Linköping Studies in Science and Technology

Dissertation No. 611

Anchoring symbols to sensory data

by

Silvia Coradeschi

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 1999

ISBN 91-7219-623-8
ISSN 0345-7524

Printed in Sweden by UniTryck
Linköping 1999

Abstract

Intelligent agents embedded in physical environments need the ability to connect, or *anchor*, the symbols used to perform abstract reasoning to the physical entities which these symbols refer to. Anchoring must deal with indexical and objective references, definite and indefinite identifiers, and a temporary impossibility to perceive physical entities. Furthermore it needs to rely on sensor data which is inherently affected by uncertainty, and to deal with ambiguities. In this thesis we outline the concept of anchoring and its functionalities. Moreover we define the general structure for an anchoring module and we present an implementation of the anchoring functionalities in two different domains: an autonomous airborne vehicle for traffic surveillance and a mobile ground vehicle performing navigation tasks.

Acknowledgments

The conclusion of a PhD gives the unique opportunity to thank the people that have contributed not only to the thesis itself, but also to the various tasks and duties accomplished during the PhD period.

First of all I would like to thank Dimiter Driankov, my main supervisor. He has not only greatly contributed to the development of this research with discussions and useful advice, but he has also taken great care to create a pleasant working environment screening me from conflicts and problems. I am greatly indebted also to Alessandro Saffiotti, my co-supervisor. He has been the main inspirer of the work on anchoring and it is thanks to him that my first ideas and examples of anchoring have grown into a more systematic generalization of the anchoring problem.

I would like to heartily thank Erik Sandewall not only for having given me first the possibility to come to Linköping University and to do my PhD, but also for having trusted and supported me in a risky enterprise such as the organization of the Third Robot World Cup Soccer Games and Conferences, RoboCup-99. I am also greatly indebted to Sture Hägglund, Anders Haraldsson, and Janete de Castro for their help and support in RoboCup-99.

The work presented in this thesis has been developed to great extent as part of the WITAS project. I would like to thank all members of the project and in particular the members of the vision group, Klas Nordberg, Thord Andersson, Gunnar Farnebäck, and Johan Wiklund for having initiated me in the mysteries of Computer Vision; the members of the autonomy group Patrik Haslum and Lars Karlsson for having helped me in the integration with the reasoning layer; and John Olsson and Tommy Persson for their patience while giving me technical support.

Lars Karlsson, my partner in work and life, has been near me during all the period of my PhD sharing with me the good and bad moments. I want to heartily thank him for his support and understanding and especially for having shared with me the heavy burden

of the organization of RoboCup-99. I do not know how I would have managed without his help.

A number of friends have been close to me during these years; I can mention just some of them: Partick Lambrix, Tim Heyer, Nahid Shahmehri, Simin Nadjm-Tehrani, Mariam Kamkar, Paul Scerri, Marcus Bjäreland, Joakim Gustafsson, Jonas Kvarnström and Peter Jonsson.

I would like to thank the administrative staff: Lise-Lott Andersson, Janete de Castro, Anne Eskilsson, and Anna Maria Uhlin for their help; Lillemor Wallgren for her support and encouragement; and Ivan Rankin for correcting the English of this thesis and for the good time while we where sharing room in the E building.

Finally I would like to thank my parents for having taught me the importance of study and self development and for supporting and encouraging me through two masters and a PhD.

This research has been fully supported by the Knut and Alice Wallenberg Foundation. A special thanks also goes to the AASS group at Örebro University for their support to the work with the mobile robot.

Preface

The work presented in this thesis has been developed to great extent as part of the WITAS project. Other members of the project have contributed to its realization and part of the thesis has been previously published as joint work with them. In particular parts of the first two chapters have been previously reported in the article [13], co-authored with Alessandro Saffiotti and part of chapter 4 has been reported in articles [12], [14], and [2] co-authored with Thord Andersson, Lars Karlsson, Klas Nordberg, and Alessandro Saffiotti. Moreover we would like to mention that Gunnar Farnebäck and Thord Andersson have greatly contributed to the implementation and design in the WITAS system of the anchoring of static objects and of the fuzzy-set representation of visual data respectively. Finally the anchoring in a mobile robot domain presented in chapter 5 has greatly benefited from the cooperation of Alessandro Saffiotti at Örebro University. Alessandro Saffiotti has also greatly contributed to the design and implementation of the fuzzy matching algorithms presented in the thesis.

Contents

Chapter 1

Introduction

> You are at a friend's house and your host asks you to go to the cellar and fetch the bottle of Barolo wine stored at the top of the green rack. You go down to the cellar, look around in order to identify the green rack, and visually scan the top of the rack to find a bottle-like object with a Barolo label. You see two of them: one is labeled "1968", the other "1993". You decide to pick up the second one, go upstairs, give it to your friend, and warn him that there was another Barolo bottle from 1968.

This vignette illustrates a mechanism that we constantly use in our everyday life: the use of words to refer to objects in the physical world, and to communicate a specific reference to another person. This example presents one peculiar instance of this mechanism, one in which the first person (the friend) "knows" which object he wants but cannot see it, while the second person (you) only has an incomplete description of the object, but can see it. Put crudely, the two persons embody two different types of processes: one who reasons about abstract representations of objects, and one who has access to perceptual data. One of the prerequisites for successful cooperation is that these two processes agree about the objects they talk about, that is, that

there is a correspondence between the abstract representations and the perceptual data that refer to the same physical object. We call *anchoring* the process of establishing this correspondence.

Not unlike our example, an autonomous system needs to incorporate processes that reason about abstract representations and processes that manipulate perceptual data; hence, it needs to perform anchoring.

A typical example of an autonomous system is an autonomous mobile robot who has to provide services in an indoor environment, or an autonomous flying vehicle performing surveillance missions. An autonomous system senses the environment, processes the sensed data and on the basis of the acquired information decides what actions to perform. Such a system needs, therefore, to integrate sensing capabilities with decision-making and acting capabilities. However, a difficulty arises in the fact that different data representations are used in different parts of the system: sensors provide quantitative data, while the decision making part of the system, which is dedicated to determine the actions to perform, needs in general more abstract forms of information. For instance, the navigation planning subsystem of a robot can establish that the best way for the robot to reach a specific office is to follow a certain corridor and to enter a room through the second door on the left. However, the execution of the planning instructions is only feasible if the robot can match the navigational landmarks (corridor, door, etc.) to the data coming from its sensors. If the robot uses, for instance, sonars, then readings indicating a continuous surface need to be matched with the corridor and readings indicating an open space need to be matched with an open door.

1.1 The anchoring problem

The need to integrate low-level and high-level data representations and processes is one of the major challenges of autonomous embedded systems and it has lead to the recent development of the layered archi-

tectures [21, 3]. In a layered architecture a hierarchy of layers is used to separate low-level sensory-motor processes using quantitative data (process layer), processes reacting to events and using discrete and symbolic data (reactive layer), and high level processes involving planning, monitoring, and diagnosis and using symbolic data (deliberative layer). Yet, separating processes of different nature is just half of the work; an essential part is also to integrate them and make it possible for them to exchange information at different levels of abstraction.

In this thesis, we focus on the anchoring problem: one particular aspect of this integration dealing with the connection between the abstract data representations used by the high-level reasoning processes (reactive and deliberative layers) to denote specific physical objects, and the data in the low-level processes that correspond to that object.

Saffiotti has given a first definition of anchoring in [39, p. 267]. To do anchoring means in his definition

> to use your perceptual capabilities to:
>
> 1) *find* the object, by matching what you are perceiving with the description you have been given;
>
> 2) *acquire* new (or more precise) information about the properties of the object [...].

This definition of anchoring outlines the initial intuition behind the anchoring process. However just one aspect of anchoring, the finding of the association the first time, is addressed. An additional issue to be considered is the keeping of the association between the symbolic description and the perceived object over time. Moreover issues such as indexical, definite, and indefinite references and the actual methods to perform the matching between description and perceived object are not addressed in the paper originally defining the anchoring concept. The aim of this dissertation is to thoroughly describe the concept of anchoring and its functionalities, to define a general structure for the concrete implementation of an anchoring module and to instantiate this general structure in anchoring modules used in realistic domains.

The nature of the anchoring problem, in fact, suggests that a general study of it must be solidly grounded in experiments performed on different robots. We have validated our research on two experimental platforms: a wheeled mobile robot, and an autonomous aerial vehicle.

The two experimental platforms share the use of a layered architecture to integrate abstract reasoning with perceptual and control processes. However, these platforms differ significantly in terms of sensory-motoric capabilities and their domains of application. The UAV is intended for the purpose of traffic surveillance over urban and rural areas, and it uses vision as its main sensor. The main aspect of anchoring in this domain is the need to connect the symbols used by a planner and a plan executor to the sensor data about specific cars which are provided by the vision system. The mobile robot uses sonars as its main sensors and moves in an office environment. The main aspect of anchoring here is the need to link descriptions of static objects, such as corridors and doors, derived from an approximate map of the environment, to the sensor data coming from the sonar. In the rest of the chapter we often refer to examples from these domains to illustrate the different aspects of the anchoring process.

A number of subtle issues are hidden in the anchoring problem. First, we need to distinguish between *definite* descriptions, like "the bottle of Barolo on the table", *indefinite* descriptions, like "a bottle of Barolo", and *indexical* descriptions, like "the bottle on my left". These descriptions need different treatments. For instance, if the robot sees two bottles on the table this is not a problem in the case of an indefinite description, but could be a problem in case of a definite description. Second, a perceived object can temporarily disappear from the sensor's view, for instance because the object moves, or because the robot's gaze is moved in a different direction. The robot needs to maintain a virtual image of the object in memory and to reidentify this object if it comes into view again. Third, perceptual information is inherently affected by uncertainty due to noise in the sensors, to poor observation conditions, or to errors in the perceptual interpretation. Anchoring must take this uncertainty into account, for instance to decide to get

a better view of an object if the perceptual data is too poor.

Summarizing, there are a number of issues that need to be investigate while tackling the anchoring problem:

- it should be possible to use both definite (the car previously seen) and indefinite identifiers (a red Mercedes);
- indexical (from the point of view of the observer) and objective references should be supported;
- it should be possible to handle a temporary impossibility to perceive entities currently referred to;
- uncertainty present in the sensory data and ambiguous situations where more than one physical object could be the referent of a symbolic identifier should be handled.

In the rest of this chapter we provide a more detailed description of these issues.

1.2 Issues related to anchoring

1.2.1 Definite and indefinite references

A definite reference denotes a specific, and in general unique, object [38], for example "the car I have seen in that position and at that time". An indefinite reference denotes an object that satisfies a number of properties, for example a "small red Mercedes". An anchoring module must be able to handle both indefinite and definite references.

The anchoring of indefinite references involves the searching of candidate objects, one of the properties, for example color, can be used for this first selection. Then each of the requested properties is checked and it is established whether each property can be considered satisfied.

The main difference between definite reference and indefinite reference is that while in an indefinite reference any object that matches

the properties well enough can be considered the right one, in the case of a definite reference there is just one object that is the correct one. One can distinguish two cases of definite references: the object has never been perceived before, and the object has been perceived before. In the first case the anchoring is performed as in the case of indefinite reference, but among the properties requested there is one that makes the object unique, for example "the small red Mercedes at coordinates (100, 100)".

The second case is the more difficult one. In fact an object previously perceived has a property that makes it unique: it is the object that has been perceived in that specific location and time, but the use of this property to reidentify the object is not straightforward. For example, when a car kept under observation drives under a bridge, the anchoring between the symbol denoting the car and the perceptual image of the car is lost and it has to be reacquired when the car exits from under the bridge. In this case a number of properties have to be combined to reidentify the car. A first property is the appearance of the car, but possible changes in perspective and illumination should be taken into consideration. A second property is the expected position of the car, calculated by combining static information (road network) and dynamic information (last seen position and velocity of the car). Finally, relationships between objects can be used as distinctive properties, for example our car was in front of a green car before disappearing.

1.2.2 Indexical and objective references

Objective references involve knowledge that is independent of the particular perspective of the observer. For example "car-1 is on road 3" is information that is interpretable by anyone having access to a map. Indexical references depend on the perspective of the observer. For instance the objects defined by "the car I am currently tracking", "the red car I have seen before" and "the corridor on my left" can be interpreted only from the point of view of the observer. Indexical

knowledge is of great help in performing actions. Actions are performed by an agent at a specific place and time and it makes sense that the knowledge required for performing the action is relative to the agent's perspective. For instance a robot that has as a goal to pick up an object needs to know mainly the relative position of the gripper and the object and not the objective knowledge about its own position and the absolute position of the object. Furthermore indexical knowledge can highly improve the computational efficiency of the system by selecting just the information that is required for determining a course of action for the agent and that is specific to the task currently being executed.

Both indexical and objective knowledge are essential for an autonomous agent and so is the possibility to combine them both. For instance a camera can track a car along a road just using the position of the car in the image and without knowing on which road the car is (indexical knowledge). If the car disappears under a bridge the knowledge about the road where the car is and the position along the road (objective knowledge) can make it possible for the system to realize the reason for the disappearance and to recover the car at the other end of the bridge.

Deictic relations between objects

A special case of indexical reference is when an object is selected as the focus of attention and the other objects are referred to in relation to it. For example, one car is currently under observation and the cars around it are referred to using deictic references such as "the car behind the observed car", "the car in front the observed car" and so on. We have used deictic references in the UAV domain while recognizing episodes involving several cars.

1.2.3 Recovery of the connection between symbol and entity

In dynamic and complex environments the fact that a symbol has been successfully anchored does not mean that the anchoring will always continue to be active. An example is when the camera has a wide view of an area, the anchoring of several cars is established, then the camera zooms in on one of the cars to check its shape and the other cars disappear from the current image. Another example is when a car that is currently under observation disappears under a bridge.

The anchoring module needs to realize whether an anchor currently referred to is not active anymore. If this is the case the anchoring module should try to reestablish it whenever possible.

1.2.4 Uncertainty in the data provided by the sensory system and in the object description

One of the difficulties of the anchoring problem is that the data provided by the sensory system is inherently affected by uncertainty. This may result in errors and ambiguities when trying to match these data to the high-level description of an object. In order to improve the reliability of the anchoring process, this uncertainty has to be taken into account.

The uncertainty can be due to actual measurements, for example pixel discretization in an image. It can also be due to the conditions in which the measurements are taken, for example the measurements on an object can contain more uncertainty if the object is seen from a great distance. The possibility to distinguish between these two kinds of uncertainty can be valuable during the anchoring process. In fact the first kind of uncertainty is always present, even in the best measurement conditions, but the second kind can be reduced by changing the measurement conditions. For example an anchoring result can be improved by a zooming action.

Moreover the high-level description of the object can be imprecise.

For instance linguistic terms such as "small" and "red" can be used in the description and these terms in general are interpreted as a range of possible values more than as an exact value.

The anchoring process is based on the matching of an object description and a perceptual signature. Due to the uncertainty of the data and the imprecision in the description, it could be appropriate to define a *degree of matching* between a perceptual signature and an object description. Being able to distinguish between objects that match a given description at different degrees can be helpful in discriminating between perceptually similar objects under poor observation conditions and can allow several possible anchors to be considered, ranked by their degree of matching. These degrees of matching can also be used to reason about the quality of an anchor and to perform higher-level decision making; for example, we can decide to engage in some active perception in order to get a better view of a candidate anchor.

1.3 Content of the thesis

Chapter 2 reviews the literature related to anchoring.

Chapter 3 gives a more detailed description of the anchoring problem and its functionalities. It also presents the structure of a general anchoring module that is then instantiated in the next two chapters in implemented anchoring modules for our two domains.

Chapter 4 presents the anchoring problem in the context of a UAV system for traffic surveillance using vision as its main sensor. Anchoring in the domain involves both static objects such as roads and dynamic objects such as cars.

Chapter 5 illustrates the anchoring problem in the context of a mobile robot performing navigation tasks in an office environment

and using sonars as its main sensors. The aspect of anchoring that we have developed in this domain is the anchoring of static objects such as corridors.

Chapter 6: Conclusions and future work.

1.3.1 Contributions

The main contribution of this dissertation is the definition of the anchoring problem and of the functionalities needed to solve it. Anchoring is a crucial problem in any systems combining symbols and perceptual data. We define the general structure for an anchoring module and we present an implementation of the anchoring functionalities in two different domains: an autonomous vehicle for traffic surveillance and a mobile robot performing navigation tasks. In both domains the clear statement of the anchoring problem and the use of the framework offered by the anchoring functionalities have facilitated the integration of the decision making and the sensory parts of the systems. Moreover uncertainty, error in the anchoring process, and disappearance of anchored objects have been taken into consideration.

Chapter 2

Related work on anchoring

Anchoring, as we have defined it, must occur in any system where symbolic reasoning about objects in the world is integrated with physical observations and manipulation of these objects. Yet, little attention has been given to the anchoring problem *per se* in the autonomous robotics and AI literature and in particular no serious systematic studies have been performed. The general principles behind the anchoring problem have, however, been investigated in other fields, including philosophy and linguistics. In what follows, we partition the existing literature that is relevant to this dissertation into three classes:

- architectures for autonomous embedded systems;
- philosophy of language;
- philosophical foundations of AI.

2.1 Architectures for autonomous embedded systems

Anchoring is a key issue in the integration of symbolic and perceptual representations, and systems where this integration arises need to deal

with it. In most systems the anchoring aspect is, however, buried in the actual code and it is impossible to examine it explicitly. In only a few systems are aspects of anchoring explicitly mentioned. One such system is Saphira in which the anchoring term was originally used. A description of the Saphira system can be found in chapter 5 where we present our results in integrating the anchoring principles presented in this dissertation in the Saphira system.

Aspects of anchoring are found in the SFB 527 project in Ulm [43], albeit without a specific awareness of the anchoring problem. The domain used in the project is a mobile robot for natural indoor environments. The tasks considered are: to locate different objects such as bottles, balls, doors, and trash cans, for example in the context of a clean-up procedure and to track and follow a person. The form of anchoring used in the location of objects is mainly anchoring of indefinite references. The robot is given the task to locate an object of a specified kind and color and it succeeds as soon as it finds a matching object. In the tracking task a simplified form of definite reference is used. The person to be followed presents himself in front of the robot and a model of the person is built. The robot can then follow the person as long as the person is in the image and he is not occluded. If the person is lost, the robot asks the person to present himself to the robot again.

Bajcsy and Košecká [4] consider a problem related to anchoring: the symbol to signal and signal to symbol transformation. Like us, they suggest that a hybrid representation of symbols and signals is needed. Their work, however, takes a direction different from our. Symbols in their work are limited to the ones that can be considered abstraction of signals and their main interest is in the selection of such symbols. Our interest is not in the selection of symbols, but rather in the connection of symbols already present at the reasoning level with perceptual data.

Finally the importance of a clear interface between the low-level perception and the high-level interpretation systems is emphasized in the designing of the PROLAB2 road vehicle [27]. This vehicle has

been developed in the framework of the Prometheus project and is equipped with a co-pilot to help the driver in dealing with road obstacles. Using cameras and proprioceptive sensors, the system interprets traffic situations involving road obstacles and it alarms the driver if an obstacle (typically another car) can represent a danger. The connection between the perceptual representation of the obstacles and the obstacle names used at the interpretation level can be considered a form of anchoring. In this project, however, the interest in the objects surrounding the car is limited. The system just needs to recognize if the object represents a danger for the car or not. Therefore several of the key aspects of anchoring are absent, such as the identification of an object as the one the system is looking for and the reidentification of the object when it disappears for some time. An additional difference with respect to our work is the use in the PROLAB2 road vehicle of several sensors and the fusion of their values. We also intend to move towards the use of multiple sensors; the current experiments have been performed using just one main sensor.

2.2 Philosophy of language

The problem of connecting linguistic descriptions of objects to their physical referents has been widely discussed in the philosophical and linguistic literature. In fact, the term *anchor* is borrowed from situation semantics [6]. Situation semantics is a semantics of natural language that tries to find meanings of sentences in the external world and in relations between situations rather than in truth values as in logic-based semantics. In the terminology of situation semantics, an anchor is an assignment of individuals, relations and locations to abstract objects.

The philosophical and linguistic tradition has also debated another notion which is of importance to the anchoring problem, namely the distinction between definite and indefinite reference, and between indexical and objective reference. The distinction between definite and

indefinite references and the semantical problems associated with definite references have been addressed by Russell [38] and Frege [20] among others. This distinction, and its impact on anchoring, does not seem to have attracted much attention in autonomous robotics and the AI literature. One of the few counter-examples is the work by Shoppers and Shu [45], who consider definite and indefinite references in a simple block world scenario. Their aim is to differentiate between goals that could be satisfied by any object of a certain kind as "obtain a state where a red block is over a blue block" and goal that can be satisfied just using specific objects as "put that (pointing) red block over a blue block". If the goal can be satisfied by any object of a certain kind the plan executor can take advantage of this opportunity. For example, if there already is a red block on top of a blue block, there is nothing that needs to be done to satisfy the goal "obtain a state where a red block is over a blue block".

2.2.1 Indexical and objective references

Indexical representations were introduced to the AI community by Agre and Chapman [1]. In the classical PENGI paper they propose the use of indexical-functional features as a way to represent and reason about a complex domain while avoiding a combinatorial explosion. PENGI is an autonomous system playing a video game where a penguin navigates in a maze and pushes blocks while trying to avoid and kill malicious bees. PENGI plays the role of the penguin. The entities in the game are not referred to with objective identifiers such as "block-25 at position (20 50)"; instead they are registered as indexical-functional entities such as "the block I am pushing" or "the bee that is closer than me to the block I intend to push". This representation is indexical because it depends on the agent perspective and it is functional because it depends on the agent's purpose.

The ideas of Agre and Chapman have independently influenced the AI community involved in formal reasoning and planning and the computer vision community.

The main contributions in including indexical references in formal reasoning are the ones of Subramanian and Woodfill introducing indexicality in Situation Calculus, [47] and Lesperance and Levesque [33] developing a theory of knowledge and action, based on modal logic and that handles indexical and objective knowledge. Subramanian and Woodfill introduce in a restricted version of Situation Calculus[1] the possibility to use indexical terms such us "now", "before", and "this-block". They also perform a computational complexity analysis attempting to establish the reason for the higher efficiency associated with the use of indexical references. They reach the conclusion that indexical references are advantageous if their number is much smaller than the total number of situations and objects in the domain. In fact indexical references provide the possibility of quantifying over just the entities involved indexically and not over all entities of the relevant type.

Lesperance and Levesque point out that the knowledge needed for performing an action is often relative to the agent perspective. Therefore they develop a quantified modal logic that can handle differences between indexical and objective knowledge and also allows indexical knowledge in the prerequisites and effects of the actions. The logic formalizes notions of knowledge, time, action, and historical necessity. Indexical and objective knowledge can also be related and an example of this relation is given in a scenario dealing with maps for navigation. Lesperance's PhD thesis, [32] offers also a detailed account of the philosophical literature involving indexical knowledge.

The need for referent identification in planning actions was already stated by Cohen in [11]. Indexical references have been introduced by Shoppers and Shu in the Universal Plan formalism [46], mainly to acquire the ability to manipulate multiple objects. They manage to manipulate a particular one of several identical objects and several identical objects simultaneously. Although this work presents some interesting points, the domain where it has been applied, a simulated

[1] Just one situation precedes any other situation.

block world, lacks most of the challenges related to perception.

In the computer vision community the ideas of Agre and Chapman have influenced the development of the *active* or *animate vision* paradigm. One of the pioneers of this area is Ballard [5] who has pointed out that vision is an active process that implies gaze control and attentional mechanisms. In contrast to traditional computer vision, active vision implies that the tasks direct the visual processing and establish which parts of the image are of interest and which features should be computed. By reducing the complexity and accelerating scene understanding, active vision opens up the possibility of constructing continuously operating real-time vision systems. The active vision paradigm is now largely used in the computer vision community, see for instance [26, 36, 19, 48].

The preceding discussion shows that both in the formal reasoning community and in the computer vision community effort has been put into integrating the notion of indexicality and task dependency. The aspect that is still missing is the connection between the indexical symbols used in the formal reasoning community and the quantitative data produced by the image processing. The selection of tasks at the reasoning level of the system must correspond to gaze control and attentional mechanisms at the processing level. The interest in this thesis is in creating this connection.

2.2.2 Deictic relations between objects

A special case of indexical reference is when an object is selected as the focus of attention and the other objects are referred to in relation to it. For example, one car is currently under observation and the cars around it are referred using deictic references such as "the car behind the observed car", "the car in front of the observed car", etc.

An interesting study in which focus of attention and deictic pointers are used to enhance the performance of the system has been developed in the Esprit project VIEWS by Buxton, Howarth and Gong [25, 10, 26]. In this work, video sequences of the traffic flow at a round-

about are examined and events such as overtaking and cars following each other are recognized. A stationary and pre-calibrated camera is used, and the system presupposes an intermediate-level image processing that detects moving objects and estimates various properties of these objects. Given this information, and the ground-plane representation, the system can recognize simple events such as a left turn and episodes such as an overtaking, which are composed of simple events using a Bayesian belief net approach.

In our work we use deictic references in the recognition of events involving more than one car in the traffic surveillance scenario. The currently anchored car is the car under observation and the other cars are referred to using deictic references such as "car in front" and "car beside". The main difference between our domain and the one used by Buxton, Howarth and Gong is the possibility to move the camera both while moving the vehicle and independently from it. However the basic aspects of the event recognition are common to both domains and we have been able to use a solution similar to their in the event recognition part. In the episode recognition part we have not found the need to use a Bayesian belief approach and we have used a simpler automaton structure.

2.3 Philosophical foundations of AI

The literature on the philosophical foundations of AI presents a wide debate on a problem which is related to anchoring: the *symbol grounding problem*, first stated by Harnard [23] and further discussed by Prem and Davidsson [35, 16] among others. Symbol grounding is the problem of how to give a formal interpretation of formal symbol systems that is based on something other than just another symbol system as in classical logic semantics. According to Harnard, meanings of symbols should be grounded bottom-up in non-symbolic iconic and categorical representations. Icons are sensor projections of the perceived objects, preserving the "shape" of the object, categorical representations are

"filtered" icons preserving just the feature of the shape that can be used reliably to distinguish members and non-members of the category. Icons are mainly used to differentiate between similar objects, for example one car from another, while categories are used to identify an object, for example to identify a car as such. In Harnard's opinion, iconic and categorical representations should be learned using neural nets.

The study of solutions to the symbol grounding problem in AI and autonomous robotic has mainly focused on how to automatically learn basic symbols given perceptual data [24, 9, 44, 34]. This is not the focus of our work. Put crudely, anchoring is not the problem of associating symbols to observable properties, but it is the problem of connecting symbolic representations of specific objects to observable physical objects.

Chapter 3

The concept of anchoring and its functionalities

3.1 The concept of anchoring

Our definition of anchoring is an extension of Saffiotti's definition of anchoring presented in the introduction.

Definition 3.1.1 *Anchoring is the connection between the abstract representations used by the high-level reasoning processes to denote a specific physical object, and the data in the low-level processes that correspond to that object. It consists of two functionalities:*

1. *Given a symbolic description of an object, create an association (anchor) between this description and an appropriate perceived object and acquire new (or more precise) information about the properties of the object;*

2. *Given an already established anchor, keep the association between the symbolic description and the perceived object over time (tracking), and keep an updated record of the values of the properties of the object.*

The first functionality is mainly used to find an object for the first time, while the second functionality is used when a regular update of the properties of the object is needed. The update requires taking into consideration not only new measurements coming form the sensors, but also extrapolation of expected properties of the object (position, velocity, color, etc.) from the last observed properties' values. The reason for extrapolating expected properties values from the observed ones lies in the need to use these properties for decision making and control even when the properties are not currently perceived. For instance, a mobile robot using sonar sensors can perceive a doorway, start crossing it and then not perceive it anymore when it is half-way through. The extrapolation of the relative position it has with respect of the doorway can allow the robot to complete crossing through the door-way successfully. Additionally the expected properties values are used to check that the currently observed object is the same as the object previously observed and to recover the anchor if the object temporarily abandons the sensor range or is occluded.

To clarify the concept of anchoring let us consider the following example. Suppose that the UAV is given the mission to find a "red Mercedes" near a specified crossing. When the car has been found, the UAV should keep it under observation and it should check the occurrence of various episodes involving this car, such as overtaking and giving way to another car.

Finding the car implies:

- to point the camera toward the position of the crossing;
- to anchor the crossing map description to the part of the image corresponding to the crossing;
- and finally to find the objects in the image that are near the crossing and that best anchor the description of a red Mercedes.

This corresponds to the first anchoring functionality. When an anchor is established, the second functionality can be used to keep

the car under observation. The properties of the car are regularly updated and the UAV can follow the car using information about its position and velocity. The reasoning system can now request to check the occurrence of episodes involving the anchored car. This can require the anchoring of cars surrounding the car under observation. These other cars can be referred to using deictic references such as "the car beside the followed car".

3.2 Functionalities of anchoring

As already mentioned, two main functionalities characterize the anchoring problem: finding an anchor and keeping track of it over time. They are expressed in terms of three kinds of internal representations for the same external, physical object.

Descriptor: a symbolic signature of the object in the world that we are talking about; a descriptor is typically represented by a set of symbolic properties which are manipulated by the symbolic layer or by an identifier of a previously perceived object. An example of a descriptor is the representation of a corridor in the map used by the mobile robot in our experiments. The corridor is represented by a name and by properties such as relation with other map objects, width and orientation.

Percept: a sensor-based description of an object in the world which is currently perceived by the sensors; a percept is typically represented by a set of values of continuous variables related to sensor's readings. An example of a percept is the representation of a corridor in terms of position and orientation of the two perceived walls delimiting the corridor.

Anchor: a reification of the association between a descriptor and a percept; an anchor gives a sensor-based description of the specific object in the world that we are talking about. An example of

an anchor is the association of the descriptor of a corridor with a percept of a corridor whose perceptual values are compatible with the properties in the corridor's descriptor.

The first functionality of anchoring is, given a descriptor for an object, to find the percept that best matches the descriptor. Once the best match is found, the perceptual properties of the object (position, color, shape, etc.) need to be stored. They can then be used both by the process requesting the anchoring of the object and to make it possible to re-identify the object even if it is going to be out of sight for a period. The second functionality, tracking, can now take over. Tracking is needed if the process requesting the anchoring needs a regular update of the properties of the object. The object is kept in the sensor range and its properties are regularly checked. In addition, during tracking the properties of the Anchor are predicted on the basis of the last observation of the object properties. The prediction serves two functions: checking that the object is still the same one as in the previous observation, in particular reidentifying the object if it temporary leaves the sensor range, and maintaining the possibility to use the properties for decision making and control even when they are not currently perceived.

The above functionalities can be summarized as follows.

Find an anchor: given a descriptor for an object, associate it with one specific percept and build the corresponding anchor. This functionality can be decomposed into two steps:

1. match the symbolic properties in the descriptor to the percepts which are currently in view, so as to select the appropriate one; and
2. build the anchor by associating the symbolic properties of the object to the actual object's perceptual signature. By perceptual signature we intend here the properties that can be perceived by the sensors and that can be used to identify the object.

Track the anchor: given an established anchor, keep an updated record of its properties. Moreover track the expected properties of the anchor (position, velocity, color, etc.) so that they can be used by the control and decision making processes; to check the identity of an object; and to recover the anchor if the object temporarily abandons the sensor range. This functionality can be further decomposed into three steps:

1. predict the properties of the anchor on the basis of their last observed values;
2. match the anchor to the current percept on the basis of the predicted properties of the anchor and the observed properties of the percept; and
3. verify that newly acquired properties of an object are still consistent with the initial properties in the descriptor.

One can notice that the first functionality of anchoring corresponds in part to the original definition of anchoring given by Saffiotti [39]. The main new aspect is the clear distinction among the three entities involved in the anchoring process: descriptor, percept, and anchor. The second functionality, the tracking of an anchored object, is completely new.

3.3 The computations of anchoring

In this section we describe the architecture for a generic anchoring module that we then instantiate in chapters 4 and 5 in an anchoring module for each of our applications.

The relations of the anchoring module with symbolic and subsymbolic processes are shown in Fig. 3.1. The anchoring module is intermediate between symbolic and subsymbolic processes. Typical examples of symbolic processes are planning systems and plan executors. The subsymbolic processes that interact with the anchoring module

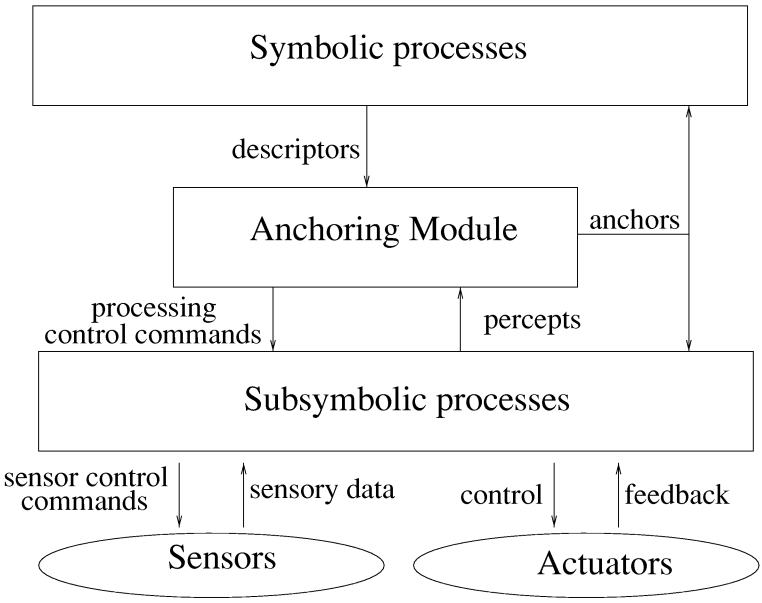


Figure 3.1: The relations of the anchoring module with symbolic and subsymbolic processes.

are the ones elaborating sensory data, for instance vision processing. The anchoring module does not interact with the sensors directly, it presupposes an intermediate subsymbolic process that recognizes objects and estimates properties of objects. The result of the elaboration of the subsymbolic process is a percept, that is, a sensory-based description of an object.

Symbolic processes request the anchoring module for the anchoring of descriptors. The descriptor can be a description of an object in terms of symbolic properties (indefinite anchoring request) or an identifier of a previously seen object (definite anchoring request). The anchoring module receives percepts from the subsymbolic processes and it then tries to find among the percepts the one that best matches the descriptor.

In sensors such as vision it is possible to focus the processing of the sensory data to extract some specific features and to direct the sensor toward a specific area. In these case the anchoring module can also send specific requests to the sub-symbolic processes indicating where to direct the sensor and what features values calculate. For other kinds of sensors, for instance sonars, the processing of the data is in general performed on all data available independently of specific requests. In our experiments we have considered both kind of sensors. In the WITAS application the vision processing module receives requests from the anchoring module stating which characteristics the object of interest has and towards which area or object the camera should be pointed. In the mobile robot application, the processing of the sonar data is done on all sonar readings available and all features are extracted. The anchoring module then selects the percepts that are of interest in the current anchoring process.

The results of the anchoring process can be used by the symbolic processes, and also by the subsymbolic processes dealing with the control of the robot. The WITAS application is an example of the first case, while the mobile robot application is an example of the second.

3.3.1 The internal representation of descriptor, anchor and percept

In this section we consider in more detail how the three main entities involved in the anchoring process are represented.

Descriptor

A descriptor can have two forms: a set of symbolic properties that describe the desired object, or an identifier of a previously perceived object. The properties can be characterized by giving the actual requested values, for instance the interval in which the width of a corridor should be included or by a linguistic term such as "red" and "small". If values or intervals of values are provided, the anchoring module uses them directly in the anchoring process. However, if linguistic terms are provided, the anchoring module needs to translate them to computable values. This translation table is stored in the anchoring module and has the following form:

(name description)

Name is the linguistic term identifying the property at the symbolic level and *description* is a representation of the property that can be used by the anchoring module to check if a perceived object has the property. For instance, in the WITAS application the property of being red is represented at the symbolic level by the linguistic term "red". The description of the property in the anchoring module is in the form of three fuzzy sets representing each measurable aspect of the color red: hue, saturation and value.

If the descriptor is the identifier of a previously anchored object, the properties of the object are stored in the anchor of the object maintained in the anchoring module. Some of the properties can become outdated. The anchoring module predicts the current value of the properties and uses them to reidentify the object.

Anchor

The anchor is a reification of the association between a descriptor and a percept. It has access to both the information originally present in the descriptor and to the sensor-based description extracted from the percept. An anchor is represented as follows:

(identifier description)

Identifier is a unique identifier for the object. In the case of static objects the identifier is in general the name of the object used at the symbolic level and extracted from the descriptor. For instance, the identifier could be the name of the road in a map of the environment. In the case of mobile objects the identifier is created by the anchoring module when the object is perceived for the first time and it is then used at the symbolic level to refer to the object at a later time. For instance, in the WITAS application the system can first request to find a car corresponding to a specific description. When a car is found, the anchoring module creates an identifier and sends it to the decision making part of the system. At a later point the identifier can be used as a descriptor to reidentify the car.

The *description* of the object is a collection of properties that can be used to identify and reidentify the object and/or that can be useful at the symbolic level. In particular it contains the perceptual data extracted from the percept at the latest time the anchoring was established. The anchor is also maintained when the anchoring connection is actually lost, for instance when the object is not in the sensor range anymore or another object is occluding it. The information in the anchor is maintained in order to make it possible to still refer to the object and to reidentify it at a later time point. There is, however, the need to predict the development over time of the values of the properties of the object in order to be able to actually use the information. For instance, the anchor of an observed car maintains the information about the last seen position and speed of the car. However, the expected current position of the car is needed to reidentify

it. The anchoring module uses two simple algorithms, one for static and one for dynamic objects, to update the values of the properties of the objects not currently anchored. In the case of static objects, the property updated is the relative position between the object and the autonomous system performing the anchoring. The update is done on the basis of the estimated movement of the system. In the case of dynamic objects, the property updated is the current position of the object with respect to the autonomous system and the map of the environment[1]. The position is calculated assuming that the object moves at constant speed. Constraints in the environment are used to help in the calculation. For instance, in the WITAS application the new position of a car is calculated considering that it is moving along the road network. There is also the possibility to consider several possible positions of an object, for example in case a car reaches a crossing. In case the object needs to be reidentified, the anchoring module handles alternative positions of an object considering one position after the other until the object is reidentified.

Percept

A percept is a sensory-based description of an object that is currently perceived by the sensors. It is the result of processing of sensory data and it requires the recognition of the actual object on the basis of the sensory data and the calculation of its properties. For instance the creation of a percept of a corridor in the mobile robot application requires the recognition of an object "corridor" on the basis of the sonar data and the estimation of properties of the corridor such as width and orientation.

[1]Other properties could also change over time, for instance the color changes due to changes in illumination. However these cases are not considered in our work.

3.3.2 Anchoring requests

The anchoring module receives requests for anchoring descriptors from the symbolic processes in the system and tries to establish the anchor of the descriptors with the percepts provided by the sensors. If the sensor supports active perception, that is, it is possible to focus the processing of the sensory data to extract some specific features and to direct the sensor toward a specific area, requests are sent by the anchoring module to the subsymbolic processes to guide the processing of the data. If the sensor does not support active perception, the anchoring module simply waits for the next perceptual data provided by subsymbolic processes. When new percepts are provided, the anchoring module tries to find among them the one best matching the descriptor.

The requests are of different kinds depending on whether they involve a definite or an indefinite reference and whether they require a continuous update of the properties of the referenced objects. In the next sections we consider the different kinds of requests.

Indefinite reference

An indefinite reference denotes an object, in general not unique, that satisfies a number of properties, for example a small red Mercedes. In a request for an indefinite reference the descriptor is a set of symbolic properties that describe the desired object. The anchoring module needs to find a percept that matches the description of the object. In the case where active perception is supported, the anchoring module sends a request to the subsymbolic processes specifying the properties characterizing the desired object in quantitative terms and waits for the result of the request. The properties stated in the anchoring requests from the symbolic processes can be described in terms of quantitative values or in linguistic terms. In this second case the anchoring module uses the description of the properties to elaborate a quantitative description of the desired object that can be used in

the request to the subsymbolic processes. For instance, if the request for finding a 'red' car is sent to the anchoring module in the WITAS application, the anchoring module uses the fuzzy set representation of 'red' to elaborate a quantitative description that is suitable for the vision processing module. If the properties are represented in terms of quantitative values, they can in general be used directly in the request to the subsymbolic processes.

When percepts relevant to the requests are provided, the anchoring module matches them with the descriptor and establishes a degree of matching for each of the percepts. If more than one percept is a good candidate for the anchoring, the anchoring module selects in general the best matching one. In some cases all possible candidates are sent back as an answer ordered with respect to the degree of matching.

Definite references

A definite reference denotes a specific, and in general, unique object. One can distinguish two cases of definite references: the object has never been perceived before, and the object has been perceived before. In the first case the anchoring is performed in the same way as an indefinite reference. However, it is assumed that among the properties requested there is one that makes the object unique, for example "the small red Mercedes at coordinates (100, 100)".

In the second case previously recorded data about the object are used for the reidentification. In the case of active perception the anchoring module retrieves information about the object and sends it to the subsymbolic processes for the purpose of reidentification. When percepts are provided, the anchoring module matches the description of the object with the percepts and selects the percept that best matches the description. Two difficulties can be encountered: no percept matches the description to a high degree and several percepts match the description to a high degree[2]. The strategies that can be

[2]The fact that several percepts match the description to a high degree can be a problem because an object denoted by a definite reference should in general be

used to solve these cases are very much application-dependent and they are influenced by considerations about the need of certainty in the anchor, the possibility and the need to inform the symbolic processes of the difficulty encountered, and so on.

Indexical references

An object is said to be referred to indexically when the reference depends on the perspective of the observer or on the relation between the object and other objects (deictic references). Examples of the first case are: "the car I am currently tracking", "the red car I have seen before", "the corridor on my left". Examples of deictic references are: "the car behind the observed car", "the car in front the observed car".

Indexical references are treated in our work as a special case of definite and indefinite references in which the properties characterizing the object are indexical. The handling of indexical properties requires the storing of these properties in the anchor. For instance in the WITAS application there is the possibility to store in the anchor of a car object additional information about the cars surrounding it. In this way references to a car beside, behind and so on can be easily handled.

Continuous update of references

The anchoring requests considered in the previous sections involve the first functionality of anchoring, that is to create an anchor between the symbolic description of the object and the perceived object. The second functionality of anchoring is to keep an updated record of the properties of an already established anchor (tracking). The anchoring module updates the property values maintained in the anchor every time new percepts are provided. If the sensor supports active perception, the anchoring module can request the subsymbolic processes to direct the sensor towards the object of interest. For instance, in the

unique.

WITAS application the vision module can move the camera in such a way that the car of interest is maintained in the center of the image.

The anchoring module needs to take into consideration the cases in which the tracked object disappears. The disappearance of the object is detected when no percept matches the description of the object stored in the anchor. The anchoring module extrapolates the expected values of the properties on the basis of the previously recorded values and it compares them with the values in the new percepts, if any are reported. If the difference between the values overcomes a specified threshold or no percept is reported, the anchoring module examines the possibility that the object has disappeared or, in some cases, the possibility that the previous anchor was incorrect. The anchoring module checks if, according to the knowledge about the environment, there is an object that can be occluding the tracked object. If this is the case and if active perception is supported, the anchoring module can direct the sensor towards the next visible position of the object. For instance, in one of the examples presented in chapter 4, the car disappears under a bridge and the anchoring module directs the camera to the area where the car is going to reappear. If the object disappears and then reappears again, the anchoring module needs to anchor the object again. The anchoring is performed on the basis of the information stored about the object before the disappearance in the same way as for a definite reference.

A case in which the anchoring module recognizes that the descriptor has been wrongly anchored is presented in chapter 5. In this case the anchoring module first anchors a corridor descriptor to a percept. The perceptual data of this percept had, however, been wrongly classified as belonging to a corridor by the subsymbolic processes. When a new percept is provided, this time one of the actual corridor, the anchoring module anchors correctly this new percept with the descriptor.

3.4 Dealing with uncertainty

One of the difficulties of anchoring descriptors to percepts is the inherent uncertainty in the measurements of perceptual data and imprecision in the definition of the properties used to characterize the descriptor.

The uncertainty of measurements is dependent on the degree of accuracy that the methods used to estimate the measurements have and on the conditions in which the measurements are taken. It is therefore in most cases unavoidable. The imprecision in the definition of the properties used to characterize the descriptor is due to the fact that the properties describe an object using linguistic terms and linguistic terms do not refer in general to a unique numerical value.

In our system we represent both measurements and properties using fuzzy sets. The reason why we have selected fuzzy sets as representation is the possibility that they support representing information at the level of precision which is available. In fact they can represent equally well precise and complete information (crisp sets) and imprecise, vague or unreliable information [40].

The anchoring process is based on the matching of a descriptor with a percept. The matching depends on the matching of the properties characterizing the descriptor and the measurements of the respective properties in the percept. Due to the uncertainty in the measurements of the perceptual data and the imprecision in the definition of the properties used to characterize the descriptor, the matching is in general better characterized by a number indicating the degree by which the measurements and the properties match. The *degree of matching* of a descriptor with a percept is computed combining the degrees of matching between the fuzzy sets representing the measurements and the fuzzy sets representing the desired properties.

Properties are represented in our system using *fuzzy set.* Therefore we briefly introduce in the next section some basic concepts of fuzzy set theory extracted from [17]. In the following section we then introduce the use of fuzzy sets for the representation of the actual properties and

describe how the degree of matching is calculated.

3.4.1 Fuzzy sets

In classical set theory a set C can be defined by a *characteristic function* $\mu_C : U \to \{0, 1\}$, that for every element of a universe of discourse U establishes whether the element belongs to the set or not. The main difference between a fuzzy set and a 'normal' set is that the elements of the universe belong to a fuzzy set just by a degree. The characteristic function of classical set theory is generalized to a *membership function* that assigns to each element in the universe a value in the interval $[0, 1]$ establishing the degree to which the element belongs to the set.

Definition 3.4.1 (Membership function) *The membership function μ_F of a fuzzy set F is a function*

$$\mu_F : U \to [0, 1]$$

Let us introduce two concepts that we will use in the rest of the chapter. The *support* of the fuzzy set F is the set of all the elements of F with non-zero degree of membership.

Definition 3.4.2 (Support) *The support of a fuzzy set F defined on the universe U is defined by:*

$$S(F) = \{u \in U | \mu_F(u) > 0\}$$

The *core* of a fuzzy set F is the set containing all the elements of F with 1 as degree of membership.

Definition 3.4.3 (Core) *The core of a fuzzy set F defined on the universe U is defined by:*

$$S(F) = \{u \in U | \mu_F(u) = 1\}$$

Although the membership function of a fuzzy set can have a number of different shapes, in this work we consider just *trapezoidal fuzzy sets*, that is, fuzzy sets with trapezoidal membership function. Examples of fuzzy sets with trapezoidal membership functions are given in Fig 3.2.

Definition 3.4.4 (Trapezoidal fuzzy set membership function) *the membership function $\mu_F : U \to [0,1]$ of a trapezoidal fuzzy set F is a function with four parameters defined as:*

$$\mu_F(u;\alpha,\beta,\gamma,\delta) = \begin{cases} 0 & u < \alpha \\ (u-\alpha)/(\beta-\alpha) & \alpha \leq u < \beta \\ 1 & \beta \leq u < \alpha \\ (\gamma-u)/(\delta-\gamma) & \gamma < u \leq \delta \\ 0 & u > \delta \end{cases}$$

The reason for choosing trapezoidal fuzzy sets is the computational efficiency of set-theoretic operations performed on them. The use of trapezoidal fuzzy sets does not represent a limitation in our domain as they allow an easy representation of all the properties we need.

Finally, let us define a number of operations over the fuzzy sets.

Definition 3.4.5 (Equality) *Two fuzzy sets are equal ($A = B$) if and only if their membership functions are identical, i.e.,*

$$\forall x \in U : \mu_A(x) = \mu_B(x)$$

Definition 3.4.6 (Subset) *A is a subset of B ($A \subseteq B$) if and only if*

$$\forall x \in U : \mu_A(x) \leq \mu_B(x)$$

The interpretation of *union* and *intersection* in fuzzy set is not as simple as in classical set theory. This is due to the fact that graded characteristic functions are used.

Definition 3.4.7 (Union) *A triangular co-norm or S-norm* $\circ$ *denotes a class of binary functions that can represent the union operation. It satisfies the following criteria:*

$a \circ b = b \circ a$;

$(a \circ b) \circ c = a \circ (b \circ c)$;

$a \leq c$ *and* $b \leq d$ *implies* $a \circ b \leq c \circ d$;

$a \circ 0 = a$.

The membership function of the union of two fuzzy sets in our system is defined as:

$$\mu_{A \cup B}(x) = \mu_A(x) \circ \mu_B(x)$$

A common choice of S-norm in the literature is the *max* operation. *Triangular norms* (T-norm and S-norm) can be considered as the most general intersection operator.

Definition 3.4.8 (Intersection) *A triangular T-norm* $\star$ *denotes a class of binary functions that can represent the intersection operation. It satisfies the following criteria:*

$a \star b = b \star a$;

$(a \star b) \star c = a \star (b \star c)$;

$a \leq c$ *and* $b \leq d$ *implies* $a \star b \leq c \star d$;

$a \star 1 = a$.

The membership function of the intersection of two fuzzy sets is defined as:

$$\mu_{A \cap B}(x) = \mu_A(x) \star \mu_B(x)$$

A common choice of T-norm in the literature is the *min* operation.

3.4.2 Representation of properties using fuzzy sets

Anchoring requests can identify the properties which describe an object in terms of linguistic terms like 'red' and 'small'. These linguistic terms do not refer to a unique numerical value and they are inherently

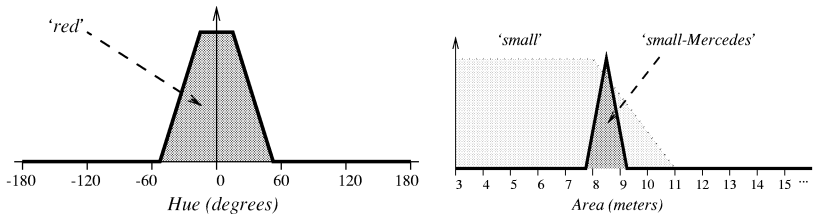


Figure 3.2: Fuzzy sets for representing the hue characterization of the symbols 'red' and the area of a 'small-Mercedes'.

imprecise. Fuzzy sets are commonly considered to be an adequate representation of linguistic terms [53, 30], so in our system we have chosen to map each symbol of this kind to a fuzzy set over the relevant universe. For example, we associate the term 'red' with three fuzzy sets: one for the hue characterizing the tint of color, one for the saturation characterizing the purity of the color, and one for the value characterizing its intensity. Fig 3.2 (left) shows the fuzzy set for the hue. This fuzzy set is interpreted as follows: for each possible value of hue h, the value of $red(h)$ measures, on a $[0, 1]$ scale, how much h can be regarded as 'red'.

A linguistic term is represented by a set of fuzzy sets over the space of the possible values of its properties. For instance the linguistic term 'small-Mercedes' is represented by a set of fuzzy sets over the space of the possible values of length, width and area of the car. Fig 4.3 (right) shows the fuzzy set for the area.

Fuzzy sets can be interpreted intuitively as follows. The values that are mapped to 1 are the values that without doubt belong to the property and that can be considered a typical example for the property, for instance, the areas of cars that can be considered without doubt proper to a small Mercedes. The values that are mapped to 0 are the ones that do not belong to the property. Finally the values that are mapped to an intermediate value are the ones that belong to the property just to a certain degree. The value to which they map can

establish an order among the objects with respect to how they satisfy the property. For instance several areas can be considered more or less typical for a small Mercedes.

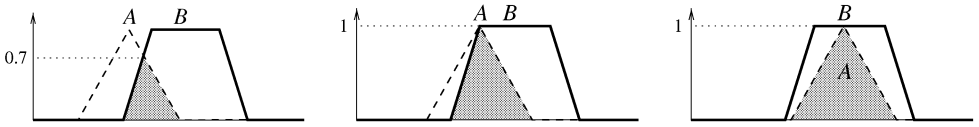
The data coming from the subsymbolic processes are in the form of fuzzy sets, each representing a property of the perceived objects. How these fuzzy sets are calculated is dependent on the kind of sensor used. In Section 4.5.1 we show how fuzzy sets are built by the vision module used in the WITAS application. The fuzzy sets are used by the anchoring module to select the objects that constitute the best answer to the anchoring request and are subsequently stored. When an object needs to be reidentified the same fuzzy sets are utilized again.

3.4.3 Fuzzy matching of one feature

The computation of the degree of matching is done using fuzzy set operations. This choice is justified in our case since fuzzy sets can be given a semantic characterization in terms of degrees of similarity [37]. There is however a subtle difference between the notion of similarity and our intended notion of matching. Consider two fuzzy sets A and B over a common domain X which respectively represent the observed data and the desired description. The degree of matching of A to B, denoted by $\text{match}(A, B)$, is the degree by which the observed value A can be one of those that satisfy B. Thus, matching implies some sort of overlap between A and B, but it does not require that A and B have a similar shape. Moreover, matching is not required to be commutative.

Once the sensors have reported to the anchoring module the selected objects and their properties in terms of fuzzy sets, the anchoring module can compute the *degree of matching* between each of the fuzzy sets given by the vision module and the desired description. The desired descriptions can be the descriptions of properties such as "red" or descriptions previously recorded for the object.

In our work, we have tried two different definitions for a degree of matching. In the first one, we measure how much A and B intersect



$\text{match}_1 = 0.7$ $\text{match}_2 = 0.4$ | $\text{match}_1 = 1.0$ $\text{match}_2 = 0.8$ | $\text{match}_1 = 1.0$ $\text{match}_2 = 1.0$

Figure 3.3: Three examples of partial matching between a set A and a reference set B.

by measuring the height of $A \cap B$. This gives us the following degree:

$$\text{match}_1(A, B) \;=\; \sup_{x \in X} \min\{A(x), B(x)\} \tag{3.1}$$

In the second definition, we measure of how much A is a (fuzzy) subset of B by comparing the area of $A \cap B$ and the area of B:

$$\text{match}_2(A, B) \;=\; \frac{\int_{x \in X} \min\{A(x), B(x)\}\, dx}{\int_{x \in X} B(x)\, dx} \tag{3.2}$$

Different definitions can be obtained using T-norm operators other than min.

The degrees of matching defined by equations (3.1) and (3.2) behave in two essentially different ways. Match_1 only depends on the existence of some common elements in A and B, while match_2 compares how much of A is inside B with how much of A is outside B. The difference is graphically illustrated in Fig. 3.3. When the cores of A and B have no common points (left), both definitions provide a degree of matching less than 1. As soon as the cores intersect (mid and right), match_1 always sanctions total matching, while match_2 gives us only a partial degree whenever A is not entirely contained into B. In a sense, definition (3.1) tells us how much the observed value *may* satisfy B; while definition (3.2) tells us how much the observed value *must* satisfy it.

Measure (3.2) is more discriminating, and it has provided superior empirical results in our experiments. We have thus adopted this

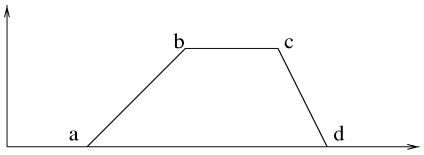


Figure 3.4: Trapezoidal fuzzy set.

measure in our system. For computational reasons, however, we approximate (3.2) by the ratio between the area of the inner trapezoidal envelope of $A \cap B$ and the area of B. These areas can be computed very easily when A and B are trapezoidal fuzzy sets.

Calculation of match$_2$

Let us consider how match$_2$ is calculated in practice.

We represent trapezoidal fuzzy sets by a list of four points *(a, b, c, d)* where a and b are the extremes of the support of the fuzzy set and b and c are the extremes of the core Fig. 3.4.

The trapezoidal envelope of $A \cap B$ and the degree by which A is a subset of B is calculated using the following algorithm where $A =$ *(a, b, c, d)* and $B =$ *(a', b', c', d')* [3]:

If the supports of the sets are disjoint ($d \leq a'$)

then the **degree of intersection** *is equal to 0 and the* **trapezoidal envelope** *of* $A \cap B$ *is empty*

else if the cores are disjoint ($c < b'$)

then **degree of intersection (deg)** $= \frac{d-a'}{(d-a')+(b'-c)}$

[3]The algorithm presupposes that A is always to the left of B, that is, $b \leq b'$. If this is not the case A and B can be switched.

trapezoidal envelope *of* $A \cap B$ = *(max{a, a'}, (d − (deg ∗ (d − c))), min{d, d'})*

else (the cores intersect)

the **degree of intersection** *is 1*

the **trapezoidal envelope** *of* $A \cap B$ = *(max{a, a'}, max{b, b'}, min{c, c'}, min{d, d'}*

The match_2 measure is then calculated as following:

$$match_2(A, B) = \frac{degree\,of\,intersection * (area\,trapezoidal\,envelope\,of\,A \cap B)}{(area\,B)}$$

3.4.4 Fuzzy matching of several features

Once we have computed a degree of matching for each individual feature, we need to combine all these degrees in order to obtain an overall degree of matching between descriptor and percept. The simplest way to combine our degrees is by using a *conjunctive* type of combination, where we require that each one of the properties of the percept matches the corresponding part in the descriptor. Conjunctive combination is typically done in fuzzy set theory by T-norm operators [50, 30]. Most used instances of these operators are min, product, and the Łukasiewicz T-norm $max(x + y - 1, 0)$. In our experiments, we have noticed that the latter operator provides the best results. (See [8] for an overview of alternative operators.)

The overall degree of matching is used by the anchoring module to select the best anchor among the candidate objects provided by the subsymbolic processes. For each candidate, the anchoring module first computes its degree of matching to the intended description; then it ranks these candidates by their degree of matching. Having a list of candidates is convenient if the currently best one later turns out not to be the one we wanted. Also, it is useful to know how much the best matching candidate is better than the other ones: if the two top

candidates have similar degrees of matching, we may decide to engage in further exploratory actions in order to disambiguate the situation before committing to one of them.

3.5 Summary

The first part of this chapter gives a detailed definition of anchoring problem and of the functionalities characterizing it: finding an anchor and tracking it over time.

In the second part of the chapter we introduce a general structure for and anchoring module. The description addresses the issues presented in Chapter 1 as essential while tackling the anchoring problem. Definite, indefinite and indexical references are explicitly treated in the section presenting the anchoring requests. Moreover cases where it is temporary impossible to perceive entities currently referred to is considered in the section presenting the continuous update of references. Finally the treatment of uncertainty using fuzzy logic in presented in the last section.

Chapter 4

Anchoring in UAV performing traffic surveillance tasks

4.1 Introduction

In the previous chapter we have defined the notion of anchoring and we have outlined a number of functionalities needed in an anchoring module. In this chapter we present an actual implementation of an anchoring module, the Scene Information Manager (SIM).

The SIM has been created in the context of WITAS, a long-term project with the aim of developing autonomous system technology and in particular an Unmanned Airborne Vehicle (UAV) for traffic surveillance.

The primary aim of the project is developing a decision-making system able to both react to sudden changes in the current status of the world and use deliberation, in particular planning, for achieving complex goals. The architecture of the decision-making system is a classical three-layered architecture: deliberative, reactive, and process layer. The SIM is part of the reactive layer and handles the anchoring

of symbolic identifiers used in the reactive and deliberative layers to sensory data produced in the process layer, and in particular in its vision processing component.

The rest of the chapter is organized as follows: Section 4 presents the WITAS project; Section 4.1.2 introduces the general architecture of the system; Section 4.2 provides a general overview of the SIM functionalities; Section 4.3 explains how static objects such as roads and crossings are anchored; Section 4.4 describes how anchoring requests are sent to the SIM, while Section 4.5 describes the treatment of sensor data using fuzzy matching. The handling of ambiguities and disappearance of objects under observation is considered in section 4.6. Section 4.7 presents on-going work in event and episode recognition. Finally Section 4.8 outlines a number of examples illustrating the issues described in the chapter.

4.1.1 The WITAS project

The WITAS project is a research cooperation project between four groups at Linköping University: computer vision, autonomous decision making, computer architecture, and software tools and simulation.

The project, initiated in January 1997, is devoted to research on information technology for autonomous systems, and more precisely for unmanned airborne vehicles used for traffic surveillance. The vehicle is to be equipped with a camera system for observation of traffic scenes, and the images will be processed online and used for the vehicle's decision-making [1].

The first three years of the project have resulted in methods and system architectures to be used in UAVs. The next four years will be dedicated to the development of an actual prototype of the vehicle.

However, the focus of the project is not in the development of the flying vehicle itself; the intention is to actually acquire a vehicle already able to take-off, fly and land autonomously. The aim is on the

[1] Additional sensors such as infrared camera and radar may be included in the system.

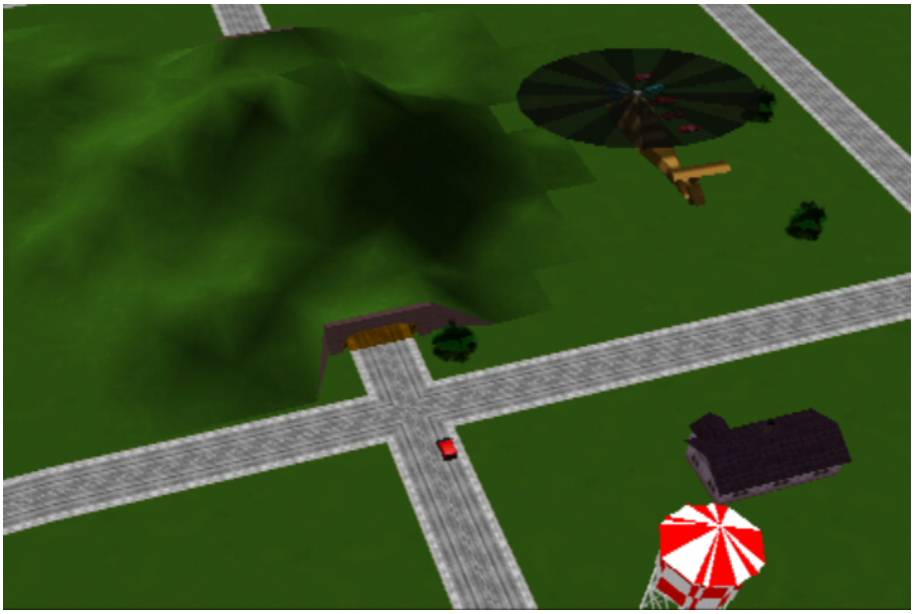


Figure 4.1: A view of the simulated environment.

development of a decision-making system capable of making complex decisions and to pursue long term goals.

Because of the safety-critical nature of the work most of the testing has being made using simulated UAVs in simulated environments, even though real image data has been used to test the vision module. In a second phase of the project, however, the testing will be made using real UAVs.

For additional information about the WITAS project see [52].

4.1.2 General system architecture

The general architecture of the system is a standard three-layered architecture consisting of:

- A deliberative layer which at run-time generates probabilistic high-level predictions of the behaviors of agents in the environment, and uses these predictions to generate conditional plans.
- A reactive layer which performs situation-driven task execution, including tasks relating to the execution of plans generated by the deliberative layer. The reactive layer has access to a library of task and behavior descriptions, which can be executed by the reactive executor. For this purpose, a new reactive language has been developed, which has some similarities to Firby's RAPS [18], but is also related to real-time and systems modeling languages such as ESTEREL [7] and StateCharts [22]. The Scene Information Manager is also part of the reactive layer and deals with anchoring of symbols to sensory data.
- A process layer which performs vision processing, sensor data acquisition and flight control.

The system is implemented and operates in a simulated environment. Figure 4.1 shows a view of the simulated environment.

4.2 The Scene Information Manager

The Scene Information Manager (SIM), is part of the reactive layer and it implements the anchoring process. Anchoring requests are sent by the reactive executor to the SIM. Some of the requests come directly from the reactive executor and some are actually requests from the deliberative layer that are transmitted through the reactive executor. A request concerns the anchoring of a specific unique object (definite request) or of a generic object satisfying a number of properties (indefinite request). We examine the two kinds of requests more closely in Section 4.4.

The requests are processed by the SIM and as a result certain image analysis procedures are activated in the vision module in order

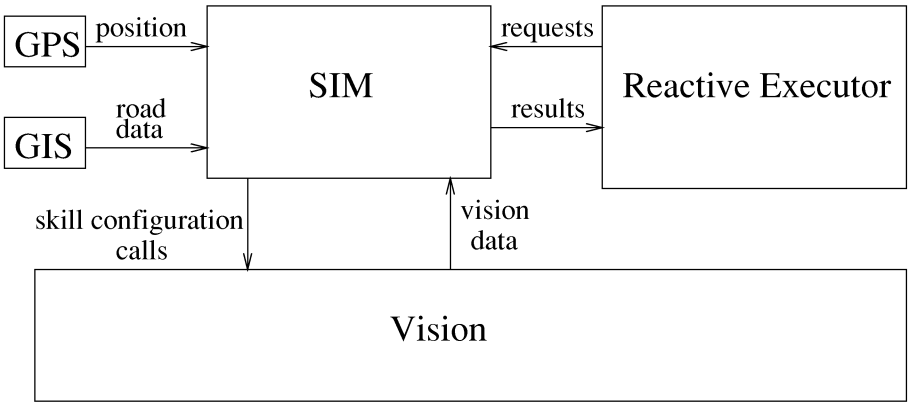


Figure 4.2: Overview of the Scene Information Manager and its interactions with decision-making and vision.

to acquire the necessary data. The SIM also supplies the appropriate parameters used by the procedures in the vision module.

In the case of indefinite requests, the parameters are calculated, mapping the symbolic properties provided in the anchoring request to visual representations. For instance, the color "red" is translated to more primitive color data such as HSV[2] values, and car models such as "Mercedes" are translated to geometrical descriptions. The visual representations are in the form of fuzzy sets to take into account the imprecision of linguistic terms like "red" and "small". In Section 4.4.1 the fuzzy sets used are described.

In case of definite requests, the parameters are retrieved, extracting information from the internal model of the current scene under observation. This model is maintained in the SIM and includes names and properties of objects in the scene, such as cars and roads, and relations between objects, for instance that a car is in a position on a specific road or one car is behind another car. It also includes infor-

[2]Hue, saturation and value of a colour.

mation about the previously observed properties of the object and the previous information, such as the last observed position of a car, can be extrapolated to current expected values.

When the SIM receives data from the vision module it processes it and stores the information in the internal model. The uncertainty of the data is handled using fuzzy matching (Section 4.5). The SIM recognizes ambiguities, that is, cases in which more than one object could be the anchor of a symbol (Section 4.5.4) and it handles simple vision failures, in particular temporary occlusion and errors in car reidentification (Section 4.6). Moreover the SIM deals with event recognition involving one or more cars, Section 4.7.

The anchoring process considered in this section concerns anchoring of dynamic objects. The successful anchoring of dynamic objects depends on the identification of where the objects are in the environment, in our case the road network. This information is required by the reactive and deliberative layers, for instance in order to follow the object. It is also used by the SIM for reidentification of an object that has been out of view. The identification of object position is done by a process that constantly anchors roads and crossings descriptions stored in a GIS with the parts of the image corresponding to them. In the next section the anchoring of static objects is described.

4.3 Anchoring of static objects

The information stored in the SIM is mainly the result of anchoring request; objects that are not in the focus of some request will simply not be registered. The only anchoring process going on continuously without requiring an explicit anchoring request is the identification of roads and crossings, based on information about the positions and geometries of roads extracted from the GIS. This information is used to find the parts of the image corresponding to specific roads, which enables determining of the position of cars relative to the roads. This is the most important example of integration of static and dynamic

knowledge in the system. The anchoring of static objects is conceptually part of the SIM functionality, but for efficiency reasons it is currently implemented in the vision module. The vision module provides, for every object reported to the SIM, on which road it is and at what distance with respect to the beginning of the road.

The anchoring of static objects can be implemented in several ways, and two quite different approaches have been tested. One is based on tracking landmarks with known world coordinates and well-defined shapes which are easily identified in an aerial image. From the world coordinates and the corresponding image coordinates of all landmarks, a global transformation from image to world coordinates (and vice versa) can be estimated assuming that the ground patch which is covered by the image is sufficiently flat. A second approach uses the shape information about each static object, e.g., roads, and measurements of the position and orientation of the UAV's camera to generate a "virtual" image. This image "looks" the same as the proper image produced by the camera, but instead of intensity values each pixel contains symbolic information, e.g., road names, position along the road, etc. The virtual image works as a look-up table which is indexed by image coordinates.

Since it relies on tracking several landmarks, the first approach is more robust but less effective and versatile than the second approach which, on the other hand, is less robust since it depends on having enough accurate measurements of the camera's position and orientation. This can, however, be managed by methods which establishes a registration between camera image and virtual image.

4.4 Handling of anchoring requests

In this section we explain how the two main kinds of anchoring requests, indefinite and definite, are treated by the SIM.

An object can be identified by a number of properties or by a description of the object stored when it was last seen. We first introduce

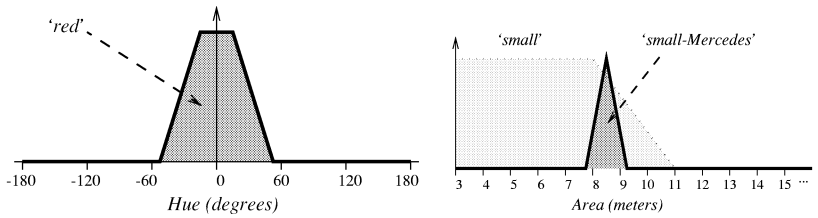


Figure 4.3: Fuzzy sets for representing the Hue characterization of the symbols 'red' and the area of a 'small-Mercedes'.

the representation of properties such as "red" and "small-Mercedes" and the representation of data about previously seen objects. Then we examine in details the handling of the actual requests.

4.4.1 Representation of properties

Properties are represented in our system using *fuzzy set*. Some basic concepts of fuzzy set theory are presented in chapter 3.

Anchoring requests identify the properties describing an object using linguistic terms like 'red' and 'small'. These linguistic terms do not refer to a unique numerical value and they are inherently imprecise.

The linguistic term 'small-Mercedes' is represented by a set of fuzzy sets over the space of the possible values of length, width and area of the car. Fig 4.3 (right) shows the fuzzy set for the area. The reason why we consider the term 'small-Mercedes' and not just 'small' is because what should be regarded as 'small' depends on the type of car we are talking about. In practice, we use a database that associates each car type to its typical length, width, and area, represented by fuzzy sets. Cars of unknown types are associated with fuzzy sets representing a generic car, like the 'small' (car) shown by the dotted lines in the picture. Intuitively the values that are mapped to 1 are the values that without doubt are proper to the property 'small car'. The values that are mapped to 0 are the ones that do not belong to the property. Finally the values that are mapped to an intermediate value are the

ones that belong to the property 'small car' just to a certain degree.

The data coming from the vision module are also in the form of fuzzy sets each representing a property of the seen object, (for details see Section 4.5.1). The fuzzy sets are used by the SIM to select the objects that constitute the best answer to the anchoring request and are then stored in the SIM. When an object needs to be reidentified, the same fuzzy sets are utilized again.

4.4.2 Indefinite references

An indefinite reference denotes an object, in general not unique, that satisfies a number of properties, for example a small red Mercedes. The request for anchoring of an indefinite reference specifies the properties of the object and, in general, it also specifies the area on the ground where the object should be searched for. A request for anchoring of an indefinite request has the following form:

(indefinite object-kind $Property_1 \ldots Property_n$ *area)*

where *object-kind* is the kind of object that is requested[3], $Property_1$ ... $Property_n$ are the requested properties of the object, for instance color and shape, and the *area* is expressed in terms of absolute ground coordinates.

The SIM processes the request by performing the following steps:

- The SIM translates the properties' symbolic names into a form that is suitable for the vision module using the fuzzy sets representation of the properties described in the previous section. What is actually sent to the vision module are intervals whose extremes correspond to the supports of the fuzzy sets of the required properties. These intervals are used by the vision module to make a first selection among the visible objects and keep only those that have properties which fit the intervals. A second more

[3]Currently just cars are supported.

accurate selection of the objects is later performed by the SIM. Details of both these selection procedures are to be found in Section 4.5.

- The translated properties and the coordinates of the area which describes where to search for the object are then transmitted to the vision module.

4.4.3 Definite references

A definite reference denotes a specific, and in general, unique object. If the object has not been perceived before, but it has a special property that makes it unique, for example "the small red Mercedes at coordinates (100, 100)", the anchoring is performed in the same way as an indefinite reference and the request has the same form.

In the case when the object has been perceived before, previously recorded data about the object are used for the reidentification. A request for an anchoring of a previously seen object has the following form:

(definite identifier)

Identifier is the identifier that was sent to the reactive executor when the object was previously anchored.

The SIM retrieves information about the object and sends it to the vision module for the purpose of reidentification. The color and shape information about the object is in the form of a trapezoidal fuzzy set, that is, the form in which the data were calculated by the vision module and then stored in the SIM. The color and shape values sent to the vision module are intervals whose extremes are the extremes of the support of the fuzzy set.

The area which defines where to look for the object is derived by extrapolating the position where the car was last seen, assuming constant speed. If the car has meanwhile reached a crossing, several areas can be candidates for search. The SIM transmits the information

about the object and one of the areas to the vision module. If the car cannot be found in that area, the SIM transmits the information about the other possible areas one at a time. In the current implementation the search of the car stops as soon as the car is found in one of the areas. An alternative possibility would be to examine all areas before making a decision, but we consider this alternative impractical due to the dynamics of our domain.

4.4.4 Continuous update of references

The anchoring requests considered in the previous sections involve the first functionality of anchoring, that is to create an anchor between the symbolic description of the object and the perceived object. The second functionality of anchoring is to keep an updated record of the properties of an already established anchor. The request to regularly get the updated values of the properties of an anchored object has the following form:

(track identifier).

Identifier is the identifier that was sent to the reactive executor when the object was anchored.

The SIM retrieves information about the object and sends it to the vision module. The vision module centers the camera on the car and tries to keep the camera centered on the object compensating for the movement of the car and of the UAV. The compensation is done using a Kalman filter. The properties of the car are regularly updated and in particular its position. The position of the object is regularly sent to the reactive executor and it is used, for instance, when the UAV is following a car.

4.4.5 Invocation of algorithms in the vision module

The invocation of the algorithms in the vision module is done by means of *skill configuration calls*. A skill configuration call has the following structure:

(Name $Parameter_1 \ldots Parameter_n$*).*

Name is the name of the collection of algorithms to be invoked by the vision module in order to acquire the needed information and $Parameter_1 \ldots Parameter_n$ are the parameters needed by the algorithms. The skill configurations currently implemented are the following:

(Look-for list-of-properties area): it moves the camera so that the image corresponds to the requested area. It then looks for objects satisfying the required properties[4] and reports the found objects to the SIM;

(Track list-of-properties position-coord): when invoked, it moves the camera so that the **position-coord** are at the center of the image. It then finds all cars satisfying the list of properties, reports the information about all found cars to the SIM ordered with respect to the distance to the **position-coord** and starts tracking the nearest car with respect to the **position-coord**[5]. The car is then continuously tracked until the skill configuration is deactivated. Tracking a car implies keeping the car in the center of the image and regularly reporting its properties. The tracking is done using a Kalman filter performing a short term prediction on the position of the car. Information on all other visible cars satisfying the properties is also reported[6].

(Track-Check-Surroundings list-of-properties position-coord): this skill configuration performs the same tasks as the **Track** skill configuration with the difference that information about all the

[4]Details on how the objects are selected are provided in Section 4.5.1.

[5]The SIM checks if the car that the vision module is starting to track is the correct one. If this is not the case, the SIM can send another **Track** request to the vision module, for instance indicating the coordinates of one of the other cars reported.

[6]This gives the SIM the ability to realize that the vision module has started tracking a different car.

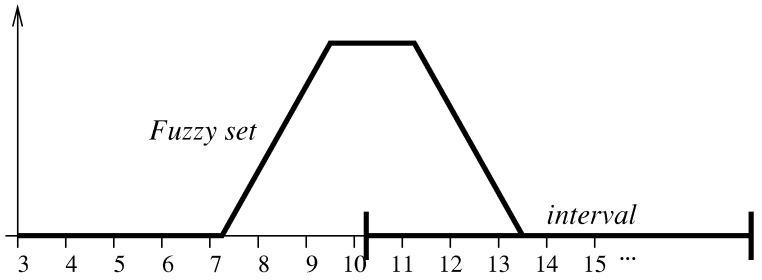


Figure 4.4: A fuzzy set compatible with an interval.

cars surrounding the tracked car is also sent to the SIM. This skill configuration is used, for instance, during event recognition involving several cars.

4.5 Treatment of sensor data

A request to the vision module to look for an object contains, as described in the previous section, a number of intervals indicating the acceptable values for each of the attributes of the object, for instance hue, length, area, and so on. For each of the visible objects the vision module calculates trapezoidal fuzzy sets for all the properties and selects the objects where the fuzzy sets of those attribute are all compatible with the required intervals. A fuzzy set is *compatible* with an interval if the support of the fuzzy set intersects the interval. Figure 4.4 shows an example of a fuzzy set compatible with an interval.

Information about the objects selected by the vision module is reported to the SIM. This information includes the fuzzy sets of color, shape, velocity, and position of the object. The SIM then makes another selection among the objects and orders them with respect to the degree by which they match the desired object. This second selection is done at the SIM level for two main reasons: it involves the comparison with previously stored information not available at the vision

level; and it involves reasoning about the relative importance of different features that is more proper of the SIM level. Each of the fuzzy sets coming from the vision module is matched with the fuzzy set describing the required property and a degree of matching is calculated. Then the degrees of matching are combined to form the total degree of matching for each object. The objects are then ordered with respect to the total degree of matching.

In the rest of the section we first explain how the fuzzy sets are created in the vision module (Section 4.5.1). Then we consider how the degrees of matching for a single feature (Section 4.5.2) and for multiple features (Section 4.5.3) are calculated. Finally answers to the reactive executor and possibility of disappearance of the object are considered in Sections 4.5.4 and 4.6 respectively.

4.5.1 Fuzzy-set representation of visual data

Data obtained from the vision system, e.g., color, shape, position and velocity, are affected by uncertainty and imprecision in several ways. In this work, we propose to explicitly represent the imprecision inherent in these data, and to take it into account when performing signature matching. In order to justify our representation, we need to analyze the way in which we extract the required parameters from the image.

Consider the measurement of the shape parameters (length, width and area) of an observed car. Roughly, the measurement starts with a segmented and labeled binary image containing our candidate cars. This binary image is created by combining and thresholding the feature images produced by the different feature channels available, e.g., orientation, color, IR and velocity (currently, we only use the color channels). For each object in the labeled image, we then compute the moment of inertia matrix. From this 2×2 matrix, we calculate the two eigenvalues which correspond to the largest and smallest moment of inertia, respectively, and convert them into the length and width of the object under the assumption that our objects (cars) are rectangular as seen from above. We also measure the area by counting the pixels that

belong to the same object. The length, width and area measures are then converted to metric measures through multiplication by a scale factor describing the meter per pixel ratio. This ratio is computed from the field-of-view angle and from the position and angles of the camera.

There are a number of factors that influence the correctness of the values measured by the above procedure. First, in the segmentation phase, the discretization of the image limits the precision of the measure.

Second, continuing the segmentation phase, we apply some binary operations (e.g., "fill" and "close") on the binary image in order to connect and "bind" segmented pixels into objects. This operation slightly alters the shape, thus limiting the precision. The above two factors together produce a *segmentation error*, denoted by ϵ_s. Third, the measurement model may be inaccurate, thus introducing an error, the *model error*, denoted by ϵ_m; for example the above assumption that cars are rectangular is almost never completely true. Note that the impact of the ϵ_s and ϵ_m errors on the quality of the measurements depends on the size of the car in the image, which in turn depends on its distance from the camera and on the focal length of the camera. A fourth factor that affects the measurement is the perspective distortion due to the angle α between the normal of the car plane and the optical axis: if the car plane is not perpendicular to the optical axis, the projection of the 3D-car on the image plane will be shorter. We denote this *perspective error* by ϵ_α. Finally, all the geometric parameters needed to compute the length may themselves be affected by errors and imprecision. For example, the distance from the camera depends on the relative position of the UAV and the car; and the α angle depends on the slope of the road; both these values may be difficult to evaluate. We summarize the impact of these factors on our measurement in a *geometric error* term, denoted by ϵ_g.[7]

[7]There are more sources of errors in this process. For example, when α increases, the car projection may seem longer due to the fact that the sides of the car will

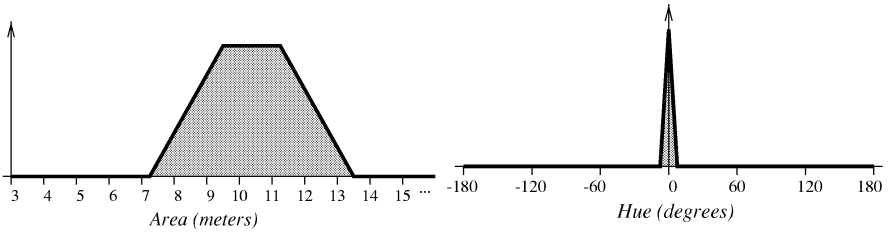


Figure 4.5: Fuzzy sets for the measured area (left) and hue (right).

The above discussion reveals that there is a great amount of uncertainty that affects the measured value, for example, the length of an object; and that this uncertainty is very difficult to precisely quantify — in other words, we do not have a *model* of the uncertainty that affects our measures. Similar observations can be made for other features measured by the vision system: for example, the measurement of the color of an object is influenced by the spectral characteristics of the light that illuminates that object. Given this nature of the uncertainty in the data coming from the vision system, we have chosen to represent these data using fuzzy sets [53]. Fuzzy sets offer a convenient way to represent inexact data whose uncertainty cannot be characterized by a precise, stochastic model — but for which we have some *heuristic* knowledge. For example, Fig 4.5 (left) shows the fuzzy set that represents a given area measurement. For each value x, the value of this fuzzy set at x is a number in the $[0, 1]$ interval that can be read as "the degree to which x can be the actual area of the object given our measurement." (See [54] for this possibilistic reading of a fuzzy set.)

In our work, we use trapezoidal fuzzy sets, both for computational reasons and for ease of construction. The possibilistic reading gives

become visible. Also, the measured value can be totally invalid if there has been an error in the segmentation and/or labeling phases. For instance, if the car has been merged with its shadow, or with another car in front of it. Accounting for these possibilities is part of our current work.

us some simple guidelines on how to build a trapezoidal fuzzy set to represent an inexact measurement. The core of the fuzzy set identifies those values x that can be fully regarded as the actual length value given our measurement. In the example in Fig 4.5 (left), these values are spread over an interval rather than concentrated in a point because of the segmentation effect: our measurement cannot tell us more than what is allowed by the pixel size. The base of the fuzzy set (its *support*) identifies those values x that can possibly be regarded as the actual length value: given the errors that may affect our measurement, the actual length may be anywhere in the support interval — but under no circumstances can it be outside this interval. Put differently, the support constitutes a sort of worst case estimate: however big the error is, the actual value must lie somewhere in this interval. While the core constitutes a best case estimate, even if the measurements are taken in the best of conditions, we cannot be more precise than this.

Let us now discuss in detail how we have built the fuzzy set in Fig 4.5. The vision system has calculated the length to 29.9 pixels, which corresponds to $l = 4.23$ meters. The segmentation error ϵ_s is estimated to a constant ± 1 pixel, which with a scale factor of $s = 0.14$ meters/pixel gives us $\epsilon_s = 0.14$ meters. This segmentation error is inherent to our measurement process, no matter how good our models and computations are, and it thus defines the core of the trapezoid in the picture, given by the interval $[l - \epsilon_s, l + \epsilon_s] = [4.09, 4.37]$.

Our estimates for the other errors are all collected in the support of the trapezoid. The model error ϵ_m is estimated in a coarse but simple way by comparing the measured area a_m with the computed area $a_c = wl$ (where w is the calculated width). The difference between these areas defines ϵ_m such that a_m will lie in the interval $[(w - \epsilon_m)(l - \epsilon_m), (w + \epsilon_m)(l + \epsilon_m)]$. If, for example, a_m is greater than a_c, ϵ_m becomes:

$$a_m - (w + \epsilon_m)(l + \epsilon_m) = 0 \implies \epsilon_m = -\frac{(w+l)}{2} + \sqrt{\frac{(w+l)^2}{4} + (a_m - a_c)}$$

which in our case gives us $\epsilon_m = 0.04$ m. As a simplification we have assumed that ϵ_m is the same for both the width and for the length.

As for the perspective error ϵ_α, in our case we have $\alpha = 40.3°$. If we assume that we measure the projected length as $l \cos\alpha$, then the worst case error due to α becomes $\epsilon_\alpha = l_{\mathrm{MAX}}\,(1 - \cos\alpha)$, where l_{MAX} is the estimation of the maximum object length. If we set $l_{\mathrm{MAX}} = 6\,\mathrm{m}$ we get $\epsilon_\alpha = 1.42\,\mathrm{m}$. Since the support of our fuzzy set must include all the values which are possible in a worst case error situation, we include all the above errors in it.[8] This gives us the interval $[l - \epsilon_s - \epsilon_m,\, l + \epsilon_s + \epsilon_m + \epsilon_\alpha] = [4.05, 5.83]$ for the base of our trapezoid. Note that ϵ_α only affect the upper bound of the interval, i.e., the car may seem smaller in the image when α increases. The correct length in our example was 4.42 m.

The construction of the fuzzy sets for the other features follows similar guidelines. For example, Fig 4.5 (right) shows the fuzzy set that represents the observed Hue value. At the current stage of development, however, we have mainly focused on the shape parameters. Although the definitions of these fuzzy sets are mostly heuristic, they have resulted in good experimental performance.

4.5.2 Fuzzy matching of one feature in the SIM

Once the vision module has reported the selected objects and their properties in terms of fuzzy sets, the SIM can compute the *degree of matching* between each of the fuzzy sets and the desired description. The desired descriptions can be the descriptions of properties such as "red" or a description of the object recorded when the object was last observed. The computation of the degree of matching is performed using fuzzy set operations.

We have tried in this domain the two different definitions for degree of matching presented in chapter 3, $match_1$ and $match_2$. The $match_2$ definition has given the best performance in this domain. In the $match_2$ definition we measure to what extent A is a (fuzzy) subset

[8] In our current experiments in the simulated environment, we have $\epsilon_g = 0$ since the position of the UAV, the camera parameters, and the road geometry are all perfectly known.

of B by comparing the area of $A \cap B$ and the area of B:

$$\text{match}_2(A, B) \;=\; \frac{\int_{x \in X} \min\{A(x), B(x)\}\, dx}{\int_{x \in X} B(x)\, dx} \tag{4.1}$$

4.5.3 Fuzzy matching of several features in the SIM

Once we have computed a degree of matching for each individual feature, we need to combine all these degrees together in order to obtain an overall degree of matching between a description and a given object perceived by the vision system. The simplest way to combine our degrees is by using a *conjunctive* type of combination. In our experiments, we have noticed that the Łukasiewicz T-norm $max(x+y-1, 0)$ operator provides the best results. The overall degree of matching is used by the SIM to select the best anchor among the candidate objects provided by the vision module. For each candidate, the SIM first computes its degree of matching to the intended description; then it ranks these candidates by their degree of matching, and returns the full ordered list to the reactive executor. The reactive executor is then responsible for deciding which candidate to choose. Knowing how much the best matching candidate is better than the other can be useful in deciding to engage in further exploratory actions in order to disambiguate the situation before committing to one candidate. For instance, we may ask the vision system to zoom in on each candidate in turn in the hope to get more precise data.

4.5.4 Answers to the reactive executor

The result of the fuzzy matching described in the previous section is an assignment of a degree of matching to the observed objects. The SIM discards the objects with 0 degree of matching and stores information about the remaining objects creating an identifier for each of the objects.

The answer to the reactive executor's anchoring request is a list of object identifiers ordered with respect to their degree of matching.

The degree of matching is also provided to the reactive executor.

$$(ObjFound(ObjId_1\, DegreeMatching_1)\ldots(ObjId_n\, DegreeMatching_n))$$

If no object is found, the string:

(no-obj-found)

is sent to the reactive executor. If the reactive executor had requested the anchoring of an indefinite reference, it simply selects one of the objects, in general the one with highest degree of matching. If the reactive executor had requested the anchoring of definite reference, the fact that there are multiple candidates could be a problem since the object referred by a definite request ideally should be unique. In this case the reactive executor can decide to zoom in with the camera or to move toward the object to gain a better view and collect more informative data. Deliberative parts of the system can also be involved in the decision about which object to consider as the correct one.

In the case of a request for tracking an object, the SIM sends to the reactive executor the following list each time the vision module sends new data to the SIM:

(obj-properties obj-id xpos ypos xvel yvel)

where *obj-id* is the identifier of the objects, *(xpos ypos)* are the absolute coordinates of the object, and *(xvel yvel)* is the velocity vector of the object. This information is used by the reactive executor, for instance to follow the object.

4.6 Handling of object disappearance

When an object is tracked, the vision module constantly tries to keep the object in the center of the image and it regularly sends updated information about the object to the SIM.

The tracking algorithm can lose the object mainly for two reasons: the UAV makes a sudden maneuver, the camera cannot compensate in time, and the object gets leaves the image; or the object is occluded by another object.

The fact that the algorithm has lost track of the object can be detected by the SIM in two ways: the vision module communicates that the tracked object is not visible anymore or the SIM detects that the vision module has started tracking the wrong object. The second case can occur for instance when the car disappears under a bridge while a similar car is passing on the bridge. The SIM can detect that the wrong object is being tracked because it compares the data coming from the vision module with the possible positions of the car calculated on the basis of the previously received data and information about the road network retrieved from the GIS. There can be several possible positions in the case where the car has meanwhile reached a crossing and they take into consideration possible change of velocity of the car and possible changes of lane. If the data coming from the vision module are incompatible with any of the possible positions, the car is considered as lost.

When the SIM determines that a car has been lost, it checks whether the car has been occluded by an object which is described in the GIS, for instance a bridge or a tunnel. If this is the case and the occlusion is expected to be short, as for instance in the case of a bridge, the SIM transmits to the vision module the description of the object and the position where the object is going to reappear. The same mechanism as for a definite reference is used[9]. If the occlusion is expected to last for a long time and/or the UAV needs to change heading or velocity in order to be able to see the object again after the occlusion, then information that the car has been lost due to an

[9] Alternatively one could use a more sophisticated model of the road network to determine the possible positions of the car that includes possibly occluding objects. In this case the SIM would know in advance that the car is going to disappear, but this would increase the complexity of the calculations without adding, we believe, a real advantage.

occlusion is transmitted to the reactive executor. This is the case, for instance, when the car disappear under a tunnel. The reactive executor can in its turn invoke the planner to take decisions about what course of action to take. The reactive executor is also informed that the car has been lost if the car does not reappear at the predicted position after an expected time. The expected time is proportional to the time that should be needed for the car to reach the end of the occluding object at the velocity that the car had before disappearing.

If the object has disappeared but there is no information in the GIS about an occluding object, the SIM continues for a pre-specified time to transmit to the vision module the information about the object and the expected position of the object given the last observed position and velocity. In this way, the SIM can recover the car in the case where it has briefly disappeared due to trees along the road or if the vision module has temporary lost the car due to a stiff turn of the UAV. If the car does not reappear after the pre-specified time, the reactive executor is informed of the disappearance of the car.

4.7 Events, activities and episodes

The recognition of traffic-related episodes is one of the tasks of the UAV in the WITAS project. Examples of episodes of interest are overtaking, giving way at a crossing, and turning. The recognition of episodes is currently implemented in the reactive executor, while the recognition of the single events and activities is placed in the SIM module. The recognition of events and activities is not anchoring in the proper sense, but since this recognition strongly relates to properties of the anchored objects, it is most naturally implemented in the SIM.

The recognition of events and activities is just a marginal part of this dissertation. The work presented in this section describes a preliminary implementation of the event and activity recognition capability in the SIM illustrated by a few examples of events, activities and episodes.

Following the work of Howarth [25] in recognition of episodes in traffic scenes, we provide the following definitions of event, activity, and episode.

Consider an object, for instance a car, and the properties of this object which we are interested in. In general these properties may be constant or variable over time. They may also assume values from different domains. If we restrict ourselves to only those object properties that take values "true" or "false", an *event* can be defined as the becoming true of a property of a specified object. For example the stopping of the car is an event, the property becoming true being the fact that the car is still.

An *activity* can be defined as the being true for a certain time of a property of a specified object. For example being near to a crossing is an activity.

An *episode* is a composition of events and activities. For instance the overtaking of one car by another is a episode that can be considered as the sequential composition of: the event "car changes lane", the activity "there is a car beside it", and the event "car goes back to the previous lane" [10].

The episode is represented in the reactive executor by an automaton. The automaton is activated when the system needs to recognize an episode. The request for recognition of the first event/activity in the automaton is sent to the SIM. The recognition of an event/activity triggers the request for the recognition of other events/activities until the episode is completely recognized or it is realized that the episode cannot be recognized.

The description of the episode includes the cases which make it possible to realize that the episode cannot be recognized. This is also expressed in terms of event/activity recognition. For instance, in the recognition of the overtaking episode given that the event *change-lane* has been recognized, recognition of the activity *car-beside* and

[10]This definition of overtaking covers the most typical cases that a human would classify as overtaking, but probably not all of them.

the event *change-lane* are requested. If the activity is recognized, the episode recognition continues. If the event is recognized, the car has returned to the original lane and the recognition of the episode is aborted.

4.7.1 Request for recognition of events and activities

The reactive executor sends requests for recognition of events and activities to the SIM. The SIM responds to the requests by activating algorithms in the vision module to process appropriate aspects of the image and activating internal functions for processing the data incoming from the vision module and recognizing events and activities.

A request for the recognition of an event has the following form:

(event event-name object)

where *event-name* is the identifier of the event of interest and *object* is the object that has to be checked with respect to the event. The object can be referred to by its symbolic identifier or using indexical references such as "the tracked car" or "the car in front of the tracked car".

When an event is produced, for example the car has changed lane, the reactive executor is notified.

An activity is recognized when the property associated with the activity is true at the moment of the request of the activity recognition. If the property is not true initially, it is regularly checked and the activity is recognized as soon as the property becomes true.

4.7.2 Examples of episodes, events, and activities

The episodes that can be currently recognized are: one car overtaking another car, and one car giving way to another car. The structures of the two episodes are shown in Fig. 4.6 and Fig. 4.7 respectively.

The overtaking episode consists of the following steps:

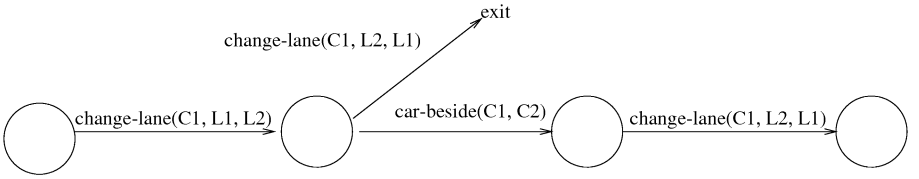


Figure 4.6: The overtaking episode.

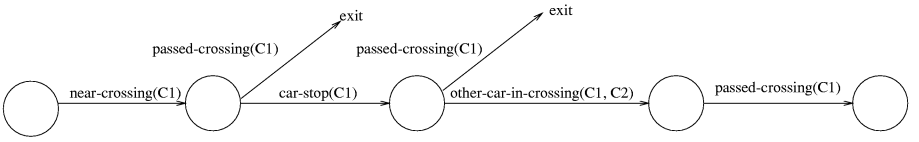


Figure 4.7: The give way episode.

- the reactive executor sends a request to the SIM to recognize the event *(change-lane car1, line1, line2)* of the car currently under observation. When the center of gravity of the car passes from one lane to another the event is recognized;
- the reactive executor now sends the request to recognize an activity *(car-beside car1, car2)* and an event *(change-lane car1, line1, line2)*. If the activity is recognized, the recognition of the episode is continued. If the event is recognized the recognition of the episode is interrupted because the car has come back to the previous lane without overtaking;
- the new requested event is *(change-lane car1, line2, line1)*. When this event is recognized the entire episode is recognized.

In the give way episode a car stops at a crossing and lets one or more cars pass before it starts moving again. The steps in the recognition of the episode are the following:

- the reactive executor sends a request to the SIM to recognize the activity *near-crossing* of the car currently under observation. When the car is less than 50 meters from the crossing, the activity is recognized;
- the reactive executor sends now the request to recognize an activity *car-stop* and an event *passed-crossing*. If the activity is recognized, the recognition of the episode is continued while if the event is recognized the recognition of the episode is interrupted;
- if the *car-stop* activity is recognized, two requests are sent: one of the activity *other-car-in-crossing* and one of the event *passed-crossing*. The first activity is recognized if there is another car in the crossing to which our car is giving way [11]. If the event is produced, the recognition of the episode is interrupted;
- if the *other-car-in-crossing* activity is recognized, the new requested event is *passed-crossing*. When this event is recognized the entire episode is recognized.

Events

The events recognized in the two episodes are the following:

- *(change-lane car previous-lane new-lane)*: it becomes true when the car changes lane from *previous-lane* till *new-lane*. The SIM receives the information about the lane the car is in from the vision module and it compares it with the previously stored value of the lane of the car;

[11] We currently consider every car in the crossing as a car to which our car is possibly giving way. Since we do not know the direction in which both cars are going to turn, it would be difficult to explicitly check if the car in the crossing is actually blocking our car.

- *(passed-crossing car)*: it becomes true when the car has passed through a crossing. The vision module sends information about where the car is with respect to the road network to the SIM. The SIM checks if the car's last position was in a crossing while the new position is on a road.

The *car* symbol in the previous list is the identifier of the car performing the event.

The properties checked are related to the speed and road position of the car. This information is regularly updated for the tracked car[12]. If the car whose properties are being checked is the tracked one, the SIM mainly activates internal functions for processing the data incoming from the vision module, checking the properties of interest and producing the requested events.

If the car whose properties are being checked is a car in the surroundings of the tracked car, then the SIM activates in the vision module an algorithm that gives information about both the tracked car and the cars surrounding it. For instance this is the case of the indexical reference "the car in front of the tracked car". Also in this case, the SIM activates internal functions for processing the data incoming from the vision module, checking the properties of interest and producing the requested events.

If the requested car is outside the current image and there are no other anchoring processes active, the SIM first tries to anchor the requested car and then checks the properties associated with the event. If the requested car is outside the current image and there are other anchoring processes active, the SIM overrides the previous anchor request and starts executing the new one.

The checking of the properties in itself is quite simple. The SIM stores the values of the properties of interest of the previous cycle and compares them with the property values currently being received.

[12] Currently just one car at a time can be tracked by the system.

Activities

The activities recognized in the two episodes are the following:

- *(car-behind car1 car2)*: the system recognizes when there is a car behind *car1* (*car2* is the identifier of the car behind). A car is considered to be behind another car if it is in the same lane and closer to the beginning of the lane[13]. The information about which lane a car is in and at what distance from the beginning of the lane is provided by the vision module;

- *(car-beside car1 car2)*: the system recognizes when there is a car beside *car1* (*car2* is the identifier of the car beside). A car is considered to be beside if it is in an adjacent lane and at a distance of a maximum of 5 meters from the car of interest.

- *(other-car-in-crossing car1 car2)*: the system recognizes when there is a car different from *car1* in the crossing (*car2* is the identifier of the car in the crossing). The information that the car is in a crossing is given by the vision module.

- *(near-crossing car1)*: the system recognizes when the car is at less than 20 meters from the crossing.

- *(car-stop car1)*: the system recognizes when the speed of the car is below a certain threshold[14].

[13]This definition is not very robust. We have adopted it for the time being as it was sufficient for our experiments.

[14]We do not require that the speed is actually 0 because of the inaccuracy in the measurement of the speed. The threshold is established considering the level of inaccuracy of the measurement.

4.8 SIM at work

4.8.1 Searching for a car ...

To illustrate the first functionality of anchoring, that is, the association of a symbolic description with a perceived object, we consider a scenario in which the reactive executor is interested in a red car of a specified model in the vicinity of a given crossing. Four cars are situated around that crossing, moving in different directions. The cars are all red, but of different models: a small van, a large Mercedes, a small Mercedes, and a Lotus. In the first example the UAV hovers over the crossing. In the second example, discriminating between the cars is made more difficult by the fact that the UAV views the crossing at an inclination of about 30 degrees (see Fig. 4.8). This results in some perspective distortions, thus introducing more uncertainty in the extraction of geometrical features.

In our first example, the reactive executor decides to follow 'Van-B', which is described as a red van. The SIM sends the prototypical signature of a red van to the vision module. Since all four cars in the image are red, and they have fairly similar shapes, the vision module returns the observed signatures of all the four cars to the SIM. These signatures are then matched against the desired signature, applying the fuzzy signature matching routine described in chapter 3. The following degrees of matching result:

ID	Color	Shape	Overall
66	1.0	0.58	0.58
67	1.0	0.38	0.38
68	1.0	1.0	1.0
69	1.0	0.0	0.0

The ID is a label assigned by the vision system to each car found in the image. The degree of matching for the color is obtained by combining the individual degrees of hue, saturation, and value; in our case, this will be 1.0 for all the cars as they are all red. The degree

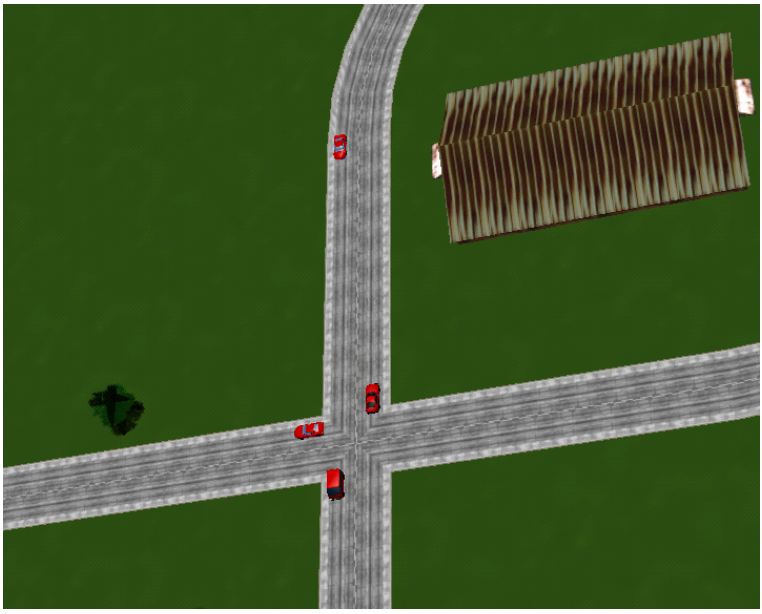


Figure 4.8: The simulated scenario for our examples.

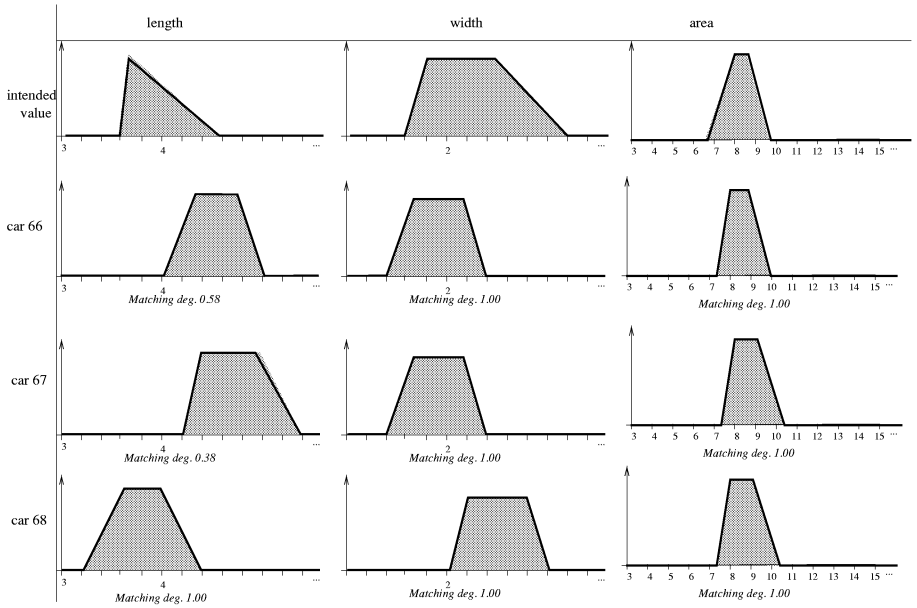


Figure 4.9: The fuzzy sets of length, width, and area of the cars of the example and the references fuzzy sets for a van.

of matching for the shape is the combination of the individual degrees of matching of length, width, and area. The overall degree is the Łukasiewicz combination of the color and shape degrees. The fuzzy sets of length, width, and area of the cars of the example and the references fuzzy sets for a van are shown in Fig. 4.9. In this case, car 68 is correctly[15] identified as the best candidate, and an anchor to that car is thus returned to the reactive executor.

In the second example, the reactive executor is interested in a small red Mercedes. The SIM sends the corresponding prototypical signature to the vision module, and again obtains the signatures of all the four cars in the image as an answer. In this case, however, the UAV is further away from the crossing and it views the crossing at an inclination of about 30 degrees. By applying the fuzzy signature matching routine, we obtain the following degrees:

ID	Color	Shape	Overall
66	1.0	0.65	0.65
67	1.0	0.84	0.84
68	1.0	0.0	0.0
69	1.0	0.97	0.97

Cars 66, 67 and 69 match the desired description to some degree, while car 68 can safely be excluded. The SIM can try to improve the quality of the data by asking the vision module to zoom on each one of cars 66, 67, and 69 in turn. Fig. 4.10 shows a car after the vision module has zoomed on it. Using the observed signatures after zooming, the SIM then obtains the new degrees of matching:

ID	Color	Shape	Overall
66	1.0	0.30	0.30
67	1.0	0.70	0.70
69	1.0	0.21	0.21

The closer view results in a smaller segmentation error, since the scale factor is smaller, and hence in more narrow fuzzy sets. As a

[15]This verification was done manually off-line.

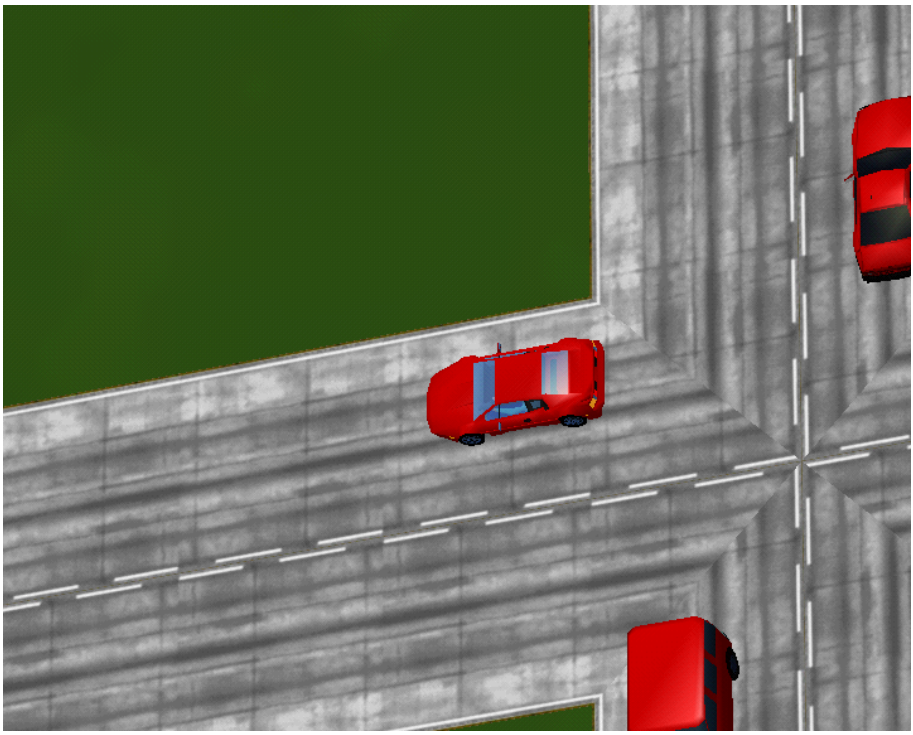


Figure 4.10: A car after the vision module has zoomed in on it.

consequence, all the degrees of matching have decreased with respect to the previous observation. What matters here, however, is the relative magnitude of the degrees obtained from comparable observations, i.e. those collected in the above table. The SIM sends the identifiers of each of the cars to the reactive executor together with their degrees of matching. These degrees allow the reactive executor to select car 67 as the best candidate.

The reactive executor now also has the option to try to further improve its choice by commanding the UAV to fly over car 67 and take another measurement from above the car — the best observation conditions for the vision system. If we do this, we finally obtain a degree of matching of 1.00 for car 67. Note that this degree could as well have decreased, thus indicating that car 67 was not really the car that we wanted. In this case, the reactive executor could have requested the SIM to go back to cars 66 and 69 to get more accurate views.

4.8.2 ... and then following it

Once a car has been found, the second anchoring functionality, i.e. keeping an updated record of the properties of the object (tracking), takes over. The tracking involves the positioning of the UAV above the car and keeping the car in the center of the image. The first is achieved by the reactive executor by adjusting the velocity and direction of the UAV depending on the car's position. The centering of the car in the image is performed by the vision module. Currently these two processes are independent, but we are considering the possibility of integrating them in one control process.

In this example we illustrate the tracking of a car in three different cases of disappearance: the car is lost because of a sudden turn of the UAV, the car disappears under a bridge, and the car disappears into a tunnel.

Let us consider the first case of disappearance. The car followed by the UAV is the car in the center of the image, Fig. 4.11. The car

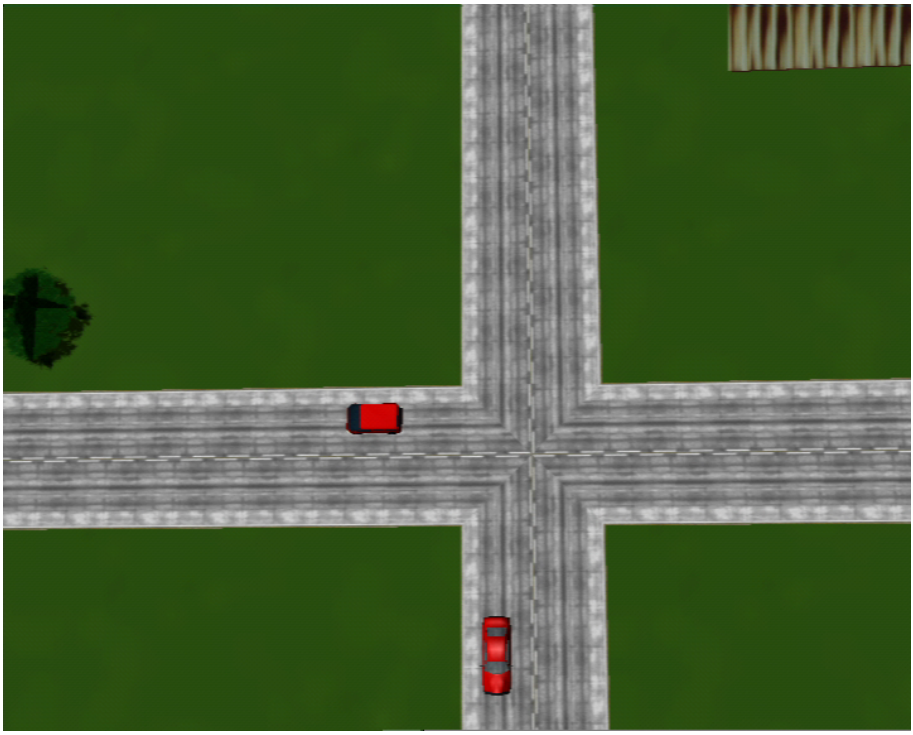


Figure 4.11: The followed car has just turned from the road starting at the bottom of the image to the road going to the left.

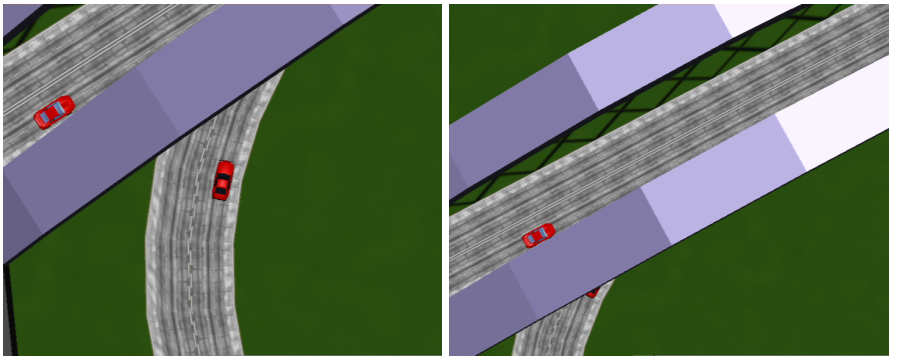


Figure 4.12: The followed car disappears under a bridge and a similar car appears at its place over the bridge.

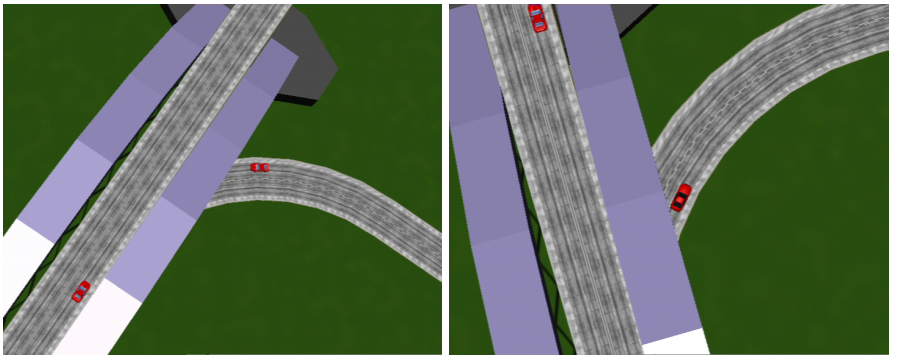


Figure 4.13: Several cars similar to the car that has disappeared move along the roads, but the SIM correctly reidentifies the car when it reappears from under the bridge.

has just turned from the road starting at the bottom of the image to the road going to the left. The UAV makes a sudden turn to follow it, the camera cannot compensate, and the car is no longer visible. The vision module reports to the SIM that the car is no longer in sight. The SIM checks the road database to see if there could be objects occluding the car. As there are no occluding objects in this case, the SIM extrapolates the last observed position of the car to a current expected position. This position together with the description of the car is sent to the vision module. The camera is then turned to the expected position and the car is reidentified.

A second example of disappearance is shown in Fig. 4.12. Two identical cars are present in the image, one traveling along a road which makes a bend under a bridge, and one which travels on the bridge. In this example, the UAV is tracking the first car which will soon disappear under the bridge and, even worse, a few moments later the second car will be in the position in the image where the first car would have been, had it not been occluded, Fig. 4.12 (left). The first car disappears and the second car is the only visible car, Fig. 4.12 (right). The vision module reports the information about the car on the bridge to the SIM. The SIM, however, regularly estimates the expected position of the car and realizes that this car cannot be the correct one as it is in a position incompatible with the previously reported position of the tracked car. The SIM checks the road database and realizes that there is a bridge covering the part of road where the car is expected to be. The predicted position where the first car will reappear is retrieved and the SIM sends this position with the description of the car to the vision module. The camera is turned toward the position waiting for the car to reappear. Several other similar cars move along the bridge and in the opposite lane with respect to the one where the car should reappear and the vision module reports information about them, Fig. 4.13 (left). However, the SIM realizes that they are in positions incompatible with the expected position of the car. The SIM reidentifies the correct car when it reappears from under the bridge, Fig. 4.13 (right).

Figure 4.14: The car disappears under a tunnel and it is reidentified at the exit of the tunnel with the intervention of the deliberative layer.

Finally let us consider the case when the car disappears into a tunnel, Fig. 4.14 (left). In this case the SIM retrieves the information that there is a tunnel at that point in the road, but instead of dealing with the occlusion itself, it reports the disappearance to the reactive executor. The reactive executor, in turn, reports the disappearance to the planning module and the planner creates an appropriate plan taking into consideration possible behaviors of the car such as slowing down in the tunnel and making a u-turn. The UAV moves to the other end of the tunnel and the car is reidentified when exiting from the tunnel, Fig. 4.14 (right).

4.8.3 Event recognition

In this last example we consider the case of recognition of an episode. The recognition of an episode is implemented in the reactive executor, however the recognition of the single events and activities is part of the SIM functionalities. The recognition of events and activities consists in checking the values of properties of anchored objects. In this sense it is related to anchoring even if it is not anchoring in the proper sense.

We present here the recognition of a "give way" episode and of the events and actions composing it. In this episode a car stops at

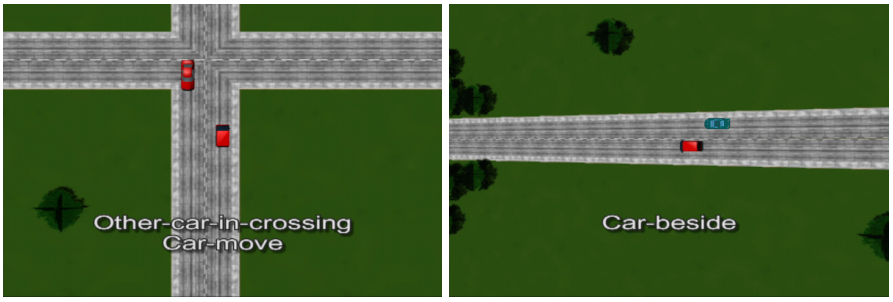


Figure 4.15: The event *other-car-in-crossing* is recognized (left). The event *car-beside* is recognized (right).

a crossing and lets one or more cars pass before starting to move again. The car needs to be anchored and the second functionality of anchoring (tracking) needs to be active. The steps in the recognition of the episode are the following:

- when the car is at less than 50 meters from the crossing, the activity *near-crossing* is recognized;
- when the speed of the car is below 0.2 m/s the *car-stop* activity is recognized;
- the activity *other-car-in-crossing* is recognized, Fig. 4.15 (left);
- finally the event *passed-crossing* is recognized.

Two of the activities and one of the events involve just the anchored and tracked car. However the last activity, *other-car-in-crossing*, involves the recognition of another car. This new car also needs to be anchored and it is referred in relation to the tracked car as being a car that is different to the tracked car and that it is at a crossing. Another example of activity recognition involving two cars is the recognition of the activity *car-beside*, see Fig. 4.15 (right). In this case the tracked car is the car in the lane above and the other car is referred by the deictic property of being the car beside the tracked car.

4.9 Open problems

In this section we state a number of open problems in the SIM that we intend to address in our future work.

Changes in orientation and illumination in definite anchoring The reidentification of an object that has been out of view is currently performed using the color and shape information stored when the object was last seen. However, if the object and the UAV have significantly changed their relative orientation, or if the illumination has changed, the color and shape information could be incorrect. In our experiments we have not experienced particular problems related to this issue, mainly because in our domain the UAV is usually not far from the objects and we presuppose that just a short time passes from when the object was last seen to when it is reidentified. However, we believe that in the general case an expected value of color and shape should be calculated depending on the changes in orientation and illumination.

Quality of the vision data The degree of matching currently calculated depends on how much the fuzzy sets received from the vision module intersect with the fuzzy sets representing the desired features. An additional aspect to take into consideration is the actual quality of the data received from the vision module. If the fuzzy set received from the vision module has a large support, that is, the quality of the data is poor, the degree of matching may not be a good indication of the "goodness" of the matching. Moreover, the knowledge about the quality of the data is important in establishing what actions are suitable to perform to improve the data's quality. For instance if the quality of the data regarding the shape is poor, SIM can request the vision module to zoom on an object to get better measurements.

Combination of features While conjunctive T-norm aggregation has produced a satisfactory behavior in our preliminary experiments,

there are a few reasons why more complex types of T-norms seem more adequate to our case. First, some of the features are more critical than others, and we would like their degree of matching to have a stronger impact on the overall degree. Second, in some situations some values are known not to be reliable and should have little impact on the overall degree of matching: for instance, the observed size of the car is not reliable when the viewing angle is large. Finally, some features have errors which are strongly correlated (e.g., length and width) and it might be wise to combine their individual degrees of matching by an idempotent operator. The search for a more adequate aggregation technique is part of our current development.

Dealing with object disappearance When an object disappears due to a known occluding object, the SIM has to decide whether to invoke the reactive executor or to deal with the disappearance at the SIM level. Currently it is specified that for certain kinds of occluding objects, such as bridges, the SIM does the reidentification, while for other occluding objects, such as tunnels, the reactive executor is invoked. An alternative solution that we intend to investigate in the future, is to establish at run-time whether the reactive executor needs to be invoked. One method to determine this could be to establish a time window inside which the SIM can act autonomously and calculate whether the car is going to reappear inside this time window. Another calculation could be whether the UAV needs to change its speed and/or direction in order to see the car again after the occluding object. In the case when change of speed and/or direction is needed, the reactive executor and the planner need to intervene.

Event and episode recognition In the event recognition part we do not take uncertainty into consideration. This has not been a problem in the experiments that we have performed up to now. We intend to study the matter further when considering more complex event recognition.

The work concerning event recognition of relations among cars is quite recent and the relations are checked using simple methods based on the road structure. The study of more advanced methods is a subject of future work. Furthermore the only cars considered are the ones present in the image: the system does not move the camera in order to cover a larger space in front or behind the car. This could be an interesting option to add to the system.

In the current system an episode can be recognized only when it is completed. In adversarial domains, for instance when the airborne vehicle is chasing a car, it could be useful to consider what episodes could possibly been going on at each time. We would like to explore this possibility in our future work and in particular we would like to apply the ideas that we have developed in a different domain, air-combat simulation, in the WITAS domain. In this work the automated pilot of one of the aircraft evaluates the utility of its actions on the basis of the early recognition of the possible strategies that the opponent could be following, see [15] and [49] for details. Similarly in the WITAS domain, the airborne vehicle could recognize the occurrence of an episode at its early stages and react appropriately.

Real vs. simulated images In our experiments we have been using simulated images. While the general anchoring mechanism should not be affected by the shift towards real images, the actual sets used in the fuzzy matching and especially the fuzzy sets produced by the vision could change when tested with real images.

We intend to proceed in two directions: creating a more realistic simulated environment and testing the algorithms in sequences of images recorded from an airborne vehicle. The reason why both directions are needed is that the testing of the decision making process and of the interaction of the airborne vehicle with the environment need to be done in simulation, while the testing of the vision algorithms requires real images.

The creation of a more realistic environment is currently in progress.

Figure 4.16: More realistic simulation environment.

Fig. 4.16 shows a simulation environment extracted from photos taken over Stockholm with a simulated car in it[16]. In few months it will also be possible to fly the unmanned helicopter that has been selected as a platform for the WITAS project and collect images of traffic scenes. The next step of our work will be to test the anchoring module with these new images.

[16]The simulated car is the one at the top of the image

Chapter 5

Anchoring in a mobile robot domain

5.1 Introduction

In this chapter we consider the application of the anchoring functionalities in a mobile robot domain. The robot performs mainly navigation tasks and uses sonars and odometry as main sensors.

The robot is a Nomad 200 model equipped with a ring of sonar sensors near to the top and two rings of contact sensors near to the bottom. It is located at Örebro University. Fig. 5.1 shows a picture of the robot.

The work in this application is more recent with respect to the work in the WITAS project and it is still in progress. The motivation to present it in the dissertation is the possibility to consider the anchoring functionalities in an application substantially different from the WITAS one. In particular the main sensors are in this case sonars and the emphasis is on the anchoring of static objects.

A typical task of the robot is to reach requested locations in an office environment following a map and avoiding static and dynamic obstacles. The map provides just a sketchy view of the environment since the robot's end-user should be able to use the robot in a new

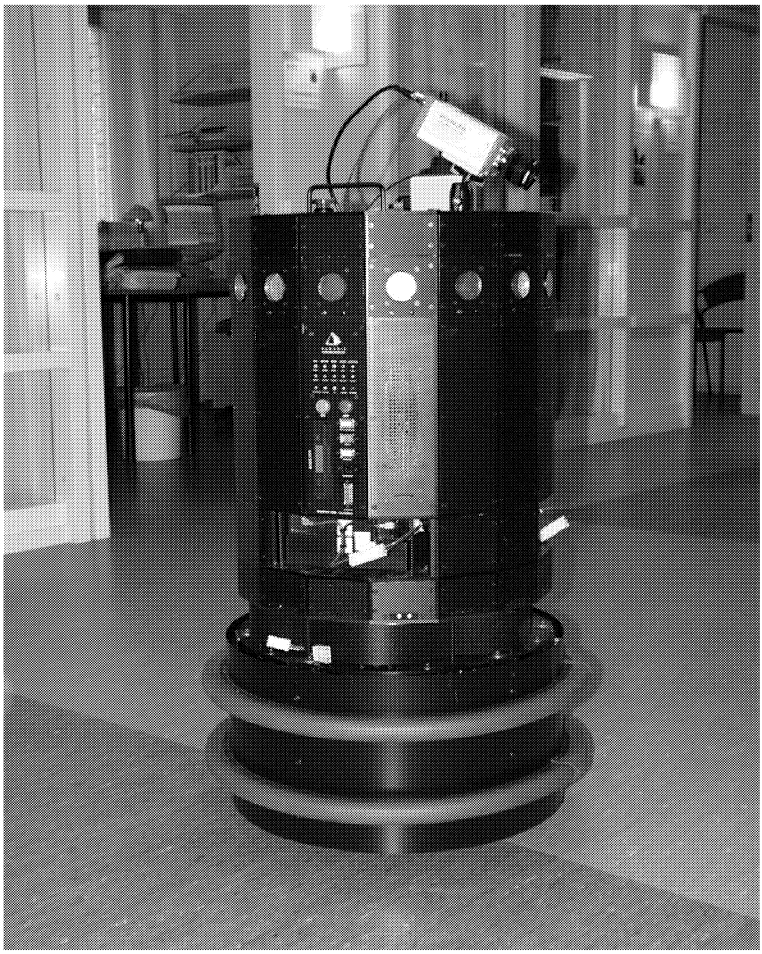


Figure 5.1: The robot used in our experiments.

environment without needing to draw a detailed map. The anchoring functionalities are used to establish a connection among elements in the map and perceptual data collected by the sonars.

The information present in the map is mainly the topological relations among the map objects, for instance which door opens in which corridor and how the corridors are connected. Moreover approximative information about the positions and shapes of objects is stored in the map, for instance a standard width for a door is in general provided and just in special cases (a very large or very small door) the width is actually measured. However, the robot needs more and more precise information about objects when acting in the environment. For instance if the planner decides on the basis of the map information that the robot needs to cross a specific door to reach its destination, the door needs to be recognized in the environment and the actual width and position of the door with respect to the robot needs to be checked using perceptual data. Therefore objects in the map need to be anchored to perceptual data. Anchoring in this domain fulfils the function to connect the perceptual data coming from the sonar to the map objects that are of interest for the robot.

When we started working with the mobile robot, a complex architecture controlling the robot was already present. The architecture included a separate anchoring process, although more primitive with respect to the one presented in this dissertation. Our work has consisted in the redesign of the anchoring module according of the principles and functionalities presented in chapter 3. In particular we have clearly identified the three entities involved in the anchoring process (descriptor, anchor, and percept) and we have separated the find and tracking functionalities. This has resulted in an improvement of the performance of the anchoring process. In particular the introduction of the descriptor entity has given us the possibility to correct, on the basis of the initial description of the desired object, an anchor wrongly established. An example of such a scenario is presented in section 5.4.3 which illustrates the case of an incorrect anchoring of a corridor due to a misinterpretation of the sonars' readings. The correct anchor is

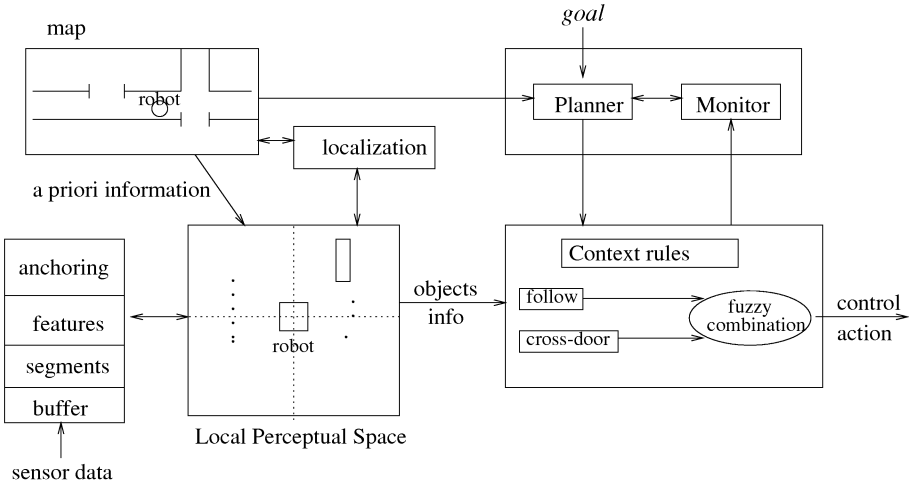


Figure 5.2: Decision making architecture

reestablished when readings are collected that better correspond to the data in the descriptor.

In the rest of the chapter we first introduce the control architecture of the robot, then we describe the anchoring process used in this domain, and finally we present an example where anchoring functionalities are used by the robot while moving along a corridor.

5.2 The control architecture

The control architecture of the robot is shown in Fig. 5.2. The planning system receives a goal to accomplish, produces a plan and activates behaviors needed to execute the plan. The monitoring system controls the effectiveness of a plan with respect to the current goals. The selection of behaviors is performed with the help of context rules. Several behaviors can be active at the same time. A fuzzy control system combines the movement directions coming from the different

behaviors and sends the appropriate commands to the lowest level robot control.

The behaviors may need perceptual and a priori information to execute. For example a "cross door" behavior needs to know the a priori approximate position of the door to move the robot toward it in the first place. When the robot is actually crossing the door more accurate information about the width of the door and its relative position with respect to the robot is needed and this information needs to be acquired through perception. The perceptual and a priori information are collected in the *Local Perceptual Space*. Map information and the perceptual data are used to update the *Local Perceptual Space*. The treatment of perception and a priori information is the focus of this work. Therefore we concentrate on this aspect in the rest of the section. For information about the planning, monitoring and execution parts see [41], [42], and [31].

5.2.1 The sonar sensors

Before going more into detail in the perceptual and a priori information handling let us briefly describe the nature of the data coming from a sonar.

The robot has 16 sonars positioned circularly around the robot. The sonars are activated in sequence. Each sonar sends an ultrasound and receives back the echo produced by the encounter of the ultrasound with the nearest object. The time needed by the echo to come back establishes the distance to the object. Each sonar covers an angle of 30 degrees. A complete cycle, that is collecting of the readings from all the sonars, takes 1 second.

The data coming from sonars can mainly give an indication that there is an object at a certain distance and angle with respect to the robot, but they cannot give any indication of the nature of the object. Moreover, the sonar data cannot be considered reliable if an object is at a distance larger than 3 meters.

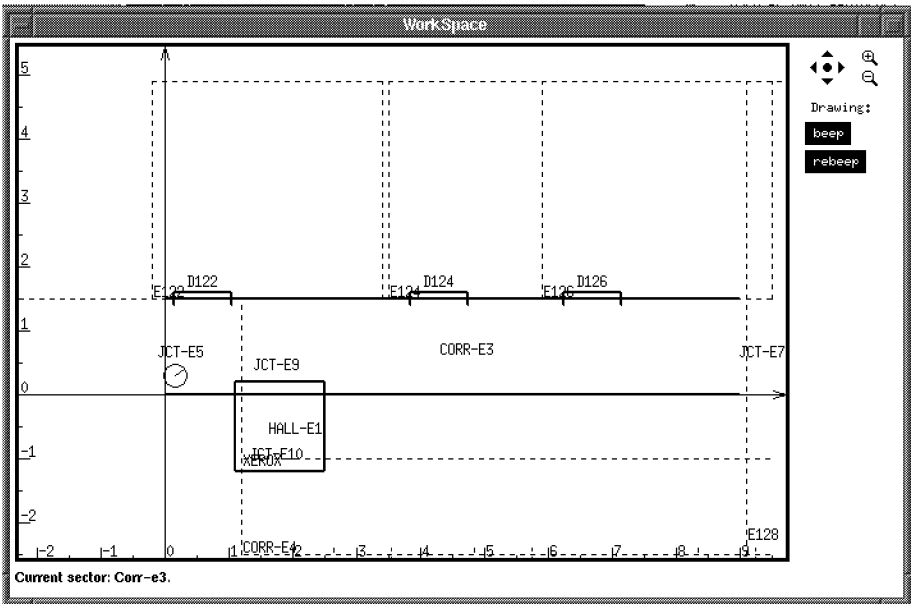


Figure 5.3: A map as represented in the debugging screen during execution.

5.2.2 The map information

The map available to the robot contains topological information such as which rooms open in a corridor and metrical information, for example length of a corridor. However the information is approximate. For instance width of corridors and doors is roughly estimated. In fact it is presupposed that the user of the robot should just need to provide a simple map of a new environment before starting to use the robot in it.

A separate process, the *localization process*, tries regularly to position the robot with respect to the map. Fig. 5.3 shows an example of a map as represented in the debugging screen during execution. The

round object is the robot.

5.2.3 The Local Perceptual Space (LPS)

The Local Perceptual Space is a blackboard-like structure showing an indexical representation of the world surrounding the robot, that is the current view of the world from the perspective of the robot. Figure 5.4 shows an example of the LPS as represented in the debugging screen during execution. At the center is the robot. The information maintained in the LPS includes:

- **Sonar readings represented by points**. Part of the points are current readings and part are update of the position of previously perceived points with respect to the movement of the robot.
- **Objects present in the global map**. In particular the double-line segments are corridor representations and the rectangular shapes are door representations. These objects are represented from the point of view of the robot and are updated while the robot is moving. The correctness of the position of these objects depends on the correctness of the self localization of the robot and on the precision of the actual map.
- **Information about the anchor of the object**, that is, where the object is believed to be according to the information stored in the anchoring structure. For instance the single parallel lines represent where the corridor is believed to be according to the information stored in the anchor. In the figure one can notice a discrepancy between the anchor and the map information (represented by double lines) with respect to the orientation of the corridor. This can be due to incorrectness in the map representation and/or in errors in the odometric estimate (used in the prediction of the positions of the objects while the robot is moving). There appears also to be a discrepancy in the width

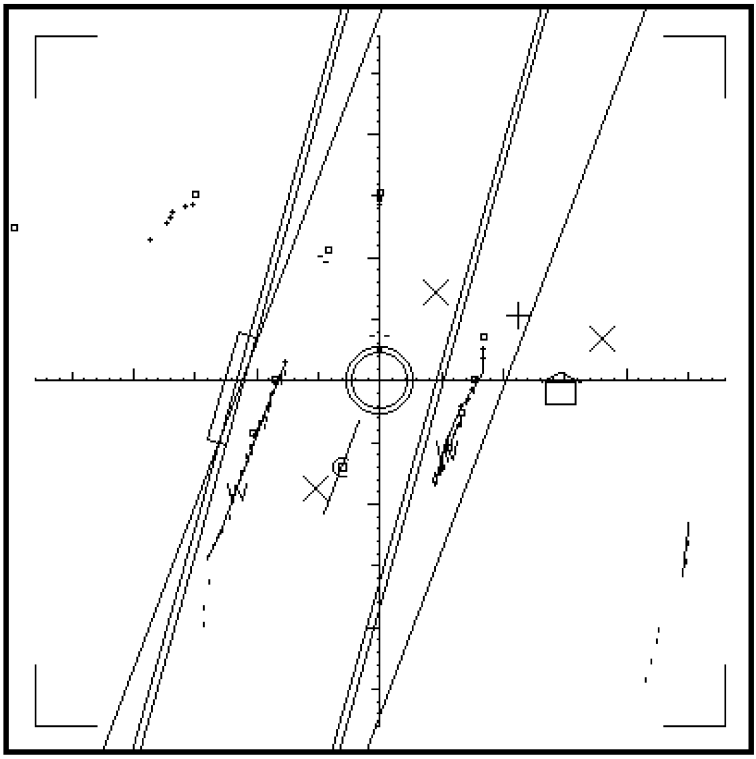


Figure 5.4: An example of the Local Perceptual Space as represented in the debugging screen during execution.

measurement, but this is not actually the case. The lines are just represented slightly moved to one side to make the figure more readable.

- **Perceptual features extracted from sensor data**. For instance, the "W" over the line formed by the perceptual points indicates the recognition of a wall and the "C" in the middle of Fig. 5.4 represents the recognition of a corridor.

5.2.4 Perception module

The *Perception module* consists of 4 processes applied in sequence every time new perceptual data are received and they together form the *perceptual cycle*:

Buffering of perceptual data: the sonar data are stored in a circular buffer where the current data and the old data (up to 8 previous perceptual cycles) are maintained;

Segments recognition: given past and current data, segments are recognized from a sequence of points. The sequence of points should be of at least a pre-specified length in order to be recognized as a segment;

Feature recognition: given the segments recognized in the previous process, features such corridors, doors, and walls are recognized. The recognition of walls and corridors is of particular interest for our examples. A wall is recognized when a segment of at least a pre-specified length is detected. The length is dependent on the environment. For instance, in an environment with long and smooth corridor the length required can be longer, while in an environment where the corridors have several recesses, as the one of our experiments (see Fig. 5.5), the length must be shorter. A corridor is recognized when two parallel walls with a suitable distance between each other are recognized;

Anchoring: the currently perceived features are anchored to symbolic identifiers such as "corridor 11" and "a door in corridor 12". The anchoring process is the topic of the following sections.

5.3 The anchoring process

The anchoring process tries to anchor every map object present in the Local Perceptual Space to a perceived object. The map objects present in the LPS are all the objects in the radius of 2.5 m from the robot and that are reported in the map representation of the environment. Each object is represented in the map by a name, the symbol used at higher levels in the system to identify the object, and by an approximate description, for instance an estimate of the width and orientation of the object.

The two functionalities of anchoring introduced in the first chapter are also present in this domain: finding an object for the first time and keeping track of it over time. The objects to be anchored are in this case static, therefore the tracking is actually necessary only if the robot moves. Tracking the object re-establishes the current position of the robot with respect to the object, and it gives to the anchoring process the opportunity to re-evaluate the correctness of the anchoring by seeing the object from a different point of view.

5.3.1 The object representations used by the anchoring process

The object representation used by the anchoring process are of three kinds:

- The **map description** stores information about a priori knowledge about the position of the object, its shape, and its connections to other objects. For instance, in the case of a corridor the information present in the map description is the coordinates of the center of the corridor, its width, the rooms opening in the

corridor and the corridors connected with it. The information is however approximate, the aim being to be able to use the robot having just a sketchy map of the environment. For example the width of the corridor is not actually measured, it is just a measure of an "average" corridor. The map description is a *descriptor* according to the terminology used in the first chapter;

- The **anchor** is created when an anchor is established between a descriptor and a perceptual object. It stores the last perceived data of the object such as shape and position of the object with respect to the robot. It also stores a value in the interval $[0, 1]$ indicating the reliability of the anchor. The reliability value is 1 when the descriptor is newly anchored and it decreases if the anchor is not reestablished in the following perceptual cycles and the robot moves;

- The **percept** is created when an object is recognized by the feature recognition module, for instance a corridor or a door and it is maintained just until the object remains in the field of view of the sensors. It stores the perceptual data of the object such as shape and position with respect to the robot.

5.3.2 The implementation of anchoring

The anchoring process considers every object in the LPS at every perceptual cycle and it checks if it has already been anchored. If the object is not anchored, the first functionality, finding the object, is applied. Otherwise the track functionality is applied.

5.3.3 Compatibility between object representations

Before presenting the finding and tracking functionalities, let us introduce the concept of compatibility between object representations.

Two object representations are *compatible* if all the measurable data stored in the representation do not differ more than certain values,

called *compatibility values*. Different compatibility values are used for different measurable data. The "largeness" of these values establishes the strictness of the compatibility: the larger the values the less strict the compatibility test.

The compatibility values are calculated using the following formula:

$$\text{Comp. value} = (\lambda * MinimumValue) + ((1 - \lambda) * MaximumValue)$$

Where *MaximumValue* and *MinimumValue* are the maximum and minimum values we wish the compatibility values to assume and are constant during execution. λ is a number in [0,1] that varies depending on how strict the compatibility needs to be. In our examples λ is 0 when the compatibility between the perceptual data and the descriptor is tested and no anchor has yet been found. In this case the compatibility value is at is maximum. We are in fact willing to accept a candidate anchor even if it is not very good. λ is 1 when an anchor has been found in the previous perceptual cycle and we compare the new perceptual data with it. In this case the compatibility value is at its minimum. Finally λ is between 0 and 1 when the anchor was established previously, but it could then not be reestablished for some time while the robot was moving. λ tends to decrease towards 0 proportionally to the space covered by the robot during the time the anchoring was not reestablished. As a consequence the compatibility value tends to increase towards is maximum value.

Finding an object

If an object is not anchored, the anchoring process tries to find an object among the perceived ones that is compatible with the descriptor. The comparison is made on the basis of the features of the object that can be perceived by the sensors. For instance, in the case of a corridor the perceived width and orientation of the corridor is compared with the width and orientation stored in the descriptor.

In general the perceptual data of several objects can be compatible with the descriptor to different degrees. The one whose measurable

data are closer to the a priori information stored in the descriptor is selected and an anchor is established. If no object whose perceptual data are compatible with the descriptor is found, the anchoring process tries again to find a compatible object when new sensor data are acquired. The control routines of the robot use, while the object is not anchored, the a priori information about the object. For instance, if the robot is supposed to follow a corridor, the control routines initially use the a priori information about the corridor to start moving the robot. While the robot is moving along the corridor, more percepts accumulate and the anchor of the descriptor to these percepts may be established.

Tracking an object

When an object has been anchored, the anchoring process keeps maintaining the anchoring between the descriptor and the percept. A difference between the visual sensor considered in the UAV domain and the sonar sensor of this domain is that in the former case it is possible to track the object by regularly moving the camera so that the object is always in the center of the image. In the case of sonar readings it is not possible to direct the sensor to the object of interest.

Every time new percepts are provided the anchoring process tries to find among them the best candidate for updating the anchor. The compatibility of the data of the percept with the data in the previous anchor and the a priori information in the descriptor is tested. The strictness in the compatibility test depends on the reliability value of the anchor: the higher the reliability value the more strict the compatibility test[1].

Let us now introduce in more detail the algorithm for tracking an object. It considers four different cases:

- If there is at least one perceived corridor whose perceptual data

[1]This is obtained assigning to the λ in the compatibility test the number indicating the reliability value.

are compatible with the current anchor and if the perceptual data are also compatible with the information in the descriptor, the anchor is updated with the new readings. If there are several perceived corridors whose perceptual data are compatible with the current anchor, the one with the best matching is selected.

- If there is at least one perceived corridor whose perceptual data are compatible with the current anchor, but they are not compatible with the information in the descriptor, several policies could be possible, privileging the perceptual and the a priori information to different degrees. We currently favor the perceptual information and we update the anchor with the newly perceived data[2].

- If there are no perceived corridors whose perceptual data are compatible with the current anchor, but there is at least a perceived corridor whose perceptual data are compatible with the descriptor, several policies could be possible. We currently update the anchor with the data of the perceived corridor. This policy gives the anchoring process the possibility to recover from erroneous anchoring as we see in the last of the following examples.

- If there are no perceived corridors whose perceptual data are compatible with either the current anchor or the descriptor, the anchoring process maintains the previous anchor. However the more the robot moves away from the place where the anchor was last established the more the anchor decreases is reliability value.

[2]This is actually one of the aspects of the algorithm that has given rise to the most intense discussions. Although this solution has performed well in the current examples, we intend to reevaluate it in the near future with the help of additional examples.

5.4 Anchoring at work

We illustrate the use of the anchoring process by presenting three examples of anchoring that have been implemented and tested in the robot. In all three examples the robot starts from a similar position and anchors the descriptor of a corridor with data perceived by the sonars. Figure 5.5 shows the robot at its starting point. The map of the environment used by the robot is shown in figure 5.6. The corridor `CORR-E3` shown in the map is the one formed by the wall on the left and the copying machine and shelf on the right. The recognition of this corridor through sonar readings presents some difficulties due to the fact that the one on the right is not a proper wall and the wall on the left is not uniform. In our last example we show a case where the Feature Recognition module fails to correctly recognize the corridor due to the peculiar form of the corridor.

The first example illustrates the "normal" case in which the robot moves along a corridor and the perceptual data of the corridor are correctly anchored to the descriptor. In the second example the robot is in the same position as in the previous example, but it has made a wrong estimate of its own position with respect to the map, in particular the estimate of the heading is off at around 20 degrees. In the third example the sonars' readings induce the Feature Recognition module to recognize a corridor where no corridor is actually present. The anchoring process first anchors the incorrect corridor to the descriptor but then, when better data are collected, the correct anchor is established.

An important aspect that needs to be emphasized is that all the three entities involved in the anchoring process: the descriptor, the anchor, and the percept need to be taken into consideration when performing the anchoring process. In the case of disagreement among the data stored in these entities, different policies need to be used giving preference to one entity over the other depending on the application, the reliability of the sensor data, and the reliability of the localization ability of the robot. In the examples we choose specific policies to deal

Figure 5.5: A photo of the environment at the beginning of the experiments.

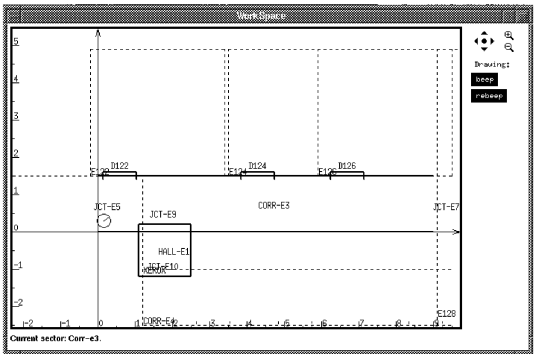


Figure 5.6: The map of the environment used by the robot with the robot in its initial position. Corridor E-3 is the corridor to the left of the xerox machine in Fig. 5.5.

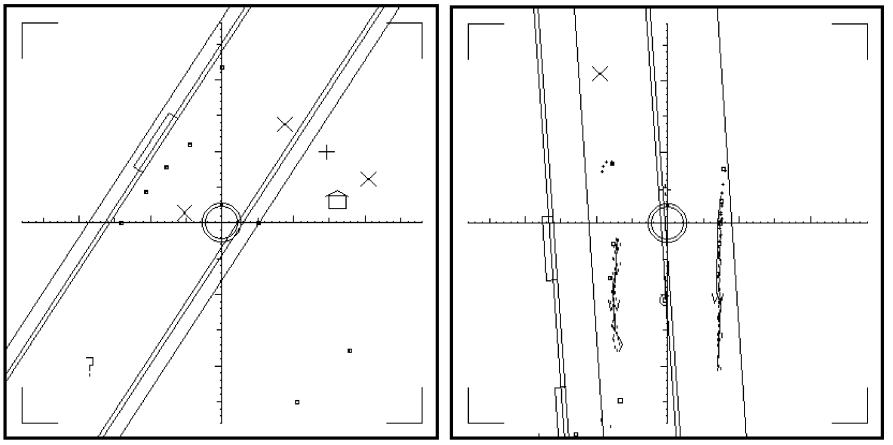


Figure 5.7: The LPS space of the robot at the beginning of the execution (left). A corridor is recognized and the anchor is established (right).

with the disagreement. However the intent of the examples is not to illustrate the policies themselves, or to claim that these policies are the optimal ones for this domain. We are aware that it could always be possible to construct specific examples where these policies do not achieve the correct results. The main intent is to illustrate the possibility to easily implement different policies in the framework offered by the anchoring functionalities.

5.4.1 Anchoring of a corridor in the "normal" case

In this example the robot is initially positioned at the beginning of corridor E3. The position of the robot can be seen in Fig. 5.6 and Fig. 5.5. The robot has the goal of moving along the corridor, so the descriptor of the corridor is added to the LPS and the anchoring process starts to try to anchor the descriptor to the perceptual data. Initially the sonars perceive a number of points. The LPS space of the robot

is shown in Figure 5.7[3]. The a priori information about the corridor is drawn with a double line. The Feature Recognition module tries to establish whether the points are part of a wall. In this case however the string of points is too short to serve as a basis for the recognition of a wall. As a consequence no corridor is recognized. The single line indicates where the corridor is according to the robot knowledge. As no perceptual data about the corridor has been collected, the only knowledge that the robot has is the a priori information [4]. The control of the robot uses the a priori information about the corridor to start moving the robot along the corridor.

Additional sonars readings are accumulated while the robot is moving and after few perceptual cycles two walls are recognized. The recognition of each wall is shown in the debugging window for the LPS by the appearance of a "W" over the string of points, see Fig. 5.7. In this case two parallel walls with a distance between each other compatible with the one of the corridor are detected. Therefore the Feature Recognition module recognizes a corridor. The anchoring process compares the percept of the corridor (orientation and width) with the a priori data present in the descriptor. In this case the data are compatible and the descriptor is anchored to the percept. The anchoring of an object for the first time implies the creation of an anchor containing: the perceived data of the object, a pointer to the descriptor, and a number in [0,1] indicating the reliability of the anchor. When the object has just been anchored, the reliability value is 1.

The control of the robot now uses the data in the anchor to direct the movement of the robot along the corridor. The newly perceived data about the corridor is compared by the anchoring process to the

[3]Notice that, while the experiments have been performed both in the real robot and in the simulator provided together with the robot, the figures are taken during the simulator execution. This is due to the difficulty of executing a step by step experiment in the real robot as needed for extracting a sequence of figures.

[4]Notice that the single line in the figure should actually coincide with the double line indicating the a priori information; however the single line is drawn slightly on one side to facilitate reading the figure.

data currently present in the anchor structure. In this example the data are compatible and are therefore used to update the anchor.

The following is the transcription of the messages produced by the program while executing the anchoring process of this example.

```
Switching sector from E115 to CORR-E3 through JCT-E5
[...]
Found first anchor for CORR-E3
Percept matches both anchor and descriptor: CORR-E3 anchored
[...]
Percept matches both anchor and descriptor: CORR-E3 anchored
[...]
```

The program records first the finding of an anchor and then the fact that in the following cycles the percept match both the data stored in the anchor and the a priori information in the descriptor.

5.4.2 Anchoring of a corridor with initially incorrect robot position

In the second example we consider the case when the robot has initially an incorrect estimation of its heading at around 20 degrees. This can be due to an accumulation of odometry errors in the previous navigation period. Similarly to the previous case, the control starts moving the robot along the corridor and a corridor is recognized by the Feature Recognition module. As one can see in figure 5.8, there is a significant discrepancy between the a priori information about the corridor (double line) and the perceptual data (single lines with "W" written over). Several policies could be taken in this case, favoring the a priori information over the perceptual data or vice versa. In our case, given that the corridor has never been anchored before, we allow a relatively large difference between the a priori information and the

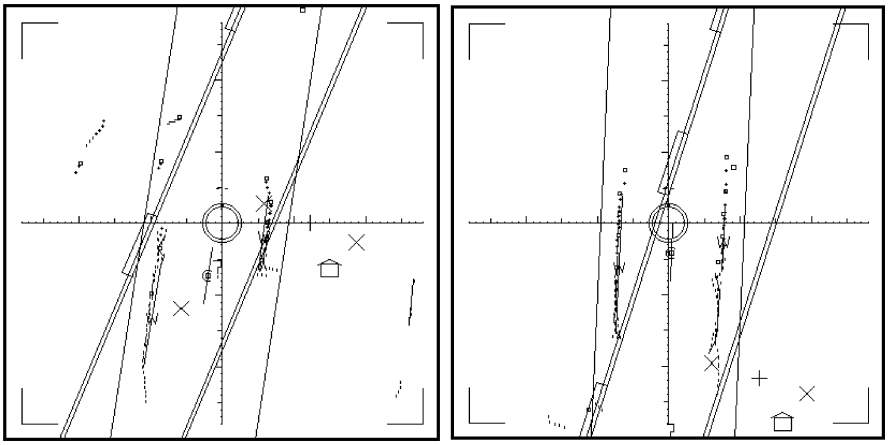


Figure 5.8: Example where there is a significant discrepancy between the a priori information and the perceptual data.

perceptual data. We adopt this policy mainly because the a priori information in this application is not very precise and it depends on the accurate localization of the robot, which can be unreliable. Once the corridor has been anchored for the first time, we adopt more strict criteria of comparison between the new percepts and the data stored in the anchor. In figure 5.8 the new perceptual data (single lines with "W" written over) are very similar to the data present in the anchor (single line) so the new perceptual data are used to update the anchor. In the following example we see a case in which the new percepts are not compatible with the data in the anchor structure.

The following is the transcription of the messages produced by the program while executing the anchoring process of this example.

```
Switching sector from E115 to CORR-E3 through JCT-E5
[...]
Found first anchor for CORR-E3
Percept matches anchor but not descriptor: CORR-E3
```

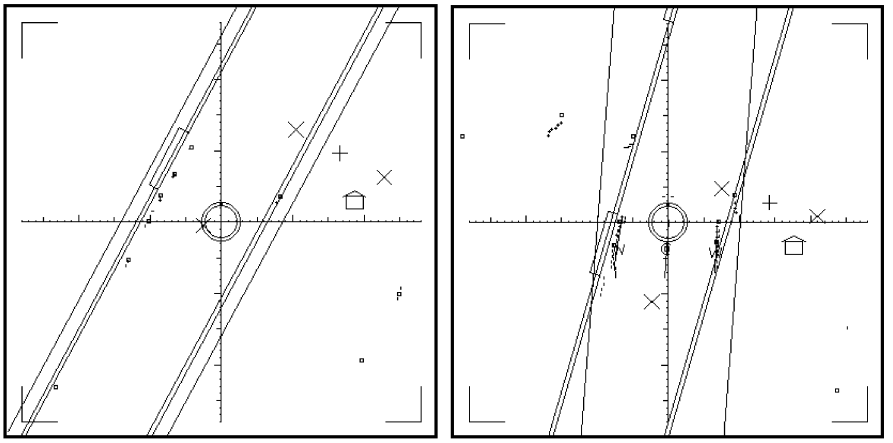


Figure 5.9: Example where the corridor is initially incorrectly anchored.

```
anchored all the same
[...]
Percept matches anchor but not descriptor: CORR-E3
anchored all the same
[...]
```

In this case the anchor is established and the following perceptual data match the anchor, but not the descriptor.

5.4.3 Anchoring a corridor in the presence of incorrect recognition by the Feature Recognition module

In this last example we consider the case when the Feature Recognition recognizes a corridor where in reality there is no corridor. This incorrect recognition is due to the fact that the wall to the left of the robot has a number of recesses that sometimes produce a number of aligned points in the sonar readings. The points can form a line that has an

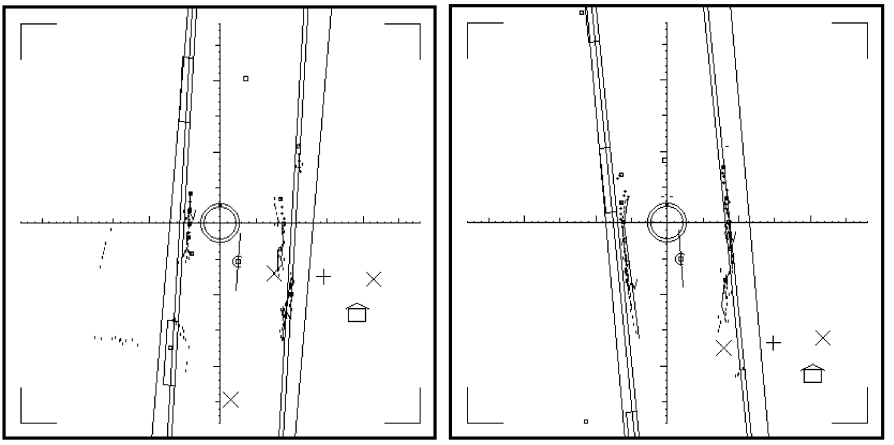


Figure 5.10: The corridor is now correctly anchored.

angle with respect to the actual wall (see figure 5.9). To the right of the robot there is a copying machine that in some cases produces a number of aligned points that form a parallel line with respect to the line on the left. Therefore in some unlucky cases two parallel walls are recognized and the Feature Recognition recognizes a corridor. Fig. 5.5 shows a photo of the corridor. The anchoring module considers this an acceptable anchor for the descriptor as this is the first time the descriptor is anchored and the comparison criteria are less strict in this case. The robot moves along the corridor and new percepts indicate a corridor with a different orientation than the previously anchored one. These new data are actually correct with respect to the real corridor.

The anchoring process compares these new data with the data stored in the anchor structure. There is a substantial difference between the data. Moreover the anchor has high reliability as it has been established very recently. Therefore the compatibility test is strict and the data are considered incompatible with the data stored in the anchor. However, a second comparison is done, this time with the a priori data in the descriptor. Given that the new data match very well

with the a priori information the anchor is updated with the new data correcting the initial anchoring error. Fig. 5.10 shows the corrected anchor. The following percepts are compatible with the data in the anchor and the data in the anchor are updated, see Fig. 5.10.

The following is the transcripted version of the messages produced be the program executing the anchoring process.

```
Switching sector from E115 to CORR-E3 through JCT-E5
[...]
Found first anchor for CORR-E3
Percept matches anchor but not descriptor:
CORR-E3 anchored all the same
[...]
Percept matches anchor but not descriptor:
CORR-E3 anchored all the same
[...]
Percept only matches descriptor: CORR-E3 anchored
Percept matches both anchor and descriptor: CORR-E3 anchored
[...]
Percept matches both anchor and descriptor: CORR-E3 anchored
[...]
```

The program records first that an anchor is established, despite the fact that the perceptual data does not match well the data in the descriptor. This is detected, however, in the subsequent perceptual cycles where the new data are compatible with the anchor but not with the descriptor. After few cycles the program records that the perceptual data are compatible only with the descriptor and that the anchor is updated with these new data. In the subsequent cycles the percepts are compatible with the data stored in both the anchor and descriptor.

To make a parallel between this application and the previously presented WITAS application, this case is comparable with the case in

which initially a car is considered to be the correct one, even if it does not match the description very well as there is no better candidate. However in following observations a car that better matches the description is detected and the system corrects itself and recognizes this car as the correct one.

5.5 Open problems

The work presented in this chapter is in progress, therefore there are a number of important issues that still remain to be addressed.

Anchoring and localization Currently two processes execute in parallel in the system: the anchoring process and the localization process. The localization process establishes the current position and heading of the robot with respect to the map using odometry information, but also recognized landmarks such us corridors and doors. Therefore the localization process is partly dependent on the correctness of the anchoring process. On the other hand, the correctness in the localization influences the correctness of the map information that in its turn influences the correctness of the anchoring process. We intend to study the interaction between these two processes and try to prevent errors in one process from generating errors in the other process.

Policies to be used in case of discrepancies among object representations The policies used in this chapter have given good results in our examples and seem to be appropriate for this particular application. However we intend to consider the performance of different policies in a wider range of examples and to study in a more systematic way the consequences of choosing one policy over another.

Fuzzy matching In this application the matching among percept, descriptor and anchor is crisp. It consists of a simple check against

lower-upper bounds and it is therefore a degenerate case of fuzzy matching. This is due to the on-going nature of the work. We intend in the near future to experiment with more complex fuzzy matching.

Integration of sonars with other sensors A robot using mainly sonar sensors is limited in the possible tasks that it can perform. The addition of other sensors can greatly expand the number of tasks that can be executed and can improve the execution of currently performed tasks [51]. For instance the crossing of a door can be greatly facilitated if the door is at the same time sensed with the sonar and seen with a video camera. The integration of the perceptual data coming from several sensors offers new challenges to the anchoring process. This is one of the directions in which we intend to direct our future research. In particular we would like to examine the integration of less "common" sensors such as the artificial mouth and artificial nose currently being developed at Örebro University.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Intelligent agents embedded in physical environments and performing complex tasks need the ability to perceive the environment, understand the current situation and make decisions about future actions. However while in general the decision making part of the system uses symbolic reasoning, the perceptual part of the system deals with quantitative data. An essential aspect for a successful understanding of the current situation and the performing of actions is the connection between the symbols used at the decision making level and the perceptual data provided by the sensors. In this dissertation we have addressed the problem of connecting the symbols used to perform abstract reasoning to the physical entities which these symbols refer to. We have called this problem the *anchoring problem*. We have then outlined a number of functionalities that we believe are needed when solving the anchoring problem and a number of requirements that should in general be satisfied when implementing an anchoring module.

The definition of anchoring and of its functionalities has been tested in two applications: an unmanned airborne vehicle (UAV) used for

traffic surveillance, and a mobile robot performing navigation tasks. The two experimental platforms share the use of a layered architecture to integrate abstract reasoning with perceptual and control processes. However, these platforms differ significantly in terms of sensory-motoric capabilities and their domains of application. Experimenting with anchoring in systems acting in different environments and using different sensors (the mobile vehicle uses sonars as its main sensors) has helped us in outlining general functionalities and requirements that we believe are common to any anchoring process.

6.2 Future work

Anchoring is a complex and multi-faceted problem. This dissertation can be considered a first study on it and a number of issues still need to be addressed. At the end of the application chapters we have proposed a number of open problems related to the domain addressed in the chapter. In this section we consider more general issues that we believe are of importance in understanding and trying to solve the anchoring problem.

6.2.1 Generalization of the anchoring concept

In this dissertation we have given a definition of the concept of anchoring and its functionality. The process of trying to find general principles behind the anchoring process has greatly benefited from being able to consider two different domains where anchoring plays an essential role and different kinds of sensory platforms. We intend to continue in our generalization process considering different, and less common, kinds of sensors, such as smell and taste sensors currently available at Örebro University. Additionally an infrared camera is will be available in the WITAS platform.

Figure 6.1: A traffic scene viewed through an infrared camera.

6.2.2 Sensor fusion

In this dissertation we have consider anchoring when one sensor is involved. In general if several sensors are present in the system there could be the additional problem of fusing the information coming from the different sensors about an object and creating a unique representation to be used as the referent to the symbolic representation of the object. In particular the object needs to be identified as the same object in all sensor representations.

In both the domains studied in this thesis we are now introducing new sensor platforms. In the WITAS project the video camera sensor will be complemented by an infrared camera. An image taken from an infrared camera is shown in Fig. 6.1. In such images it is quite easy to differentiate the cars from the road and they could therefore be useful for the calculation of for instance the car velocity, although additional information such as color and shape needs to be extracted using a regular video camera image. If a car is identified as the same one in both images, the properties extracted from both images can be combined in one common car representation.

One of the aims of the research on the robotic platform developed at Örebro University is to integrate a number of sensors that will allow the robot to move in the environment, pick up objects, and smell and taste them. This long-term goal is illustrated in an amusing picture showing the robot participating in a Swedish traditional buffet, Fig. 6.2. The challenge in this domain is to integrate sensors traditionally used in robotics such as video cameras and sonars, with an artificial mouth and nose. For instance the recognition of a food can involve the combination of visual perceptions with the perception of its taste and smell.

6.2.3 Anchoring in communication between agents

The notion of anchoring can be extended to address a general aspect of communication between agents. Anchoring is currently conceived

Figure 6.2: The Örebro University robot participating in a Swedish traditional buffet. Courtesy of AASS, http://www.oru.se/forsk/aass/, Örebro University.

as a mechanism for creating a correspondence between the internal representation used by the perceptual system and those used by the reactive reasoning system to denote the same external object. In more general terms, we can see anchoring as (one side of) the problem of establishing the correspondence between the representations used by two different systems embedded in the same physical environment to refer to the same objects in the environment. In particular, in the next phase of the WITAS project an autonomous ground vehicle will provide services to the airborne vehicle such as re-tanking. The two vehicles will need to coordinate and communicate using shared references of objects. Moreover a new aspect of the WITAS project will be the involvement of a human operator that will communicate with the airborne vehicle giving additional instructions and/or changing the current mission. Even in this case the need to refer to the same objects in the environment is clear.

There is also an interest in cooperation and communication among robots at Örebro University. In particular, we intend to study the possibility of cooperation among different kinds of robots in performing tasks. An interesting case is the cooperation among Sony legged robots in the RoboCup domain. RoboCup is an international soccer competition among robots and among simulated agents; see [29] and [28] for additional information on the RoboCup initiative. Örebro University has been part of the Swedish team for the Sony legged robot league of this year's competition, RoboCup-99, and will probably be part of next year's competition, too. The Sony legged robots are equipped with a color video camera and can also hear. Fig. 6.3 shows a picture of the robot. The effort in this year's RoboCup has been mainly to program each robot so that it could perform the basic functionality; however, to get a good team, cooperation among robots is the next necessary step: this cooperation will involve some communication about the objects in the environment, and this will require the sort of reference sharing that is a property of anchoring.

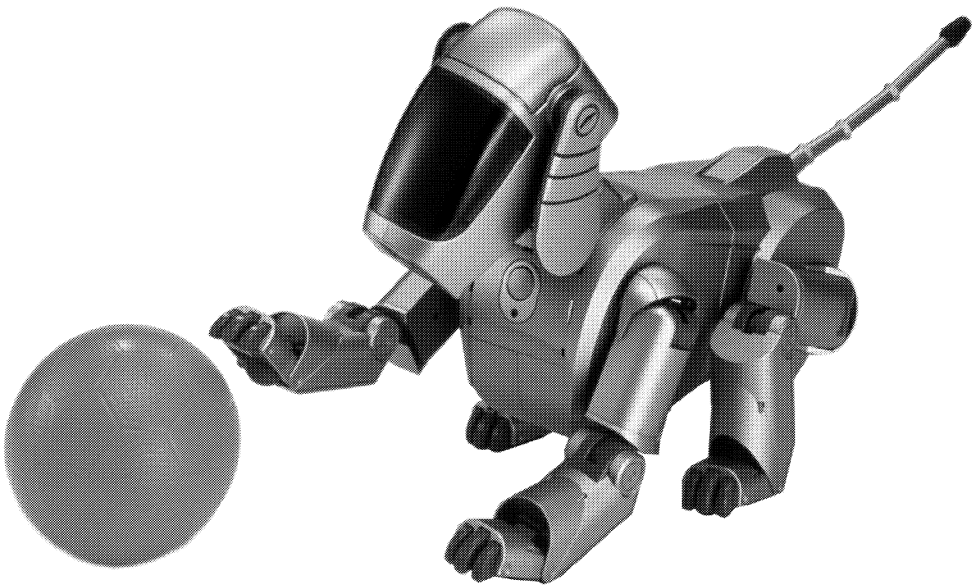


Figure 6.3: The Sony legged robot. Courtesy of the RoboCup Federation.

References

[1] P. Agre and D. Chapman. Pengi: an implementation of a theory of activity. In *AAAI87*, pages 268–272. Seattle, WA, 1987.

[2] T. Andersson, S. Coradeschi, and A. Saffiotti. Fuzzy matching of visual cues in an unmanned airborne vehicle. Linköping Electronic Articles in Computer and Information Science, Vol. 4 (1999): no. 8. `http://www.ep.liu.se/ea/cis/1999/008`.

[3] R. C. Arkin. *Behavior-Based Robotics.* MIT Press, Cambridge, MA, 1998.

[4] R. Bajcsy and J. Košecká. The problem of signal and symbol integration: a study of cooperative mobile autonomous agent behaviors. In *Proceedings of KI-95: Advances in Artificial Intelligence, 19th Annual German Conference on Artificial Intelligence*, volume 981 of *LNCS*, pages 49–64, Berlin, Germany, 1994. Springer.

[5] D.H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–87, 1991.

[6] J. Barwise and J. Perry. *Situations and Attitudes.* The MIT Press, 1983.

[7] G. Berry. *The Esterel v5 Language Primer*, version 5.10, release 2.0 edition, 1998. Included in the Esterel distribution available from `http://www-sop.inria.fr/meije/esterel/`.

[8] I. Bloch. Information combination operators for data fusion: A comparative review with classification. *IEEE Transactions on Systems, Man, and Cybernetics*, A-26(1):52–67, 1996.

[9] A. Bonarini. Symbol grounding and a neuro-fuzzy architecture for multisensor fusion. In *Proceedings of the World Automation Conference (WAC)*, pages 75–80, Montpellier, FR, 1996. TSI Press.

[10] H. Buxton and Shaogang Gong. Visual surveillance in a dynamic and uncertain world. *Artificial Intelligence*, 78:431–459, 1995.

[11] P. Cohen. The need for identification as a planned action. In *IJCAI81*, pages 31–36. 1981.

[12] S. Coradeschi, L. Karlsson, and K. Nordberg. Integration of vision and decision-making in an autonomous airborne vehicle for traffic surveillance. In H. I. Christiansen, editor, *Computer Vision Systems*, volume 1542 of *LNCS*, pages 216–230, Berlin, Germany, 1999. Springer.

[13] S. Coradeschi and A. Saffiotti. Anchoring symbolic object descriptions to sensor data. Problem statement. Linköping Electronic Articles in Computer and Information Science, Vol. 4 (1999): no. 9. `http://www.ep.liu.se/ea/cis/1999/009`, 1999.

[14] S. Coradeschi and A. Saffiotti. Anchoring symbols to vision data by fuzzy logic. In A. Hunter and S. Parsons, editors, *Qualitative and Quantitative Approaches to Reasoning with Uncertainty*, LNAI, pages 104–115. Springer, Berlin, Germany, 1999.

[15] S. Coradeschi and T. Vidal. Accounting for temporal evolutions in highly reactive decision-making. In L. Khatib and R. Morris, editors, *Proceedings of the Fifth International Workshop on Temporal Representation and Reasoning (TIME98)*, pages 3–10, Los Alamitos, California, 1998. IEEE Computer Society.

[16] P. Davidsson. Toward a general solution to the symbol grounding problem: combining machine learning and computer vision. In *AAAI Fall Symposium Series, Machine Learning in Computer Vision: What, Why and How?*, pages 191–202. AAAI Press, 1993.

[17] D Driankov, H. Hellendoorn, and M. Reinfrank. *An introduction to fuzzy control.* Springer, 1993.

[18] J. Firby. Task networks for controlling continuous processes. In *Proceedings of the Second International Conference on AI Planning Systems*, 1994.

[19] R. J. Firby. The RAP language manual. Technical Report AAP-6, University of Chicago, 1995.

[20] F.L.G. Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, pages 25–50, 1892.

[21] E. Gat. Three-layer architectures. In R.P. Bonasso D. Kortenkamp and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*, chapter 8, pages 195–210. MIT Press, Cambridge, MA, 1998.

[22] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231 – 274, 1987.

[23] S. Harnard. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[24] H. Hexmoor, J. Lammens, and S. C. Shapiro. Embodiment in GLAIR: A grounded layered architecture with integrated reasoning for autonomous agents. In D. Dankel, editor, *Proceedings of the Florida AI Research Symposium*, pages 325–329, 1993.

[25] R. Howarth. *Spatial representation, reasoning and control for a surveillance system.* PhD thesis, Queen Mary and Westfield College, 1994.

[26] R. Howarth. Interpreting a dynamic and uncertain world: task-based control. *Artificial Intelligence*, 100:5–85, 1998.

[27] D. Hutber, S. Moisan, C. Shekhar, and M. Thonnat. Perception-interpretation interfacing for the prolab2 road vehicle. In *Proceedings of 7th Symposium on Transportation Systems: Theory and Application of Advanced Technology*, Tianjin, China, August 1994.

[28] H. Kitano. Robocup as a research program. In *Proceedings of IROS-97*, Grenoble, 1997.

[29] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. Robocup synthetic agent challenge 97. In *Proceedings of IJCAI'97*, pages 24–29. Nagoya, Japan, 1997.

[30] G. Klir and T. Folger. *Fuzzy sets, uncertainty, and information.* Prentice-Hall, 1988.

[31] K. Konolige, K.L. Myers, E.H. Ruspini, and A. Saffiotti. The Saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):215–235, 1997.

[32] Y. Lespérance. *A formal theory of indexical knowledge and actions.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, ON, January 1991.

[33] Y. Lespérance and J. Levesque. Indexical knowledge and robot action - a logical account. *Artificial Intelligence*, 73:69–115, 1995.

[34] K.F. MacDorman. Grounding symbols through sensorimotor integration. *Journal of the Robotic Society of Japan*, 17:20–24, 1999.

[35] E. Prem. Dynamic symbol grounding, state construction and the problem of teleology. In Mira J. and Sandoval F., editors, *From*

Natural to Artificial Neural Computation, Proceedings International Workshop on Artificial Neural Networks, pages 619–626. Springer Verlag, 1995.

[36] D.A. Reece and S.A. Shafer. Control of perceptual attention in robot driving. *Artificial Intelligence*, 78:397–430, 1995.

[37] E. H. Ruspini. On the semantics of fuzzy logic. *International Journal of Approximate Reasoning*, 5:45–88, 1991.

[38] B. Russell. On denoting. In *Mind XIV*, pages 479–493. 1905.

[39] A. Saffiotti. Pick-up what? In C. Bäckström and E. Sandewall, editors, *Current trends in AI Planning*, pages 266–277. IOS Press, Amsterdam, Netherlands, 1994.

[40] A. Saffiotti. The uses of fuzzy logic in autonomous robotics: a catalogue raisonné. *Soft Computing*, 1(4):180–197, 1997.

[41] A. Saffiotti. *Autonomous Robot Navigation: a fuzzy logic approach.* PhD thesis, Université Libre de Bruxelles, IRIDIA, Bruxelles, Belgium, 1998.

[42] A. Saffiotti, K. Konolige, and E. H. Ruspini. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*, 76(1-2):481–526, 1995.

[43] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Wötz. Integrating vision based behaviours with an autonomous robot. In *Proceedings of the International Conference on Vision Systems (ICVS99)*, pages 1–20, Las Palmas de Gran Canaria, Spain, 13-15 January 1999. Springer Verlag.

[44] S.C. Shapiro. Embodied cassie. In *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium*, pages 36–143. AAAI Press, Menlo Park, CA, 1998.

[45] M. Shoppers and R. Shu. General indexical-functional reference. In *Proceedings of AAAI-96*, pages 1153–1159. Portland OR, 1996.

[46] M. Shoppers and R. Shu. An implementation of indexical-functional reference for the embedded execution of symbolic plans. In *Proceedings DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 490–496. Portland OR, 1996.

[47] D. Subramanian and J. Woodfill. Making situation calculus indexical. In *Proceedings of KR-89*, pages 467–474. 1989.

[48] T. Uhlin and J. Eklundh. Animate vision in a rich environment. In *Proceedings of IJCAI'95*, pages 27–35. Montreal, Canada, 1995.

[49] T. Vidal and S Coradeschi. Highly reactive decision making: a game with time. In *Proceedings of IJCAI-99*, pages 1002–1007, Stockholm, August 1999.

[50] S. Weber. A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms. *Fuzzy sets and systems*, 11:115–134, 1983.

[51] P. Wide, A. Saffiotti, and H.H. Bothe. Environmental exploration: an autonomous sensory systems approach. *IEEE Instrumentation & Measurement Magazine*, 2(3):28–42, 1999.

[52] WITAS web page: http://www.ida.liu.se/ext/witas/.

[53] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[54] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

Department of Computer and Information Science
Linköpings universitet

Dissertations

Linköping Studies in Science and Technology

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977, ISBN 91-7372-144-1.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977, ISBN 91-7372-157-3.

No 18 **Mats Cedwall:** Semantisk analys av processbeskrivningar i naturligt språk, 1977, ISBN 91-7372-168-9.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978, ISBN 91-7372-188-3.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System 1978, ISBN 91-7372-232-4.

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980, ISBN 91-7372-387-8.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980, ISBN 91-7372-404-1.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980, ISBN 91-7372-403-3.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981, ISBN 91-7372-414-9.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981, ISBN 91-7372-479-3.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981, ISBN 91-7372-489-0.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982, ISBN 91-7372-527-7.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983, ISBN 91-7372-652-4.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983, ISBN 91-7372-660-5.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation,1984, ISBN 91-7372-801-2.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984, ISBN 91-7372-805-5.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987, ISBN 91-7870-133-3.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987, ISBN 91-7870-183-X.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987, ISBN 91-7870-225-9.

No 174 **Johan Fagerström:** A Paradigm and System for Design of Distributed Systems, 1988, ISBN 91-7870-301-8.

No 192 **Dimiter Driankov:** Towards a Many Valued Logic of Quantified Belief, 1988, ISBN 91-7870-374-3.

No 213 **Lin Padgham:** Non-Monotonic Inheritance for an Object Oriented Knowledge Base, 1989, ISBN 91-7870-485-5.

No 214 **Tony Larsson:** A Formal Hardware Description and Verification Method, 1989, ISBN 91-7870-517-7.

No 221 **Michael Reinfrank:** Fundamentals and Logical Foundations of Truth Maintenance, 1989, ISBN 91-7870-546-0.

No 239 **Jonas Löwgren:** Knowledge-Based Design Support and Discourse Management in User Interface Management Systems, 1991, ISBN 91-7870-720-X.

No 244 **Henrik Eriksson:** Meta-Tool Support for Knowledge Acquisition, 1991, ISBN 91-7870-746-3.

No 252 **Peter Eklund:** An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarchies,1991, ISBN 91-7870-784-6.

No 258 **Patrick Doherty:** NML3 - A Non-Monotonic Formalism with Explicit Defaults, 1991, ISBN 91-7870-816-8.

No 260 **Nahid Shahmehri:** Generalized Algorithmic Debugging, 1991, ISBN 91-7870-828-1.

No 264 **Nils Dahlbäck:** Representation of Discourse-Cognitive and Computational Aspects, 1992, ISBN 91-7870-850-8.

No 265 **Ulf Nilsson:** Abstract Interpretations and Abstract Machines: Contributions to a Methodology for the Implementation of Logic Programs, 1992, ISBN 91-7870-858-3.

No 270 **Ralph Rönnquist:** Theory and Practice of Tense-bound Object References, 1992, ISBN 91-7870-873-7.

No 273 **Björn Fjellborg:** Pipeline Extraction for VLSI Data Path Synthesis, 1992, ISBN 91-7870-880-X.

No 276 **Staffan Bonnier:** A Formal Basis for Horn Clause Logic with External Polymorphic Functions, 1992, ISBN 91-7870-896-6.

No 277 **Kristian Sandahl:** Developing Knowledge Management Systems with an Active Expert Methodology, 1992, ISBN 91-7870-897-4.

No 281 **Christer Bäckström:** Computational Complexity of Reasoning about Plans, 1992, ISBN 91-7870-979-2.

No 292 **Mats Wirén:** Studies in Incremental Natural Language Analysis, 1992, ISBN 91-7871-027-8.

No 297 **Mariam Kamkar:** Interprocedural Dynamic Slicing with Applications to Debugging and Testing, 1993, ISBN 91-7871-065-0.

No 302 **Tingting Zhang:** A Study in Diagnosis Using Classification and Defaults, 1993, ISBN 91-7871-078-2.

No 312 **Arne Jönsson:** Dialogue Management for Natural Language Interfaces - An Empirical Approach, 1993, ISBN 91-7871-110-X.

No 338 **Simin Nadjm-Tehrani**: Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification, 1994, ISBN 91-7871-237-8.

No 371 **Bengt Savén:** Business Models for Decision Support and Learning. A Study of Discrete-Event Manufacturing Simulation at Asea/ABB 1968-1993, 1995, ISBN 91-7871-494-X.

No 375 **Ulf Söderman:** Conceptual Modelling of Mode Switching Physical Systems, 1995, ISBN 91-7871-516-4.

No 383 **Andreas Kågedal:** Exploiting Groundness in Logic Programs, 1995, ISBN 91-7871-538-5.

No 396 **George Fodor:** Ontological Control, Description, Identification and Recovery from Problematic Control Situations, 1995, ISBN 91-7871-603-9.

No 413 **Mikael Pettersson:** Compiling Natural Semantics, 1995, ISBN 91-7871-641-1.

No 414 **Xinli Gu:** RT Level Testability Improvement by Testability Analysis and Transformations, 1996, ISBN 91-7871-654-3.

No 416 **Hua Shu:** Distributed Default Reasoning, 1996, ISBN 91-7871-665-9.

No 429 **Jaime Villegas:** Simulation Supported Industrial Training from an Organisational Learning Perspective - Development and Evaluation of the SSIT Method, 1996, ISBN 91-7871-700-0.

No 431 **Peter Jonsson:** Studies in Action Planning: Algorithms and Complexity, 1996, ISBN 91-7871-704-3.

No 437 **Johan Boye:** Directional Types in Logic Programming, 1996, ISBN 91-7871-725-6.

No 439 **Cecilia Sjöberg:** Activities, Voices and Arenas: Participatory Design in Practice, 1996, ISBN 91-7871-728-0.

No 448 **Patrick Lambrix:** Part-Whole Reasoning in Description Logics, 1996, ISBN 91-7871-820-1.

No 452 **Kjell Orsborn:** On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications, 1996, ISBN 91-7871-827-9.

No 459 **Olof Johansson:** Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4.

No 461 **Lena Strömbäck:** User-Defined Constructions in Unification-Based Formalisms,1997, ISBN 91-7871-857-0.

No 462 **Lars Degerstedt:** Tabulation-based Logic Programming: A Multi-Level View of Query Answering, 1996, ISBN 91-7871-858-9.

No 475 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av hur ekonomiska styrsystem utformas och används efter företagsförvärv, 1997, ISBN 91-7871-914-3.

No 480 **Mikael Lindvall:** An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution, 1997, ISBN 91-7871-927-5.

No 485 **Göran Forslund**: Opinion-Based Systems: The Cooperative Perspective on Knowledge-Based Decision Support, 1997, ISBN 91-7871-938-0.

No 494 **Martin Sköld**: Active Database Management Systems for Monitoring and Control, 1997, ISBN 91-7219-002-7.

No 495 **Hans Olsén**: Automatic Verification of Petri Nets in a CLP framework, 1997, ISBN 91-7219-011-6.

No 498 **Thomas Drakengren:** Algorithms and Complexity for Temporal and Spatial Formalisms, 1997, ISBN 91-7219-019-1.

No 502 **Jakob Axelsson:** Analysis and Synthesis of Heterogeneous Real-Time Systems, 1997, ISBN 91-7219-035-3.

No 503 **Johan Ringström:** Compiler Generation for Data-Parallel Programming Langugaes from Two-Level Semantics Specifications, 1997, ISBN 91-7219-045-0.

No 512 **Anna Moberg:** Närhet och distans - Studier av kommunikationsmmönster i satellitkontor och flexibla kontor, 1997, ISBN 91-7219-119-8.

No 520 **Mikael Ronström:** Design and Modelling of a Parallel Data Server for Telecom Applications, 1998, ISBN 91-7219-169-4.

No 522 **Niclas Ohlsson:** Towards Effective Fault Prevention - An Empirical Study in Software Engineering, 1998, ISBN 91-7219-176-7.

No 526 **Joachim Karlsson:** A Systematic Approach for Prioritizing Software Requirements, 1998, ISBN 91-7219-184-8.

No 530 **Henrik Nilsson:** Declarative Debugging for Lazy Functional Languages, 1998, ISBN 91-7219-197-x.

No 555 **Jonas Hallberg:** Timing Issues in High-Level Synthesis,1998, ISBN 91-7219-369-7.

No 561 **Ling Lin:** Management of 1-D Sequence Data - From Discrete to Continuous, 1999, ISBN 91-7219-402-2.

No 563 **Eva L Ragnemalm:** Student Modelling based on Collaborative Dialogue with a Learning Companion, 1999, ISBN 91-7219-412-X.

No 567 **Jörgen Lindström:** Does Distance matter? On geographical dispersion in organisations, 1999, ISBN 91-7219-439-1.

No 582 **Vanja Josifovski:** Design, Implementation and Evaluation of a Distributed Mediator System for Data Integration, 1999, ISBN 91-7219-482-0.

No 589 **Rita Kovordányi**: Modeling and Simulating Inhibitory Mechanisms in Mental Image Reinterpretation - Towards Cooperative Human-Computer Creativity, 1999, ISBN 91-7219-506-1.

No 592 **Mikael Ericsson:** Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, 1999, ISBN 91-7219-532-0.

No 593 **Lars Karlsson:** Actions, Interactions and Narratives, 1999, ISBN 91-7219-534-7.

No 594 **C. G. Mikael Johansson:** Social and Organizational Aspects of Requirements Engineering Methods - A practice-oriented approach, 1999, ISBN 91-7219-541-X.

No 595 **Jörgen Hansson:** Value-Driven Multi-Class Overload Management in Real-Time Database Systems, 1999, ISBN 91-7219-542-8.

No 596 **Niklas Hallberg:** Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach, 1999, ISBN 91-7219-543-6.

No 597 **Vivian Vimarlund:** An Economic Perspective on the Analysis of Impacts of Information Technology: From Case Studies in Health-Care towards General Models and Theories, 1999, ISBN 91-7219-544-4.

No 598 **Johan Jenvald:** Methods and Tools in Computer-Supported Taskforce Training, 1999, ISBN 91-7219-547-9.

No 607 **Magnus Merkel:** Understanding and enhancing translation by parallel text processing, 1999, ISBN 91-7219-614-9.

No 611 **Silvia Coradeschi:** Anchoring symbols to sensory data, 1999, ISBN 91-7219-623-8.

Linköping Studies in Information Science

No 1 **Karin Axelsson:** Metodisk systemstrukturering - att skapa samstämmighet mellan informationssystemarkitektur och verksamhet, 1998. ISBN-9172-19-296-8.

No 2 **Stefan Cronholm:** Metodverktyg och användbarhet - en studie av datorstödd metodbaserad systemutveckling, 1998. ISBN-9172-19-299-2.