be classified into two general categories: the top-down approach termed *integrated cognitive architectures* and the bottom-up approach coined *integrated planning and control architectures.* Such a classification does not indicate that a hybrid architecture belongs exclusively to a category. It simply suggests the hybrid architecture maps more closely to one category than the other.

2.1.1 Top-Down Approach: Integrated Cognitive Architectures (Laird, 1991)

The top-down approach emphasizes high-level deliberative task planning, which is sometimes acknowledged as cognitive robotics (Baral and McIlraith, 2002; de Giacomo, 1998; Lakemeyer, 2000); the major bulk of the robot system's workload is on deliberative functionalities, which are primarily characterized by activities of symbolic reasoning and task scheduling. Advocates of this faction believe that deliberative planning should guide the reactive components in executing the plans robustly. The interface strategies of many hybrid participants hinges on this notion. Arkin (1998) and Lyons (1992) have classified the integrated cognitive architectures according to the type of interface/coordination strategies that they employ. Four principal representatives are noteworthy:

- **Selection: Planning is viewed as behavioral configuration.**

  The planning component determines the behavioral composition and parameters used during execution, which is consistent with the "plans-as-constraints" (Grosz and Sidner, 1988; Konolige and Pollack, 1989; Pollack, 1992) or "plans-as-rating-criteria" (Hayes-Roth, 1985) view. The planner may reconfigure them, when necessary, to cope with system failures.

  Integrated cognitive architectures that employ this interface strategy are Autonomous Robot Architecture (AuRA) (Arkin, 1989a, 1990; Arkin and Balch, 1997), Animate Agent Architecture (Bonasso *et al.*, 1997; Firby, 1993, 1994, 1995; Firby and Slack, 1995; Firby *et al.*, 1995), MORIA (Surmann and Peters, 2000), Servo-Subsumption-Symbolic (SSS) (Connell, 1992), Reactive and Preplanned Control (Soldo, 1990), Generic Robot Architecture (Chatila, 1995; Chatila *et al.*, 1992; Noreils and Chatila, 1989a,b, 1995;

Noreils, 1991), Agent Architecture (Hayes-Roth *et al.*, 1993, 1995a,b; Michaud *et al.*, 1996), Purposive Map (PM) (Zelinsky *et al.*, 1994, 1995; Zelinsky and Kuniyoshi, 1996), SOMASS Hybrid Assembly System (Malcolm and Smithers, 1990), RL/POMDP (Decugis and Ferber, 1998), Dynamic Systems (Bicho and Schöners, 1997; Engels and Schöners, 1995; Large *et al.*, 1997a,b, 1999; Menzner *et al.*, 2000; Schöners and Dose, 1992; Schöners *et al.*, 1995; Steinhage and Bergener, 1998), and Teleoreactive Agent Architecture (Benson and Nilsson, 1995).

- **Advising: Planning is viewed as advice giving.**

  The deliberative planner suggests changes that the reactive system may or may not use. This is consistent with the "plans-as-advice" (Bresina and Drummond, 1990; Payton *et al.*, 1990; Payton, 1990) or "plans-as-communications" (Agre and Chapman, 1990) view in which plans offer courses of actions but the reactive agent determines whether each is advisable.

  Integrated cognitive architectures that employ this interface strategy are A Three-Layer Architecture for Navigating Through Intricate Situations (ATLANTIS) (Bonasso *et al.*, 1997; Gat, 1991b, 1992, 1993a, 1998), and Task Control Architecture (TCA) (Simmons, 1991, 1994; Simmons and Mitchell, 1989; Simmons *et al.*, 1997a,b, 2000).

- **Adaptation: Planning is viewed as reactor adaptation.**

  The planner continuously improves the performance of the ongoing sub-optimal reactor in the light of changing environmental conditions and task requirements. It provides approximate answers in a time-critical manner such that at any point, a plan is available for execution and the quality of the available plan increases over time. This is consistent with the "plans-as-resource-bounds" (Pollack, 1992) view or anytime planning (Dean and Boddy, 1988; Dean and Wellman, 1991; Boddy and Dean, 1993; Mouaddib and Zilberstein, 1995; Zilberstein, 1996).

Integrated cognitive architectures that employ this interface strategy are Planner-Reactor (Lyons and Hendriks, 1992a,b, 1994, 1995), Goals As Parallel Program Specifications (GAPPS) with Anytime Planner (Kaelbling, 1986; Kaelbling and Rosenschein, 1990), and Entropy Reduction Engine (ERE) (Bresina, 1993; Bresina and Drummond, 1990; Drummond *et al.*, 1991, 1993).

- **Postponing: Planning is viewed as a least commitment process.**

  The planner defers making decisions on actions until as late as possible. This enables recent sensor data, by postponing reactive actions until absolutely necessary, to provide a more effective course of action than would be developed if an initial plan was generated at the beginning. Plans are elaborated only when necessary.

  Integrated cognitive architectures that employ this interface strategy are Procedural Reasoning System (PRS) (Georgeff and Lansky, 1987; Georgeff and Ingrand, 1989), Saphira (Congdon *et al.*, 1993; Konolige *et al.*, 1997; Saffiotti, 1993; Saffiotti *et al.*, 1993, 1994, 1995, 1997), Cypress (Wilkins *et al.*, 1995), Theo-Agent Architecture (Blythe and Mitchell, 1989; Mitchell, 1990), Cooperative Intelligent Real-Time Control Architecture (CIRCA) (Musliner *et al.*, 1993), and SOAR (Laird, 1990; Laird and Rosenbloom, 1990; Laird *et al.*, 1987).

The abstract plan produced by the deliberative planner in my proposed architectures can be considered advice giving; it "advises" the target reaching module in the reactive model to move the robot through the sequence of generated checkpoints to reach the goal. The reactive model can, however, ignore the planner's advice and function independently as a pure reactive architecture. This causes the mobile robot to be less capable of complex motion tasks (Section 1.1.1).

My proposed architectures only possess motion planning capabilities, which is a minor subset of task planning that handles other higher-level cognitive activities as well (e.g., symbolic reasoning and task scheduling). Instead of accentuating on task planning, these architectures concentrate on real-time motion control capabilities, which is the key notion of integrated planning and control discussed in the next section.

2.1.2 Bottom-Up Approach: Integrated Planning and Control Architectures (Khatib *et al.*, 1997)

The bottom-up design methodology focuses on how very low-level reactive control can be integrated with motion planning to provide real-time enhanced motion control capabilities (Section 1.2). Advocates of this faction, on the other hand, deem reactive control as a vital counterpart to motion planning as it helps to accommodate dynamism in the environment and most importantly, achieve real-time performance.

The following representatives of integrated planning and control align closely to the classification of the motion planning techniques (Section 2.2):

- **Roadmap: Motion planning constructs a graph of connected low-dimensional curves representing free space and generates a path via graph search.**

  An integrated planning and control architecture that uses this motion planning technique is Fuzzy Modular Control (Zhang and Knoll, 2000), which combines tangent-graph with fuzzy logic as reactive control.

- **Cell Decomposition: Motion planning constructs a graph of connected cells**

representing free space and generates a path via graph search.

Integrated planning and control architectures that use this motion planning technique are Elastic Strips (Brock and Kavraki, 2000, 2001; Brock and Khatib, 1997, 1998, 1999b, 2000), Elastic Bands (Quinlan and Khatib, 1993a,b), and GOFER navigation system (Choi and Latombe, 1991), which employ navigation functions (Barraquand and Latombe, 1991) in a tunnel or channel of cells (e.g., spherical and rectanguloid) to move the robot towards the goal and potential fields (Khatib, 1985, 1986) to avoid unexpected obstacles.

- **Potential Field: Motion planning computes a local minima free potential function and generates a path via gradient descent.**

  Integrated planning and control architectures that use this motion planning technique are Global Dynamic Window Approach (Brock and Khatib, 1999a), which combines a navigation functions (Barraquand and Latombe, 1991) for moving the robot to the goal with a dynamic window technique (Fox *et al.*, 1996, 1997) for reactive obstacle avoidance; Internalized Plans (Payton *et al.*, 1990; Payton, 1990; Stentz and Hebert, 1995), which combines path transform (Zelinsky, 1992, 1994) with subsumption based reactive behaviors (Brooks, 1986); SPOTT architecture (Zelek and Levine, 1996), which combines harmonic potentials (Connolly and Grupen, 1993; Kim and Khosla, 1992) with teleo-reactive trees method (Nilsson, 1994); Integrated Path Planning and Dynamic Steering Control (Krogh and Thorpe, 1986), which uses a grid-based potential function to produce and adapt a sequence of intermediate checkpoints (Thorpe, 1984) and potential fields (Khatib, 1985, 1986) to move from one checkpoint to the next.

My proposed integrated planning and control architectures coalesce a variant of approximate cell decomposition known as *slippery cells* (Quinlan and Khatib, 1993a) (Chapter 3) with reactive control methods for target reaching (indirect-mapping EKM) and obstacle avoidance (Braitenberg Type-3C vehicle (Braitenberg, 1984) or direct-mapping EKMs). However, the emphasis on fully exploiting the reactive competences to simplify the planning process significantly distinguishes my approaches from the abovementioned fellow architectures in several aspects, which have already been discussed in Section 1.2.

2.1.3 Continuous vs. Discrete Response Encoding

Hybrid architectures can also be differentiated by their encodings of responses. *Continuous response encoding* maps from a stimulus domain to an infinite space of potential reactions. It can therefore encode very low-level velocity/torque control of motor/joint actuators, which makes fine, smooth motion control possible. By tightly coupling high-level deliberative planning with reactive control at the lowest level, sophisticated tasks (Section 1.1) that require a high degree of smoothness, flexibility and precision in motion control can be accomplished.

Discrete response encoding, on the other hand, transforms a sensory input to a finite, enumerated set of responses (e.g., {forward,backward,left,right,speedup,slowdown,...,etc.}) (Arkin, 1998). The encoding of these high-level motor control commands de-emphasizes precision and smoothness in motion control, thus greatly reducing the robot's capabilities in managing complex motion tasks.

My proposed architectures are among the few hybrid systems (Arkin and Balch, 1997; Brock and Khatib, 1998; Choi and Latombe, 1991; Firby and Slack, 1995; Gat, 1992; Khatib *et al.*, 1997; Krogh and Thorpe, 1986; Michaud *et al.*, 1996; Saffiotti *et al.*, 1995; Schöners and Dose, 1992; Simmons *et al.*, 1997a; Surmann and Peters, 2000; Zhang and Knoll, 2000) that perform continuous response encoding of very low-level control to produce fine, smooth motion. This is in contrast to behavior-based architectures (Agre and Chapman, 1987; Bonasso, 1991; Brooks, 1986; Connell, 1990; Kaelbling and Rosenschein, 1990; Maes, 1990; Matarić, 1992; Rosenblatt, 1997; Sahota, 1994) and several hybrid architectures (Connell, 1992; Decugis and Ferber, 1998; Donnart and Meyer, 1994; Drummond *et al.*, 1991; Firby, 1995; Gat, 1998; Georgeff and Lansky, 1987; Hayes-Roth *et al.*, 1995a; Kaelbling and Rosenschein, 1990; Laird and Rosenbloom, 1990; Lyons and Hendriks, 1995; Malcolm and Smithers, 1990; Mitchell, 1990; Musliner *et al.*, 1993; Benson and Nilsson, 1995; Noreils and Chatila, 1995; Payton, 1990; Soldo, 1990; Wilkins *et al.*, 1995; Zelek and Levine, 1996; Zelinsky and Kuniyoshi, 1996) that employ discrete response encoding.

However, instead of using pre-wired reactive sensorimotor controllers such as Artificial Potential Fields (Khatib, 1985, 1986), Navigation Templates (NaTs) (Gat, 1993b; Slack, 1990, 1993), Motor Schemas (Arkin, 1987, 1989b, 1992b), Deformation Zone for collision avoidance (Zapata *et al.*, 1991), Curvature Velocity Method (CVM) (Ko and Simmons, 1998; Simmons, 1996), and Fuzzy Logic (Driankov and Saffiotti, 2000) in the hybrid systems to perform continuous response encoding, a novel self-organizing neural network technique known as indirect-mapping EKM is embedded in both my architectures to learn the low-level reactive motion control.

2.2 High-Level Deliberative Motion Planning

Though there exists a large number of methods for solving the basic motion planning problem, these methods can be generalized into a few general approaches: *roadmap*, *cell decomposition* and *potential field*. These methods will be briefly described with emphasis on their connectivity computation and path generation. Interested readers are referred to the references of

Hwang and Ahuja (1992); Latombe (1991, 1999); Laumond (1998); Li and Canny (1993).

2.2.1 Roadmap

This class of algorithms captures the connectivity of the robot's free space in a graph of partially connected low-dimensional curves called a *roadmap*, which is used as a set of standardized paths. The construction of the roadmap is the most computationally expensive operation in this algorithm class. A path is generated by connecting the initial and final robot positions or configurations to points in the roadmap and subsequently connecting these points by a graph search of the roadmap. Roadmap algorithms include *visibility graph* (Nilsson, 1969), *Voronoi diagram* (Aurenhammer, 1991), *freeway net* (Brooks, 1983), *silhouette* (Canny, 1988), *subgoal network* (Faverjon and Tournassoud, 1987), *retraction* (Ó'Dúnlaing and Yap, 1982; Ó'Dúnlaing *et al.*, 1983) and *probabilistic roadmap* (Kavraki and Latombe, 1994; Kavraki *et al.*, 1996).

2.2.2 Cell Decomposition

Cell decomposition techniques divide the robot's free space into simple regions called *cells* such that a motion path between any two positions or configurations in a cell can be easily generated using a local method. A connectivity graph is constructed to represent the adjacency relations (arcs) between cells (nodes). A path can be generated by searching the graph to produce a sequence of adjacent cells called a *channel* or *tunnel* and points at which the transition from one cell to another occurs.

This family of algorithms can be further categorized into *exact* and *approximate* methods. For exact cell decomposition (Schwartz and Sharir, 1987), the union of the cells corresponds exactly to the entire free space. Approximate cell decomposition methods (Brooks and Lozano-Pérez, 1985; Faverjon, 1984, 1986; Kambhampati and Davis, 1986; Lozano-Pérez, 1981; Laugier and Germain, 1985; Zhu and Latombe, 1991), however, produce cells of predefined shape (e.g., rectanguloids and spheres) whose union is a subset of the free space. The approximation becomes more accurate if the free space is represented to a high resolution. A poor approximation may result in the planner failing to find a path when one exists.

The deliberative motion planner in my proposed integrated planning and control architectures uses a variant of the approximate cell decomposition method known as *slippery cells* (Quinlan and Khatib, 1993a) (Chapter 3). In contrast to the other approximate methods, the shape of a slippery cell is general, yet it maintains the property of quick and easy path generation within the cell. This allows the free space to be decomposed into much fewer cells (less nodes), thus decreasing the amount of search time in the reduced connectivity graph.

2.2.3 Potential Field

The simplest approach to motion planning is to discretize the entire space into a fine regular grid of states or configurations and search this grid for a path connecting the initial and final positions without crossing any obstacles. To explore this typically enormous grid, two sets of heuristics are available: the local reactive method known as *artificial potential fields*[1] and the global deliberative method called *local minima free potential functions.*

Artificial potential fields (Arkin, 1989b; Khatib, 1985, 1986) represents the robot as a point in the state or configuration space moving under the influence of an artificial potential, which combines the attractive force pulling the robot towards the goal and repulsive forces pushing the robot away from the obstacles. This corresponds to applying gradient descent to the potential function. The path is not generated in advance; rather, it is implicitly rep-

[1] Artificial potential fields is described here rather than in Section 2.3 to facilitate its comparison with local minima free potential functions.

resented by the potential function, which continuously computes the next motion command at each instantaneous robot position. This results in a low computational complexity but incurs the local minima problem (Koren and Borenstein, 1991); a resultant force of zero at a position other than the goal traps the robot at that position.

To overcome this problem, local minima free potential functions are introduced. They utilize gradient descent to generate a guaranteed, complete motion path from the initial position to the goal prior to execution. These potential functions include harmonic potentials (Connolly, 1994; Connolly and Grupen, 1993; Connolly *et al.*, 1990; Feder and Slotine, 1997; Kim and Khosla, 1991, 1992), circulatory fields (Singh *et al.*, 1996), analytical navigation functions (Koditschek, 1987; Rimon and Koditschek, 1992), and numerical, grid-based navigation functions such as wavefront expansion (Barraquand and Latombe, 1991), distance transform (Jarvis and Byrne, 1986), and path transform (Zelinsky, 1991, 1992, 1994; Zelinsky and Yuta, 1993a,b; Zelinsky *et al.*, 1993).

2.3 Low-Level Reactive Motion Control

Low-level reactive motion control or execution, which is sometimes used interchangeably with sensorimotor coordination or control, typically requires only two basic behaviors, namely target or goal reaching and obstacle avoidance. These two fundamental sensorimotor control tasks have evolved many numerical solutions as well as posed as fine candidates for the application of several learning techniques. The following paradigms will be briefly discussed in the context of sensorimotor coordination: numerical techniques, fuzzy logic, multivariate regression, reinforcement learning, and feature maps. Readers interested in sensorimotor learning are referred to the references of Bekey and Goldberg (1993); Brooks *et al.* (1998);

Connell and Mahadevan (1993); Dorigo (1996); Franklin *et al.* (1996); Gaussier and Zrehen (1995); Hexmoor and Matarić (1998); Omidvar and van der Smagt (1997); Sharkey (1997); Sharkey and Heemskerk (1997); Ziemke and Sharkey (1999).

2.3.1 Numerical Techniques

Math-based sensorimotor controllers are employed widely for a broad span of tasks. Continuous responses are usually encoded and unlike the learning methods, they can be utilized immediately without training. Nevertheless, they have to be specifically tailored to different types of robots and modelling of the robot dynamics can become intractable when the control problem approaches nonlinearity. Furthermore, numerical methods do not tolerate noise that exists in the environment, robot sensors and actuators.

The mathematical models for goal reaching can be *biologically inspired*; they transform the difference between the current and the desired location signatures to a direction of travel. These models include *Snapshot Model* (Cartwright and Collett, 1983), *Proportional Vector Model* (Lambrinos *et al.*, 2000), *Difference Vector Model* (Lambrinos *et al.*, 2000), *Partial Image Matching Model* (Möller *et al.*, 1998), *Average Landmark Vector Model* (Lambrinos *et al.*, 1998), *Distribution Model* (Anderson, 1977), *Feature Matching Model* (Hong *et al.*, 1991), and *Warping Model* (Franz *et al.*, 1998).

Numerous math-based motion execution algorithms for real-time, local collision avoidance in an unpredictable environment have also been developed. They incorporate the most recent sensor data to determine the next motion command, which is executed instantaneously. *Artificial potential fields* (Arkin, 1989b; Khatib, 1985, 1986), which has been explicated in Section 2.2.3, and *Braitenberg Vehicles* (Braitenberg, 1984) are by far the most applauded reactive paradigms due to their low computational complexity. However, they cause a series of limitations (Koren and Borenstein, 1991) such as trap situations due to local minima, no passage between closely spaced obstacles, and oscillations (unstable motion) in the presence of obstacles and narrow passages.

Though high-level deliberative motion planning techniques (Section 2.2) can be used to remedy these problems, they are often computationally expensive and cannot be used to overcome unforeseen local minima. As a result, a range of reactive strategies has been developed to "partially" circumvent this problem.

While artificial potential fields and Braitenberg vehicles only consider the repulsive forces from obstacles acting on the robot to determine a resultant motion command, a contrasting class of *search space approaches* incorporates more sensor information from the environment to construct a local state or configuration space. This space is searched for a feasible motion command. The methods that belong to this class include *Vector Field Histograms* (Borenstein and Koren, 1991a,b; Ulrich and Borenstein, 1998, 2000), and *Steer Angle Fields* such as *Parameterized Path Families* (Feiten *et al.*, 1994), *Curvature-Velocity Methods* (Ko and Simmons, 1998; Simmons, 1996), and *Dynamic Window Approach* (Fox *et al.*, 1996, 1997).

Local path planners (Gat, 1993b; Lagoudakis and Maida, 1999; Slack, 1990, 1993; Zelek and Levine, 1996) attempt to negotiate unforeseen local minimas by using gradient ascent/descent to construct a shortest motion path on a local sensor-based space. This space has to be larger than that used in search space approaches since a motion path is plotted instead of a motion command.

Lastly, *heuristical methods* have also been employed. They include backtracking from the local minima regions and cordoning of these areas (Balch and Arkin, 1993; Choi *et al.*, 1989; Choi and Latombe, 1991; Gambardella and Haex, 1993; Gambardella and Versino, 1994; Liu *et al.*, 2000), and the use of noise as a form of 'reactive grease' (Arkin, 1987, 1989b). These heuristical techniques usually require considerable amount of time and effort to detect and escape from the local minima, and often result in long winding paths.

The reactive mechanisms in the "dynamic neural fields" architecture involved in negotiating unforeseen concave and extended obstacles (Chapter 5) fall into the category of search space approaches. However, my method differs from these approaches in that it is based on behavior-based neural fields dynamically interacting on the local sensory space of the indirect-mapping EKM after self-organized training. The motion command is determined by searching the EKM for the neuron with the highest activation.

2.3.2 Fuzzy Logic

Fuzzy sensorimotor controllers (Driankov and Saffiotti, 2000; Goodridge and Luo, 1994; Maeda *et al.*, 1991; Saffiotti *et al.*, 1993, 1994, 1997; Song and Tai, 1992; Sugeno and Nishida, 1985; Takeuchi *et al.*, 1988; Tunstel *et al.*, 1997; von Altrok *et al.*, 1992) have been designed to perform target reaching and obstacle avoidance robustly in the presence of uncertainty. They differ from conventional discrete rule-based methods (Drummond, 1989; Kaelbling, 1988; Nilsson, 1994; Schoppers, 1987; Suchman, 1987) by operating on noncrisp meanings of sensor readings and actuator commands and a set of fuzzy rules; continuous responses can therefore be encoded.

Like the numerical techniques, knowledge of the robot dynamics is required to formulate the fuzzy rules. To overcome this problem, efforts have been made to automatically generate these rules using various methods of learning (Bonarini, 1997; Castellano *et al.*, 1996; Hagras *et al.*, 2000; Homaifar and McCormick, 1995; Hsu *et al.*, 1995; Jou and Wang, 1993; Reignier, 1995; Wang and Mendel, 1992).

2.3.3 Multilayer Perceptrons

The sensorimotor control problem can be formulated as a nonlinear multivariate regression problem such that a multilayer perceptron is trained to perform this mapping (Berns *et al.*, 1991; Nagata *et al.*, 1990; Pomerleau, 1989, 1991, 1993; Sekiguchi *et al.*, 1989; Nehmzow and McGonigle, 1994; Nehmzow *et al.*, 1993; Nguyen and Widrow, 1989, 1990; Sharkey, 1998; Tani and Fukumura, 1994). It offers good generalization capabilities. However, it requires the prior collection of training samples for every time step so that quantitative error signals can be defined for network training, which can be a very tedious task.

2.3.4 Reinforcement Learning

The reinforcement learning approach (Kröse, 1995; Sutton, 1998) circumvents the difficulty in multilayer perceptrons by providing a qualitative success/failure feedback only at the end of executing the motor control sequence. Two main reinforcement learning strategies are commonly used to solve sensorimotor control problems (Kaelbling *et al.*, 1996): *genetic/evolutionary algorithms* (Floreano and Mondada, 1996; Harvey *et al.*, 1997) and *temporal difference methods* (Sutton, 1988). The former searches a population of weight combinations to find one that performs well according to a pre-defined fitness rule. The latter estimates how well each previously executed motor control vector contribute to the overall success/failure of achieving the desired goal state and modifies the weights accordingly through rewards or penalties. The training process tends to converge slowly due to sparse reinforcements and the imprecise estimate of each motor control vector's contribution.

Another difficulty faced by reinforcement learning is that of generalization. Many reinforcement learning algorithms encode discrete states and responses, which cannot apply directly to the continuous sensorimotor domains of the real-world control tasks. *A priori* discretization of the continuous spaces may introduce hidden states and weak generalization, if done poorly. By combining with function approximators (e.g., multilayer perceptron or feature map) that are capable of generalizing across continuous state and action spaces, this limitation can be overcome (Baird, 1995; Berns *et al.*, 1992; Chapman and Kaelbling, 1991; Gaskett *et al.*, 1999; Gullapalli, 1990; Hougen *et al.*, 2000; Iske *et al.*, 2000; Lin, 1992; Mahadevan and Connell, 1991, 1992; Millán, 1996, 1997; Millán and Torras, 1992; Millán *et al.*, 2002; Smart and Kaelbling, 2000; Sutton, 1996; Touzet, 1997).

2.3.5 Feature Maps

Feature maps such as the Self-Organizing Map (SOM) (Kohonen, 2000) and Extended Kohonen Map (EKM) (Ritter and Schulten, 1986) are often used to learn sensorimotor coordination (Berthouze and Kuniyoshi, 1998; Heikkonen *et al.*, 1993; Heikkonen and Koikkalainen, 1997; Rao and Fuentes, 1995, 1996, 1998; Sorouchyari, 1990; Touzet, 1997; Versino and Gambardella, 1995). They are trained to partition the continuous input and/or output space into localized regions. Their generalization capabilities arise from its self-organization during training such that each neuron is trained to map a localized sensory region to a desired motor control output. The models of Kuperstein (1991) and Zalama *et al.* (1995) also employ feature maps but the partitioning of their input spaces is pre-defined rather than self-organized. Unlike a multilayer perceptron, interpretation of the weights is possible (Section 6.1).

In Chapter 4, a new method for the feature map approach called the *indirect-mapping EKM* is introduced. Its indirect sensorimotor mapping contrasts with existing methods, which perform direct mapping. This modified property evolves significant benefits (Section 6.1).

Chapter 3

High-Level Deliberative Motion Planning

In this chapter, the deliberative model of the two integrated planning and control architectures is elaborated (Fig. 1.4). Two improvements are proposed to reduce the workload of the deliberative motion planner (second thesis objective), thereby boosting its real-time performance; they are motion planning (1) in *workspace* rather than in configuration space, and (2) with a variant of the approximate cell decomposition technique known as *slippery cells*.

3.1 Motion Planning: Workspace vs. Configuration Space

Workspace refers to the physical, real-world space that the robot moves in. Its dimensionality is constant: 2D plane for mobile robots and a 3D volume for robot manipulators. Configuration space, on the other hand, is the motor control space (e.g., velocity, torque or joint space) of the robot (e.g., mobile robots, robot manipulators, etc...). Therefore, its dimensionality corresponds to the controllable degrees of freedom of the robot (de Berg *et al.*, 1997; Hwang and Ahuja, 1992), which increases with the versatility of the robot.

The first improvement suggests global motion planning to be conducted in workspace rather than in configuration space. Operation in the workspace is more computationally efficient than that in the configuration space, which is particularly beneficial for robot motion planning involving higher degrees of freedom (implying higher dimensional configuration space) or kinematic constraints (implying complicated model of robot sensorimotor dynamics, e.g., nonholonomicity). The exploration of the configuration space is delayed until runtime whereby it can be performed very efficiently using reactive control algorithms and local sensor information from the environment (Brock and Kavraki, 2000, 2001; Brock and Khatib, 1997, 1998, 1999b, 2000).

3.2 Approximate Cell Decomposition: Slippery Cells

Conventional motion planners are hardly ever designed to emphasize real-time performance, needless to say its suitability for integration with reactive motion control. Most plot paths that often provide highly redundant motion information to reactive control for execution, thus incurring a huge computational burden. If instead, the *minimalistic*[1] resources required for reactive control to navigate successfully can be ascertained, I can exploit this knowledge to build a planner that provides just enough resources for task completion. In this manner, the computational burden can be considerably relieved.

The second improvement to the planner is based on the above notion. The deliberative motion planner operates on an approximate cell decomposition technique, which decomposes the free space into a new class of cells called slippery cells (Quinlan and Khatib, 1993a).

[1] Minimalism attempts to determine the minimal configuration of resources required to solve a given robotics task (Donald *et al.*, 1997). For example, if task A can be accomplished without resource B, it proves that B is somehow inessential to the information structure of the task.

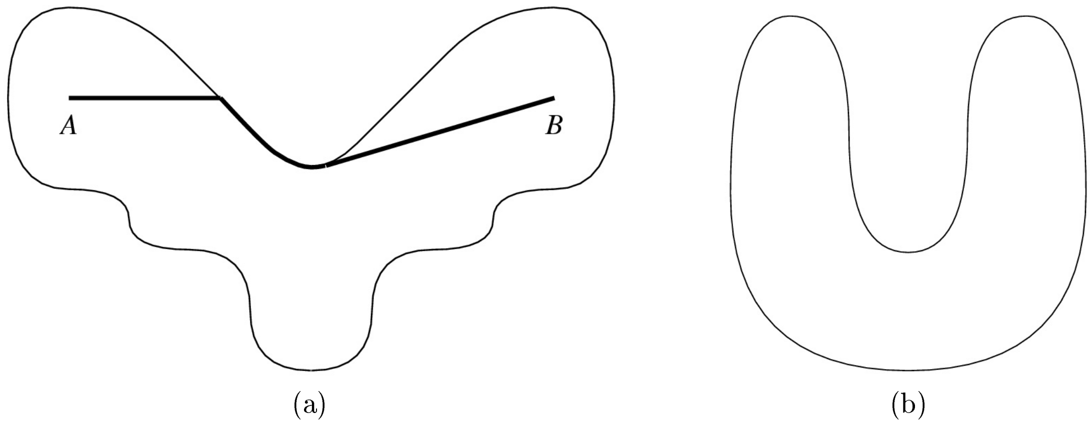


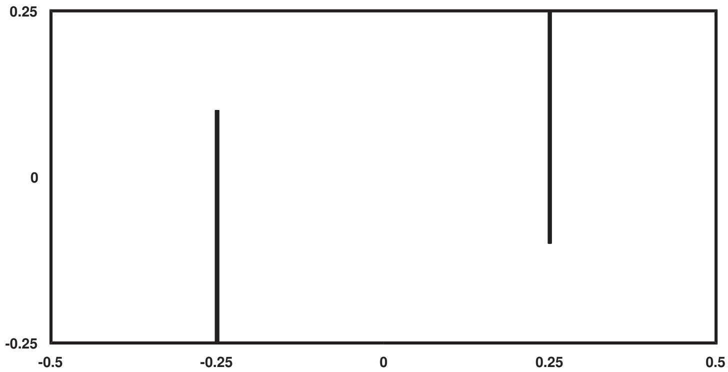
Figure 3.1: Slippery vs. non-slippery cell. (a) Slippery cell. (b) Non-slippery cell.

In contrast to existing approximate methods that evolve cells of pre-defined shape (e.g., rectanguloids and spheres), the shape of a slippery cell is general, yet it maintains the property that a path between any two points in the cell can be easily generated with a local reactive method. This new approach can usually decompose the free space into fewer cells than the other approximate methods. Fewer cells entail fewer nodes in the graph, thus reducing the search time.

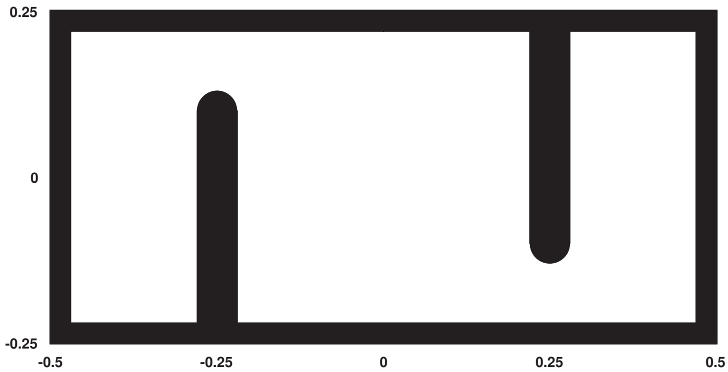
3.2.1 What is a Slippery Cell?

> "A cell is defined to be slippery (Fig. 3.1) if the outward normal from any point on its boundary does not intersect the same cell."

To demonstrate that a path is feasible between any two points in the slippery cell with a local reactive algorithm, consider the following example. To navigate from point A to point B (Fig. 3.1(a)), move straight towards B until either B or the cell boundary is reached. If the boundary is reached first, slide along the boundary by projecting the desired motion towards B into a direction tangential to the boundary. If the motion vector towards B points



(a)



(b)

Figure 3.2: Robot workspace. The environment consists of three rooms that are connected by two open doorways. (a) Original workspace. (b) “Padded” workspace.

away from the boundary, then stop sliding and move directly towards B. Since the outward normal does not intersect the cell, the vector of desired motion towards B will always contain a component tangential to the boundary. Hence, it is impossible to be stuck while sliding on the surface; the cell is slippery. It can be shown that the motion monotonically decreases the distance to B and the robot will definitely reach its goal.

3.2.2 Cell Representation of Free Space

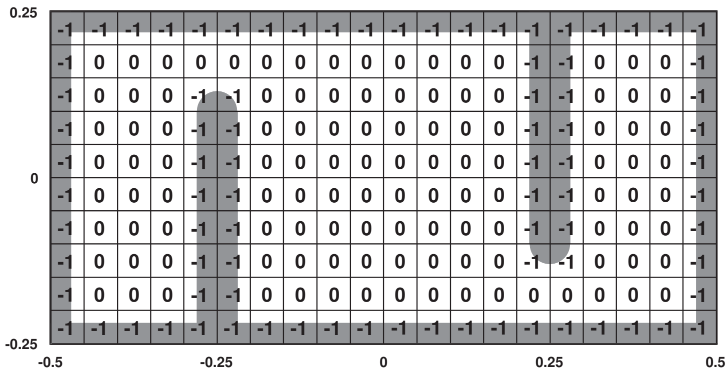
The free space in the *a priori* world map has to be decomposed into a set of disjoint slippery cells such that its union closely represents the free space. Since the robot is embodied rather than point-based, the obstacles on the map or workspace (Fig. 3.2(a)) have to be padded with a virtual layer of cushion[2] to eliminate the risk of possible impact. The entire "padded" workspace (Fig. 3.2(b)) is then converted into a bitmap cell representation before decomposing into slippery cells.

The bitmap cell representation (Fig. 3.3(a)) is formed by dividing the workspace into a grid of uniform square cells. A cell that is free is labeled '0' while a cell that is occupied is labeled '-1'. Free cells are entirely within the free space while occupied cells intersect obstacles or the cushion. Such a representation is also an approximate cell decomposition method, which is accurate to the resolution of the grid.

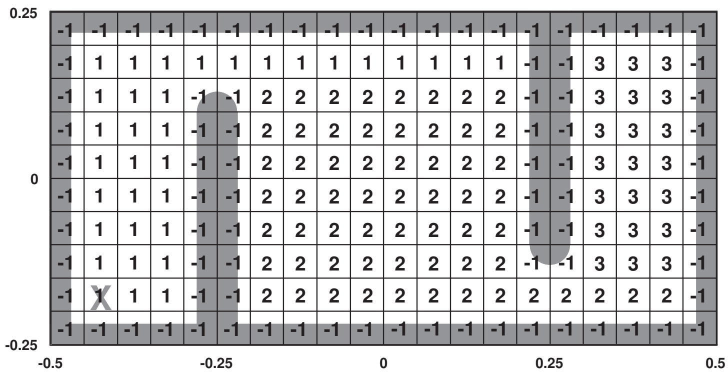
To transform the bitmap cell representation into a set of disjoint slippery cells (Fig. 3.3(b)), each free bitmap cell is labeled with an integer indicating the slippery cell to which it belongs. In principle, the labeling algorithm works as follows: first find a free bitmap cell that has not been labeled as part of a slippery cell. This cell becomes the seed of a new slippery cell. The seed is grown by adding neighboring unlabeled bitmap cells that are in the free space, while preserving the slippery cell property (Section 3.2.1). When no more bitmap cells can be added, the process is repeated to generate another slippery cell. If no unlabeled free bitmap cells are left, the algorithm to label slippery cells is complete.

The intricate aspect of this algorithm lies in determining whether an unlabeled free bitmap cell can be added to a slippery cell (Fig. 3.4). In essence, the procedure is as follows: first determine the axes along which the free cell is connected to the slippery cell. If it is connected

[2] The cushion width approximates the robot radius.



(a)



(b)

Figure 3.3: Approximate cell decomposition methods. The entire workspace is divided into a grid of 20 by 10 uniform square cells. (a) Bitmap cell representation: A free cell is labeled '0' while an occupied cell is labeled '-1'. (b) Slippery cell representation: The labeling of the first slippery cell '1' starts at the robot's initial position marked by 'X'. When the algorithm has finished labeling all the free cells, three slippery cells are formed.

only vertically (horizontally) to the slippery cell and the row (column) it resides in contains a labeled cell that belongs to the same slippery cell, then this free cell cannot be added to the slippery cell. However, if it is connected both horizontally and vertically to the same slippery cell, then this free cell can be added to the slippery cell.

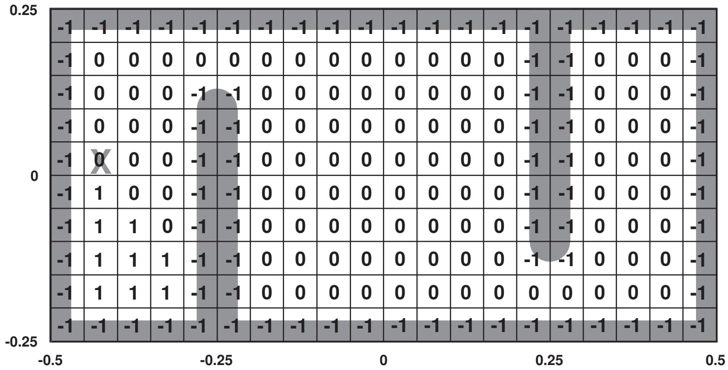
The pseudo code for the algorithm that labels slippery cells is provided in Fig. 3.5. The input to the algorithm is a two dimensional array $\mathbf{W}$ describing the workspace as a set of free and occupied bitmap cells (Fig. 3.3(a)), i.e., $\mathbf{W}[\mathbf{q}] = 0$ or -1 where $\mathbf{q} = (q_x, q_y)$. The output is the same array with each free bitmap cell labeled by the slippery cell to which it belongs (Fig. 3.3(b)), i.e., $\mathbf{W}[\mathbf{q}] = cell$ where *cell* is the slippery cell number. Two one dimensional arrays $\mathbf{X}$ and $\mathbf{Y}$ record the projection of the current slippery cell onto the x-axis and y-axis respectively, i.e., if $\mathbf{W}[\mathbf{q}] = cell$, $\mathbf{X}[q_x] = cell$ and $\mathbf{Y}[q_y] = cell$. Lastly, $\mathbf{L}$ is a list that queues the free bitmap cells to be examined.

A thorough test experiment has been conducted by Quinlan and Khatib (1993a), which reveals the number of cells generated by the slippery cells approach is very much smaller than that generated by the Octree approximate cell decomposition method (Faverjon, 1984). This quantitative evaluation is omitted from this thesis to avoid duplication of work.

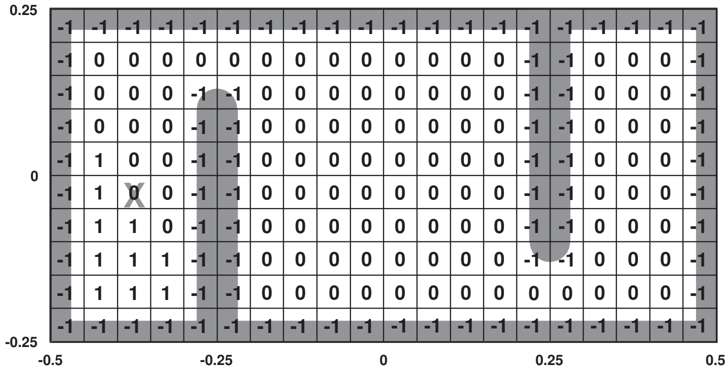
3.2.3 Checkpoints Generation

After generating a set of slippery cells that represent the free space, a connectivity graph can be constructed; the graph nodes represent the slippery cells and the graph arcs represent the adjacent cells. Using this graph, a series of checkpoints between the start position of the robot and the goal can be generated (Fig. 3.6) and provided to the reactive target reaching module to perform goal-directed motion (Fig. 1.4).

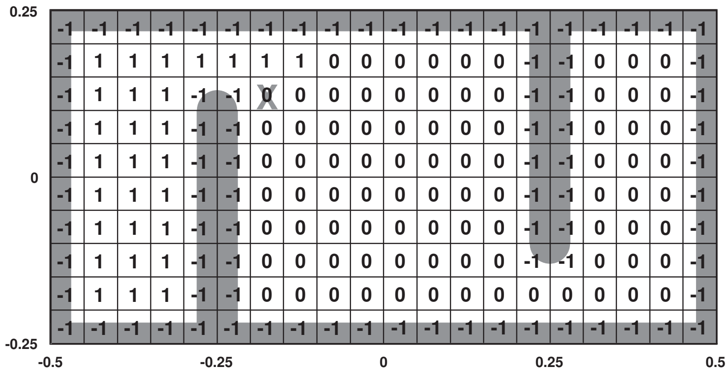
To do so, a sequence of adjacent cells must first be found via graph search such that the



(a)



(b)



(c)

Figure 3.4: Procedure to add a free bitmap cell to a slippery cell. (a) The free bitmap cell at 'X' is connected vertically to the first slippery cell. Since the row it resides in does not contain a cell that is labeled under the first slippery cell, it can be added to the first slippery cell. (b) The free bitmap cell at 'X' is connected both vertically and horizontally to the first slippery cell. Therefore, it can be added to the first slippery cell. (c) The free bitmap cell at 'X' is connected vertically to the first slippery cell. However the row it resides in already contains cells that are labelled under the first slippery cell. Thus, it cannot be added to the first slippery cell.

Algorithm *SlipperyCellLabeling*($\mathbf{W}$)
$cell \leftarrow 0$;
for each element $\mathbf{q}$ in $\mathbf{W}$
  do if $\mathbf{W}[\mathbf{q}] = 0$
    then $cell \leftarrow cell + 1$;
      initialize $\mathbf{L}$ to $\mathbf{q}$;
      while $\mathbf{L}$ is not empty
        do $\mathbf{p} \leftarrow$first element of $\mathbf{L}$;
          if $AddBitmapCell(\mathbf{p}, cell, \mathbf{W}, \mathbf{X}, \mathbf{Y})$
            then $\mathbf{W}[\mathbf{p}] \leftarrow cell$;
              $\mathbf{X}[p_x] \leftarrow cell$;
              $\mathbf{Y}[p_y] \leftarrow cell$;
              for $\mathbf{r} \leftarrow \mathbf{q} + (0,1), \mathbf{q} + (1,0), \mathbf{q} + (0,-1), \mathbf{q} + (-1,0)$
                do if $\mathbf{W}[\mathbf{r}] = 0$ **then** append $\mathbf{r}$ to $\mathbf{L}$;

Algorithm *AddBitmapCell*($\mathbf{p}, cell, \mathbf{W}, \mathbf{X}, \mathbf{Y}$)
if $\mathbf{W}[\mathbf{p} + (1,0)] = cell$ **or** $\mathbf{W}[\mathbf{p} + (-1,0)] = cell$
  then return $\mathbf{X}[\mathbf{p}_x] \neq cell$;
if $\mathbf{W}[\mathbf{p} + (0,1)] = cell$ **or** $\mathbf{W}[\mathbf{p} + (0,-1)] = cell$
  then return $\mathbf{Y}[\mathbf{p}_y] \neq cell$;
return true;

Figure 3.5: Algorithm to label slippery cells. The symbols are explained in the text.

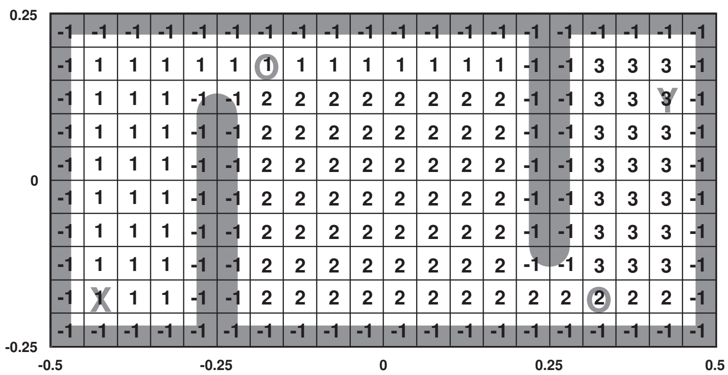


Figure 3.6: Checkpoints generation. If the robot's initial position is at 'X' and its goal is at 'Y', two checkpoints, marked by 'O', will be generated at the boundaries of the adjacent slippery cells.

start and goal positions are enclosed within the first and last cell in the sequence respectively. Subsequently, for each pair of adjacent cells in the sequence, a checkpoint that lies on the boundary of both cells is located. These checkpoints are determined using greedy search. From the start position, the closest point is found on the boundary between the current cell and the next cell in the sequence. This point becomes the first checkpoint. The second checkpoint is found in the same manner, except that the first checkpoint is now the position from which the distance is calculated. This is repeated until the last cell is reached.

3.3 Summary

The work in this chapter reveals two improvements that can be made to reduce the workload of the deliberative motion planner. By planning the robot's motion in the workspace instead of the configuration space, computational efficiency can be enhanced. An approximate cell decomposition technique known as slippery cells is introduced to decompose the free workspace into much fewer cells or graph nodes than the other approximate methods, resulting in reduced search time. Yet any two points in the cell can still be successfully traversed by reactive control mechanisms. Using this graph, the planner generates a series of checkpoints from the start point to the goal (Section 3.2.3) for the reactive models in both architectures (Chapters 4 and 5) to perform goal-directed, collision-free motion from one checkpoint to the next.

Chapter 4

Low-Level Reactive Motion Control I

This chapter discusses the reactive model of the "command fusion" architecture (Fig. 1.4(a)). In this model, the *target reaching* module contains an *indirect-mapping EKM*, which is trained to move between checkpoints provided by the deliberative motion planner (Chapter 3). While reaching targets, the *obstacle avoidance* module uses the *Braitenberg Type-3C vehicle* method to avoid collision with obstacles during goal-directed motion. The motor control outputs of the reactive modules are integrated via *command fusion* into a final command that is sent to the actuators.

4.1 Target Reaching: Direct-Mapping EKM

The direct-mapping EKM adopts an egocentric representation of the sensory input vector $\mathbf{u}_p = (\alpha, d)^T$ where α and d are the direction and the distance of a checkpoint relative to the robot's current location and heading. At the goal state at time T, $\mathbf{u}_p(T) = (\alpha, 0)^T$ for any

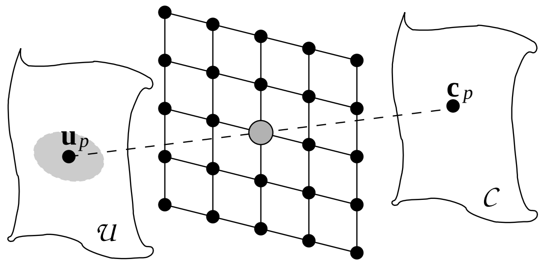


Figure 4.1: Target reaching with direct-mapping EKM. EKM neurons map the sensory input space $\mathcal{U}$ directly to discretized points in the motor control space $\mathcal{C}$.

α.

If sensorimotor coordination is a linear problem, then the motor control vector $\mathbf{c}_p$ would be related to the sensory input vector $\mathbf{u}_p$ by the linear equation

$$\mathbf{c}_p = \mathbf{M}\mathbf{u}_p \tag{4.1}$$

where $\mathbf{M}$ is a matrix of motor control parameters. The control problem would be reduced to one of determining $\mathbf{M}$ from training samples.

In practice, however, sensorimotor coordination is a nonlinear problem because a real motor takes a finite but non-zero amount of time to accelerate or decelerate in order to change speed. This nonlinear problem is further complicated in nonholonomic robots. The direct-mapping EKM has been widely embraced as a viable solution to this nonlinear problem (Berthouze and Kuniyoshi, 1998; Heikkonen *et al.*, 1993; Heikkonen and Koikkalainen, 1997; Rao and Fuentes, 1995, 1996, 1998; Touzet, 1997; Versino and Gambardella, 1995). In principle, it is trained to partition the sensory input space $\mathcal{U}$ into localized regions, which is similar to that of SOM. Each neuron i in the EKM has a sensory weight vector $\mathbf{w}_i$ that encodes a localized region in $\mathcal{U}$ centered at $\mathbf{w}_i$. It also has a set of output weights $\mathbf{c}_i$ which encode the motor control vector (Fig. 4.1).

A careful scrutiny of the direct-mapping EKM would reveal that since it maps all the sensory inputs $\mathbf{u}_p$ in a region in the sensory input space $\mathcal{U}$, represented by a neuron k, to the same discrete point $\mathbf{c}_k$ in the motor output space $\mathcal{C}$ (Fig. 4.1) (i.e., $\mathbf{c}_p = \mathbf{c}_k$), only a small number of points in $\mathcal{C}$ are represented by the neurons' outputs, i.e., the motor output space is very sparsely sampled. Consequently, the sparse, discretized sampling of the motor control space greatly compromises the smoothness and precision in the sensorimotor control of the robot's target-reaching motion. This motivates the need to devise a new technique that can sample the motor control space continuously, so as to improve its motion control capabilities.

4.2 Target Reaching: Indirect-Mapping EKM

My proposed indirect-mapping EKM (Fig. 4.2) revises the neural network architecture in the previous section to accommodate to the continuous sampling of the motor control space, thus providing finer and smoother motion control. In essence, the indirect-mapping EKM is trained to partition the sensory input space $\mathcal{U}$ into *locally linear* regions. Each neuron i in the EKM has a sensory weight vector $\mathbf{w}_i$ that encodes a localized region in $\mathcal{U}$ centered at $\mathbf{w}_i$, which is similar to that of direct-mapping EKM. However, unlike the direct-mapping EKM, the output weights $\mathbf{M}_i$ of neuron i represent control parameters in the parameter space $\mathcal{M}$ instead of the motor control vector. The control parameter matrix $\mathbf{M}_i$ is mapped to the actual motor control vector $\mathbf{c}_p$ by the linear model of Eq. 4.1.

Analytically, the indirect-mapping approach maps each $\mathbf{u}_p$ in a locally linear region in $\mathcal{U}$ to a different point $\mathbf{c}_p$ in $\mathcal{C}$ through Eq. 4.5. Since this mapping is linear and continuous, the indirect-mapping approach maps a region in $\mathcal{U}$ to a region in $\mathcal{C}$, thus providing finer and smoother sensorimotor control of the robot's target-reaching motion than does direct mapping. A quantitative comparison has been conducted (see Section 6.1) to substantiate

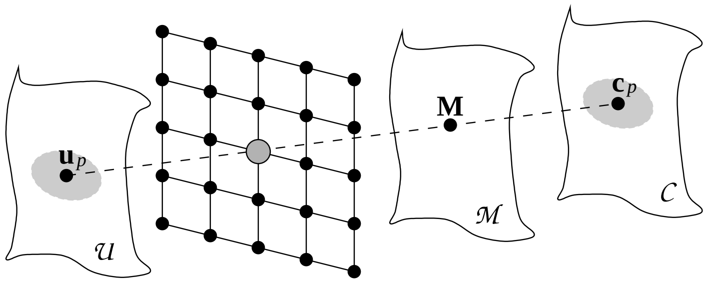


Figure 4.2: Target reaching with indirect-mapping EKM. EKM neurons map the sensory input space $\mathcal{U}$ indirectly to the continuous motor control space $\mathcal{C}$ through the control parameter space $\mathcal{M}$.

this claim.

With indirect-mapping EKM, motor control is performed as follows:

Motor Control

Given a sensory input vector $\mathbf{u}_p$,

1. Determine the winning neuron k.

   The winning neuron k is the neuron whose sensory weight vector $\mathbf{w}_k = (\alpha_k, d_k)^T$ is nearest to the input $\mathbf{u}_p = (\alpha, d)^T$:

$$D(\mathbf{u}_p, \mathbf{w}_k) = \min_{i \in \mathcal{A}(\alpha)} D(\mathbf{u}_p, \mathbf{w}_i) \ . \tag{4.2}$$

   The difference $D(\mathbf{u}_p, \mathbf{w}_i)$ is a weighted difference between $\mathbf{u}_p$ and $\mathbf{w}_i$:

$$D(\mathbf{u}_p, \mathbf{w}_i) = (\gamma_\alpha(\alpha - \alpha_i)^2 + \gamma_d(d - d_i)^2)^{1/2} \tag{4.3}$$

   where γ_α and γ_d are constant parameters. The minimum in Eq. 4.2 is taken over the set $\mathcal{A}(\alpha)$ of neurons encoding very similar angles as α:

$$|\alpha - \alpha_i| \leq |\alpha - \alpha_j|, \quad \text{for each pair } i \in \mathcal{A}(\alpha), j \notin \mathcal{A}(\alpha) \ . \tag{4.4}$$

   In other words, direction has priority over distance in the competition between EKM neurons. This method allows the robot to quickly orientate itself to face the target while moving towards it (Versino and Gambardella, 1995).

2. Compute motor control vector $\mathbf{c}_p$ for target reaching:

$$\mathbf{c}_p = \begin{cases} \mathbf{M}_k \mathbf{u}_p & \text{if } -\mathbf{c}^* \leq \mathbf{M}_k \mathbf{u}_p \leq \mathbf{c}^* \\ \mathbf{M}_k \mathbf{w}_k & \text{otherwise.} \end{cases} \tag{4.5}$$

The constant vector $\mathbf{c}^*$ denotes the upper limit of physically realizable motor control signal. For instance, for the Khepera robots, $\mathbf{c}_p$ consists of the motor speeds v_l and v_r of the robot's left and right wheels. In this case, we define $\mathbf{c}_p \leq \mathbf{c}^*$ if $v_l \leq v_l^*$ and $v_r \leq v_r^*$. Note that if $\mathbf{c}_p$ is beyond $\mathbf{c}^*$, simply saturating the wheel speeds does not work. For example, if the target is far away and not aligned with the robot's heading, then saturating both wheel speeds only moves the robot forward. Without correcting the robot's heading, the robot will not be able to reach the target. Hence, the winning neuron's input weights $\mathbf{w}_k$ are used to generate the physically realizable motor control output. This motor control would be the best substitution for the sensory input $\mathbf{u}_p$ because $\mathbf{w}_k$ is closest to $\mathbf{u}_p$ compared to other weights $\mathbf{w}_i, i \neq k$.

The motor control algorithm is applied at each time step t to compute the motor control vector $\mathbf{c}_p(t)$ for the current sensory input $\mathbf{u}_p(t)$. It is repeated until the robot reaches the goal state $\mathbf{u}_p(T)$ at time step T.

4.3 Self-Organized Training of Indirect-Mapping EKM

In contrast to most existing methods, online training is adopted for the indirect-mapping EKM. Initially, the EKM has not been trained and the motor control vectors $\mathbf{c}_p$ generated are inaccurate. Nevertheless, the EKM self-organizes, using these control vectors $\mathbf{c}_p$ and the corresponding robot displacements $\mathbf{v}$ produced by $\mathbf{c}_p$, to map $\mathbf{v}$ to $\mathbf{c}_p$ indirectly. As the robot moves around and learns the correct mapping, its sensorimotor control becomes more accurate. At this stage, the same online training can still be performed, and it mainly fine tunes the indirect mapping. The self-organized training algorithm (in an obstacle-free environment) can be summarized as follows:

Self-Organized Training

Repeat

1. Get sensory input $\mathbf{u}_p$.

2. Execute motor control algorithm and move robot.

3. Get new sensory input $\mathbf{u}'_p$ and compute actual displacement $\mathbf{v}$ as a vector difference between $\mathbf{u}'_p$ and $\mathbf{u}_p$.

$$\mathbf{v} = \mathbf{u}'_p - \mathbf{u}_p \tag{4.6}$$

4. Use $\mathbf{v}$ as the training input to determine the winning neuron k (same as Step 1 of Motor Control).

5. Adjust the weights $\mathbf{w}_i$ of neurons i in the neighborhood $\mathcal{N}_k$ of the winning neuron k towards $\mathbf{v}$:

$$\Delta\mathbf{w}_i = \eta\, G(k,i)(\mathbf{v} - \mathbf{w}_i) \tag{4.7}$$

where $G(k,i)$ is a Gaussian function of the distance between the positions of neurons k and i in the EKM, and η is a constant learning rate.

6. Update the weights $\mathbf{M}_i$ of neurons i in the neighborhood $\mathcal{N}_k$ to minimize the error e:

$$e = \frac{1}{2} G(k,i) \|\mathbf{c}_p - \mathbf{M}_i \mathbf{v}\|^2 \ . \tag{4.8}$$

That is, apply gradient descent to obtain

$$\Delta \mathbf{M}_i = -\eta \frac{\partial e}{\partial \mathbf{M}_i} = \eta\, G(k,i)(\mathbf{c}_p - \mathbf{M}_i \mathbf{v})\mathbf{v}^T \ . \tag{4.9}$$

At each training cycle, the weights of the winning neuron k and its neighboring neurons i are modified. The amount of modification is proportional to the distance $G(k,i)$ between the neurons in the EKM. The input weights $\mathbf{w}_i$ are updated towards the actual displacement $\mathbf{v}$ and the control parameters $\mathbf{M}_i$ are updated so that they map the displacement $\mathbf{v}$ to the corresponding motor control $\mathbf{c}_p$.

After self-organization has converged, the neurons will stabilize in a state such that $\mathbf{v} = \mathbf{w}_i$ and $\mathbf{c}_p = \mathbf{M}_i\mathbf{v} = \mathbf{M}_i\mathbf{w}_i$. For any winning neuron k, given the sensory input $\mathbf{u}_p = \mathbf{w}_k$, the neuron will produce a motor control output $\mathbf{c}_p = \mathbf{M}_k\mathbf{w}_k$ which yields a desired displacement of $\mathbf{v} = \mathbf{w}_k$. For a sensory input $\mathbf{u}_p \neq \mathbf{w}_k$ but close to $\mathbf{w}_k$, the motor control output $\mathbf{c}_p = \mathbf{M}_k\mathbf{u}_p$ produced by neuron k will still yield the correct displacement if linearity holds within the input region that activates neuron k. Therefore, given enough neurons to produce an approximate linearization of the sensory input space $\mathcal{U}$, the indirect-mapping EKM can produce finer and smoother motion control than that of direct-mapping EKM.

4.4 Obstacle Avoidance: Braitenberg Type-3C Vehicle

The reactive obstacle avoidance module adopts the architecture of Braitenberg's Type-3C vehicle (Braitenberg, 1984). Given a set $\mathbf{u}_o$ of sensor inputs, the motor velocity $\mathbf{c}_o$ for obstacle avoidance is computed as:

$$\mathbf{c}_o = \mathbf{Z}\,\mathbf{u}_o \tag{4.10}$$

where $\mathbf{Z} = [z_{ij}]$ is the control matrix. The matrix elements z_{ij} that link the forward-facing sensors on one side of the robot's body with the motor on the opposite side have large

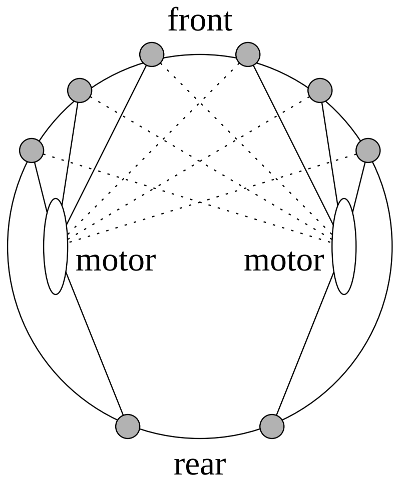


Figure 4.3: Obstacle avoidance with Braitenberg Type-3C vehicle. The connections between the forward-facing sensors on one side of the robot's body with the motor on the opposite side have large negative values (dotted lines), while the other connections have small positive values (solid lines).

negative values, while the other matrix elements have small positive values (Fig. 4.3). When the robot senses the presence of an obstacle, say, in front and on the left, the right motor will rotate backward faster than the left motor's rotation forward, thus turning the robot away from the obstacle.

A similar approach is adopted in the architecture of Decugis and Ferber (1998). The main difference is that Decugis and Ferber (1998) assigned different sets of weights to different behaviors including obstacle avoidance, left wall following, right wall following, corridor following, left static turn, right static turn, and forward move. On the other hand, our method uses only one set of weights. It removes the need to recognize different situations and select different behaviors, and thus simplifies the reactive modules. As will be seen in Section 6.2, different behaviors can still emerge naturally from the interaction between the robot and the environment.

4.5 Behavioral Coordination: Command Fusion

The motor control for obstacle avoidance $\mathbf{c}_o$ is added to the motor control for target reaching $\mathbf{c}_p$ to produce the final motor control signal $\mathbf{c}$:

$$\mathbf{c} = \beta\,\mathbf{c}_p + (1-\beta)\,\mathbf{c}_o \tag{4.11}$$

where β is a constant parameter. The homeostatic control $\mathbf{c}_h$ of the motors is omitted. Equation 4.11 is analogous to the potential fields method for obstacle avoidance (Khatib, 1985, 1986) and is able to overcome small unforeseen obstacles and non-adversarial moving obstacles.

Recall that the target reaching and obstacle avoidance modules run at different rates. Each time one of the modules produces a new motor control signal, it updates a global motor state, which then causes the combined motor control signal $\mathbf{c}$ to be sent to the robot's wheels to drive the robot. In the absence of obstacles, the motor control signal will be sent at regular intervals. In the presence of obstacles, additional control signal may be sent as and when obstacles are detected. This method allows the robot to run as smoothly as possible and to make adjustments only when necessary.

4.6 Summary

This chapter illustrates how the target reaching and obstacle avoidance modules are fused *indirectly at the action level* via command fusion. In the next chapter, the reactive model in the "dynamic neural fields" architecture will be examined. It differs from the model here by using multiple cooperative EKMs to fuse the reactive modules *directly with their neural representations*, resulting in further enhanced reactive motion control capabilities.

Chapter 5

Low-Level Reactive Motion Control II

This chapter discusses the reactive model of the "dynamic neural fields" architecture (Fig. 1.4(b)). In this model, the reactive modules are tightly integrated using *multiple cooperative EKMs*. The *target reaching* module operates in a similar fashion as that in the "command fusion" architecture. The *obstacle avoidance* module, contains EKMs, which provide inhibitory inputs to the target-reaching EKMs at and around the locations where obstacles are detected. The intimate coupling of the two modules enables the robot to negotiate unforeseen concave and extended obstacles during goal-directed motion, which can trap robots controlled by other local obstacle avoidance methods (Arkin, 1989b; Braitenberg, 1984; Khatib, 1986).

5.1 Cooperative Extended Kohonen Maps

Multiple EKMs can cooperate to enable a mobile robot to perform a more complex sensorimotor control task, that is to negotiate unforeseen concave or extended obstacles. These

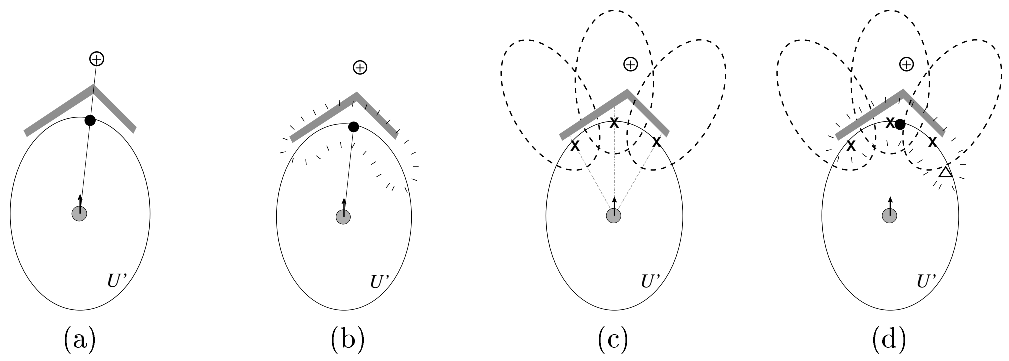


Figure 5.1: Cooperative EKMs with indirect mapping. (a) In response to the target $\oplus$, the nearest neuron (black dot) in the target localization EKM (ellipse) of the robot (gray circle) is activated. (b) The activated neuron produces a target field (dotted ellipse) in the motor control EKM. (c) Three of the robot's sensors detect obstacles and activate three neurons (crosses) in the obstacle localization EKM, which produce the obstacle fields (dashed ellipses). (d) Subtraction of the obstacle fields from the target field results in the neuron at $\triangle$ to become the winner in the motor control EKM, which moves the robot away from the obstacle.

EKMs are categorized into two main types based on their functionality: (1) target reaching and (2) obstacle avoidance.

5.2 Target Reaching EKMs

Two EKMs are used to perform target reaching, one for target localization (Step 1) and the other for motor control (Steps 2–4).

Target Reaching

Given a sensory input vector $\mathbf{u}_p$ of a target location,

1. Determine the winning neuron s in the target localization EKM.

   Neuron s is the one whose sensory weight vector $\mathbf{w}_s = (\alpha_s, d_s)^T$ is nearest to the input $\mathbf{u}_p = (\alpha, d)^T$:

$$D(\mathbf{u}_p, \mathbf{w}_s) = \min_{i \in \mathcal{A}(\alpha)} D(\mathbf{u}_p, \mathbf{w}_i). \tag{5.1}$$

The difference $D(\mathbf{u}_p, \mathbf{w}_i)$ is computed in the same manner as Equation 4.3 in Step 1 of Motor Control in Section 4.2. In the EKM, each neuron encodes a location $\mathbf{w}_i$ in the sensory input space $\mathcal{U}$. The region of $\mathcal{U}$ that encloses all the neurons is called the *local sensory space* $\mathcal{U}'$. Even if the target falls outside $\mathcal{U}'$, the nearest neuron can still be activated (Fig. 5.1a).

2. Compute the neuronal activities of the motor control EKM.

The activity a_i of neuron i depends on the integration of target localization EKM and obstacle field EKM:

$$a_i = G_a(\mathbf{w}_s, \mathbf{w}_i) - b_i \tag{5.2}$$

where b_i is the inhibitory input from the neuron in the obstacle field EKM (Section 5.3). The function G_a is an elongated Gaussian:

$$\begin{aligned} G_a(\mathbf{w}_s, \mathbf{w}_i) &= \exp(-D_a(\mathbf{w}_s, \mathbf{w}_i)/\sigma_a^2) \\ D_a(\mathbf{w}_s, \mathbf{w}_i) &= \gamma_\alpha(\alpha_s - \alpha_i)^2 + \gamma_d(d_s - d_i)^2 \ . \end{aligned} \tag{5.3}$$

Parameter γ_d is much larger than γ_α, making the Gaussian distance-sensitive and angle-insensitive. Moreover, they elongate the Gaussian along the direction perpendicular to the target direction α_s (Fig. 5.1b). This elongated Gaussian is the *target field*, which plays an important role in avoiding local minima during obstacle avoidance (Section 5.3).

3. Determine the winning neuron k in motor control EKM.

Neuron k is the one with the largest activity:

$$a_k = \max_i a_i \ . \tag{5.4}$$

4. Compute motor control vector $\mathbf{c}$ for target reaching:

$$\mathbf{c} = \begin{cases} \mathbf{M}_k \mathbf{u}_p & \text{if } -\mathbf{c}^* \leq \mathbf{M}_k \mathbf{u}_p \leq \mathbf{c}^* \text{ and } k = s \\ \\ \mathbf{M}_k \mathbf{w}_k & \text{otherwise.} \end{cases} \tag{5.5}$$

This step is similar to Step 2 of Motor Control in Section 4.2.

The target reaching procedure is applied at each time step t to compute the motor control vector $\mathbf{c}(t)$ for the current sensory input $\mathbf{u}_p(t)$. It is repeated until the robot reaches the goal state $\mathbf{u}_p(T)$ at time step T.

The self-organized training of the motor control EKM is explained in Section 4.3. The target localization EKM self-organizes in the same manner as the motor control EKM except that Step 6 is omitted.

5.3 Obstacle Avoidance EKMs

The obstacle avoidance module contains multiple obstacle localization EKMs (Step 1) and an obstacle field EKM (Step 2) that are self-organized in the same way as the target reaching EKMs. As a result, each neuron i in the obstacle avoidance EKMs has the same input weight vector $\mathbf{w}_i$ as the neuron i in the target reaching EKMs. The robot has h directed distance sensors around its body for detecting the presence of obstacles. Therefore, each activated sensor encodes a fixed direction α_j and a variable distance d_j of the obstacle relative to the robot's heading and location.

Obstacle Avoidance

Given the sensed inputs $\mathbf{u}_j$, $j = 1, \ldots, h$,

1. Find the set of winning neurons $\mathcal{S}$ in the obstacle localization EKMs.

   Each sensor input $\mathbf{u}_j$ activates a winning neuron in the jth obstacle localization EKM, which is activated in the same manner as Step 1 of Target Reaching.

2. Compute activity b_i of neuron i in the obstacle field EKM:

$$b_i = \sum_{s \in \mathcal{S}} G_b(\mathbf{w}_s, \mathbf{w}_i) \tag{5.6}$$

where

$$\begin{aligned} G_b(\mathbf{w}_s, \mathbf{w}_i) = &\quad \exp(-D_b(\mathbf{w}_s, \mathbf{w}_i)/\sigma_b^2) \\ D_b(\mathbf{w}_s, \mathbf{w}_i) = &\quad \delta_\alpha(\alpha_s - \alpha_i)^2 + \delta_d(d_s, d_i)(d_s - d_i)^2 \\ \delta_d(d_s, d_i) = &\quad \begin{cases} 1 - (d_i - d_s)^{0.01} & \text{if } d_i \geq d_s \\ 1 + ((d_s - d_i)/\tau)^4 & \text{otherwise} \end{cases} \end{aligned}$$

where δ_α and τ are constant parameters. $G_b(\mathbf{w}_s, \mathbf{w}_i)$ is a Gaussian stretched along the obstacle direction α_s so that motor control EKM neurons beyond the obstacle locations are also inhibited to indicate inaccessibility (Fig. 5.1c). If no obstacle is detected, $G_b = 0$. In the presence of an obstacle, the neurons in the obstacle field EKM at and near the obstacle locations will be activated to produce *obstacle fields* (Eq. 5.6). The neurons nearest to the obstacle locations have the strongest activities.

In activating the motor control EKM (Fig. 5.1d), the obstacle field is subtracted from the target field (Eq. 5.2). If the target lies within the obstacle field, the activation of the motor control EKM neurons close to the target location will be suppressed. Consequently, another neuron at a location that is not inhibited by the obstacle field becomes most highly activated (Fig. 5.1d). This neuron produces a control parameter that moves the robot away from the obstacle. While the robot moves around the obstacle, the target and obstacle localization EKMs are continuously updated with the current locations and directions of the target and obstacles. Their interactions with the motor control and obstacle field EKMs produce fine, smooth, and accurate control of the robot to negotiate the obstacle and move towards the target.

5.4 Summary

This chapter describes a reactive model that is built upon a multitude of behavior-based (target reaching and obstacle avoiding) neural fields dynamically interacting on the local sensory space of the self-organized, indirect-mapping EKM (Fig. 5.1). Each motion command that is sent to the actuators is determined by searching for the neuron with the highest activation. It renovates the reactive model of the "command fusion" architecture by fusing the reactive modules *directly at the level of neural representation*, instead of *indirectly at the action level*. This modification further augments the reactive motion control capabilities by enabling the robot to negotiate unforeseen concave and extended obstacles. The next chapter compares the motion control capabilities of the two integrated planning and control architectures qualitatively.

Chapter 6

Experiments and Discussion

In this chapter, the indirect-mapping EKM is quantitatively assessed with respect to the direct-mapping EKM. The two integrated planning and control architectures (Fig. 1.4), which are coined the "command fusion" architecture (Chapter 4) and the "dynamic neural fields" architecture (Chapter 5), are also qualitatively compared.

6.1 Quantitative Evaluation

This section presents a quantitative evaluation of the indirect-mapping EKM. The tests were performed using Webots (http://www.cyberbotics.com), Khepera mobile robot simulator, which incorporated 10% noise in its sensors and actuators. In the experiments, EKMs with 15×15 neurons were trained in an obstacle-free environment. Each training/testing trial took 100,000 time steps and each time step for target reaching control lasted 1.024 sec. During training, the weights of the EKM were initialized to correspond to regularly spaced locations in the sensory input space $\mathcal{U}$. The robot began the training at the center of the environment and a randomly selected sequence of targets were presented. The robot's task was to move to the targets, one at a time, and weight modification was performed at each time step after the robot had made a move. At each time interval of 10,000 steps during training, a fixed testing procedure was conducted. In each test, the robot began at the center of the environment and was presented with 50 random target locations in sequence. The robot's task was to move to each of the target locations (this time, no training was performed). The above training/testing trial was repeated 5 times and testing performance was averaged over the 5 trials.

The testing performance index measured in the above trials is the *mean positioning error* E, which is the average distance ε_i between the center of the robot and the ith target location after it has come to a stop (i.e., motor control $\mathbf{c} = \mathbf{0}$):

$$E = \frac{1}{RN} \sum_i \varepsilon_i \tag{6.1}$$

where R is the number of trials and N is the number of testing target locations. Test results (Fig. 6.1(a)) show that, with indirect mapping, the self-organization of EKM began to stabilize at 40,000 time steps. At the end of 100,000 time steps, the robot driven by the trained EKM with indirect mapping had a mean positioning error of 3 mm. In comparison, the EKM that adopted direct mapping stabilized at about 50,000 time steps and had a mean positioning error of 8 mm.

The radius of the robot is 25 mm. So, it is reasonable to regard the robot to have reached (and touched) a target if the distance-to-target ε is less than 25 mm. The next three performance indices are based on this target reaching criterion after the robot has been trained.

The *target reaching probability* $P(\varepsilon)$ measures the probability of the robot reaching closer than a distance of ε (with or without stopping) from the target locations. The *normalized*

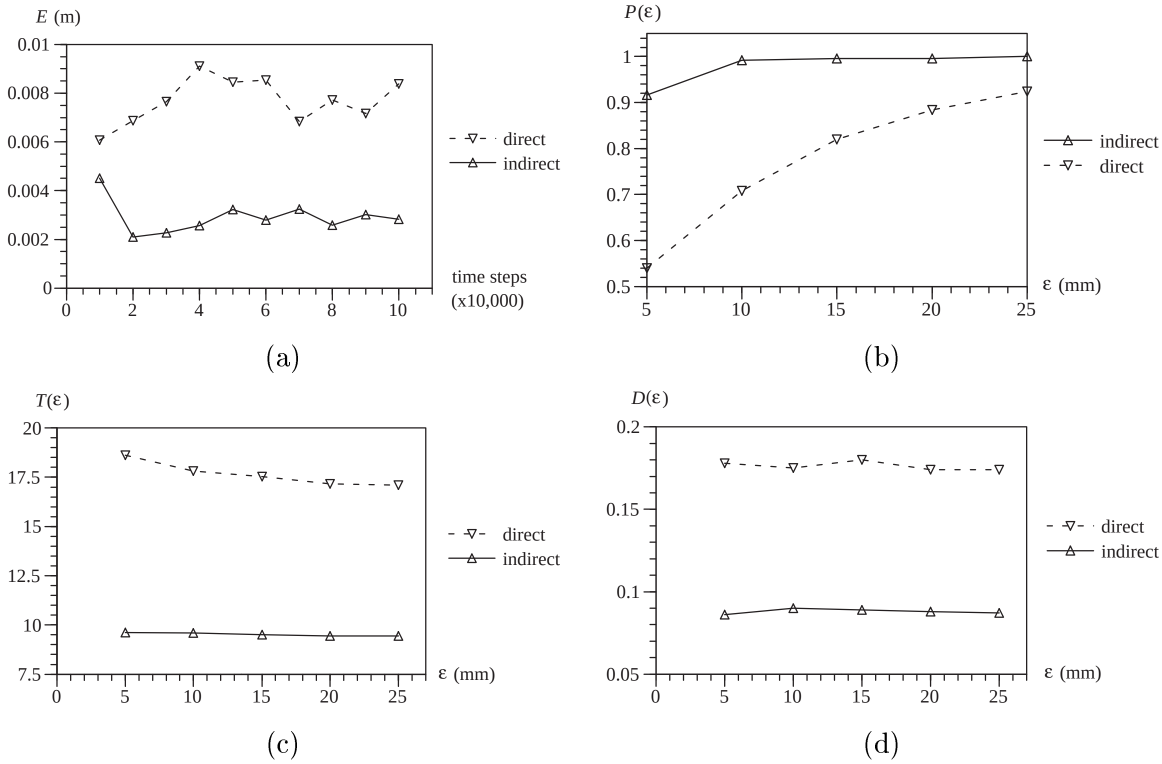


Figure 6.1: Performance comparison. (a) Mean positioning error during training. (b) Target reaching probability, (c) normalized time-to-target, and (d) mean deviation from straight line trajectory after training.

time-to-target $T(\varepsilon)$, measures how long it takes the robot to reach closer than a distance of ε (with or without stopping) from the target locations:

$$T(\varepsilon) = \frac{1}{RN} \sum_i \tilde{t}_i(\varepsilon) \ , \quad \tilde{t}_i(\varepsilon) = \frac{t_i(\varepsilon)}{l_i} \tag{6.2}$$

where $t_i(\varepsilon)$ is the earliest time it takes the robot to reach closer than a distance of ε from the ith target, l_i is the straight line distance between targets $i-1$ and i, and $\tilde{t}_i$ is the normalized time taken to reach target i. That is, normalized time-to-target measures the average amount of time the robot takes to travel a distance of 1 m towards a target. The *mean deviation from straight-line trajectory* $D(\varepsilon)$ measures how straight or wavy is the robot's trajectory:

$$D(\varepsilon) = \frac{1}{RN} \sum_i \tilde{\delta}_i(\varepsilon) \ , \quad \tilde{\delta}_i(\varepsilon) = \frac{\|d_i(\varepsilon) - l_i\|}{l_i} \tag{6.3}$$

where $d_i(\varepsilon)$ is the distance traveled to reach closer than a distance of ε from target location i, and $\tilde{\delta}_i$ is the deviation from straight-line trajectory for target i.

Test results (Fig. 6.1) show that, with indirect mapping, the robot could get very close to the targets with high probability (> 0.9), reached the targets in about 9 time steps, and its trajectories deviated by less than 9% from straight line trajectories. In contrast, with direct mapping, the robot had a lower probability (< 0.9) of reaching close to the targets, took about 17.5 time steps to reach them, and its trajectories deviated by about 18% from straight line. These test results show that, with indirect mapping, the robot can stop closer to the targets, reach the targets faster and travel in straighter paths than with direct mapping.

The advantages of indirect mapping can also be assessed from the results of self-organization. Figure 6.2 illustrates the EKMs at the end of one of the five training trials. The neurons in the direct-mapping EKM were clustered into four clusters: $d = 0$ and $\alpha = -3, 0, +3$ radian. Although the neurons in the indirect-mapping EKM also clustered in a similar manner,

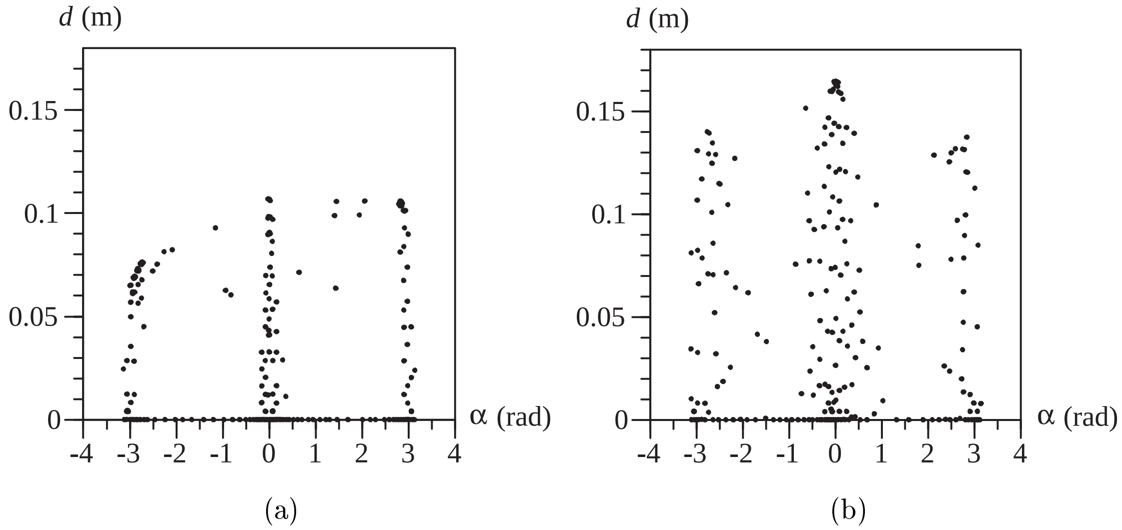


Figure 6.2: Self-organization results of EKM using (a) direct and (b) indirect mapping. Each dot denotes the weights $\mathbf{w}_i = (\alpha_i, d_i)^T$ of a neuron.

its neurons were more spread out. Moreover, they sampled distances up to 0.16 m whereas direct-mapping neurons sampled distances only up to 0.11 m. Note that 0.16 m is the furthest that a Khepera robot can move in a single time step of 1 second. Thus, indirect-mapping EKM samples the sensory input space more completely than does direct-mapping EKM, and produces finer and smoother motor control.

6.2 Qualitative Evaluation

This section details the qualitative comparison between the two integrated planning and control architectures. Experimental results reveal that the “command fusion” architecture cannot be used to overcome unforeseen concave or extended obstacles. It would therefore be meaningful to preempt the comparison with an additional section to evaluate the “command fusion” architecture under normal circumstances, which include negotiation of small unforeseen obstacles, non-adversarial moving obstacles and unexpected change in environment.

6.2.1 Command Fusion: Target Reaching with Short-Range Obstacle Avoidance

The performance of the “command fusion” architecture (Fig. 1.4(a)) was qualitatively assessed in an environment under three unforeseen conditions: (1) static obstacle, (2) moving obstacle, and (3) unexpected change in environment. The environment consisted of three rooms connected by two doorways (Figs. 6.3–6.5). The robot began in the left-most room and was tasked to move to the right-most room via three checkpoints. The robot was regarded to have reached a checkpoint if it was less than 5 mm from the checkpoint. The robot was required to stop at the goal. The target reaching module ran at 1.024 sec interval while the obstacle avoidance module ran at 0.128 sec interval. 8 short-range IR sensors (5 cm range) were modelled around the robot's body (Fig. 4.3). 10% noise was also added into the sensor readings by the Webots simulator.

In the first test (Fig. 6.3), an unforeseen static obstacle was placed in the middle room between the first and second checkpoint. Just before reaching the first checkpoint, the robot detected a wall on its right and deviated slightly to the left to reach the first checkpoint. While moving towards the second checkpoint, the robot detected the unforeseen static obstacle. It reacted to the presence of the obstacle by going around the obstacle to reach the second checkpoint. At the second checkpoint, the robot made a sharp turn due to the direction of the goal and headed towards it.

In the second test (Fig. 6.4), a mobile robot, following an anti-clockwise circular path, served as the moving obstacle. When the target reaching robot first met the obstacle on its left, it tried to avoid by turning right. Subsequently, it encountered the obstacle on its right and was diverted to the left before it moved out of the obstacle's path, headed towards the

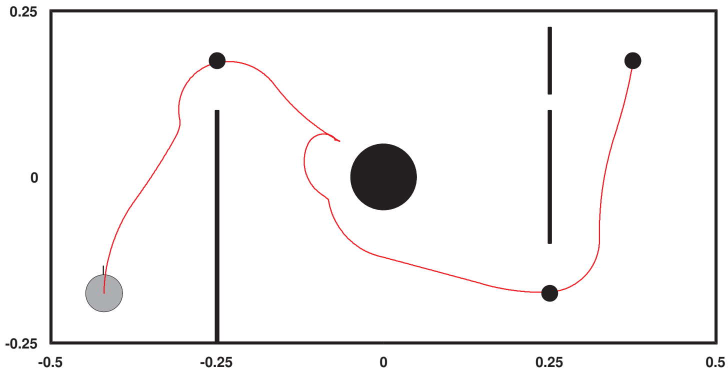


Figure 6.3: Motion of robot (gray) in an environment with an unforeseen static obstacle (black). The checkpoints (small black dots) are located at the doorways and the goal position. The robot was able to go around the obstacle to reach the goal. The robot, obstacle, and rooms are drawn to the same scale.

second checkpoint, and finally towards the goal.

In the last test (Fig. 6.5), the same checkpoints as the previous tests were initially planned for the robot. However, while the robot was en route to the second checkpoint, the high-level planning module realized that the environment had changed. A new checkpoint was planned and given to the target reaching module. Consequently, the target reaching module changed the heading of the robot en route, so that it could reach the goal through the new checkpoint. This test clearly demonstrates the advantage of our integrated planning and control approach. First, a plan can be easily modified by changing only the checkpoints. Second, the target reaching module can react immediately to the change of a checkpoint, and produce a course change at ease. In contrast, existing architectures that plan the entire path need to make detailed modifications to the path such as how to turn the robot around, taking into account the robot's current motion.

In all cases, the robot under the control of the trained EKM was able to move to the checkpoints successfully. The paths taken by the robot between checkpoints were not per-

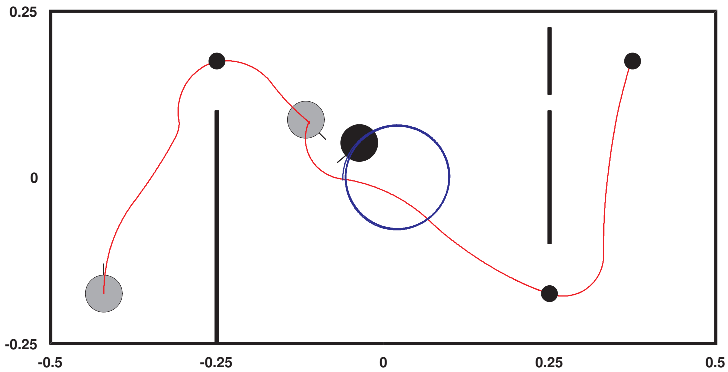


Figure 6.4: Motion of robot (gray) in an environment with an obstacle (black) moving in an anti-clockwise circular path. The robot was able to negotiate past the moving obstacle and proceed to the second checkpoint.

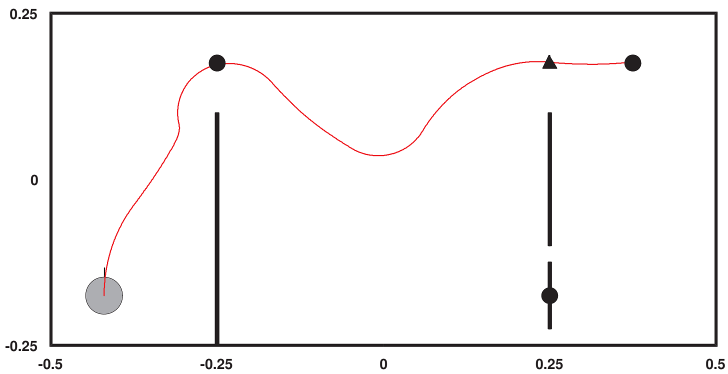


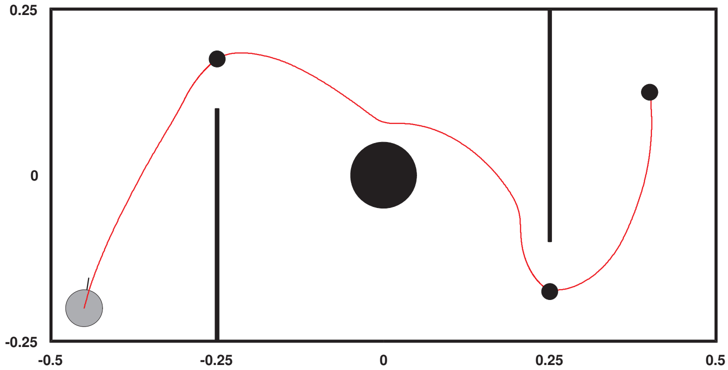
Figure 6.5: Motion of robot (gray) in an environment that changed. The initial checkpoints were planned with the assumption that the lower doorway was opened. The planning module changed the second checkpoint to a new checkpoint (triangle) while the robot was en route to the old one. The target reaching module was able to react immediately and it changed the robot's heading.

fectly straight due to several realistic constraints. The two-wheeled robot was nonholonomic; it was much harder to control and achieve smoothness in motion (Section 1.3). The motor control injections by the obstacle avoidance and target reaching modules were not strictly continuous, but at discrete servo intervals. Furthermore, the Webots simulator automatically injected noise into the sensor inputs and motor outputs, which offered realistic simulation of low-level robot control. Nevertheless, the paths taken were quite close to straight lines.

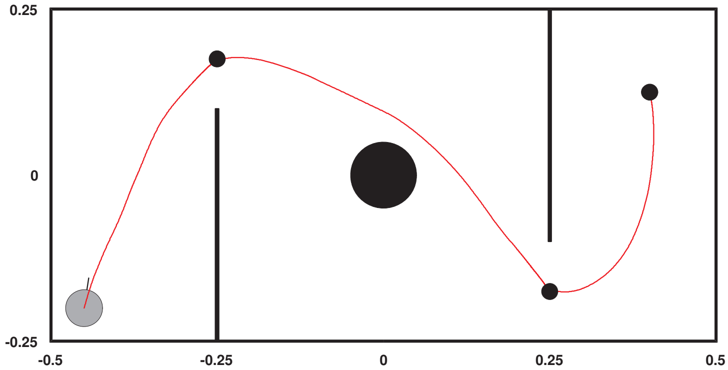
6.2.2 Behavioral Coordination: Command Fusion vs. Dynamic Neural Fields

Three tests were performed to compare the two integrated planning and control architectures. For both architectures, the target reaching and obstacle avoidance modules ran at 0.128 sec intervals. 12 directed long-range sensors were modelled around the robot's body of radius 2.5 cm. Each sensor had a range of 17.5 cm, enabling the detection of obstacles at 20 cm or nearer from the robot center. To simulate noise, the sensors have a resolution of $\tau = 0.5$ cm. The robot's performance is assessed in an environment under three *unforeseen* conditions: (1) stationary obstacle, (2) concave obstacle, and (3) extended obstacle.

The environment and conditions for the first test with the unforeseen static obstacle is the same as the one used in the previous section (Fig. 6.3). The robot fitted with the "command fusion" architecture using long-range sensors (17.5 cm range) failed its task. It could not pass through the first narrow doorway as the repulsive forces from neighboring walls were too strong. Reducing effect of these forces allowed the robot to pass through the doorway but resulted in it getting stuck at the static obstacle. However, the robot with the "command fusion" architecture using short-range sensors (5cm range) could complete its task as shown in Fig. 6.6(a). These results indicate that the "command fusion" architecture could only



(a)



(b)

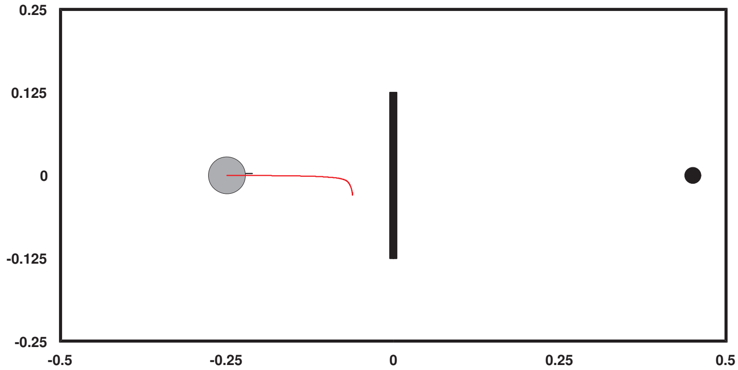
Figure 6.6: Comparison of robot (gray) motions in an environment with an unforeseen stationary obstacle (black) between "command fusion" and "dynamic neural fields" architectures. The checkpoints (small black dots) were located at the doorways and the goal position. (a) The robot endowed with the "command fusion" architecture using short-range sensors negotiated past the obstacle to reach the goal. (b) The robot endowed with the "dynamic neural fields" architecture using longer-range sensors detected the unforeseen stationary obstacle earlier and moved along a shorter path to reach the goal.

work well with short-range sensors.

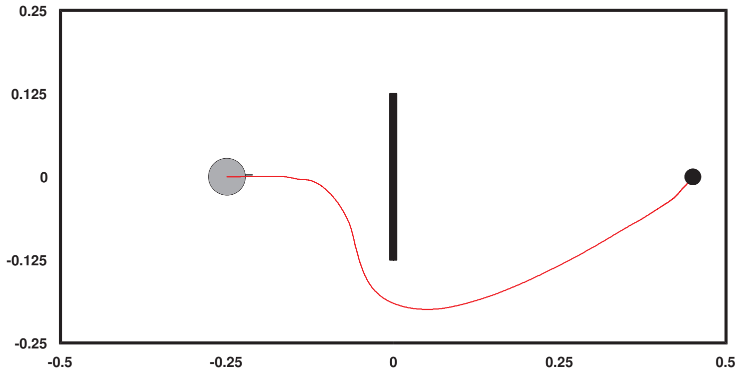
The above phenomenon can be explained by the following: by fusing the reactive modules indirectly at the action level, the target reaching module could not communicate to the Braitenberg obstacle avoidance module that the narrow opening among the walls offered a through passage. Recall from Section 4.5 that command fusion is analogous to the combination of attractive and repulsive forces. Therefore, to pass through the doorway, the repulsive forces from the walls acting on the robot must somehow be lower than the attractive force to the designated goal. This could only be effectuated by either manually reducing the repulsive effect or shortening the range of the distance sensors so that the walls would rarely be detected unless being approached.

Figure 6.6b demonstrated how the robot endowed with the “dynamic neural fields” architecture using long-range sensors completed its task. The robot detected and avoided the unforeseen static obstacle early due to its long-range sensors. Consequently, it moved along a shorter path to reach the goal. The doorways were also detected, thus allowing the robot to reach the checkpoints. By fusing the reactive modules directly at the level of neural representations, the target reaching and obstacle avoidance modules could negotiate dynamically via the common local sensory space about the free space for the robot to move to and the forbidden space to cordon due to obstacles using neural fields, which are produced by the EKMs.

The next two tests assessed the performance of the robot in negotiating unforeseen extended (Fig. 6.7) and concave obstacles (Fig. 6.8). The robot that adopted the “command fusion” architecture got trapped by the obstacles regardless of the sensor range (Figs. 6.7a, 6.8a). The lack of direct interaction between the two reactive modules caused a conflict of interest:

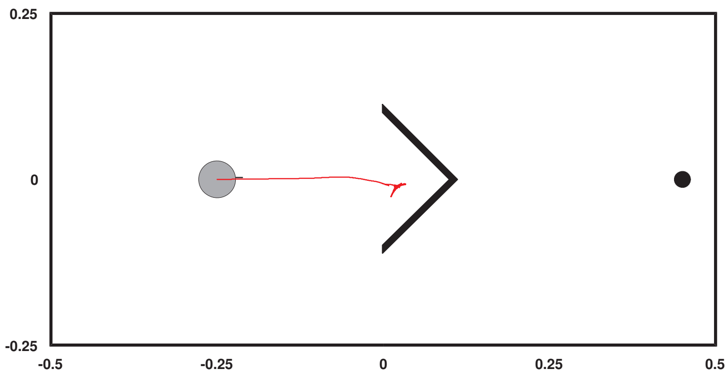


(a)

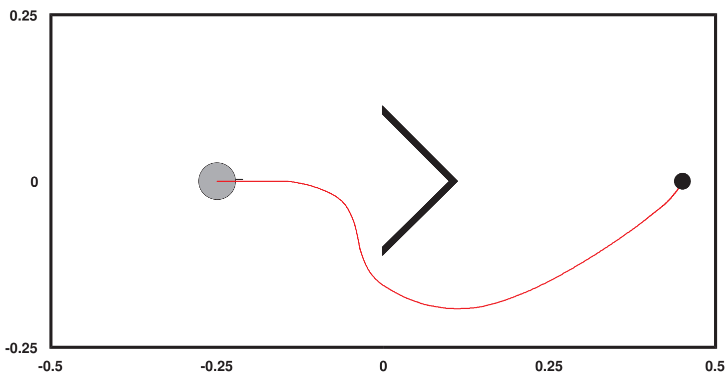


(b)

Figure 6.7: Negotiating unforeseen extended obstacle. (a) The robot using the "command fusion" architecture was trapped but (b) the one adopting the "dynamic neural fields" architecture successfully negotiated the obstacle.



(a)



(b)

Figure 6.8: Negotiating unforeseen concave obstacle. (a) The robot using the "command fusion" architecture was trapped but (b) the one adopting the "dynamic neural fields" architecture successfully negotiated the obstacle.

the target reaching module provided an attractive force to move the robot forward to reach the target while the obstacle avoidance module created repulsive forces to move it backward to avoid the obstacle. The combined output cancelled out each other's actions. On the other hand, the robot that adopted the "dynamic neural fields" architecture successfully negotiated the obstacles to reach the goal (Figs. 6.7b, 6.8b). The "dynamic neural fields" architecture facilitated the direct interaction between the target reaching and obstacle avoidance modules to resolve their conflicts on the common local sensory space using neural fields and produce more appropriate actions. All these results illustrate the advantage of the "dynamic neural fields" architecture over the "command fusion" architecture. In other words, the direct coupling of the reactive modules at the level of neural representations offers better qualitative performance than the indirect coupling of the reactive modules at the action level.

It is noted that the robot with the "dynamic neural fields" architecture can still get trapped if the obstacle is so concave that the obstacle fields cannot completely inhibit the neurons at or near the target location. However this limitation does not diminish the significance of the method as it is simpler than many existing reactive methods for overcoming unexpected concave obstacles (Balch and Arkin, 1993; Choi *et al.*, 1989; Choi and Latombe, 1991; Gambardella and Haex, 1993; Gambardella and Versino, 1994; Gat, 1993b; Lagoudakis and Maida, 1999; Liu *et al.*, 2000; Slack, 1990, 1993; Zelek and Levine, 1996). In particular, it maintains only local information about the checkpoint and the obstacles. Moreover, it is possible to train the neural networks to adapt to different kinds of obstacles while optimizing its performance.

Chapter 7

Future Work

This chapter reveals the limitations of the current work in the thesis and presents possible improvements, applications and extensions to the existing integrated planning and control architectures.

7.1 Improvements

7.1.1 Reactive Cognizant Failure Capabilities

A robot has to be robust to both failures and errors related to the task under execution; this can be effectuated by providing the robot with fast error recovery procedures that can be executed before or instead of replanning (Noreils, 1990; Noreils and Chatila, 1995). Nevertheless, the robot must be able to detect (as opposed to avoid) and recognize on its own any failures to complete its task via monitor routines (e.g., time-out) in order to call the appropriate recovery procedure. This is known as *cognizant failure* (Firby, 1989; Gat, 1991a; Gat and Dorais, 1994; Howe and Cohen, 1991). Instead of attempting to design algorithms in the robot controller that never fail (which is impossible on real robots), one can instead design algorithms that never fail to detect a failure. This allows other components of the system to take corrective action to recover from the failure. Therefore, endowing the integrated planning and control architectures with reactive cognizant failure capabilities would greatly enhance their real-time motion control capabilities.

7.1.2 Graph Connectivity and Search Algorithms

To make the high-level deliberative motion planner complete, the algorithms for graph connectivity and search can be implemented to automatically generate the sequence of checkpoints for low-level reactive motion control to perform goal-directed, collision-free motion.

7.2 Extensions

The integrated planning and control architectures should ultimately be built on real mobile robots to investigate their aptness for real-world applications. Real-time issues are often the primary cause of axing software robot architectures that are too complicated.

7.3 Applications

The integrated planning and control architectures can be applied to static and mobile robot manipulators (Brock and Kavraki, 2000, 2001; Brock and Khatib, 1997, 1998, 1999b, 2000) as well as collective robotics (Hannebauer *et al.*, 2001).

Chapter 8

Conclusion

This thesis presents two novel integrated planning and control architectures for a nonholonomic mobile robot, which are termed "command fusion" architecture and "dynamic neural fields" architecture. The unification of high-level deliberative motion planning and low-level reactive motion control into one coherent integrated hybrid framework can solve the problem of goal-directed, collision-free motion in an unstructured environment. Furthermore, to be able to execute a sophisticated task in our complex and unpredictable world, real-time performance and fine, smooth control have to be accentuated. The following objectives have been satisfied to endow the integrated planning and control architectures with real-time enhanced motion control capabilities.

1. **Enhancing the capabilities of low-level reactive motion control**

   (a) **Enriching the repertoire of reactive behaviors**

   Both architectures perform target reaching as well as obstacle avoidance. This is in contrast to many hybrid systems (Brock and Khatib, 1999a; Brock and Kavraki, 2000, 2001; Fox *et al.*, 1998; Hu and Brady, 1994; Pyeatt and Howe, 1999; Simmons *et al.*, 1997a; Thrun *et al.*, 1998a) that employ the deliberative planner to plot the exact motion path with detailed sequence of actions for execution by the actuators. Their reactive controllers perform only a single task, i.e., simple obstacle avoidance, by making minor modifications to an otherwise good course of action.

(b) Performing fine, smooth and efficient motion control

A high degree of smoothness, flexibility and precision in motion control is essential for the efficient execution of sophisticated tasks as well as physical interaction with humans. This prerequisite can only be achieved with continuous response encoding of very low-level velocity/torque control of motor/joint actuators. Hence, by tightly coupling high-level deliberative motion planning with reactive control at the lowest level, these complex tasks can be accomplished.

Both architectures are among the few hybrid systems (Arkin and Balch, 1997; Brock and Khatib, 1998; Choi and Latombe, 1991; Firby and Slack, 1995; Gat, 1992; Khatib *et al.*, 1997; Krogh and Thorpe, 1986; Saffiotti *et al.*, 1995; Schöners and Dose, 1992; Simmons *et al.*, 1997a) that perform continuous response encoding of very low-level control to produce fine, smooth motion. This is in contrast to behavior-based architectures (Agre and Chapman, 1987; Bonasso, 1991; Brooks, 1986; Connell, 1990; Kaelbling and Rosenschein, 1990; Maes, 1990; Matarić, 1992; Rosenblatt, 1997; Sahota, 1994) and several hybrid architectures (Connell, 1992; Decugis and Ferber, 1998; Drummond *et al.*, 1991; Firby, 1995; Gat, 1998; Georgeff and Lansky, 1987; Hayes-Roth *et al.*, 1995a; Kaelbling and Rosenschein, 1990; Laird and Rosenbloom, 1990; Lyons and Hendriks, 1995; Malcolm and Smithers, 1990; Mitchell, 1990; Musliner *et al.*, 1993; Benson and Nilsson, 1995; Noreils and Chatila, 1995; Payton, 1990; Wilkins *et al.*, 1995; Zelek and Levine, 1996; Zelinsky and Kuniyoshi, 1996) that employ discrete response encoding.

However, instead of using pre-wired reactive sensorimotor controllers such as Artificial Potential Fields (Khatib, 1985, 1986), Navigation Templates (Gat, 1993b; Slack, 1990, 1993), Motor Schemas (Arkin, 1987, 1989b, 1992b), Deformation Zone (Zapata *et al.*, 1991), Curvature Velocity Method (Ko and Simmons, 1998; Simmons, 1996), and Fuzzy Logic (Driankov and Saffiotti, 2000) in the hybrid systems to perform continuous response encoding, a novel self-organizing neural network technique known as indirect-mapping EKM is embedded in both architectures to learn the low-level reactive motion control. Its indirect sensorimotor mapping contrasts with many existing methods, which perform direct mapping. This modified property evolves significant quantitative benefits (Section 6.1).

(c) Negotiating unforeseen, complex obstacles

The "dynamic neural fields" architecture (Chapter 5) uses multiple cooperative EKMs to fuse the reactive modules directly at the level of neural representation, rather than indirectly at the action level for the "command fusion" architecture. In essence, it is a multitude of behavior-based (target reaching and obstacle avoiding) neural fields produced by EKMs dynamically interacting on the local sensory space of the self-organized, indirect-mapping EKM (Fig. 5.1). This dynamic interaction of neural fields constantly negotiates the free space for the robot to move to and the forbidden space cordoned by the obstacles during runtime. This form of direct fusion further augments the reactive motion control capabilities by enabling the robot to negotiate unforeseen concave and extended obstacles.

2. **Reducing the workload of high-level deliberative motion planner**

   Two improvements have been made to reduce the workload of the deliberative motion planner. By planning the robot's motion in the workspace instead of the configuration space, computational efficiency can be enhanced. An approximate cell decomposition technique known as slippery cells is introduced to decompose the free workspace into much fewer cells or graph nodes than the other approximate methods, resulting in reduced search time. Yet any two points in the cell can still be successfully traversed by reactive control mechanisms. Using this graph, the planner generates a series of checkpoints from the start point to the goal (Section 3.2.3), which is required by the reactive models in both integrated planning and control architectures (Chapters 4 and 5) to perform goal-directed, collision-free motion from one checkpoint to the next.

Bibliography

Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 268–272.

Agre, P. E. and Chapman, D. (1990). What are plans for? In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 17–34. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Albus, J. S. (1991a). Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, **21**(3), 473–509.

Albus, J. S. (1991b). A theory of intelligent machine systems. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 3–9.

Albus, J. S. (1995). RCS: A reference model architecture for intelligent systems. In *Working Notes: AAAI Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents*, pages 1–6.

Albus, J. S. (1997). The NIST real-time control system (RCS): An approach to intelligent systems research. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 157–174.

Albus, J. S., McCain, H. G., and Lumia, R. (1989). NASA/NBS standard reference model for telerobot control system architecture (NASREM). Technical Report 1235, National Institute of Standards and Technology, Gaithersburg, MD.

Anderson, A. M. (1977). A model for landmark learning in the honeybee. *Journal of Comparative Physiology A*, **114**, 335–355.

Arkin, R. C. (1987). Motor schema based mobile robot navigation: An approach to programming by behavior. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 264–271.

Arkin, R. C. (1989a). Towards the unification of navigational planning and reactive control. In *Working Notes: AAAI Spring Symposium on Robot Navigation*, pages 1–5.

Arkin, R. C. (1989b). Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, **8**(4), 92–112.

Arkin, R. C. (1990). Integrating behavioral, perceptual, and world knowledge in reactive navigation. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From*

Biology to Engineering and Back, pages 105–122. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Arkin, R. C. (1992a). Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments. *Journal of Robotic Systems*, **9**(2), 197–214.

Arkin, R. C. (1992b). Behavior-based robot navigation for extended domains. *Adaptive Behavior*, **1**(2), 201–225.

Arkin, R. C. (1995). Reactive robotic systems. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 793–796. MIT Press, Cambridge, MA.

Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge, MA.

Arkin, R. C. and Balch, T. (1997). AuRA: Principles and practices in review. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 175–189.

Aurenhammer, F. (1991). Voronoi diagrams – a survey of fundamental geometric data structure. *ACM Computing Surveys*, **23**(3), 345–405.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of International Conference on Machine Learning*, pages 30–37.

Balch, T. R. and Arkin, R. C. (1993). Avoiding the past: A simple but effective strategy for reactive navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 678–685.

Baral, C. and McIlraith, S., cochairs. (2002). AAAI Workshop on cognitive robotics. Technical Report WS-02-07.

Barraquand, J. and Latombe, J.-C. (1991). Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, **10**(6), 628–649.

Bekey, G. and Goldberg, K., editors (1993). *Neural Networks in Robotics*. Kluwer, Boston, MA.

Benson, S. and Nilsson, N. J. (1995). Reacting, planning and learning in an autonomous agent. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 14*. Oxford: The Clarendon Press.

Berns, K., Dillmann, R., and Hofstetter, R. (1991). An application of a backpropagation network for the control of a tracking behavior. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2426–2431.

Berns, K., Dillmann, R., and Zachmann, U. (1992). Reinforcement learning for the control of an autonomous mobile robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1808–1815.

Berthouze, L. and Kuniyoshi, Y. (1998). Emergence and categorization of coordinated visual behavior through embodied interaction. *Machine Learning*, **31**(1-3), 187–200.

Bicho, E. and Schöners, G. (1997). The dynamic approach to autonomous robotics demonstrated on a low-level vehicle platform. *Robotics and Autonomous Systems*, **21**(1), 23–35.

Blythe, J. and Mitchell, T. (1989). On becoming reactive. In *Proceedings of International Workshop on Machine Learning*, pages 255–259. Morgan Kaufmann.

Boddy, M. and Dean, T. (1993). Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, **67**(2), 245–285.

Bonarini, A. (1997). Anytime learning and adaptation of hierarchical fuzzy logic behaviors. *Adaptive Behavior*, **5**(3-4), 281–315.

Bonasso, R. P. (1991). Integrating reaction plans and layered competences through synchronous control. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D., and Slack, M. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 237–256.

Borenstein, J. and Koren, Y. (1991a). The vector field histogram: Fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, **7**(3), 278–288.

Borenstein, J. and Koren, Y. (1991b). Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, **7**(4), 535–539.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA.

Bresina, J. (1993). Design of a reactive system based on classical planning. In *Working Notes: AAAI Spring Symposium on Foundations of Automatic Planning: The Classical Approach and Beyond*.

Bresina, J. and Drummond, M. (1990). Integrating planning and reaction: A preliminary report. In *Working Notes: AAAI Spring Symposium on Planning in Uncertain, Unpredictable or Changing Environments*.

Brock, O. and Kavraki, L. E. (2000). Towards real-time motion planning in high-dimensional spaces. In *Proceedings of International Symposium on Robotics and Automation*.

Brock, O. and Kavraki, L. E. (2001). Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1469–1474.

Brock, O. and Khatib, O. (1997). Elastic strips: Real-time path modification for mobile manipulation. In *Proceedings of International Symposium of Robotics Research*.

Brock, O. and Khatib, O. (1998). Executing motion plans for robots with many degrees of freedom in dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 1–6.

Brock, O. and Khatib, O. (1999a). High-speed navigation using the global dynamic window approach. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 341–346.

Brock, O. and Khatib, O. (1999b). Elastic strips: A framework for integrated planning and execution. In *Proceedings of International Symposium on Experimental Robotics*, volume 1, pages 341–346.

Brock, O. and Khatib, O. (2000). Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 550–555.

Brooks, R. (1983). Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, and Cybernetics*, **13**(3), 190–197.

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

Brooks, R. (1989). A robot that walks: Emergent behavior from a carefully evolved network. *Neural Computation*, **1**(2), 253–262.

Brooks, R. (1990). Elephants don't play chess. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 3–15. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Brooks, R. and Lozano-Pérez, T. (1985). A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, **15**(2), 224–233. Also appeared in *Proceedings of International Joint Conference on Artificial Intelligence*, 799–806, 1983.

Brooks, R., Grossberg, S., and Optican, L., editors. (1998). Neural control and robotics: Biology and technology. *Special Issue of Neural Networks*, **11**(7-8).

Burgard, W., Cremers, A. B., Fox, D., Hhnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1998). The interactive museum tour-guide robot. In *Proceedings of National Conference on Artificial Intelligence (AAAI).*

Burgard, W., Cremers, A. B., Fox, D., Hhnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, **114**(1-2), 3–55.

Canny, J. F. (1988). *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, Massachusetts.

Cartwright, B. A. and Collett, T. S. (1983). Landmark learning in bees. *Journal of Comparative Physiology A*, **151**, 521–543.

Castellano, G., Attolico, G., Stella, E., and Distante, A. (1996). Automatic generation of rules for a fuzzy robotic controller. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1179–1186.

Chapman, D. and Kaelbling, L. P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Chatila, R. (1995). Deliberation and reactivity in autonomous mobile robots. *Robotics and Autonomous Systems*, **16**(2-4), 197–211.

Chatila, R. and Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 138–145.

Chatila, R., Alami, R., Degallaix, B., and Laruelle, H. (1992). Integrated planning and execution control of autonomous robot actions. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2689–2696.

Choi, W. and Latombe, J. C. (1991). A reactive architecture for planning and executing robot motions with incomplete knowledge. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 1, pages 24–29.

Choi, W., Zhu, D., and Latombe, J. C. (1989). Contingency-tolerant motion planning and control. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 78–85.

Collett, T. S. (1996). Insect navigation en route to the goal: Multiple strategies for the use of landmarks. *Journal of Experimental Biology*, **199**, 227–235.

Congdon, C., Huber, M., Kortenkamp, D., Konolige, K., Myers, K., Ruspini, E. H., and Saffiotti, A. (1993). Carmel versus flakey: A comparison of two winners. *AI Magazine*, **14**(1), 49–57.

Connell, J. H. (1987). Creature building with the subsumption architecture. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1124–1126.

Connell, J. H. (1990). *Minimalistic Mobile Robotics: A Colony Architecture for an Artificial Creature*. Academic Press.

Connell, J. H. (1992). SSS: A hybrid architecture applied to robot navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2719–2724.

Connell, J. H. and Mahadevan, S., editors (1993). *Robot Learning*. Kluwer Academic Publishers, Reading, MA.

Connolly, C. I. (1994). Harmonic functions and collision probabilities. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3015–3019.

Connolly, C. I. and Grupen, R. A. (1993). On the applications of harmonic functions to robotics. *Journal of Robotic Systems*, **10**(7), 931–946.

Connolly, C. I., Burns, J. B., and Weiss, R. (1990). Path planning using Laplace's equation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2101–2106.

Davids, A. (2002). Urban search and rescue robots: From tragedy to technology. *IEEE Intelligent Systems*, **17**(2), 81–83.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (1997). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg.

de Giacomo, G., chair. (1998). AAAI Fall Symposium on cognitive robotics. Technical Report FS-98-02.

Dean, T. and Boddy, M. (1988). An analysis of time-dependent planning. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*.

Dean, T. and Wellman, M. (1991). *Planning and Control*. Morgan Kaufmann, San Mateo, CA.

Decugis, V. and Ferber, J. (1998). Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In *Proceedings of 2nd International Conference on Autonomous Agents*, pages 354–361.

Donald, B., Jennings, J., and Rus, D. (1997). Minimalism + Distribution = Supermodularity. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 293–321.

Donnart, J.-Y. and Meyer, J.-A. (1994). A hierarchical classifier system implementing a motivationally autonomous animat. In *From Animals To Animats 3: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Dorigo, M., editor. (1996). Learning autonomous robots. *Special Issue of IEEE Transactions on Systems, Man, and Cybernetics – Part B*, **26**(3).

Driankov, D. and Saffiotti, A. (2000). *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*. Physica-Verlag.

Drummond, M. (1989). Situated control rules. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning*, pages 103–113.

Drummond, M., Bresina, J., and Kedar, S. (1991). The entropy reduction engine: Integrating planning, scheduling, and control. In *Working Notes: AAAI Spring Symposium on Integrated Cognitive Architectures*. Also appeared in *SIGART Bulletin*, **2**(4), 61–65.

Drummond, M., Swanson, K., Bresina, J., and Levinson, R. (1993). Reaction-first search. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Durrant-Whyte, H., Majumder, S., Thrun, S., de Battista, M., and Scheding, S. (2001). A bayesian algorithm for simultaneous localization and map building. In *Proceedings of International Symposium of Robotics Research*.

Elfes, A. (1986). A sonar-based mapping and navigation system. In *Proceedings of IEEE International Conference on Robotics and Automation*.

Engels, C. and Schöners, G. (1995). Dynamic fields endow behavior-based robots with representations. *Robotics and Autonomous Systems*, **14**(1), 55–77.

Everett, H. R. and Gage, D. W. (1999). From laboratory to warehouse: Security robots meet the real world. *International Journal of Robotics Research*, **18**(7), 760–768.

Faverjon, B. (1984). Obstacle avoidance using an octree in the configuration space of a manipulator. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 504–512.

Faverjon, B. (1986). Object level programming of industrial robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1406–1412.

Faverjon, B. and Tournassoud, P. (1987). A local approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1152–1159.

Feder, H. J. S. and Slotine, J.-J. E. (1997). Real-time path planning using harmonic potentials in dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 874–881.

Feiten, W., Bauer, R., and Lawitzky, G. (1994). Robust obstacle avoidance in unknown and cramped environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2412–2417.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, **2**, 251–288.

Fiorini, P., Kawamura, K., and Prassler, E., editors. (2000). Cleaning and housekeeping robots. *Special Issue of Autonomous Robots*, **9**(3).

Firby, R. J. (1989). *Adaptive Execution in Complex Dynamic Worlds*. Ph.D. Dissertation, Yale University, New Haven, CT. Technical Report YALEU/CSD/RR #672.

Firby, R. J. (1993). An architecture for a synthetic vacuum cleaner. In *Working Notes: AAAI Fall Symposium on Instantiating Real-World Agents*, pages 55–63.

Firby, R. J. (1994). Architecture, representation and integration: An example from robot navigation. In *Working Notes: AAAI Fall Symposium on Control of the Physical World by Intelligent Agents*.

Firby, R. J. (1995). Lessons learned from the animate agent project (so far). In *Working Notes: AAAI Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents*, pages 92–96.

Firby, R. J. and Slack, M. G. (1995). Task execution: Interfacing to reactive skill networks. In *Working Notes: AAAI Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents*, pages 97–111.

Firby, R. J., Kahn, R. E., Prokopowicz, P. N., and Swain, M. J. (1995). An architecture for vision and action. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 72–79.

Floreano, D. and Mondada, F. (1996). Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, **26**(3), 396–407.

Fox, D., Burgard, W., and Thrun, S. (1996). Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1280–1287.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, **4**(1), 23–33.

Fox, D., Burgard, W., Thrun, S., and Cremers, A. B. (1998). A hybrid collision avoidance method for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1238–1243.

Franklin, J. A., Mitchell, T. M., and Thrun, S., editors. (1996). Recent advances in robot learning. *Special Issue of Machine Learning*, **23**(2-3).

Franz, M. O., Schölkopf, B., Mallot, H. A., and Bülthoff, H. H. (1998). Where did i take that snapshot? scene-based homing by image matching. *Biological Cybernetics*, **79**, 191–202.

Gambao, E. and Balaguer, C., editors. (2002). Robotics and automation in construction. *IEEE Robotics and Automation Magazine*, **9**(1).

Gambardella, L. M. and Haex, M. (1993). Incorporating learning in motion planning techniques. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 712–715.

Gambardella, L. M. and Versino, C. (1994). Robot motion planning: Integrating planning strategies and learning methods. In *Proceedings of International Conference on AI Planning Systems*.

Gaskett, C., Wettergreen, D., and Zelinsky, A. (1999). Q-learning in continuous state and action spaces. In *Proceedings of Australian Joint Conference on Artificial Intelligence*.

Gat, E. (1991a). Robust low-computation sensor-driven control for task-directed navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2484–2489.

Gat, E. (1991b). Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation. In *Working Notes: AAAI Spring Symposium on Integrated Cognitive Architectures*. Also appeared in *SIGART Bulletin*, **2**(4), 70–74.

Gat, E. (1992). Integrating planning and reaction in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*.

Gat, E. (1993a). Design for an autonomous agent. In *Working Notes: AAAI Fall Symposium on Instantiating Real-World Agents*, pages 64–65.

Gat, E. (1993b). Navigation templates: Enhancements, extensions, and experiments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 541–547.

Gat, E. (1998). On three-layer architectures. In D. Kortenkamp, R. P. Bonnasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pages 195–210. AAAI Press.

Gat, E. and Dorais, G. (1994). Robot navigation by conditional sequencing. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1293–1299.

Gat, E., Desai, R. S., Ivlev, R., Loch, J., and Miller, D. P. (1994). Behavior control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation*, **10**(4), 490–503.

Gaussier, P. and Zrehen, S., editors. (1995). Moving the frontiers between robotics and biology. *Special Issue of Robotics and Autonomous Systems*, **16**(2-4).

Georgeff, M. P. and Ingrand, F. F. (1989). Decision-making in an embedded reasoning system. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 972–978.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 677–682.

Giralt, G., Sobek, R., and Chatila, R. (1979). A multi-level planning and navigation system for a mobile robot: A first approach to hilare. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Giralt, G., Chatila, R., and Vaisset, M. (1984). An integrated navigation and motion control system for autonomous multi-sensory mobile robots. In *Proceedings of 1st International Symposium on Robotics Research*, pages 191–214.

Goodridge, S. G. and Luo, R. C. (1994). Fuzzy behavior fusion for reactive control of an autonomous mobile robot: MARGE. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1622–1627.

Grosz, B. J. and Sidner, C. L. (1988). Plans for discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*. MIT Press, Cambridge, MA.

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, **3**, 671–692.

Hagras, H., Callaghan, V., and Colley, M. (2000). Online learning of the sensors fuzzy membership functions in autonomous mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3233–3238.

Hannebauer, M., Wendler, J., and Pagello, E., editors. (2001). *Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From RoboCup to Real-World Applications*. Lecture Notes in AI 2103, Springer-Verlag Berlin Heidelberg.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, **20**(2–4), 205–224.

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, **26**(3), 251–321.

Hayes-Roth, B., Lalanda, P., Morignot, P., Pfleger, K., and Balabanovic, M. (1993). Plans and behavior in intelligent agents. KSL Technical Report 93-43, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Hayes-Roth, B., Pfleger, K., Lalanda, P., Morignot, P., and Balabanovic, M. (1995a). A domain-specific software architecture for adaptive intelligent systems. *IEEE Transactions on Software Engineering*, **21**(4), 288–301.

Hayes-Roth, B., Pfleger, K., Morignot, P., and Lalanda, P. (1995b). Plans and behavior in intelligent agents. KSL Technical Report 95-35, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Heikkonen, J. and Koikkalainen, P. (1997). Self-organization and autonomous robots. In O. Omidvar and P. van der Smagt, editors, *Neural Systems for Robotics*, pages 297–337. Academic Press.

Heikkonen, J., Koikkalainen, P., and Oja, E. (1993). From situations to actions: Motion behavior learning by self-organization. In *Proceedings of International Conference on Artificial Neural Networks*, pages 262–267.

Hertzberg, J., Christaller, T., Kirchner, F., Licht, U., and Rome, E. (1998). Sewer robotics. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proceedings of International Conference on Simulation of Adaptive Behavior*, pages 427–436, Cambridge, MA. MIT Press.

Hexmoor, H. and Kortenkamp, D., cochairs. (1995a). AAAI Spring Symposium on lessons learned from implemented software architectures for physical agents. Technical Report SS-95-02.

Hexmoor, H. and Kortenkamp, D. (1995b). Issues on building software for hardware agents. *Knowledge Engineering Review*, **10**(3), 301–304.

Hexmoor, H. and Matarić, M., editors. (1998). Learning in autonomous robots. *Joint Special Issue of Machine Learning*, **31**(1-3). Also appeared in *Autonomous Robots*, **5**(3-4).

Hexmoor, H., Kortenkamp, D., and Horswill, I., editors. (1997). Software architectures for hardware agents. *Special Issue of Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3).

Homaifar, A. and McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, **3**(2), 129–139.

Hong, J., Tan, X., Pinette, B., Weiss, R., and Riseman, E. M. (1991). Image-based homing. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 620–625.

Hougen, D. F., Gini, M., and Slagle, J. (2000). An integrated connectionist approach to reinforcement learning for robotic control: The advantages of indexed partitioning. In *Proceedings of International Conference on Machine Learning*, pages 383–390.

Howe, A. E. and Cohen, P. R. (1991). Failure recovery: A model and experiments. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*.

Hsu, S. C., Hsu, J. Y., and Chiang, I. J. (1995). Automatic generation of fuzzy control rule by machine learning methods. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 287–292.

Hu, H. and Brady, M. (1994). A Bayesian approach to real-time obstacle avoidance for a mobile robot. *Autonomous Robots*, **1**, 69–92.

Hwang, Y. K. and Ahuja, N. (1992). Gross motion planning – a survey. *ACM Computing Surveys*, **24**(3), 219–291.

Iske, B., Ruckert, U., Malmstrom, K., and Sitte, J. (2000). A bootstrapping method for autonomous and in-site learning of generic navigation behavior. In *Proceedings of International Conference on Pattern Recognition*, volume 4, pages 656–659.

Jamisola, R., Lim, T. M., Ang Jr., M. H., Oetomo, D. N., Khatib, O., and Lim, S. Y. (2002). Operational space formulation implementation to aircraft canopy polishing application using a mobile manipulator. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 400–405.

Jarvis, R. A. and Byrne, J. C. (1986). Robot navigation: Touching, seeing and knowing. In *Proceedings of Australian Conference on Artificial Intelligence*.

Jou, C.-C. and Wang, N.-C. (1993). Training a fuzzy controller to back up an autonomous vehicle. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 923–928.

Kaelbling, L. P. (1986). An architecture for intelligent reactive systems. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning About Actions and Plans*, pages 395–410. Morgan Kaufmann.

Kaelbling, L. P. (1988). Goals as parallel program specifications. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*.

Kaelbling, L. P. and Rosenschein, S. J. (1990). Action and planning in embedded agents. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 35–48. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, **4**, 237–285.

Kambhampati, S. and Davis, L. S. (1986). Multiresolution path planning for mobile robots. *IEEE Journal of Robotics and Automation*, **2**(3), 135–145.

Kavraki, L. E. and Latombe, J.-C. (1994). Randomized preprocessing of configuration space for fast path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2138–2145.

Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, **12**(4), 566–580.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 500–505.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, **5**(1), 90–98.

Khatib, O. (1987). A unified approach to motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, **3**(1), 43–53.

Khatib, O. (1999). Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems*, **26**, 175–183.

Khatib, O., Quinlan, S., and Williams, D. (1997). Robot planning and control. *Robotics and Autonomous Systems*, **21**, 249–261.

Kim, J.-O. and Khosla, P. (1991). Real-time obstacle avoidance using harmonic potential functions. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 790–796.

Kim, J.-O. and Khosla, P. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, **8**(3), 338–349.

Ko, N. Y. and Simmons, R. (1998). The lane-curvature method for local obstacle avoidance. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1615–1621.

Koditschek, D. E. (1987). Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1–6.

Kohonen, T. (2000). *Self-Organizing Maps*. Springer, New York, third edition.

Konolige, K. and Pollack, M. E. (1989). Ascribing plans to agents: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 924–930.

Konolige, K., Myers, K., Ruspini, E., and Saffiotti, A. (1997). The saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 215–235.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1394–1404.

Krogh, B. H. and Thorpe, C. E. (1986). Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1664–1669.

Kröse, B. J. A., editor. (1995). Reinforcement learning and robotics. *Special Issue of Robotics and Autonomous Systems*, **15**(4).

Kuperstein, M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks*, **4**, 131–145.

Lagoudakis, M. G. and Maida, A. S. (1999). Robot navigation with a polar neural map, Student Abstract. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, page 965.

Laird, J. E., editor. (1991). Integrated cognitive architectures (AAAI Spring Symposium). *Special Issue of SIGART Bulletin*, **2**(4).

Laird, J. E. (1990). Integrating planning and reaction in SOAR. In *Working Notes: AAAI Spring Symposium on Planning in Uncertain, Unpredictable or Changing Environments*.

Laird, J. E. and Rosenbloom, P. S. (1990). Integrating execution, planning, and learning in SOAR for external environments. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 1022–1029.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, **33**, 1–64.

Lakemeyer, G., chair. (2000). The Second International Workshop on Cognitive Robotics (held in conjunction with ECAI2000).

Lambrinos, D., Möller, R., Pfeifer, R., and Wehner, R. (1998). Modeling visual landmark navigation with autonomous agents. In N. Elsner and R. Wehner, editors, *Proceedings of Neurobiology Conference Göttingen*, page 19, Stuttgart. Georg Thieme Verlag.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., and Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, **30**, 39–64.

Large, E. W., Christensen, H. I., and Bajcsy, R. (1997a). Dynamic robot planning: Cooperation through competition. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2306–2312.

Large, E. W., Christensen, H. I., and Bajcsy, R. (1997b). Scaling the dynamic approach to autonomous path planning: Planning horizon dynamics. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1360–1365.

Large, E. W., Christensen, H. I., and Bajcsy, R. (1999). Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. *International Journal of Robotics Research*, **18**(1), 37–58.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Boston.

Latombe, J.-C. (1999). Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, **18**(11), 1119–1128.

Laugier, C. and Germain, F. (1985). An adaptative collision-free trajectory planner. In *Proceedings of IEEE International Conference on Advanced Robotics*.

Laumond, J.-P. (1998). *Robot Motion Planning and Control*, volume 229. Springer-Verlag, London.

Li, Z. and Canny, J. F. (1993). *Nonholonomic Motion Planning*. Kluwer Academic Publishers, Boston.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, **8**, 293–321.

Liu, C., Ang Jr., M. H., Krishnan, H., and Lim, S. Y. (2000). Virtual obstacle concept for local-minimum-recovery in potential-field based navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 983–988.

Lozano-Pérez, T. (1981). Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, **11**(10), 681–698.

Lozano-Pérez, T. (1987). A simple motion-planning algorithm for general robot manipulation. *IEEE Journal of Robotics and Automation*, **3**(3), 224–238.

Lyons, D. (1992). Planning, reactive. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 1171–1182. John Wiley and Sons, New York.

Lyons, D. and Hendriks, A. (1992a). Planning for reactive robot behavior. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2675–2680.

Lyons, D. and Hendriks, A. (1992b). A practical approach to integrating reaction and deliberation. In *Proceedings of International Conference on Artificial Intelligence Planning Systems*, pages 153–162.

Lyons, D. and Hendriks, A. (1994). Planning by adaptation: Experimental results. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 855–860.

Lyons, D. and Hendriks, A. (1995). Planning as incremental adaptation of a reactive system. *Robotics and Autonomous Systems*, **14**(4), 255–288.

Maeda, M., Maeda, Y., and Murakami, S. (1991). Fuzzy drive control of an autonomous mobile robot. *Fuzzy Sets and Systems*, **39**, 195–204.

Maes, P. (1989). The dynamics of action selection. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 991–997.

Maes, P. (1990). Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 49–70. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Mahadevan, S. and Connell, J. (1991). Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 768–773.

Mahadevan, S. and Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, **55**(2-3), 311–365.

Malcolm, C. and Smithers, T. (1990). Symbol grounding via a hybrid architecture in an autonomous assembly system. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 123–144. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Massios, N., Dorst, L., and Voorbraak, F. (2001). A strategy for robot surveillance using the hierarchical structure of the environment. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 43–50.

Matarić, M. J. (1990). *A Distributed Model for Mobile Robot Environment-Learning and Navigation*. MIT EECS Master's Thesis, MIT AI Lab. Technical Report AITR-1228.

Matarić, M. J. (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, **8**(3), 304–312.

Matarić, M. J. (1997). Behaviour-based control: Examples from navigation, learning, and group behaviour. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 323–336.

Menzner, R., Steinhage, A., and Erlhagen, W. (2000). Generating interactive robot behavior: A mathematical approach. In *From Animals To Animats 6: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Michaud, F., Lachiver, G., and Dinh, C. T. L. (1996). A new control architecture combining reactivity, planning, deliberation and motivation for situated autonomous agent. In *From Animals To Animats 4: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Millán, J. del R. (1996). Rapid, safe, and incremental learning of navigation strategies. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, **26**, 408–420.

Millán, J. del R. (1997). Incremental acquisition of local networks for the control of autonomous robots. In *Proceedings of International Conference on Artificial Neural Networks*, pages 739–744.

Millán, J. del R. and Torras, C. (1992). A reinforcement connectionish approach to robot path finding in non-maze-like environments. *Machine Learning*, **8**, 363–395.

Millán, J. del R., Posenato, D., and Dedieu, E. (2002). Continuous-action Q-learning. *Machine Learning*, **49**, 249–265.

Miller, D. P., Desai, R. S., Gat, E., Ivlev, R., and Loch, J. (1992). Reactive navigation through rough terrain: Experimental results. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 823–828.

Mitchell, T. (1990). Becoming increasingly reactive. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 1051–1058.

Möller, R., Lambrinos, D., Pfeifer, R., Labhart, T., and Wehner, R. (1998). Modeling ant navigation with an autonomous agent. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 5: Proceedings of International Conference on Simulation of Adaptive Behavior*, pages 185–194, Cambridge, MA. MIT Press.

Moravec, H. P. and Cho, D. W. (1989). A Bayesian method for certainty grids. In *Working Notes: AAAI Spring Symposium on Robot Navigation*, pages 57–60.

Mouaddib, A. I. and Zilberstein, S. (1995). Knowledge-based anytime computation. In *Proceedings of International Joint Conference on Artificial Intelligence*, volume 1, pages 775–781.

Murphy, R. R., Blitch, J., and Casper, J. (2002). AAAI/Robocup-2001 urban search and rescue events: Reality and competition. *AI Magazine*, **23**(1), 37–42.

Musliner, D. J., Durfee, E., and Shin, K. (1993). Circa : A cooperative, intelligent, real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**, 1561–1574.

Nagata, S., Sekiguchi, M., and Asakawa, K. (1990). Mobile robot control by a structured hierarchical neural network. *IEEE Control Systems Magazine*, **10**(3), 69–76.

Nehmzow, U. and McGonigle, B. (1994). Achieving rapid adaptations in robots by means of external tuition. In *From Animals To Animats 3: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Nehmzow, U., Smithers, T., and McGonigle, B. (1993). Increasing behavioural repertoire in a mobile robot. In *From Animals To Animats 2: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Nguyen, D. and Widrow, B. (1989). The truck backer-upper: An example of self-learning in neural networks. In *Proceedings of International Joint Conference on Neural Networks*, volume 2, pages 357–363.

Nguyen, D. and Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, **10**, 18–23.

Nilsson, N. J. (1969). A mobile automaton: An application of artificial intelligence techniques. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 509–520.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA.

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, **1**, 139–158.

Noreils, F. R. (1990). Integrating error recovery in a mobile robot control system. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 396–401.

Noreils, F. R. (1991). From planning to execution monitoring control for indoor mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1510–1518.

Noreils, F. R. and Chatila, R. (1989a). Control of mobile robot actions. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 701–712.

Noreils, F. R. and Chatila, R. (1989b). A general structure for mobile robot action control. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 550–556.

Noreils, F. R. and Chatila, R. (1995). Plan execution monitoring and control architecture for mobile robots. *IEEE Transactions on on Robotics and Automation*, **11**(2), 255–266.

Ó'Dúnlaing, C. and Yap, C. K. (1982). A retraction method for planningthe motion of a disc. *Journal of Algorithms*, **6**, 104–111.

Ó'Dúnlaing, C., Sharir, M., and Yap, C. K. (1983). Retraction: A new approach to motion planning. In *Proceedings of 15th ACM Symposium on the Theory of Computing*, pages 207–220.

Omidvar, O. and van der Smagt, P., editors (1997). *Neural Systems for Robotics*. Academic Press.

Payton, D. W. (1986). An architecture for reflexive autonomous vehicle control. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 1838–1845.

Payton, D. W. (1990). Internalized plans: A representation for action resources. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 89–103. MIT Press, Cambridge, MA. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

Payton, D. W., Rosenblatt, J. K., and Keirsey, D. M. (1990). Plan guided reaction. *IEEE Transactions on Systems, Man, and Cybernetics*, **20**(6), 1370–1382.

Pollack, M. E. (1992). The uses of plans. *Artificial Intelligence*, **57**(1), 43–68.

Pomerleau, D. A. (1989). ALVINN: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 305–313. Morgan Kaufmann, San Mateo.

Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, **3**(1), 88–97.

Pomerleau, D. A. (1993). *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Boston, MA.

Pyeatt, L. D. and Howe, A. E. (1999). Integrating POMDP and reinforcement learning for a two layer simulated robot architecture. In *Proceedings of 3rd International Conference on Autonomous Agents*, pages 168–174.

Quinlan, S. and Khatib, O. (1993a). Towards real-time execution of motion tasks. In R. Chatila and G. Hirzinger, editors, *Experimental Robotics 2*. Springer-Verlag.

Quinlan, S. and Khatib, O. (1993b). Elastic bands: Connecting path planning and robot control. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 802–807.

Rao, R. P. H. and Fuentes, O. (1995). Perceptual homing by an autonomous mobile robot using sparse self-organizing sensory-motor maps. In *Proceedings of World Congress on Neural Networks*.

Rao, R. P. H. and Fuentes, O. (1996). Learning navigational behaviors using a predictive sparse distributed memory. In *From Animals To Animats 4: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Rao, R. P. H. and Fuentes, O. (1998). Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Machine Learning*, **31**(1-3), 87–113. Also appeared in *Autonomous Robots*, **5**(3-4), 297–316.

Reignier, P. (1995). Supervised incremental learning of fuzzy rules. *Robotics and Autonomous Systems*, **16**, 57–71.

Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, **8**(5), 501–518.

Ritter, H. and Schulten, K. (1986). Topology conserving mappings for learning motor tasks. In J. S. Denker, editor, *Neural Networks for Computing, AIP Conference Proceedings 151*. Snowbird, Utah.

Rome, E., Hertzberg, J., Kirchner, F., Licht, U., Streich, H., and Christaller, T. (1999). Towards autonomous sewer robots: the MAKRO project. *Urban Water*, **1**, 57–70.

Rosenblatt, J. K. (1995). DAMN: A distributed architecture for mobile navigation. In *Working Notes: AAAI Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents*, pages 167–178.

Rosenblatt, J. K. (1997). DAMN: A distributed architecture for mobile navigation. *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2-3), 339–360.

Rosenblatt, J. K. and Payton, D. (1989). A fine-grained alternative to the subsumption architecture for mobile robot control. In *Proceedings of International Joint Conference on Neural Networks*, pages 317–323.

Rosenblatt, J. K. and Thorpe, C. (1995). Combining multiple goals in a behavior-based architecture. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 136–141.

Rosenschein, S. J. and Kaelbling, L. P. (1986). The synthesis of machines with provable epistemic properties. In J. Halpern, editor, *Proceedings of Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–98. Morgan Kaufmann, San Mateo, CA.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ.

Saffiotti, A. (1993). Some notes on the integration of planning and reactivity in autonomous mobile robots. In A. Lansky, editor, *Working Notes: AAAI Spring Symposium on Foundations of Automatic Planning*, number SS-93-03, pages 122–126, Menlo Park, CA. AAAI Press.

Saffiotti, A., Ruspini, E. H., and Konolige, K. (1993). Integrating reactivity and goal-directedness in a fuzzy controller. In *Proceedings of IEEE International Conference on Fuzzy Systems*, pages 134–139.

Saffiotti, A., Ruspini, E. H., and Konolige, K. (1994). Robust execution of robot plans using fuzzy logic. In A. L. Ralescu, editor, *Fuzzy Logic in Artificial Intelligence, IJCAI'93 Workshop*, pages 24–37. Lecture Notes in AI 847, Springer-Verlag, Berlin, Germany.

Saffiotti, A., Konolige, K., and Ruspini, E. (1995). A multi-valued logic approach to integrating planning and control. *Artificial Intelligence*, **76**(1-2), 481–526.

Saffiotti, A., Ruspini, E. H., and Konolige, K. (1997). Using fuzzy logic for mobile robot control. In D. Dubois, H. Prade, and H. J. Zimmermann, editors, *International Handbook of Fuzzy Sets and Possibility Theory*, volume 5. Kluwer Academic Publishers Group, Norwell, MA.

Sahota, M. (1994). Reactive deliberation: An architecture for real-time intelligent control in dynamic environments. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 1303–1308.

Schöners, G. and Dose, M. (1992). A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. *Robotics and Autonomous Systems*, **10**, 253–267.

Schöners, G., Dose, M., and Engels, C. (1995). Dynamics of behavior: Theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, **16**(2-4), 213–245.

Schoppers, M. J. (1987). Universal plans for reactive robots in unpredictable domains. In *Proceedings of International Joint Conference on Artificial Intelligence*, volume 2, pages 1039–1046.

Schoppers, M. J. (1989). In defense of reaction plans as caches. *AI Magazine*, **10**(4), 51–60.

Schraft, R. D. and Schraft, G. (1999). *Service Robot - Products, Scenarios, Vision.* AK Peters.

Schwartz, J. T. and Sharir, M. (1987). *Planning, Geometry, and Complexity of Robot Motion.* Ablex, Norwood, NJ.

Sekiguchi, M., Nagata, S., and Asakawa, K. (1989). Behavior control for a mobile robot by multi-hierarchical neural network. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1578–1583.

Sharkey, N. E., editor. (1997). The new wave in robot learning. *Special Issue of Robotics and Autonomous Systems*, **22**(3-4).

Sharkey, N. E. (1998). Learning from innate behaviors: A quantitative evaluation of neural network controllers. *Machine Learning*, **31**(1-3), 115–139. Also appeared in *Autonomous Robots*, **5**(3-4), 317–334.

Sharkey, N. E. and Heemskerk, J. N. H. (1997). The neural mind and the robot. In A. J. Browne, editor, *Neural Network Perspective on Cognition and Adaptive Robotics*. Institute of Physics Press, London.

Shastri, S. V., editor. (1999). Field and service robotics. *Special Issue of International Journal of Robotics Research*, **18**(7).

Simmons, R. (1991). Coordinating planning, perception, and action for mobile robots. In *Working Notes: AAAI Spring Symposium on Integrated Cognitive Architectures*. Also appeared in *SIGART Bulletin*, **2**(4), 156–159.

Simmons, R. (1994). Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, **10**(1), 34–43.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3375–3382.

Simmons, R. and Mitchell, T. (1989). A task control architecture for mobile robots. In *Working Notes: AAAI Spring Symposium on Robot Navigation*, pages 85–89.

Simmons, R., Goodwin, R., Haigh, K. Z., Koenig, S., and O'Sullivan, J. (1997a). A layered architecture for office delivery robots. In *Proceedings of 1st International Conference on Autonomous Agents*, pages 245–252.

Simmons, R., Goodwin, R., Haigh, K. Z., Koenig, S., O'Sullivan, J., and Veloso, M. M. (1997b). Xavier: Experience with a layered robot architecture. *ACM SIGART Bulletin*, **8**(1-4), 22–23.

Simmons, R., Fernandez, J., Goodwin, R., Koenig, S., and O'Sullivan, J. (2000). Lessons learned from xavier. *IEEE Robotics and Automation Magazine*, **7**(2), 33–39.

Singh, L., Stephanou, H., and Wen, J. (1996). Real-time robot motion control with circulatory fields. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2737–2742.

Slack, M. (1990). Situationally driven local navigation for mobile robots. JPL Publication No. 90-17, NASA Jet Propulsion Laboratory, Pasadena, CA.

Slack, M. G. (1993). Navigation templates: Mediating qualitative guidance and quantitative control in mobile robots. *IEEE Transactions on Robotics and Automation*, **23**(2), 452–466.

Smart, W. D. and Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *Proceedings of International Conference on Machine Learning*.

Soldo, M. H. (1990). Reactive and preplanned control in a mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1128–1132.

Song, K. and Tai, J. (1992). Fuzzy navigation of a mobile robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 621–627.

Sorouchyari, E. (1990). Self-organizing neural network for trajectory control and task coordination of a mobile robot. *Connection Science*, **2**.

Steinhage, A. and Bergener, T. (1998). Dynamical systems for the behavioral organization of an anthropomorphic mobile robot. In *From Animals To Animats 5: Proceedings of International Conference on Simulation of Adaptive Behavior*.

Stentz, A. and Hebert, M. (1995). A complete navigation system for goal acquisition in unknown environments. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 425–432.

Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human Machine Communication*. Cambridge University Press, Cambridge, MA.

Sugeno, M. and Nishida, M. (1985). Fuzzy control of model car. *Fuzzy Sets and Systems*, **16**, 103–113.

Surmann, H. and Peters, L. (2000). MORIA: A robot with fuzzy controlled behavior. In D. Driankov and A. Saffiotti, editors, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, pages 343–365. Physica-Verlag.

Sutton, R. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press.

Sutton, R. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, **3**(1), 9–44.

Takeuchi, T., Nagai, Y., and Enomoto, N. (1988). Fuzzy control of a mobile robot for obstacle avoidance. *Information Sciences*, **43**, 231–248.

Tani, J. and Fukumura, N. (1994). Learning goal-directed sensory-based navigation of a mobile robot. *Neural Networks*, **7**(3), 553–563.

Thorpe, C. E. (1984). Path relaxation: Path planning for a mobile robot. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 318–321.

Thrun, S., Buecken, A., Burgard, W., Fox, D., Froehlinghaus, T., Henning, D., Hofmann, T., Krell, M., and Schmidt, T. (1998a). Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pages 21–52. AAAI Press.

Thrun, S., Fox, D., and Burgard, W. (1998b). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, **31**(1-3), 29–53. Also appeared in *Autonomous Robots*, **5**(3-4), 253–271.

Touzet, C. (1997). Neural reinforcement learning for behavior synthesis. *Robotics and Autonomous Systems*, **22**(3-4), 251–281.

Tunstel, E., Danny, H., Lippincott, T., and Jamshidi, M. (1997). Adaptive fuzzy-behavior hierarchy for autonomous navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 829–834.

Ulrich, I. and Borenstein, J. (1998). VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1572–1577.

Ulrich, I. and Borenstein, J. (2000). VFH*: Local obstacle avoidance with look-ahead verification. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2505–2511.

Versino, C. and Gambardella, L. M. (1995). Learning the visuomotor coordination of a mobile robot by using the invertible Kohonen map. In *Proceedings of International Workshop on Neural Networks*.

von Altrok, C., Krause, B., and Zimmermann, H. J. (1992). Advanced fuzzy logic control of a model car in extreme situations. *Fuzzy Sets and Systems*, **48**, 41–52.

Wang, L.-X. and Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, **22**(6), 1414–1427.

Wilkins, D. E., Myers, K. L., Lowrance, J. D., and Wesley, L. P. (1995). Planning and reacting in uncertain dynamic environments. *Journal of Experimental and Theoretical Artificial Intelligence*, **7**, 197–227.

Zalama, E., Gaudiano, P., and Coronado, J. L. (1995). A real-time, unsupervised neural network for the low-level control of a mobile robot in a non-stationary environment. *Neural Networks*, **8**(1), 103–123.

Zapata, R., Perrier, M., Lepinay, P., Thompson, P., and Jouvencal, B. (1991). Fast mobile robots in ill-structured environments. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 793–798.

Zelek, J. S. and Levine, M. D. (1996). Spott: A mobile robot control architecture for unknown or partially known environments. In *Working Notes: AAAI Fall Symposium on Planning with Incomplete Information for Robot Problems*, pages 129–140.

Zelinsky, A. (1991). Navigation by learning. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 331–338.

Zelinsky, A. (1992). A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, **8**(6), 707–717.

Zelinsky, A. (1994). Using path transforms to guide the search for findpath in 2D. *International Journal of Robotics Research*, **13**(4), 315–325.

Zelinsky, A. (1997). *Field and Service Robotics*. Springer Verlag.

Zelinsky, A. and Kuniyoshi, Y. (1996). Learning to coordinate behaviors for robot navigation. *Advanced Robotics*, **10**(2), 143–159.

Zelinsky, A. and Yuta, S. (1993a). Reactive path planning for a mobile robot using numeric potential fields. In *Proceedings of International Conference on Intelligent Autonomous Systems*.

Zelinsky, A. and Yuta, S. (1993b). A unified approach to planning, sensing and navigation for mobile robots. In *Proceedings of International Symposium on Experimental Robotics*, pages 444–455.

Zelinsky, A., Jarvis, R. A., Byrne, J. C., and Yuta, S. (1993). Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of IEEE International Conference on Advanced Robotics*.

Zelinsky, A., Kuniyoshi, Y., and Tsukune, H. (1994). Monitoring and co-ordinating behaviours for purposive robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 894–901.

Zelinsky, A., Kuniyoshi, Y., Suehiro, T., and Tsukune, H. (1995). Using an augmentable resource to robustly and purposefully navigate a robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2586–2592.

Zhang, J. and Knoll, A. (2000). Integrating deliberative and reactive strategies via fuzzy modular control. In D. Driankov and A. Saffiotti, editors, *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, pages 367–386. Physica-Verlag.

Zhu, D. and Latombe, J.-C. (1991). New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, **7**(1), 9–20.

Ziemke, T. and Sharkey, N. E., editors. (1999). Artificial neural networks for robot learning. *Special Issue of Autonomous Robots*, **7**(1).

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, **17**(3), 73–83.