FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Self-Localization and Navigation in Bionically Inspired Agents

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Ivan Šakić

Matrikelnummer 0927619

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ.-Prof. Dr. Dietmar Dietrich
Mitwirkung: Dipl.-Ing. Christian Brandstätter

Wien, 01.08.2014

(Unterschrift Ivan Šakić) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Self-Localization and Navigation in Bionically Inspired Agents

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computational Intelligence

by

Ivan Šakić

Registration Number 0927619

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ.-Prof. Dr. Dietmar Dietrich
Assistance: Dipl.-Ing. Christian Brandstätter

Vienna, 01.08.2014

(Signature of Author) (Signature of Advisor)

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Ivan Šakić
Lassallestraße 6/13 1020 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschlieSSlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Ivan Šakić)

Abstract

The Artificial Recognition System (ARS) project is an ongoing attempt to create a bionic cognitive model for autonomous intelligent agents. The ARS model adopts a holistic approach to perception and high-level reasoning based on drives and emotions that is inspired by the psychoanalytical functional model of the human mind, pushing further the boundaries of the current state of the art and opening new possibilities for solving many problems in the field of artificial intelligence.

The purpose of this thesis is to provide a description of a symbolic landmark-based navigation model that is based on the ARS cognitive model. Utilizing advanced cognitive capabilities provided by the underlying model allows for novel approaches to the problem of agent localization, map construction, and route planning.

The thesis explores various qualitative symbolic approaches to agent localization in an effort to identify the core principles behind this process, thus forming a basic localization capability that can be augmented by additional capabilities as the research progresses. Furthermore, the concept of a hierarchical cognitive map as a mean of spatial knowledge representation is considered, with particular emphasis on the integration into the common knowledge representation of the ARS model, with the aim to organically facilitate high-level spatial reasoning. Finally, the process of route planning is considered based on the integrated cognitive map, exploring the possibilities and advantages of route calculation and selection guided by the decision-making process.

The implementation of the navigation model has been evaluated in a simulated environment using various scenarios and has demonstrated behavior consistent with the aims of the ARS project. In addition, the results of the evaluation have provided a great amount of information that paves the way for further research and development of the navigation model as a part of the ARS model.

Kurzfassung

Im Artifical Recognition System (ARS)-Projekt wird aktuell versucht ein bionisch-kognitives Modell für intelligente autonome Agenten zu entwickeln. Das ARS-Modell nutzt einen ganzheitlichen Ansatz der Wahrnehmung und des logischen Denkens, welcher vom psychoanalytisch-funktionalen Modell des menschlichen Verstands inspiriert ist und auf Trieben und Emotionen basiert. Dies treibt den Stand der Wissenschaft voran und eröffnet neue Möglichkeiten, Probleme auf dem Gebiet der künstlichen Intelligenz zu lösen.

Das Ziel dieser Arbeit ist es, eine Beschreibung des symbolischen Wegpunkt-basierten Navigationsmodells zu erstellen, welches auf dem kognitiven ARS-Modell beruht. Unter Zuhilfenahme der fortschrittlichen kognitiven Fähigkeiten, die das zugrunde liegende Modell bietet, können neue Lösungen zu Problemen wie die Lokalisation, der Kartenerstellung und der Routenplanung der Agenten entwickelt werden.

Diese Thesis untersucht verschiedene qualitativ-symbolische Ansätze der Agentenlokalisation, um die Grundprinzipien dieses Prozesses zu bestimmen. Dadurch kann eine grundlegende Fähigkeit zur Lokalisation hergestellt werden, die durch zusätzliche Entwicklungen erweitert werden kann. Zusätzlich wird das Konzept der hierarchischen kognitiven Karte auf ihre Tauglichkeit als Darstellung einer räumlichen Wissensrepräsentation untersucht. Dabei wird ein besonderes Augenmerk auf die Integration in die Repräsentation des gemeinsamen Wissens des ARS-Modells gelegt, um räumliche Schlussfolgerung auf hoher Ebene zu ermöglichen. Schließlich wird der Prozess der Routenplanung mit Hilfe der integrierten kognitiven Karte analysiert. Dabei werden die Möglichkeiten und Vorteile der Routenberechnung und -auswahl unter Zuhilfenahme des Entscheidungsfindungsprozesses untersucht.

Die Implementierung des Navigationsmodells wurde mit Hilfe einer Simulation unter der Nutzung verschiedener Szenarios evaluiert. Sie zeigte dabei ein Verhalten, das mit den Zielen des ARS-Projekts übereinstimmt. Zusätzlich hat die Evaluierung eine Menge an Daten geliefert, die die Forschung und Weiterentwicklung des Navigationsmodells als Teil des ARS-Modells vorantreiben können.

Acronyms

ARS Artificial Recognition System. 1–4, 13, 17, 25–32, 35, 39, 49–53, 55, 56, 58–60, 76–82

JUNG Java Universal Network/Graph Framework). 59

MASON Multi-Agent Simulator of Neighborhoods (or Networks). 4, 56, 63, 68

SLAM Simultaneous Localization and Mapping. 5, 9, 12, 17–20, 24

SSH Semantic Spatial Hierarchy. 21, 22, 24

TP thing presentation. 28

TPM thing presentation mesh. 28

WP word presentation. 29

WPM word presentation mesh. 29, 50, 52, 53, 59

Contents

CHAPTER 1

Introduction

In a world dominated and shaped by humans, the ability of autonomous agents to traverse and interact such an environment independently is steadily gaining priority. Moreover, since humans shape the environment to suit their needs, perception, mobility, and reasoning above all else, it stands to reason to equip autonomous agents with similar abilities. This would in turn facilitate incorporation of the said agents into the human world by allowing them to function independently of the human input.

An autonomous agent is a system situated within and a part of an environment and acts on it, over time, in pursuit of its agenda [FG97]. As autonomous intelligent agents steadily become commonplace in today's world, their capability of navigating their environment effectively and without external input or help is becoming a very important feature of their design. There have been many successful navigation models developed in the past that perform efficiently and accurately within their respective areas of application, yet lately there has been almost no novelty in approaching the problem; instead, most of the practical research seemed to be focused on improving the already developed methods.

One of the shared features of these models is their level of isolation from the rest of the agent's reasoning system, and many are designed to perform without such system in the first place. While all of them provide feedback to the agent in the form of the current location and the path that needs to be taken, the interaction between the two parts rarely goes beyond that. Such separation is not mandatory, however, if one takes a different approach to the general problem of intelligent behavior.

Artificial general intelligence models aspire to emulate the functionality of the human mind as closely as possible, in an attempt to harness the potential that domain-specific artificial intelligence model do not possess. These models, such as the Artificial Recognition System (ARS) [DFZB09], are still lacking a functional navigation model that utilizes the benefits of a holistic approach to reasoning.

Their symbolic approach to modeling of human-like intelligence poses a great obstacle to successful implementation of such capabilities. None of the contemporary navigation models have been developed with symbolical representation of information and knowledge in mind. Thus, in order to create a suitable solution for models such as the ARS one must take a different approach.

1.1 Problem statement

The aim of this thesis is to conceptualize and develop a navigation model for the ARS. This model is to be responsible for determining the location of the agent in the environment and navigating the said environment by comparing the sensory input with the internal representation of the environment, assisted by the existing functionality of the ARS model. To achieve this goal several features of such module have been postulated:

- functionality - localizing the agent within the environment with acceptable accuracy and to calculate or retrieve the proper route to the current goal
- efficiency - performs within acceptable time and memory constraints
- integration - shares the same knowledge base, input, and is itself a vital part of the decision-making process
- intuitivity - performs in a simple manner that is inspired by human behavior and therefore easily understood by humans
- extensibility - designed to allow growth and improvement along with the rest of the system

Functionality and efficiency are self-explanatory features that are necessary for any usable model in general. In particular, it is important to strike a proper balance between the two, so that the resulting model is simple enough to allow efficient computation, yet powerful enough to perform under set conditions in given environments. Simplicity of the model is also governed by the intuitivity feature explained below.

Integration is the crucial feature that separates approach used in this model from approaches used in other available models. By allowing deeper integration with ARS one can utilize its advanced reasoning and decision making to develop novel approaches and methods that otherwise would not be available or feasible.

Intuitivity serves as a guideline that leads closer to the human spatial reasoning by identifying and favoring human-like behavior among the set of considered approaches in any part of the model. This feature acts as a safeguard against overstepping the psychoanalytical limits imposed by the ARS.

Extensibility is a crucial feature of the model due to the current state of the ARS implementation. The implementation is still largely a work in progress [BDM$^+$13], with several features this model depends on either missing or only partially developed. In order to prevent major redesigns and modifications of the model as the implementation of ARS progresses further, great emphasis will be placed on identifying those features early on and designing the model around them, providing proper abstractions and following the interfaces defined by the model of ARS. Comments and recommendations regarding the implementation of these features based on the result of the thesis will be provided in the future work section of the thesis (see section 6.6).

The model will be developed by firstly researching different state-of-the-art approaches and identifying features conceptually compatible with the inner workings of the ARS; specifically, its knowledge and information representation, and the decision making process.

Additionally, once the model is properly developed an implementation of the model will be made as a proof of concept, in order to ensure the performance of the model within a simple test environment. This should be implement to correspond as closely as possible to the interfaces defined by the ARS model, in order to eliminate any problems that might arise due to incompatibility of the models.

1.2 Methodology

In order to develop an applicable and scientifically satisfactory model, several rules meant to help properly guide the research and development towards the desired goal have been defined:

Rule 1. The resulting model should attempt to model human spatial cognition as close as possible within the given parameters of the thesis and particular test scenarios.[1] This means that any existing method or process used to accomplish a certain task must be discarded if there is no sufficient evidence these are available to a healthy human being without any external influence. Sufficient evidence is obtained from published and peer-reviewed research provided by the experts in a given field. If no definitive research is found, processes should be designed to adhere to the intuitivity feature of the model as described in the previous section, preferably by emulating apparent human behavior.

Rule 2. Any feature deemed currently impossible to model or implement according to rule 1 will be modeled to serve as an intermediate solution until the situation allows for a better or a more fitting solution to be developed.

Rule 3. The implementation is a proof of concept. Relating to the rule 1 above, the model will disregard any optimization of processes it contains, even when those might come to be as a result of human practice or expertize regarding the particular task. The implementation of the model should at least be able to provide valid results, not necessary optimal ones.

Rule 4. The model presented in this thesis is designed specifically with ARS architecture in mind. This means that any input and output must be compatible with architecture's declared

[1] Parameters refer to the current state of the environment and the agent's internal state, as well as any conditions defined by a test scenario.

interfaces and data types as stated in its model. Any deviations must be identified and properly documented.

Rule 5. The scope of the thesis limits the complexity and details of the model. Any novel approaches or possibilities discovered that fall outside of the scope of the thesis should be mentioned in order to provide a reference for future work. The actual parameters defining the scope of the thesis are mentioned in chapter 4.

Rule 6. Tools that are used to develop the ARS implementation and the associated simulation will be used in this thesis as well. This includes the choice of the programming language (Java) and the choice of the simulation suite, Multi-Agent Simulator of Neighborhoods (or Networks) (MASON). MASON simulation suite is further described in chapter 5.

1.3 Structure of the work

The structure of this thesis contains 6 chapters.

Chapter 1 gives an introduction into the problematics of navigation in autonomous intelligent agents. It also provides the problem statement, and an overview of methodology used and principles followed during research and development of the solution.

Chapter 2 presents an overview of general concepts relevant to the topic of navigation for autonomous agents. It defines concepts such as localization, route planning, and mapping processes, as well as describing their dynamics and interactions. This provides a theoretical foundation needed for understanding the rest of the thesis matter.

Chapter 3 gives an overview of the current state of the art pertaining to the navigation in autonomous agents, as well as deeper look into the ideas and concepts behind the ARS model, which represents the infrastructure of the model developed in this thesis, in order to provide justification for certain design choices made when creating the model.

Chapter 4 provides a more specific description of conceptual foundation of the developed model and gives a detailed description of the model itself. Here the design choices are described in detail and the proposed integration into the ARS model is provided.

Chapter 5 describes the proof-of-concept implementation of the developed model, along with the tools, either already existing or developed, that were used in its realization.

Chapter 6 presents the results obtained through testing of various processes of the model performed on its implementation, along with their interpretation. It also contains the conclusion and the discussion of future work.

CHAPTER 2

Concepts

The problem of navigation in autonomous agents is a broad one, with a large number of concepts and definitions that aren't easy to grasp, due to many different approaches one can take based on other fields of science – neurology, cognitive science, psychology, to name a few – which have different concepts and definitions of phenomena. This chapter will therefore provide a clear explanation of principles, concepts, and definitions that are essential for general understanding of the topic of navigation in autonomous agents, and understanding of the navigation model described in chapter 4 in particular.

2.1 Navigation in autonomous agents

Navigation is a process that allows an autonomous mobile agent to move through and interact with its environment it in a purposeful and intelligent way. This process necessitates that the agent be capable of perceiving the environment and moving with sufficient degrees of freedom. In the rest of the thesis the term autonomous agent will assume these capabilities unless explicitly stated otherwise.

When observed as a black box, navigation takes as input agent's perception and a goal, giving as output a description of the action that needs to be executed to reach that goal. Internally, navigation is a rather complex process that can be conceptually divided into three processes: *localization*, *mapping*, and *route planning*. These processes are separated mostly theoretically for the sake of clarity and are in most models integrated in one way or another.

The prevalent type of navigation model today, in theory and in practice, combines the localization and mapping process into a single process which performs both functions concurrently. This model is commonly referred to as the Simultaneous Localization and Mapping (SLAM), and is further explained in the chapter 3. Nevertheless, these two processes will be described separately as the navigation model of the thesis distinguishes between for reasons that will be described in chapter 4.

Some navigation models eschew the mapping process entirely, consisting only of localization and navigation processes. This is possible, and in most cases preferable, when the agent possesses *a priori* knowledge of the environment provided before deployment. This is usually done in cases of largely static environments where any changes in the map are very unlikely; this is done in order to reduce complexity of development and operation. Some models posses limited mapping capability such as online map updates in rare cases of environmental change, but since alterations of the map are done externally, they would not be categorized as true mapping processes.

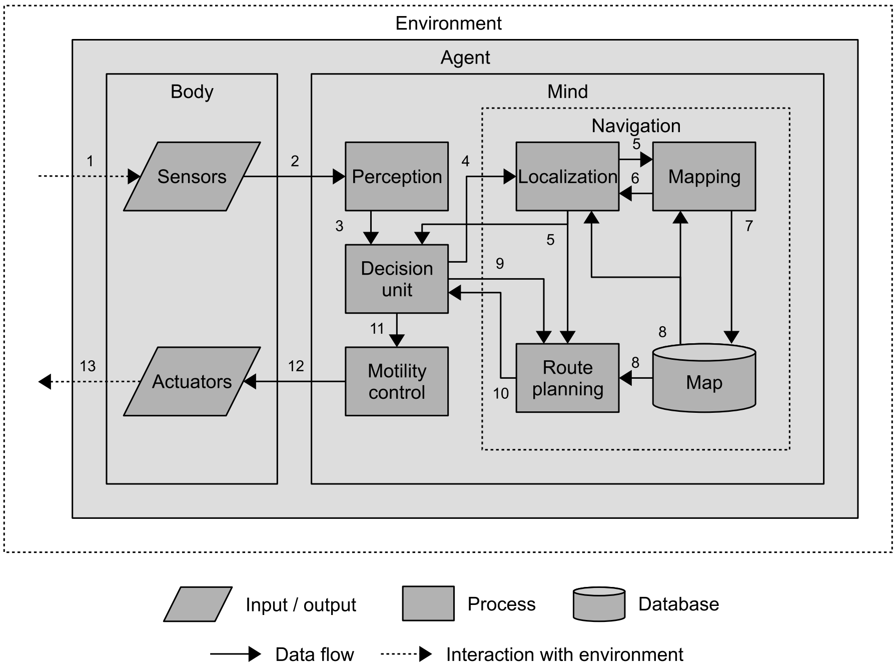


Figure 2.1: Navigation model as a part of an autonomous agent.

Figure 2.1 depicts the navigation model integrated into a mind of an autonomous agent. The figure depicts the concept of intelligent navigation by showing interaction of processes within the agent's mind. The numbers represent concepts as follows: 1 – environmental features, 2 – raw sensor data, 3 – perceptual input, 4 – relevant perceptual input, 5 – current location, 6 – new location, 7 – map update, 8 – map data, 9 – destination, 10 – route, 11 – actuator instructions, 12 – action (motion). The rest of this chapter will describe processes involved, their functionality and interaction, as well as the principles of representing and managing the spatial knowledge.

2.2 Spatial knowledge representation

In order for any intelligent agent to be able to reason about the environment, it must have an internal representation of that environment. The concept of such spatial knowledge representation comes from the field of cognitive sciences, first mentioned in 1948 in a paper by Tolman [Tol48], where it was referred to as a *cognitive map*, and has since been the subject of extensive research. Despite that, a general consensus regarding its definition hasn't yet been reached. As this concept has over time become an important part of many other fields of science, such as geography, urban planning, and psychology. A paper by Kitchin [Kit94] identified four different viewpoints that govern its interpretation with respect to human cognition:

- Cognitive map is a map - as hypothesized by O'Keefe and Nadel [ON78] based on their experiments on rats, the region of brain called *hippocampus* is a three-dimensional, euclidean model of the world with rigid geometrical properties.
- Cognitive map is analogous to a map - the cognitive map has map-like properties. While there might not be a specific region of the brain the environment is mapped onto, there is a correspondence between input-output behavior of the storage and retrieval functions of the two representations.
- Cognitive map is a metaphor of a map - based on the belief humans act as if they had a map in their minds. This interpretation provides no detail regarding the physical manifestation of the map.
- Cognitive map is merely a hypothetical construct - the term map has no literal meaning, and is used to represent a set of processes which are believed to exist and affect everyday spatial behavior of humans. This supports the non-euclidean spatial products that occur in cognitive maps such as intransitivity and asymmetry.

This disagreement about the definition caused by the lack of conclusive evidence is carried over into the field of artificial intelligence, resulting in different approaches to the modeling of this concept. The same paper touches upon these computational models, identifying two main types of computational models models: *cognitive models*, and *bionic (connectivist) models*. Cognitive models are centered around memory structures and information processing, whereas bionic models are centered around behavior of neurons and neural networks. Both types of models attempt to obtain outcomes that mimic human behavior. However, where cognitive models usually adopt cognitive science approaches in an attempt to model *perceived* human mental processes, bionic models search for direct links between the environment and the behavior instead, and focus on emergent behavior [Kit94].

Spatial knowledge representation is the part of the agent's knowledge base that contains the total knowledge about the environment, regardless of its actual structure or the types of knowledge representation. This means that it might consist of several unrelated data structures that are distributed across the system, using different data types and having different interfaces. The definition of the spatial knowledge representation is stated in such way in order to guide the process of design towards a fully integrated, common knowledge base, preferably integrated into the general knowledge base in a meaningful and useful way.

There exist many terms today to refer to this concept, such as abstract map, cognitive configuration, cognitive schema, environmental image, spatial representation, to name a few [Kit94]. The term *cognitive map* will be used to refer to this concept throughout this thesis for the sake of brevity, referring to the cognitive computational model of the spatial knowledge representation in particular, unless stated otherwise.

2.2.1 Mapping

Mapping is a term that describes the various kinds of interaction with the cognitive map. A more general definition of mapping describes it as “a process composed of series of psychological transformations by which an individual acquires, stores, recalls, and decodes information about the relative locations and attributes of the phenomena in their everyday spatial environment” [DS74].

There are two main ways of interacting with the spatial knowledge representation, regardless of the taken approach to modeling: knowledge retrieval and knowledge update. Both can occur either through direct manipulation of the data structures comprising the knowledge representation, or through a knowledge management system that provides a functional interface to the navigation process.

Knowledge retrieval describes a set of operations that facilitate the navigation process by providing necessary knowledge about the current state of the environment, as stored within the spatial knowledge representation, to the underlying processes. This involves several operations:

- Retrieval of information about current location
- Retrieval of encodings of locations based on arbitrary criteria
- Retrieval of parts of the environment and its structure
- Retrieval of known routes

Knowledge update describes a set of operations that modify the current knowledge of the environment based on exploration of unknown parts of the environment, and perceived changes in the known part of the environment. Additionally, should the spatial knowledge representation be subject to certain logical or structural constraints, knowledge update may also pertain to changing the internal structure of the spatial knowledge representation in order to satisfy those constraints. Within knowledge update two kinds of processes can be discerned.

First process is termed *map learning*. As the concept of “an unknown part of the environment” can be defined as a part of the environment that has no representation within the cognitive map, an agent may assume it is located within an unknown part of the environment when it cannot localize itself, that is, find a known location that matches the perceptual input, with sufficient certainty. In other words, learning is never initiated as long as the agent stays within the known part of the environment. Map learning process is dependent on the results of the localization process, and this is the reason they are commonly integrated into a single process, as is the case with the SLAM navigation models.

Second important process is *map maintenance*. This process focuses on the cognitive map, ensuring the representations of parts of the environment are up-to-date and valid, as well as maintaining the overall structure, if one exists.

Representations of parts of the environment consist of environmental features and their configuration, which describe that particular part. In real world environments these can often change: features might change appearance, appear or disappear, and their configuration might change. Such changes can be highly detrimental to proper execution of the localization process, and as such must be handled properly.

It may occur that a certain location changes beyond recognition, where it would be completely unrecognizable to the agent. As there would be no accurate localization result, map maintenance could not be performed. In such cases, knowledge of the last known location(s) in conjunction with the map structure could be used in an attempt to ascertain the actual current position of the agent.

Changes in the environment can also affect the traversability of the environment, by blocking or destroying previously known routes. These changes are also updated within the map when necessary.

Some cognitive maps may have constraints placed on their structure in order to ensure algorithms that are performed on it can function properly. Map maintenance verifies this whenever a change occurs in order to maintain the proper structure of the map.

2.3 Localization

The term *localization* refers to an agent's process of determining where it is located within its environment [WFJL08]. The result of this process is referred to as the *current position* or *current location* of the agent. In cases where the environment is represented by values within a coordinate system, and where the agent's rotation is important, the result is referred to as *pose*, which is a joint term for the current position and rotation of the agent.

The crucial aspect of localization is the solving of the correspondence problem. The *correspondence problem* pertains to finding out which part of the environment stored in the memory best corresponds to the current sensory input. There are four factors that must be taken into account when designing an effective localization solution:

- Feature representation - Which features are made explicit in the map? (sensor reflection points, extracted feature points, . . .)
- Representation of configurations - Which spatial relations are made explicit in the map? (qualitative knowledge, metric data, . . .)
- Spatial reasoning / configuration matching - Which matching algorithm is used? (Iterative Closest Point, shape matching, . . .)
- Temporal reasoning - How is history information handled? (stochastic estimators, conceptual neighborhoods, . . .)

Feature representation governs the interpretation of the sensory input. It specifies which features present in the environment are to be detected, extracted, and stored by the model. The choice of features depends primarily on the type of sensors used and the algorithms used for their extraction. Commonly selected features are landmarks, paths, and free space. Landmarks in particular form the foundation of the model described in this thesis and are addressed in section 2.5 in greater detail.

Representation of configurations describes the types of configurations features. A configuration describes spatial arrangements of perceived features in the environment. Configurations can be separated into two main categories. *Qualitative configurations* employ a finite set of relations to model spatial information in an abstract way. These relations usually describe the position of two features relative to one another or to a third reference point, abstracting vector properties such as distance or direction. Relations such as “north of”, “left of”, “behind”, “near” are examples that belong to this category. *Quantitative configurations* employ no such abstractions apart from reduction in resolution, using absolute and uniform scales to describe position of features. As with qualitative representations, these can be expressed as relations between two features, such as exact distance, or coordinates/vectors within a coordinate system.

Spatial reasoning, also termed *matching*, is a process that establishes correspondence of the perceived features and their configurations with the features and configurations in the internal representation. There are three main problems pertaining to this factor: obtaining a feasible solution, handling uncertainty, and integrating spatio-temporal knowledge.

Feasibility pertains to the amount of data that needs to be processed, as well as the complexity of the algorithms involved. The matching process is quite susceptible to the *combinatorial explosion* [Vel08], where extreme amount of input data prevents processing within a reasonable time. This complexity problem arises when observing objects in the environment that have minute perceived differences that depend from the viewpoint or the particular instance of the object, which would normally be encoded along with the rest of the object, thereby leading to an explosion in numbers of encoded representations of objects within the knowledge base. To prevent this, incoming data may be *filtered* to remove less-valuable data. Additionally, *heuristic approaches* can also be used, such as removing the features that are *expected* to be invisible to the agent based on its pose. The choice of these methods again depends on the overall model. Regardless of the approach, the model must be able to tolerate uncertainty caused by false negatives[1] as well. In short, one should attempt to *obtain usable results whilst minimizing the computation costs*, and the selection of the features and the algorithm must reflect this principle.

Uncertainty is inherent in processing real-world data, which makes perfect correspondences almost impossible to obtain. This means that any kind reasoning an agent might employ for self-localization purpose must be able to tolerate uncertainty in some way. Many solutions employ various stochastic models of the environment to account for those uncertainties, employing approaches such as Markov models and Monte Carlo localizations (see related work in chapter 3).

Spatio-temporal reasoning combines spatial with temporal and causal information to facilitate more accurate self-localization. By taking into account consequences of agent's movement through space as well as stored information about previous traversals one can predict incoming features as well as account for their absence at a certain point in time. Spatio-temporal reasoning is often reinforced by stochastic models mentioned above. This type of reasoning is particularly useful in self-localization models that are more dependent on the accurate estimates of agent's pose.

2.3.1 Current position

Current position represents the agent's knowledge or belief about its relation to the environment at a certain moment. It is the result of the self-localization process and is needed by the remaining two processes, and, depending on the model, the self-localization process itself as a system feedback. This makes it the most crucial result of the overall process.

Its actual representation in the model is not strictly defined, and depends on the nature of the cognitive map. In case of a location-based model it is usually a memory pointer or an object reference to the location object in the cognitive map that is currently considered current, or its identified. In case of coordinate-based models it can be a set of coordinates or a whole pose of the agent. Whether this piece of information is stored in the common storage or passed via interfaces depends on whether it can be used elsewhere in the system that utilizes the model.

In case of a location-based model current position is instead referred to as *current location*, differentiating it from the current position, which instead refers to the actual position in the environment relative to some external coordinate system the agent itself is unaware of, such as the coordinates within a simulated environment.

2.4 Route planning

Route planning is a process of calculating a route between two distinct locations in the environment. The various definitions of a *route* can be summarized into a static way from a starting

[1] False negatives in this case pertain to useful features being falsely recognized as less-important and thus filtered out.

position to a target place [WKBM^{+}97]. In the context of autonomous agent, particularly the one pertaining to this thesis, a route can be more broadly defined as a *sequence of actions* that an agent needs to perform to reach given destination from its current location. This definition considers the fact that any calculated route, regardless of its encoding, is ultimately translated into a sequence of actions during the traversal. The definition is a proper generalization of commonly accepted definition of a route, as any trajectory derived from the encoding, such that can be visualized with a line in a multidimensional space, also assumes *movement* along it in a certain direction. This definition is also supported by the way humans convey routes to each other, most commonly by describing it with actions that one should perform to reach the intended destination, instead of describing the route statically in the sense of the first definition.

It also facilitates a more complex description of connections between distinct locations in the environment by allowing non-symmetric connections, that is, when a route from location B to location A cannot be obtained by inverting the route from A to B. The necessity of such definition becomes clear in section 4.5, where actions are defined with respect to the developed navigation model.

All navigation models, regardless of the choice of the spatial representation, ultimately segment the environment into parts that are functionally equivalent to locations as defined in this thesis. Even if these parts are encoded as ends of known routes, they still represent unique places of interest in the sense that is covered by the term location and can be subsumed under it. In case of models such as Simultaneous Localization and Mapping (SLAM), described in section 3.1, which generate a topographical spatial representation, various algorithms, such as algorithms based on generalized Voronoi graphs [CN01] can be employed to obtain non-colliding trajectories in a multidimensional spatial representation which can be used for route computation.

Further expanding on this idea, one can assume that all spatial representations that are capable of facilitating navigation can ultimately be abstracted into a structure equivalent to a graph, at least when considering global navigation (section 4.1). This allows the route planning problem to be considered in terms of graphs and graph-based algorithms, where well-established routing algorithms exist, such as A^* or other variations of Dijkstra's algorithm, that can be applied in solving the problem.

2.4.1 Route evaluation

Route planning as described above refers to finding any valid route, without any constraints or scores placed on the result. If such constraints or scores do exist, they usually give an estimate of the quality of a route, and it is therefore in one's best interest to calculate the feasibly best route available by factoring these into the route planning algorithms.

Historically most relevant and widespread of these factors is the length of a route, or the distance that needs to be traveled towards a destination. The well-known Dijkstra's algorithm was developed specifically to solve the shortest route problem. In real world scenarios of human navigation, which rarely involves exact calculation, the problem much more complicated as the set of factors grows significantly. Factors such as safety, convenience, familiarity, as well as a complex interaction of the internal state of the agent, and the currently perceived and recalled state of the environment play a predominant role in route evaluation and selection. With ongoing development of artificial general intelligence models such as ARS, these factors and their benefits are becoming increasingly more important to include in any reasoning process, including navigation.

2.4.2 Route encoding and retrieval

As it is the case with any other process whose results are often reused, routes that result from the route planning process are commonly stored for later use, as the retrieval of stored results is always preferable to repeating the process. *Route encoding* is the process of storing discovered or learned routes into the agent's spatial representation. It involves storing information about the route in a manner so that it can be later sufficiently reconstructed and translated into sequences of movement actions. It can involve storing exact displacement and rotations, which is used in models that rely on geometrically accurate spatial representations, or it can use environmental features as anchoring points for movement actions, used in feature-oriented models with topographical spatial representations.

Route retrieval deals with obtaining stored routes according to the agent's current goal. The most important factor that influences the retrieval process is the *indexing* of stored routes. During directed navigation, agent has knowledge of its current location, provided by the localization process, and the destination location, provided by the decision-making process. These two pieces of information are analogous to the beginning and the end of a route, either known or unknown, making them the natural choice for indexing of encoded routes. This is the most common approach to indexing, allowing for computationally efficient retrieval of routes. More than one route can be indexed in such manner, in which case a decision on which route is to be taken must be made.

2.5 Landmarks

Section 2.3 mentioned environmental features as a critical factor in localization. Since this thesis primarily deals with landmark-based navigation, landmarks as feature of the environment will be discussed here in detail.

The concept of landmarks is a rather broad one, and can have quite different meanings depending on the context. In order to provide a clear and concise definition pertaining to the navigation problem, in which landmarks primarily have a role of a navigation tool, Sorrows et al. [SH99] have provided the following definition:

A *landmark* may be any element in an environment that is external to the observer and that serves to define the location of other objects or locations. A landmark may have particular visual characteristics, a unique purpose or meaning, or be in a central or prominent location that makes it effective as a landmark.

The most crucial concept pertaining to landmarks is *salience*. Salience, also referred to as *singularity*, is the state or quality of an object by which it stands out relative to its neighbors. Salience is not necessarily solely a visual property of a landmark. The same article by Sorrows et al. divides landmarks into three categories based on the type of the salience: [SH99, p. 45–46]

- *visual landmarks*, which are considered landmarks due to their visual characteristics, such as singularity and prominence described above, and are considered most important type of landmarks in physical environments
- *cognitive landmarks*, which stand out due to their meaning, usually of some personal importance and therefore not universally perceived as landmarks
- *structural landmarks*, which are important due to their role or position in structural organization of the environment, such as intersections or town squares.

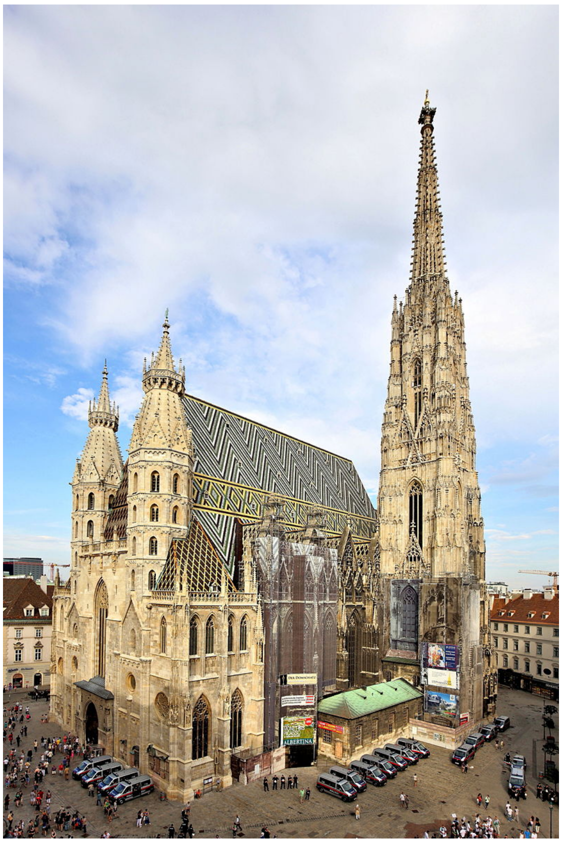


Figure 2.2: St. Stephen's Cathedral is an example of a highly salient landmark. It stands out from its surroundings by its size and noticeably different and rich gothic architecture [Bwa].

Landmarks can belong to more than one of the above mentioned categories. In reality no landmark or landmark category is universally salient, that is, there is no landmark that is salient in and of itself without its context. For example, a skyscraper such as the DC Tower One in Vienna is an extremely salient landmark, as the number of skyscrapers in Vienna is very low, but if it were located in an environment such as Manhattan, New York, where there is an abundance of skyscrapers, its salience would be significantly lower. Same reasoning can be applied to a house in a field versus a house in a city. Having sufficient ontological knowledge of the environment and its features, as well as the ability of high-level reasoning at disposal, one can discern which types of landmarks that are usually salient and use them for localization and location encoding whenever possible.

2.5.1 Landmark recognition

Landmark recognition is usually a task delegated to the agent's perception processes. Since the navigation model directly depends on proper recognition of landmarks makes the navigation usually tightly coupled with the perception. Landmark recognition primarily involves processing visual sensory input and matching it to existing encoded patterns in the long-term memory The processing itself involves image processing and pattern matching methods, in conjunction with disambiguation facilitated through reasoning.

2.5.2 Filtering in dynamic environments

Real world environments, as opposed to environments created for domain-specific purposes, are dynamic and constantly changing. In spite of great advances made in the field of localization and mapping for autonomous agents, published solutions still have at most very limited ability to cope with the complete range of changes that may occur in any real world environment.

If a solution uses a landmark recognition strategy based on salience, which is the most rational and widely used option [WRN04, SH99], it may still recognize transient or changing objects as such and incorporate them into the cognitive map. Objects such as vehicles, construction sites, advertising boards, and trees are all examples of objects that may posses high salience in their context, but are not eligible as landmarks due to some or all mentioned reasons. This necessitates a filtering strategy that will recognize these objects as ineligible and prevent their integration into the internal spatial representation. Such strategies should be based on pattern matching, object categorization and some level of reasoning based upon them.

The ability to recognize, classify, and filter perceived objects and entities allows for simpler implementation of localization and navigation routines, as the most complex issues regarding them are assumed to be satisfactorily handled by other parts of the system.

2.5.3 Landmark specificity

Landmarks do not need to be perceived as globally unique; on the contrary, many landmarks may share the same description across different scenes. Specificity of a landmark is not a constant and can change as new information about it is provided, or the known information is lost.

Based on the availability of the landmarks in the agent's current surrounding an object will be recognized as a landmark depending on its relative usefulness to other perceived landmarks. Landmarks that posses insufficient salience, such as one of many similar looking houses, should be ignored during localization. There exist an abstraction limit below which no object are admissible as landmarks for navigation purposes. Such objects are still important in the obstacle avoidance task, which is a part of the local navigation, which is functionally distinct from global navigation and therefore not covered by this thesis.

A landmark can move forward on the specificity spectrum as soon as any information is obtained that would give it more significance for the purpose of localization. A landmark loses it's value the more it approaches the lower extreme, as the number of objects in the environment corresponding to it increases.

While the usefulness of less specifically recognized landmarks as localization tools might be diminished, they are still very important navigation tools, as actions use landmarks as reference points. Actions and their anchoring points are discussed in section 4.5.

CHAPTER 3

Related work

The model developed during this thesis is inspired by previously developed ideas and solutions in the field of self-localization and navigation for autonomous intelligent agents. This chapter describes some of the most successful, as well as most relevant approaches in the field and thus provide a broader picture of the current state of the art.

Additionally, the background of the Artificial Recognition System (ARS) project will be described. An ongoing project in the field of the artificial general intelligence, ARS itself necessitates an appropriate solution for self-localization and navigation functionality, which is the main topic of this thesis.

3.1 Simultaneous localization and mapping

Simultaneous Localization and Mapping (SLAM) is by far the most prevalent approach to solving the problem of robotic navigation today. Originally developed by Hugh Durrant-Whyte and John J. Leonard [LDW91] based on earlier work on stochastic map-based localization by Smith, Self and Cheeseman [SSC87], SLAM is concerned with the problem of building a map of an unknown environment by a mobile robot while simultaneously navigating the environment using the same map. The models using this approach are supposed to be able to function without any previous knowledge about the environment.

The SLAM approach consists of several routines or processes: landmark extraction, data association, state prediction, state update and landmark update. Since there are no strict rules imposed on their design or functionality, there are many ways to solve each of the processes, which has resulted in a variety of different SLAM models which are either an improvement of an older model, or tailored to a particular purpose or environment type. Landmarks are also not precisely defined and can be any salient feature of the environment.

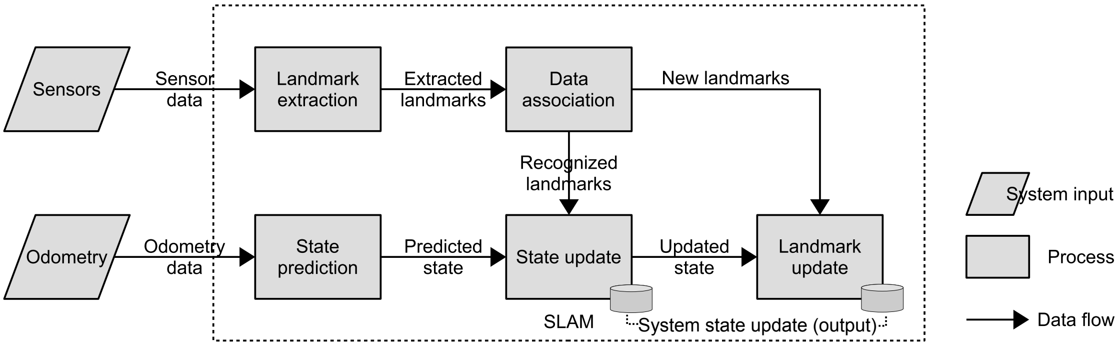


Figure 3.1: Diagram of SLAM subprocesses

3.1.1 Senors and odometry

In order to diminish the effect of accumulated errors over time SLAM models often rely on several different types of sensors [MB13] to acquire data with statistically independent errors. This is necessary in most cases, as this method is otherwise very susceptible to accumulated error. Commonly used sensors are laser sensors, radars and sonars, but sometimes less commonly used sensors such as temperature sensors are used as well.

Odometry data is used to provide an approximate pose of the agent. It is obtained by tracking the motion of motility actuators. Unfortunately, it is also a source of errors, so it alone is not sufficient to facilitate reliable localization. The odometry data and the sensor readings should be obtained simultaneously whenever possible, in order to prevent inconsistencies resulting from displacement that may occur in between.

3.1.2 Landmark extraction and association

Processing sensor input in SLAM involves recognizing and encoding the landmarks in the environment. Landmarks are expected to be easily detectable from any or most angles and above all else static in nature. They are encoded by their position in the environment and stored in an array. Because of this they cannot be qualitatively distinguished from one another, which necessitates the whole array be updated along with the agent's pose at every step. They are also subject to uncertainty, as a slightly displacement of a landmark on revisit is always expected, caused by errors in odometry, and should not be recognized as a new landmark. This uncertainty is also tracked and updated continuously by the state update process.

Landmarks are first extracted from the sensor data. Landmark extraction largely depends on what types of landmarks are extracted as well as on what kinds of sensors are used. For example, spike landmark method is used for extraction of point landmarks, while the Random Sampling Consensus (RANSAC). method is used to extract line landmarks.

Once all landmarks have been extracted, they need to be cross-referenced with the landmarks stored in the map in order to determine if any of the currently extracted landmarks have been observed before, and if so, to associate them properly. One of the methods to do this is to pair every extracted landmark with the stored landmark from the map nearest to it. Every pair then undergoes verification by determining if the extracted landmark lies within the area of uncertainty of the stored landmark, which is calculated by the state update process. If this is true, the extracted landmark is associated with the stored landmark and designated as recognized. If not, it is designated as a new landmark.

3.1.3 State prediction and update

The defining parts of the SLAM approach are state estimation and state update, which rely on the probabilistic filtering methods. The *system state* is usually represented by an array that consists of the current pose of the agent and the positions of all observed landmarks.

The goal of the state prediction is to calculate the expected system state based on the odometry data and the previous state. This is done by applying the translation and rotation determined from odometry on the system state. Since odometry data is always subject to uncertainty, the the error estimate is also extracted, which plays an important role in the next step.

Once the predicted state has be calculated, state update is performed using a probabilistic filter method. These methods calculate and store uncertainty data across subsequent executions of SLAM, updating it continuously based on re-observation data. This data is applied to the system state in order to create probabilistically best possible estimate of the new system state.

The are two types of methods for state estimation that are most commonly used in SLAM systems: methods based on Kalman filter, and particle filter methods. Both of these methods are an example of sensor and data fusion, where the accuracy of sensor readings is improved using the stored data about the environment.

Kalman filter is a statistical method used for estimating position of an object whose movement can be modeled by a linear system. This method can be also applied for detection and tracking of moving objects, especially for separation of moving objects from the static ones. In case of movement modeled by non-linear systems, methods such as the Extended Kalman filter are used instead, which models the movement by approximating it with a linear system [WB95].

Particle filter methods, such as the Rao-Blackwellized particle filter method [SGB05] or the Monte Carlo localization method [TFBD01], employ a larger set of possible poses, called particles, that have probabilities assigned to them which signifies how likely that particular pose is likely to be the correct one, called weights. In the beginning, particles are normally distributed across the known space with equal weights. The weights are then adjusted for each particle based on the match between sensor reading expected in the pose represented by that particle to the actual sensor reading. Particles above a certain threshold are retained, and the others discarded. Upon performed movement the particle set is repopulated based on the retained set by applying the movement to them with the addition of random errors based on the expected error distribution for such movement (see figure 3.2 for an example of population update without removal of particles). This step is important as it allows the method to account for certain errors.

Once the particle step has been repopulated, the process repeats itself.

The main deficiency of this method is that the number of particles can become very large as the map grows. Another problem is the depletion of particles in otherwise viable areas, resulting in the method getting stuck in local minima.

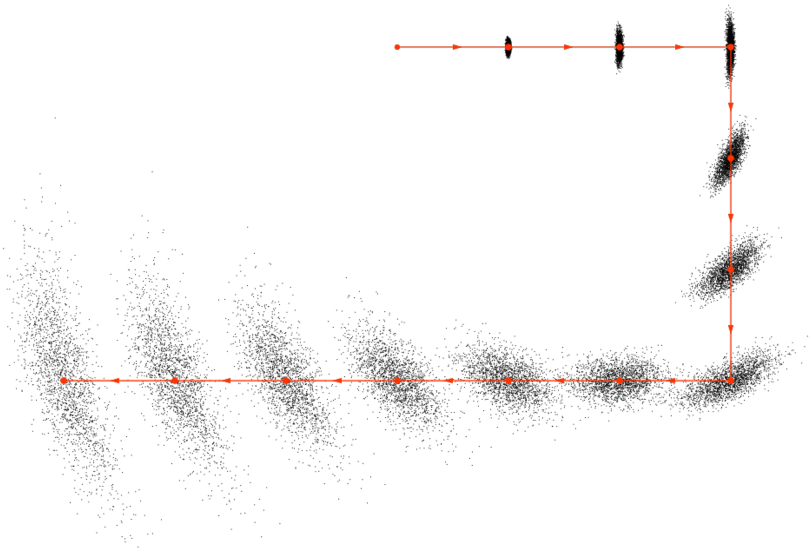


Figure 3.2: A visualization of the particle population in several discrete steps in a Monte Carlo particle model with no state correction [Lu].

3.1.4 Landmark update

Once the system state has been updated, it remains to add the newly observed landmarks to the system state. This is done by expanding the system state array with positions of new landmarks, along with expansion of filter method matrices that perform state updates with new uncertainty data. This step concludes the SLAM process, which is then restarted upon subsequent movement of the agent.

3.1.5 Closing remarks

As already stated, SLAM approach has found many uses due to its versatility and adaptability and continues to be a productive field of research, mostly focused on improving reliability versus computational efficiency. Unfortunately, it's analytical and probabilistic approach to localization and mapping makes it incompatible with the ARS. This incompatibility is further exacerbated by the fundamental difference in knowledge representation and handling.

3.2 Semantic Spatial Hierarchy

The Spatial Semantic Hierarchy (Semantic Spatial Hierarchy (SSH)) is a comprehensive theoretical model of knowledge of large-scale space developed by Benjamin Kuipers of the University of Texas [Kui00]. Inspired by the properties of the human cognitive map, it models knowledge of large-scale space through multiple interacting representations, which can be either qualitative or quantitative in nature. It can thus serve both as a model of the human cognitive map, and as a method used in robot navigation. The structure of the SSH permits handling of partial knowledge, enabling it to deal robustly with uncertainty during spatial learning and problem-solving.

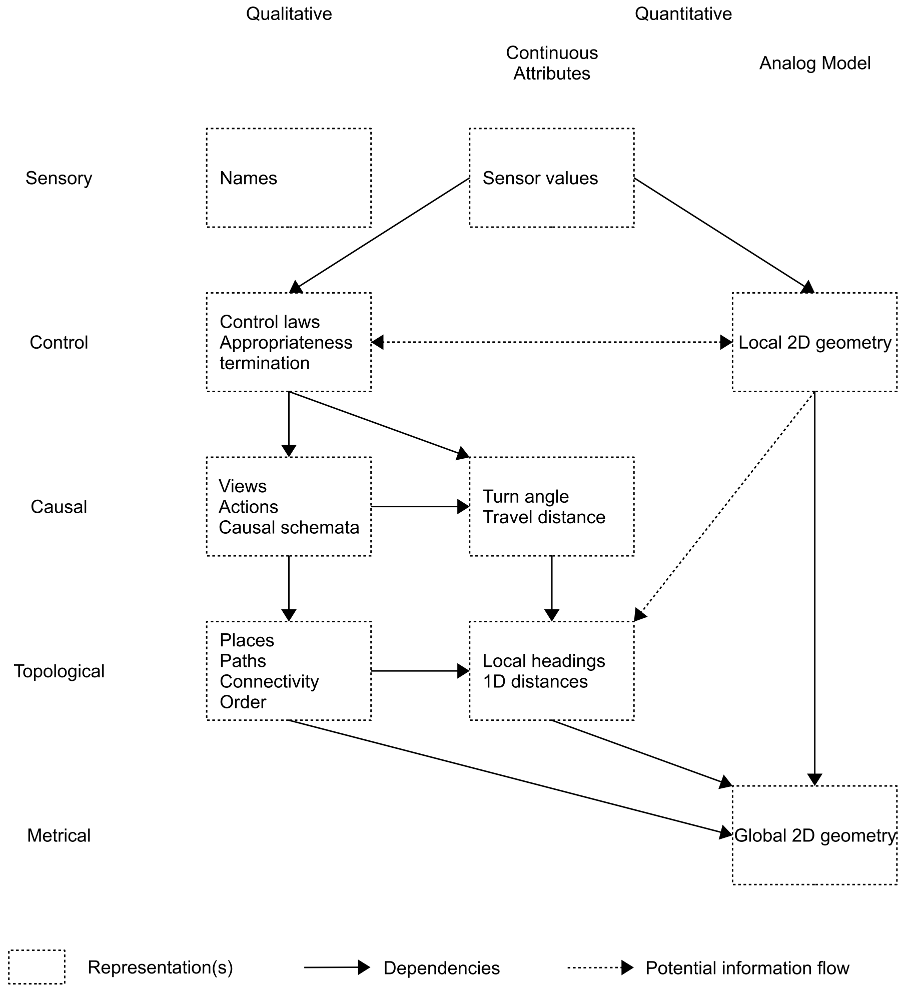


Figure 3.3: Diagram of distinct representations of SSH [Kui00].

Representations in the SSH, their dependencies and relationships are shown in a diagram in figure 3.3, where the particular representations are represented as nodes. Representations are classified as either qualitative and quantitative based on the type of knowledge they represent.

Each representation has an ontology and a set of axioms and rules. The ontology specifies the set of objects and relations that can be represented within a particular representation, while axioms and rules determine what conclusions can be inferred and what actions taken based on the represented knowledge. Representations are grouped into hierarchy levels based on the ontology they employ. Each level further down depends on the knowledge of the previous levels. The information obtained through sensors "trickles" down through the levels of the hierarchy. On each level, different kind of knowledge is extracted and stored, forming a basis for knowledge on the lower levels.

3.2.1 Sensor level

The sensory level is the interface to the agents sensory system. This level is primarily focused on motion and exploration guided by continuous sensors, making it a traditional sensor-model interface. An interesting feature of this part of the model is that it accommodates for structured communication through maps or verbal commands. This would enable an agent employing the SSH to obtain spatial information directly from graphical maps or verbal directions, translating it into knowledge on appropriate levels of the hierarchy.

For example, graphical maps can be straightforwardly translated into knowledge to the metrical level, where local metrical maps can align frames of reference with the graphical map, allowing for more accurate reasoning on the metric level. Additionally, routes can be visually recognized on the map and stored on the topological level. Spoken and written verbal route directions are frequently sequences of imperatives and their results which naturally corresponds to knowledge at the SSH causal level. These advanced features notably depend quite heavily on high-level reasoning.

3.2.2 Control level

The control level describes the world in terms of continuous control laws that bind the agent and its environment into a dynamical system throughout a qualitatively uniform segment of the environment. On this level, the agent can be modeled and observed as a continuous dynamical system with a feedback control loop.

Control level uses states as means of representing agent's pose in the environment. A locally distinctive state within a neighborhood is a uniquely encoded and recognizable state within the representation that the agent can converge toward by following a control law. Two main control laws are defined: trajectory-following control laws, which bring the agent from one distinctive state to the neighborhood of the next state, and hill-climbing control laws, which bring the agent to a locally distinctive state from any state within the local neighborhood. Each control law has conditions for its appropriateness, and for its termination once it has been selected. Local geometric maps can also be created at the control level, which further reinforce the already available sensor input in order to further ensure proper execution of the control laws.

3.2.3 Causal level

Sequence of control laws that reliably takes the agent from one distinctive state to another can be abstracted on the causal level into an action, and the involved states into views. The association between them is represented as a schema, which is a triple V, A, V' with the interpretation "Action A takes agent from view V to view V'". View representations contain the sensory image and its description along with the agents pose, while the action representation denotes a sequence of applications of one or more control laws on the control level.

Such representation allows the continuous state space in which the agent is described as following the trajectories of a dynamical system to be abstracted abstracted to a discrete state space in which the agent is described as performing a sequence of discrete actions resulting in state transitions. The reasoning on this level is formally performed by applying situation calculus on views, actions, and schemata, resulting in more complex routines.

3.2.4 Topological level

Spatial knowledge on the topological level is represented by an analogue of a topological map. The knowledge on this level is represented as a collection of places, paths and regions, connected by topological relations such as connectivity and containment. This topological map is created by abduction, positing the minimal set of places, paths, and regions required to explain the knowledge extracted on the causal level in terms of views and actions.

Places are descriptions of the environment as points and may lie on one or more paths. Paths represent parts of the environment as a one-dimensional subspace an agent can move along, taking the agent from one place to another. Regions represent sets of places and form a two-dimensional subset of the environment. These representations of knowledge allow for the environment to be viewed as a graph, which permits route planning using appropriate algorithms.

The ontology of the topological level allows for relations to be defined between them, such as "at", "along", "left-of", and so forth. New relations can be logically extracted from existing relations or the knowledge on upper levels of the model. Additionally, hierarchy of regions may be constructed which facilitates spatial reasoning using abstracted representations of the environment.

3.2.5 Metrical level

The final level of the hierarchy represents a global geometric map of the environment. Knowledge from the local 2D geometry representation is placed within a single frame of reference and "patched together" using the knowledge about the topological structure of the environment. The resulting map is incomplete, meaning that not every part of the environment must or is represented, but sufficiently consistent with respect to a single frame of reference to allow for simple geometrical analysis to be performed upon it. This is the only level in the hierarchy that is not necessary for navigation, but can be beneficial in some cases.

3.2.6 Closing remarks

As mentioned in section 4.1, the concept of a cognitive map is highly ambiguous, as evident by a multitude of different definitions provided. The approach behind the Spatial Semantic Hierarchy holds this heterogeneity is a real feature of the phenomenon, and a source of the flexibility, power and robustness of the cognitive map. This makes the SSH one of the most comprehensive models of spatial knowledge available today.

Unlike the SLAM approach, the SSH handles uncertainty by decomposing it into components that are handled effectively by the different representations. This shows that a probabilistic representation of the environment is not mandatory for an effective navigation model and can be replaced by high-level reasoning and the appropriate spatial representation.

The navigation model developed in this thesis was highly influenced by the SSH, which provided a great amount of insight into the problem of spatial knowledge representation, especially regarding the abstraction of the environment and the ontologies and rules on the causal and topological levels of the hierarchy.

3.3 Artificial Recognition System

The ARS is an ongoing artificial general intelligence project of the Institute of Computer Technology (ICT) at the Vienna University of Technology.

Launched in year 2003, the ARS project features researchers from the fields of computer science and psychology working together toward a goal of creating automation systems that would be able to efficiently process large amounts of various sensor data and make decisions based upon them in a human-like manner.

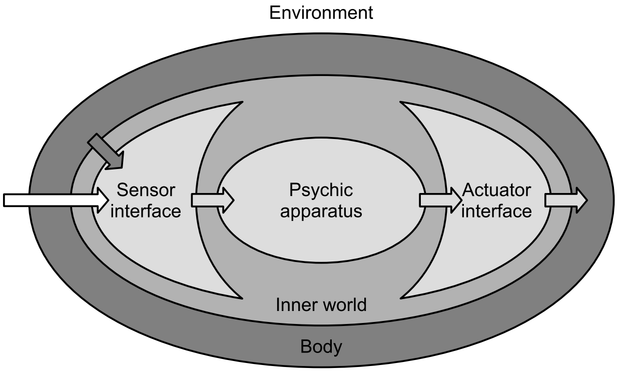


Figure 3.4: Agent architecture of the ARS

The ARS project consisted of two sub-projects: the ARS-PC (ARS Perception), which focused on modeling processes that involve perception of both internal and external phenomena, and ARS-PA (ARS Psychoanalysis), which focused on modeling the reasoning and decision making processes. The results of these projects is the ARS model whose current state is described below.

3.3.1 Perception and symbolization

ARS attempts to model human-like perception system that can handle vast amounts of heterogeneous sensory input from a variety of internal and external sensors, processing, filtering, and storing them in a reliable and efficient way that facilitates human-like reasoning and decision making. The key process behind this task is *neurosymbolization* [VB08], which essentially processes the raw input information and assembles it into more abstract pieces of information termed symbols.

A *symbol* is a piece of information resulting from neurosymbolization that carries a certain meaning and represents the basic unit of information that can be processed by the system. Neurosymbolization occurs on several levels of symbolic abstraction, from less to more abstract: feature symbols, sub-unimodal symbols, unimodal symbols and multimodal or representation layer symbols, each resulting from processing and combining symbols from lower levels into more complex symbols. Multimodal symbols represent the final form of the interpretation of the input data [Gru07].

Symbols have a different lifespan on each level – low-level symbols are constantly created and destroyed as the sensory input changes, but the high-level symbols persist as long as the entities represented by those symbols are perceived, but are constantly modified as the low-level symbols they are associated with are created and destroyed.

Symbols are categorized and recognized following a memory-based approach, which entails comparing the incoming symbol to symbols stored in agent's memory. Identified symbols comprising the perceptual input allow the agent to recognize its situation and to retrieve scenarios that allow decision making to take place. These high-level symbols therefore represent the input to the decision making process.

3.3.2 Reasoning and decision making

The ARS features a model of decision-making and control capable of high-level reasoning based on Sigmund Freud's "Ego – Super-ego – Id" psychoanalytical personality model. It is intended to handle the high-level symbolic information created by the perception process, retrieve and evaluate scenarios and possible actions based on that input, and ultimately pass chosen actions translated into commands to the actuators of the agent. As stated before, all information ultimately comes from either the environment or internally from the agent itself.

Inspired by human decision making, the ARS models an internal state of the agent comprised of drives, emotions and desires which have a strong influence on the action evaluation process. These are termed *internal stimuli* and the main task of the decision-making process is to keep them in balance.

This is best seen on an example of an autonomous agent that must eat to sustain itself. Starvation is a condition represented by a great internal imbalance and as such will place great value on actions by which the agent can obtain food. Similarly, perceived threats of a dangerous situation increase the basic emotion of fear, which causes actions to be chosen which remove those threats or the agent from them, preserving the well-being of the agent. In short: the bigger the internal imbalance, the greater the priority of actions that counter it.

These internal stimuli are directly handled by the part of the model termed *Id*. It responds to internal stimuli and suggests actions to fulfill these basic needs in order to restore the internal balance. It is a purely reactive system that is activated when a physiological stimulus arises, and does not consider the consequences any of the proposed actions might have. These actions are then passed to the decision making process performed by the the part termed *Ego*. Ego receives action proposals from Id and decides on the next course of action. Before executing any action it will check the proposal with the third part of the model called Super-ego. *Super-ego* is the part of the model that contains norms and prohibitions imposed on the agent. These are usually externally defined and allow humans to give sets of behavioral rules that are important to the agent's purpose, but can be constructed by the agent itself through experience. The super-ego overrides actions that would violate those rules, such as those that would result in harm of humans or even other agent through pursuit of actions that satisfy an urgent need. The Ego therefore mediates between two other parts and is responsible with finding a course of action that both Id and Super-ego allow. That action is then passed to movement control that executes it.

Following a top-down approach to modeling, the ARS model was designed in several layers of complexity. The first layer corresponds to the personality model as described above, and serves as a basic outline of the complex underlying functionality of the ARS (see figure3.5). From this, each part has been further decomposed into smaller functional parts in several layers, resulting in the fourth layer of the functional model, shown in figure 3.6, in which basic functionalities of the mind have been identified, described and separated into functional modules that communicate with one using defined interfaces.

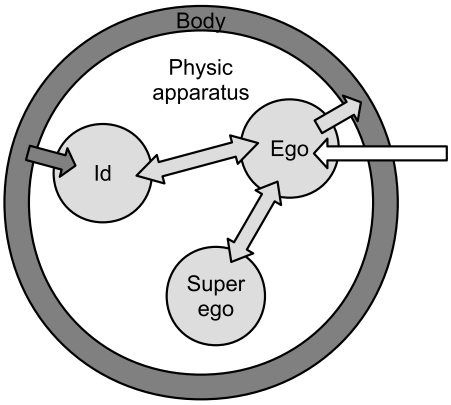


Figure 3.5: Diagram of the ARS functional model - first level

The model described in this thesis is a part of the decision-making strand of the ARS functional model. As stated before, the ARS model is inspired by Freud's psychoanalytical models of the human mind and thus inherits many of its concepts. Those that are necessary for understanding of the navigation model developed in this thesis are presented below.

3.3.3 Primary and secondary process

The cognitive process of the mind as modeled by the ARS is divided into two parts based on how the information is represented and handled: the primary and the secondary process.

The so-called *primary process* processes information based solely on the internal drive demands and is hence independent of the current context and the situation the agent is found in. Information is also processed completely, regardless of possible conflicts and contradictions contained within. The primary process is therefore said to follow the so-called "pleasure principle", as it driven by the aim to satisfy the basics needs of the agent no matter what. The primary process manipulates things presentation and affects. Since there are no logical relations defined on these data structures, no reasoning can take place within the primary process.

The *secondary process* is the part of the model that processes the information and assigns logical, local and temporal associations to its contents, as well as resolving the conflicts and contradictions that are found within the information content during the processing. Additionally, it takes into consideration the current situation and rules and is said to follow the so-called "reality principle". The secondary process manipulates distinct content types termed word presentations which, unlike the thing presentations, support above mentioned associations, thus allowing high-level reasoning to be performed.

The decision-making strand of the cognition belongs to the secondary process, making it the environment in which localization and navigation routines take place. The following section will describe the above mentioned content types along with some additional content types in order to lay a foundation needed to describe these routines within the context of the ARS architecture.

3.3.4 Information and knowledge representation

Psychoanalytical basis of the ARS model provides an abstract concept of memory structures, which defines the information and knowledge representation used by the ARS.

The central concept of information representation in psychoanalysis is the *memory trace*. It is defined within the ARS project as "a psycho-physiological concept of representing memories in the psyche' ' [DFZB09, p. 424]. Memory trace are patterns upon which all psychic data structures are based. Incoming perceptual information is matched against existing memory traces, activating data structures of the best match. If no matches are found, new memory traces are created and stored. Memory traces give rise to thing presentations, but are of no further importance the topic of the thesis, and are mentioned here for the sake of completion only.

A *thing presentation (TP)* is a symbolic representation of a piece of information about an object. It is defined as "the psychic representation of an objects sensory characteristics in the form of acoustic, visual, olfactory, haptic, and gustatory modalities" ' [DFZB09, p. 426]. A thing presentation, in its narrow definition, consist of the modality of the information, and its value, for example: "Color: green", "Shape: circle", "Temperature: cold". In this way it is comparable to the concept of a program variable for primitive types, in that it has a type and a value associated with it.

Thing presentations form the content of the primary process. Accordingly, thing presentations can be connected to one another using associations that describe similarity and co-occurrence. A set of interconnected thing presentations is termed *thing presentation mesh (TPM)*.

Thing presentations associated to a certain mesh are called *attributes*. Every attribute association has a weight associated to it representing the strength of that association, that is, the importance of that particular attribute to the mesh. The model distinguishes between class attributes and instance attributes. A *class attribute* is a TP that pertains to a whole class of objects and is the same across all instances of that class, while a *instance attribute* pertains to a particular instance, distinguishing it from the other objects. The distinction exists to limit the allowed changes of predefined TPMs, preventing changes that would alter them in a way that would make them incongruous with the reality, as the ARS model at the time can only perceive and understand predefined classes and is unable to create new classes itself. An instance attribute can be converted into a class attribute if the association weights are strong across all instances of an object class [Zei10].

Thing presentation meshes are divided into three categories depending on their source: environmental sensations, bodily sensations, and homeostatic state. Thing presentation mesh has no equivalent in the psychoanalytical theory; instead, it is a technical concept crucial for the implementation of the model.

A *word presentation (WP)* is a representation of a concept as a set of symbols. Any information represented by a thing presentation that needs to be processed in the secondary process must be converted into a word presentation. Word presentations are used for representation of basic psychic concepts or properties of a perceivable object. Word presentations are connected to one another using a kind of association distinct from the kind used for connecting thing presentations. These associations allow temporal and logical relations to be defined between word presentation; this allows for construction of more complex symbolic structures.

A *word presentation mesh (WPM)* is a complex content type of the secondary process consisting of one or more connected word presentations. Word presentation meshes are used to represent complex concepts of entities and objects, but also of drive aims, feelings, goals, situations, acts, and scenarios in the secondary process. Similar to thing presentation meshes, they are a technical concept and have no equivalent in the psychoanalytical theory.

Every concept used by any part of the secondary process of the ARS must be represented by word presentations and word presentation meshes. Such strict specification of the content representation carries many benefits to the reasoning process, but also represents an additional challenge to the implementation of any functionality within the secondary process, as it also places restrictions on all content used by process, particularly regarding the interfaces with the rest of the model.

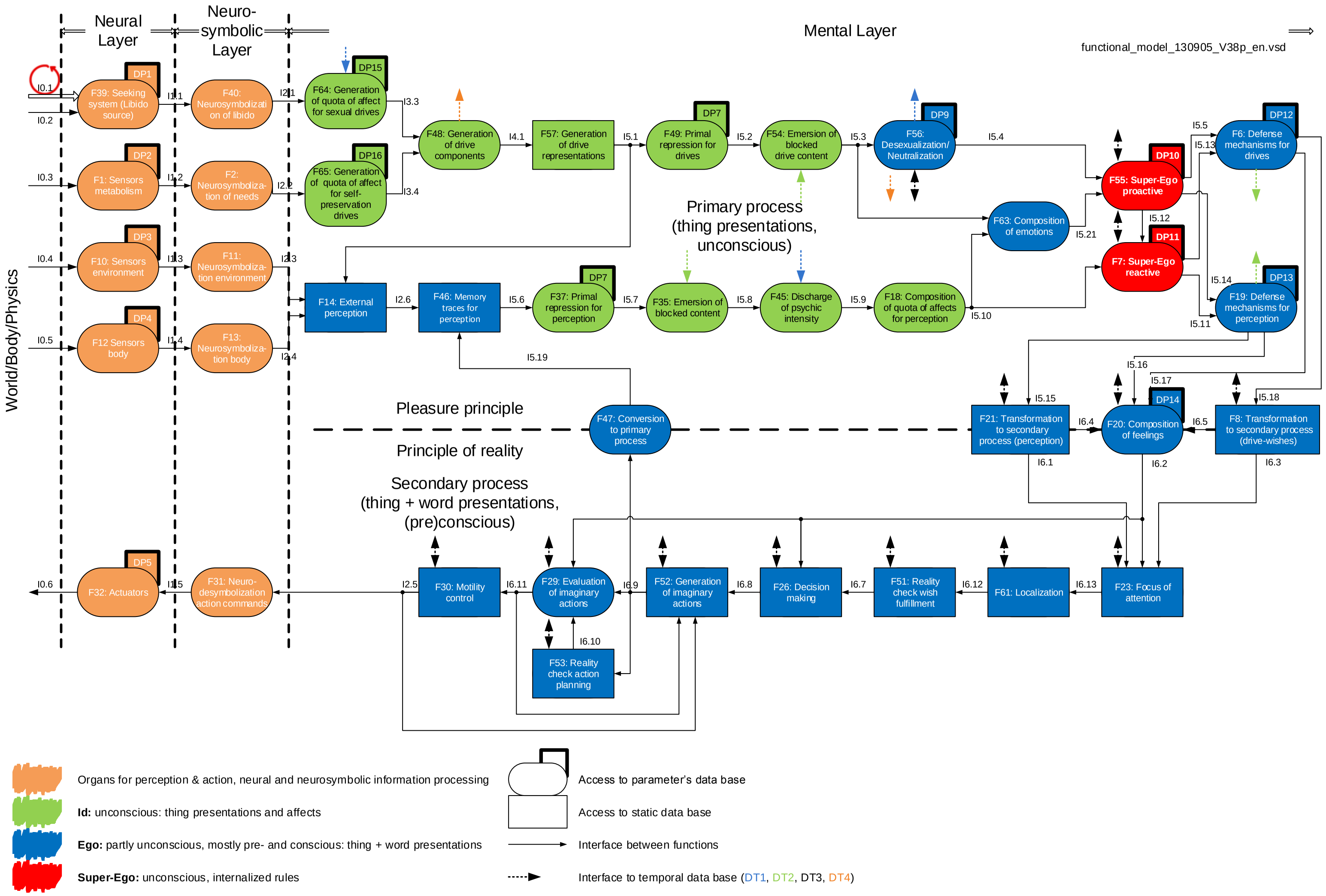

Neural Layer
Neuro-symbolic Layer
Mental Layer
functional_model_130905_V38p_en.vsd
World/Body/Physics
I0.1
I0.2
I0.3
I0.4
I0.5
I0.6
F39: Seeking system (Libido source)
DP1
F1: Sensors metabolism
DP2
F10: Sensors environment
DP3
F12 Sensors body
DP4
F32: Actuators
DP5
I1.1
I1.2
I1.3
I1.4
I1.5
F40: Neurosymbolizati on of libido
F2: Neurosymbolization of needs
F11: Neurosymbolization environment
F13: Neurosymbolization body
F31: Neuro-desymbolization action commands
I2.1
I2.2
I2.3
I2.4
I2.5
F64: Generation of quota of affect for sexual drives
DP15
F65: Generation of quota of affect for self-preservation drives
DP16
I3.3
I3.4
F48: Generation of drive components
I4.1
F57: Generation of drive representations
I5.1
F49: Primal repression for drives
DP7
I5.2
F54: Emersion of blocked drive content
I5.3
F56: Desexualization/ Neutralization
DP9
I5.4
F63: Composition of emotions
I5.21
F55: Super-Ego proactive
DP10
F7: Super-Ego reactive
DP11
I5.12
I5.5
I5.13
F6: Defense mechanisms for drives
DP12
I5.14
I5.11
F19: Defense mechanisms for perception
DP13
Primary process
(thing presentations,
unconscious)
F14: External perception
I2.6
F46: Memory traces for perception
I5.6
F37: Primal repression for perception
DP7
I5.7
F35: Emersion of blocked content
I5.8
F45: Discharge of psychic intensity
I5.9
F18: Composition of quota of affects for perception
I5.10
I5.19
Pleasure principle
F47: Conversion to primary process
Principle of reality
Secondary process
(thing + word presentations,
(pre)conscious)
I5.15
I5.16
I5.17
I5.18
F21: Transformation to secondary process (perception)
I6.4
F20: Composition of feelings
DP14
I6.5
F8: Transformation to secondary process (drive-wishes)
I6.1
I6.2
I6.3
F30: Motility control
I6.11
F29: Evaluation of imaginary actions
I6.9
F52: Generation of imaginary actions
I6.8
F26: Decision making
I6.7
F51: Reality check wish fulfillment
I6.12
F61: Localization
I6.13
F23: Focus of attention
I6.10
F53: Reality check action planning
Organs for perception & action, neural and neurosymbolic information processing
Id: unconscious: thing presentations and affects
Ego: partly unconscious, mostly pre- and conscious: thing + word presentations
Super-Ego: unconscious, internalized rules
Access to parameter's data base
Access to static data base
Interface between functions
Interface to temporal data base (DT1, DT2, DT3, DT4)

Figure 3.6: Diagram of the ARS functional model - fourth level

CHAPTER 4

Model

This chapter will describe the model of the navigation module for the ARS designed in this thesis. The model follows the cognitive approach to navigation (chapter 2). The reason for this decision lies in the nature of the ARS, itself being a cognitive model of the human mind in which the observable functionality of the mind is identified and replicated. This eliminates the possibility of following a bionic connectivist approach outright, as it would preclude any interoperability with the rest of the system on the symbolic level, thereby defeating its purpose.

4.1 Cognitive map

This approach distinguishes navigation *between* locations and navigation *within* a location. This obviates splitting environment beyond necessity. When present at a certain location an agent should be able to

- transition into another location
- perceive and reach its intended goal, if present at the current location.

In first case, navigation is considered to still be in progress, and the agent must be able to perceive anchoring features in order to progress further. In the second case the agent has reached its destination. The process of navigation ceases and the agent instead goes about accomplishing its intended task. This can include further movement, such as approaching the goal itself or avoiding dangers. This kind of behavior will be termed “local navigation”, as opposed to the other kind navigation discussed in the rest of the thesis, termed “global navigation”.

In the context of planning and decision making, such behavior is distinct from global navigation as it additionally relies on objects and entities within the environment that are abstracted in the latter, such as other mobile agents, movable and non-salient obstacles, and so on. Another important distinction is that it dynamically produces routes that are not constant and very likely to change in the future, and thus never committed to the memory. Such behavior will always take precedence over the long-range navigation, as it ensures its proper and safe execution. For example, while traversing a long range route obstacles are sure to be encountered. Whenever this occurs the decision-making process suspends long-range navigation in favor of action that will go around the obstacle which would usually prevent further movement, continuing it once the obstacle has been cleared. This can happen many times over the course of a single route traversal at any point in time and space, depending on the state of the environment.

The example above demonstrates how the ARS architecture allows these two types of navigation to be functionally largely independent from one another. Short range navigation has been abstracted away in this thesis as it bears no impact on the model described in this thesis. This allows for a simple model of the internal representation of the environment that needs not take into account non-constant objects and features found in the environment, while still permitting obstacle-avoiding behavior that is triggered when need arises.

4.1.1 Map properties and structure

The cognitive map is the basic data structure navigation algorithm works with. In order to ensure its proper functioning some restrictions on the structure and behavior of the cognitive map must be imposed.

When considering large-scale environments, which contain a large number of distinct places over greater distances, and which are commonplace in everyday world, the complexity of traversing such environments successfully without external assistance becomes apparent.

There exists sufficient evidence to suggest humans use abstractions called regions to efficiently navigate environments with even very large amounts of distinct locations [SV06]. These abstractions enable simplified representation of parts of the environment which allows spatial reasoning to be performed without risking information overload due to complexity of the problem.

As this navigation model is expected to be able to operate in environments of any size, implementing spatial abstraction becomes a major factor in the design of the cognitive map and the algorithms that work with it.

The most important concept related to spatial abstraction is the concept of a region. A *region* is a unit of a cognitive map which can contain arbitrarily many other map units (locations or other regions) based on various criteria. Regions and their contents form a structure called *region hierarchy*, which is superimposed over a simple cognitive map, thus facilitating abstraction on a symbolic level. Regions and locations together are referred to as *map units* in this thesis. There exist models of cognitive maps which allow multiple independent hierarchies to be defined for the same environment [FG02]. The model described in this thesis has been limited to a single hierarchy for the sake of simplicity, but can be expanded to include multiple hierarchies in the future, if necessary.

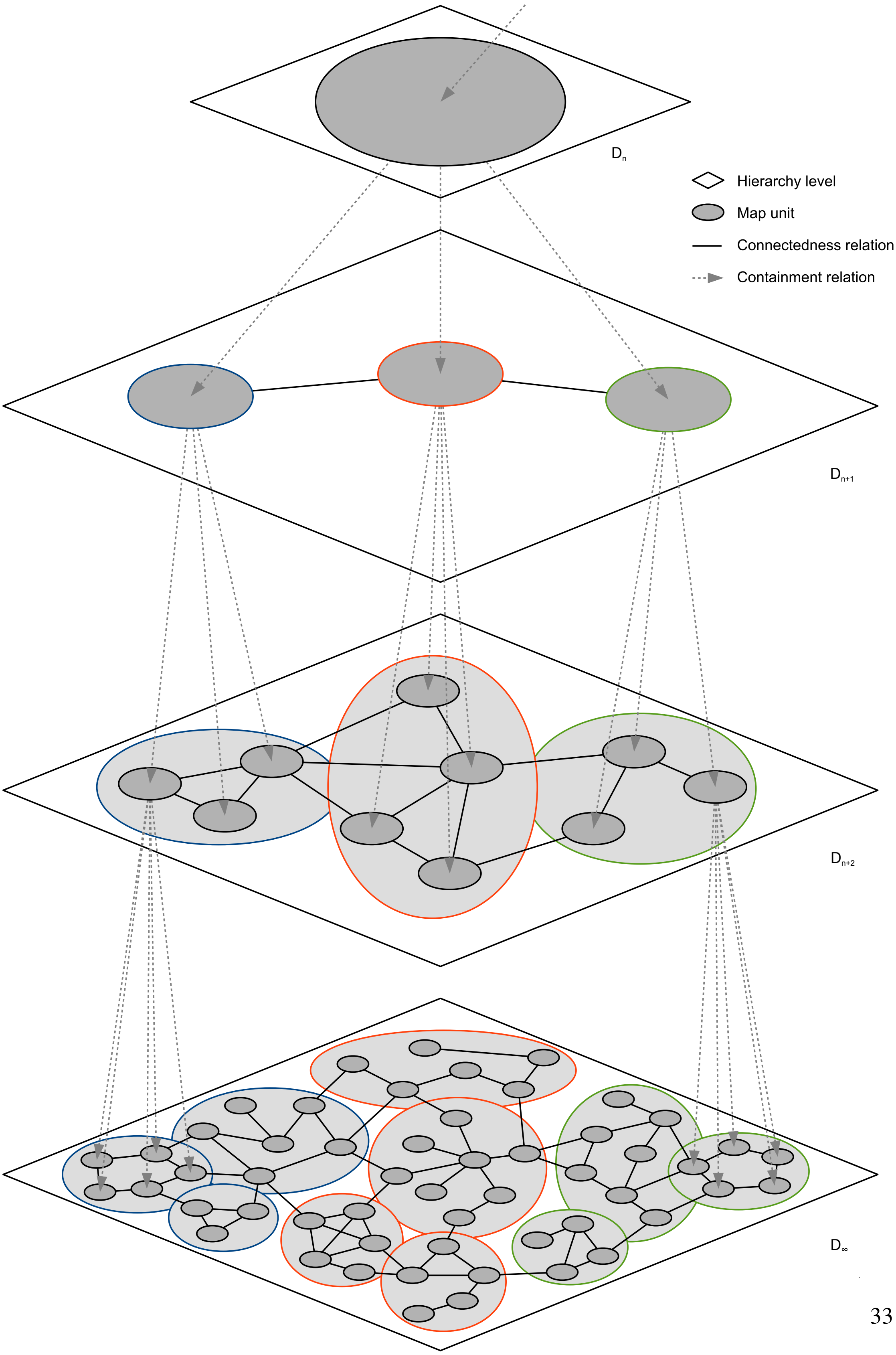


Figure 4.1: Section of the hierarchical cognitive map.

The so-called Universe region[1] is the only mandatory region in the hierarchy. It is the top-level region of the map, meaning it has no parent region and it contains all other map units in the map. This property is crucial for the route planning algorithm, which depends upon the property that any two locations in the map have a common super-region. The term "top-level region" may be used interchangeably with the term "Universe".

Hierarchy is divided into levels, starting from level 0, which must only contain the top-level region. From this level on, the depth of a level a region belongs to is determined by the distance of that region from the top-level region. This distance is termed *depth* of a region, and is equal to the number of the containment relations between that region and the top-level region. All locations of the map belong to the bottommost level, which by definition has an infinite (∞) depth. Having all locations placed within a single bottommost level is necessary as all connectedness relations between regions on any level above are ultimately inferred from the connectedness of locations in the bottommost level. This level represents the flattened version of the hierarchical map. This allows the navigation algorithm to perform even in cases where no other regions are defined.

Figure 4.1 depicts a part of the cognitive map that is contained by a region on level n. Regions on the level are color-coded to better show the containment relation across multiple levels, as the arrows depicting containment relation between the two bottommost layers are not all shown in order to maintain visibility. Additionally, connectedness relations, which are properly represented with arcs akin to those in directed graphs, have been abstracted to simple lines for the same reason.

Regions also have a restriction on their underlying structure. When observing the cognitive map at a certain level, the agent's location is abstracted into the containing region at that level. This means the agent's actual location within the region should not influence the connectivity on that region's level. In order to assure this property, all subunits must be connected to all exit subunits of the region.

The importance of this property can be easily shown on an example. Figure 4.2 shows a inconsistent region Dn in which the exit map unit G is not reachable from map units A, C, D, E. The agent's current location is in A, and the navigation algorithm currently running on the level D_n finds the best path, which happens to be passing through region Y. This is based on the fact that G is connected to a map unit in region Y on level D_{n+1}, which implies that X is connected to Y. Since the agent is in A, which is *not* connected to G within X, the algorithm will ultimately fail to find a path on the level D_{n+1}.

This applies to the top-level region as well, which implies that the every location within the environment must ultimately be accessible from any other location. This notion is a more general case of the problem of finding strongly connected components of a simple directed graph, the

[1] The term "Universe region" was chosen in order to distinguish the top-level region in the internal representation of the environment from the actual environment. The word is capitalized to further avoid ambiguity with any other possible meaning of the term "universe".

difference being that only a subset of vertices, that is, the exit nodes of a component need be reachable from all others, as opposed to all vertices being reachable from all others. This is a non-issue if all connections are bi-directional, as is mostly the case in the real world.

Should the cognitive map break any of the rules at any point, it is considered to be *inconsistent* and must be brought back to a consistent state using the map maintenance algorithm in order to ensure the proper execution of the navigation algorithm. The algorithm in question is described in section 4.2.2.

4.2 Mapping

Despite the lack of the learning capability of the ARS model, this section will cover the principles of map learning pertaining to this model in particular, as well as describe certain map update algorithms designed to maintain the proper structure of the map, which is vital to the whole navigation process.

4.2.1 Map learning

As stated before, the main purpose of regions is the division of the environment into more manageable units. This gives rise to the *utility principle* of region management: to construct and modify regions in a way that provides the best possible hierarchical structure of the map.

In contrast, the set of factors that influences the formation of regions in human spatial behavior is much larger and more diverse. Factors such as geographical or man-made features, cultural and political divisions, and history are seldom considered in computational models do to the complexity of their cognition and reasoning based on them. Constructs such as rooms, buildings, streets, town squares, etc. provide an implicit separation of the environment which humans readily and efficiently use in everyday situations. It would thus be beneficial to adapt the model to these features to maximally utilize the state of the environment.

This adds another facet to the problem of mapping in models based on hierarchical representations of the environment: not only should the newly discovered locations be properly recognized and encoded, but also properly integrated into the hierarchy in a useful and meaningful way.

An example of a situation where this balance comes into play: The agent is located in an urban environment, in a town square. As it moves into a previously unvisited street, a new part of the environment is recognized and encoded. The current state of its cognitive map favors the integration of the new location into the smallest region he was in up to that moment, following the utility principle. However, since it entered a street, which is semantically different from a town square, the semantic principle states it would be preferable to create a new region that is separate, but adjacent to the region that represents that town square. Following the utility principle one would end up with a map that is more efficient, but semantically incoherent. On the other hand, following the semantic principle will result in a region containing only one location, which is adding additional complexity without providing any utility. This conflict necessitates a decision being made as to which principle to follow at which moment.

Finding a balance between the utility and semantic principles is non-trivial despite the existing research into human behavior and further research toward finding a proper hierarchy integration model is needed.

4.2.2 Map maintenance

Creating and maintaining the cognitive map is a complex learning task that involves processing various types of inputs that may be available to the agent. The process of segmenting the environment into regions falls outside the scope of the thesis. This is primarily due to the existence of various non-spatial factors that govern the division of environment into regions, such as political, cultural, even internal factors. Additional abilities of the agent that influence the structure of the cognitive map are exploration of the unmapped part of the environment and detection of changes in the mapped part of the environment, both of which are beyond the scope of this thesis.

Assuming, however,i' that the agent does posses such abilities, actions that change the cognitive map must be executed in a way that maintains the valid structure of the cognitive map. One change in connectedness in the bottommost level can trigger a wave of changes that can propagate up to the top-level region. The changes that can occur are: addition and removal of either a location, a region, or a connection between locations. In addition, joining and splitting regions that are on the same depth level is also possible, should the need arise, but such changes are expected to be based on qualitative reasoning about spatial relations.

Adding a location or a connection between existing locations to the map cannot break its consistency and therefore entails no other structural change. On the other hand, removing a connection can lead to inconsistency by breaking the connectedness rule. Removal of a connection subsumes removal of a location, as it effectively means removing all connections of that location to the rest of the map. They will therefore be discussed together.

Removal of a region *does not* imply the removal of the contained map units. Instead, removal of a region is seen as restructuring of the map's hierarchy. When removing a region on some level D_n, then all of the contained regions will be shifted up by one level. The new connections can then be calculated from the connections on the level D_{n+1} and below.

Conversely, adding a region consists of selecting the region's contents, then replacing them with the new region. This action can also lead to inconsistencies, as the new created region has its own, different exit subunits. Taking the example from figure (REF) again, grouping map units A, C, D, E, G, H into a new region is permissible, but A, B, C, D, E, F isn't, as that would make F an exit unit of the region, which is not reachable from A, C, D, E within the new group.

This algorithm will not change the structure of the map if it is consistent. It can therefore be used after every change of this type to verify consistency, as it can take a large amount of time

Algorithm 4.1: Hierarchy repair algorithm

```
Input: deepest region affected by change $R_{current}$
do
    Let $S$ be an empty region stack;
    Push $R_{current}$ onto the stack;
    while stack not empty do
        Pop region $R$ from the stack;
        Find set of exit map units $S_{exit}$ within $R$;
        Let $M$ be an array of map unit sets indexed by map unit sets;
        foreach map unit $U$ contained by $R$ do
            Determine map unit subset $S'$ of $S_{exit}$ reachable from that unit;
            Add map units in $S'$ to set in $M$ indexed by $S_{exit}$;
        end
        foreach set $S_{units}$ in $M$ do
            Group $S_{units}$ into a new region $R_{new}$;
            push $R_{new}$ onto the stack;
        end
    end
    if $R_{current}$ is the top-level region then
        Exit;
    else
        Set $R_{current} :=$ parent region of $R_{current}$;
    end
loop
```

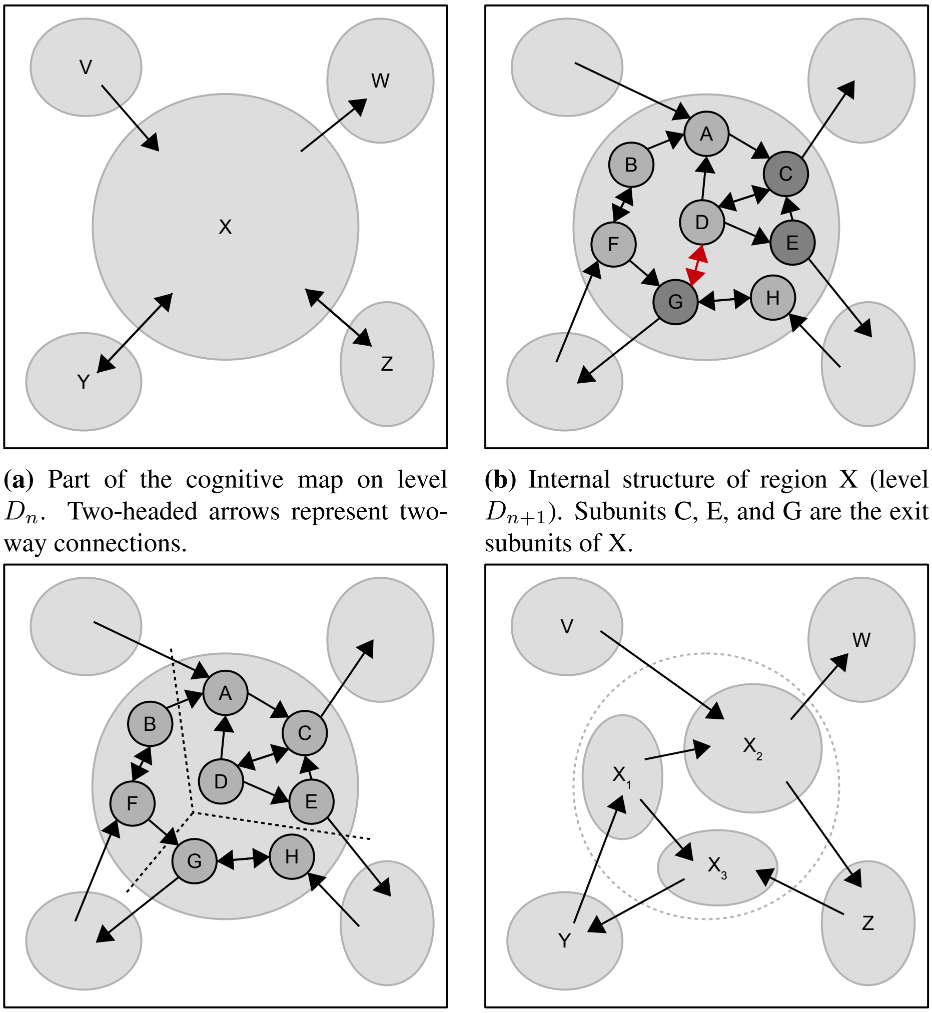
for large maps, especially if applied to the whole map. However, if one assumes the cognitive map was consistent prior to the change, running this algorithm on the affected portion of the map will always bring it back to a consistent state. This in turn means that the algorithm need not be run constantly, but only after changes. Additionally, a whole-map check should be performed at start-up in order to ascertain the consistency of the initial cognitive map.

4.3 Localization

Localization process, as described in section 2.3, is essentially a matching algorithm that compares a set of possible solutions from a set of all know solutions against an input pattern, outputting the best-matching solution as the end result.

Localization process of the navigation model features following steps:

- Input preprocessing - Perceptual input is filtered, extracting relevant environmental features and their configurations. The filtered input data constitutes the *matching pattern*.



(a) Part of the cognitive map on level D_n. Two-headed arrows represent two-way connections.

(b) Internal structure of region X (level D_{n+1}). Subunits C, E, and G are the exit subunits of X.

(c) Split caused by removing connection marked red in figure (b). Note that the exit subunits of new regions are universally reachable from within their region.

(d) New structure on level D_n after running one step of the algorithm (REF). Further changes might occur on upper levels, depending on the rest of the structure.

Figure 4.2: Visualized example of one iteration of algorithm 4.1. Map units are represented by cricles, regions by ellipses, connectedness by arrows.

- Candidate selection - The cognitive map is searched to obtain a limited set of potentially matching location encodings, called *scenes* based on the received input.
- Matching - Scenes from the candidate set are matched one by one against the matching pattern, thereby obtaining the *similarity score*. Finally, the location encoded by the scene with the best matching score is returned as the result.

The rest of this section will describe the steps listed above in greater detail.

4.3.1 Input pre-processing

As a part of the ARS model, the localization process receives the input from the primary process, where it undergoes most of the necessary preprocessing (see section 3.3.3) which makes it ready for use in the secondary process. This processed input is expected to preserve and represent environmental features as symbols and their configuration as associations between them. In addition, a symbolic representation of the agent should also be provided to allow representation of configurations with respect to the agent.

This input data may also contain information preserved in the short-term memory in order to maintain a consistent representation of the perceived environment across shorter time frames in which localization takes place. The reasons for including short-term memory information are further explained in section 5.3.3.

4.3.2 Candidate selection

The running time of the matching process, at least in this case, depends primarily on two factors: the complexity of the patterns and the size of the solution set. The first factor remains relatively constant over time, as the set of recognized environmental features and their configurations is expected not to change as the agent gathers knowledge about the environment.[2]

On the other hand, as the agent explores and learns new parts of the environment, the cognitive map, which takes the role of the solution set in this context, will logically grow as well. Cognitive map are expected to become extremely large in case of real-world environments. Since the matching needs to be performed for *every* admissible location encoding at least once, it follows that the run time grows in a linear fashion as the size of the cognitive map increases.

By itself this would not present a significant problem, if the localization were seldom performed during the normal operation of the agent. However, since this process is performed *at every execution step*, long run times should be avoided at all costs. In order to achieve this two heuristic methods were developed. These constitute the candidate selection process.

Adjacent location heuristic works by reducing the initial candidate set to the set containing scenes of locations directly reachable from the current location, together with the scene of the current location itself. This is based on the *continuous movement assumption*, which supposes that the maximum possible distance traversed by the agent between two consecutive localizations is smaller than the size of the smallest location. This assumption assures that at every distinct moment in time the agent is either still in the last known current location or has moved into an adjacent one, but never farther than that.

This is an optional part of the localization process, but has a big impact on the performance by significantly limiting the size of the candidate set. Since there is currently no known way of reducing the candidate set *during* the matching, this heuristic is the best way of improving the performance without sacrificing the validity of the process.

[2] This statement refers to the *types* of environmental features and their configurations, not their instances.

The heuristic described above is not applicable if the agent is not localized, which is always the case when agent is first initialized, but can also happen when the localization fails due factors such as various errors, loss of data, or an unexpected change in the environment. For these cases another heuristic was developed, termed *specific landmark heuristic*.

Specific landmark heuristic assumes the associations between landmarks and scenes are traversable in both directions, that is, that one or more scenes can be accessed from a landmark that defines them and vice-versa. If this is the case, the most specific perceived landmark, which is most likely to have a limited number of scenes associated with it, is used to define the candidate set. The candidate set is then the set of scenes associated with it.

4.3.3 Matching

The development of the scene matching algorithm presents a particular challenge due to the symbolic nature as well as the amount of the data provided by the perception field.

Algorithm 4.2: Localization matching algorithm

```
Input: perceptual information, last location, expected next step
Output: new current location
Get visible landmark set from the scene in the perception field;
if agent is localized then
    Create a candidate set from the current location and adjacent locations;
    Retain candidates that are associated with any of the visible landmarks;
else
    Create a candidate set from locations in the environment that are associated with any
    of the visible landmarks;
end
foreach candidate location in the candidate set do
    Match the remembered scene of the candidate location to the perceived scene and
    obtain the similarity score;
    if the candidate location is the current location then
        Apply the current location bonus to the score;
    end
    if the current candidate location has the best score then
        Make the current candidate the best candidate;
    end
end
return the best matching candidate location;
```

4.3.4 Similarity score

In order to be able to select the candidate scene that best matches the perceived scene, one must establish a proper numerical measure with which similarity of scenes can be expressed and compared. This measure of similarity is termed *similarity score*, or simply *score*, which is a real number within the range between zero and one inclusive, where *zero* represents complete dissimilarity of two scenes, and *one* represents a perfect match. This allows the best match to be decided by simple numerical comparison.

Similarity score is calculated using the similarity routine, which takes two scenes as arguments and outputs the similarity score. The similarity routine is the heart of the matching algorithm, and works by performing several comparison heuristics on the two scenes, weighting their results and calculating the normalized aggregate score. For a matching algorithm using n different heuristics, where V_i is the score of the i-th heuristic and x_i is its weight, the normalized aggregate score V_{total} is calculated using the following formula:

$$V_{total} = \frac{\sum_{i=1}^{n} V_i x_i}{\sum_{i=1}^{n} x_i}$$

All variables in the formula can only contain values between zero and one. Heuristics are designed to be mutually independent, and this formula allows them to be combined them as desired, individually amplifying or reducing their weights in order to test their efficacy and the and the overall effect on the localization process. Heuristics that were chosen for this model are presented below.

4.3.5 Landmark set heuristic

Landmark set heuristic is a simple heuristic that compares the unordered set of landmarks within of a scene with that of another scene, while ignoring all other available information. The heuristic first removes all landmarks from the set of the perceived scene that are not present in the set of the recalled scene. Then, the set difference between the recalled and the perceived scene sets is calculated. Finally, the cardinality of the set difference is normalized onto the interval $[0, 1]$ by dividing it with the cardinality of the recalled scene set. The final formula for obtaining the score is:

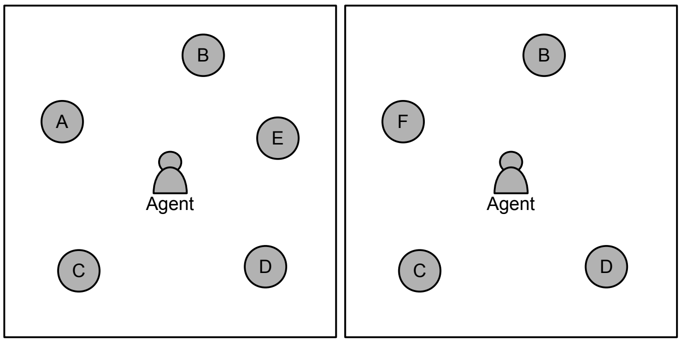
$$V_{set} = \frac{|(S_{perceived} \cap S_{recalled})|}{|S_{recalled}|}$$

This formula assumes that the recalled scene cannot be empty, which is reasonable since such scene would be devoid of information and therefore useless to the agent.

Considering two scenes depicted in figure 4.3, their similarity score according to the formula above is:

$$S_{perceived} = \{A, B, C, D, E\}, S_{recalled} = \{B, C, D, F\}$$

$$V_{set} = \frac{|(S_{perceived} \cap S_{recalled})|}{|S_{recalled}|} = \frac{|(\{A, B, C, D, E\} \cap \{B, C, D, F\})|}{|\{B, C, D, F\}|} = \frac{|\{B, C, D\}|}{|\{B, C, D, F\}|}$$
$$= \frac{3}{4} = 0,75$$



(a) Perceived scene **(b)** Recalled scene

Figure 4.3: Two scenes to be compared. Landmarks are represented by circles with labels inside.

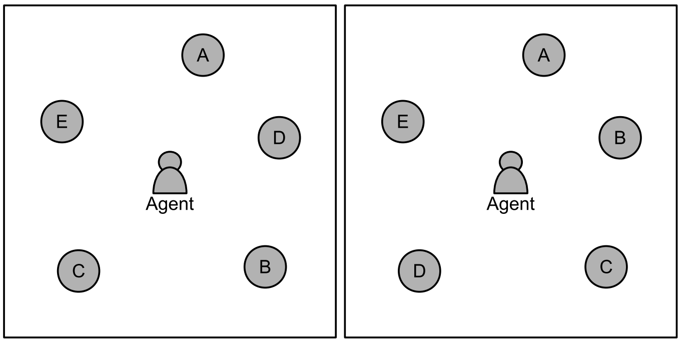
It must be noted that this comparison is not symmetric. Indeed, by swapping the perceived and the recalled scene in the example above one would obtain a significantly different result of $0,6$.

The scene that is obtained from the perceptual input is different from scenes used to encode locations. It may contain significantly greater number of features and configurations than remembered scenes. The reason for this asymmetry lies in the expectation that only a few landmarks, those that are most salient, will be remembered and thus encoded in a scene of a particular location. Since the resulting recalled scene is therefore necessarily smaller information-wise, is it reasonable to use it as basis for normalization. For these reasons the perceived scene must be treated differently by the localization algorithm, which is reflected in the design of this and other heuristics.

4.3.6 Landmark order heuristic

Landmark order heuristic was inspired by the problem where two landmarks positioned next to one another should allow an agent to determine which side they are being viewed from. This is impossible to achieve using set or distance heuristics, as neither takes into account relations between landmarks in a scene. Since solving such a problem was deemed important in the beginning, a heuristic method was developed that would allow an agent to distinguish those cases. Existing landmark triangulation approaches [Bor09] gave some indication that such method might be crucial for proper localization, at least for computer models.

This method works by comparing ordered lists of landmarks of two scenes. First, both ordered lists are trimmed so that only those landmarks present in both lists remain. Afterward, a *Kendall tau difference* [Ken38] is calculated, which is based on the number of swaps a bubble sort algorithm would need to make in order to reorder one list in the same order as the other list.



(a) Perceived scene **(b)** Recalled scene

Figure 4.4: Two scenes to be compared. Landmarks are represented by circles with labels inside.

Considering two scenes depicted in figure 4.4, their similarity score would be calculated as follows: the two lists are in order as follows: A, D, B, C, E compared to A, B, C, D, E. Using the bubble sort analogy, one would have to perform *two* swaps in the first list: B–D, followed by C–D, in order to reach the same order as in the second list. The normalized similarity score is then obtained by the formula:

$$V_{order} = 1 - \frac{n_{swaps}}{\frac{n_{length}(n_{length}-1)}{2}}$$

which in this case yields 0,8.

4.3.7 Landmark distance heuristic

Landmark distance heuristic is based on the idea that the distance is intrinsically related to the localization. This heuristic assumes qualitative distinction of absolute distances, or in other words, that there is a certain distance beyond which all objects are perceived as “far” as opposed to “near”, and that this distinction can be utilized in the scene recognition.

This heuristic works by categorizing the landmarks based on their absolute distance to the agent, and then applying penalties to the score for each distant defining landmark of the scene. Invisible defining landmarks are not penalized, as this is already done by the landmark set heuristic. If S_{all} represents the set of all visible landmarks, $S_{near}, S_{mid}, S_{far}$ the sets of near, mid-far, and far landmarks, respectively, and P_{mid} and P_{far} score penalties for mid-far and far landmarks, respectively, the formula for calculating the score is:

$$V_{dist} = \frac{|S_{near}| + (1 - P_{mid})|S_{mid}| + (1 - P_{far})|S_{far}|}{|S_{all}|}$$

In the example depicted in figure 4.5, landmark B is categorized as near, A and E as mid-far, C as far, while D is not visible. Supposing the penalties are 0,15 for mid-far landmarks, and 0,30 for far landmarks, and the defining landmarks of a location are A, B, and C, the landmark distance score for that location according to the formula above is

$$V_{dist} = \frac{1 + (1 - 0,15) \cdot 1 + (1 - 0,3) \cdot 1}{3} = 0,85$$

The landmark distance heuristic was introduced it became clear that there exist many cases in which other heuristics alone would give high scores to locations that were further away from the agent's current position, sometimes even getting higher score than the location that should have been recognized as the current location due to their having the same score, forcing the algorithm to choose the first one that came up as a candidate. By making distance a relevant factor in localization significantly improved localization results, severely reducing the instances of erroneous localization, and proving crucial to localization process.

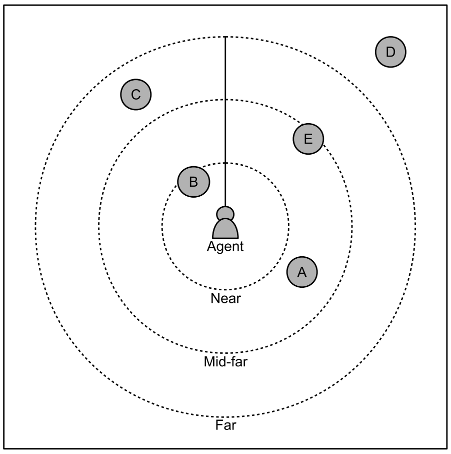


Figure 4.5: Diagram of vision ranges. Landmarks are represented by circles with labels inside.

4.3.8 Current location heuristic

The initial testing of the navigation model showed that the agent was prone to erratic behavior during transitions between two locations. The agent would often turn in an unexpected direction and go off the expected route. The cause was first clearly demonstrated in localization test results that showed highly irregular, that is, concave borders between regions which resulted in agent incorrectly assuming it was in another location, thus selecting the action appropriate for the following transition.

To remedy this a new heuristic was introduced, termed *current location heuristic*. This heuristic works by increasing the score of the current location during the matching process by a certain margin. Doing this was shown to effectively increase the area of the current location, covering the problematic border area. This allows the agent to cross the problematic location border by forcing it to move further into the next location before the location change is perceived.

The change is applied by raising the base score of the current location to the power of $(1 - x)$, where x is a real number between zero and one, inclusive. This was designed in such way in order to keep the augmented score within the normalization range. If x is set to zero, the score is raised to the power of one, providing no bonus; if it is set to one, the score becomes equal to one. This means that the bonus factor should not be set to 1 as it would break the localization process by constantly localizing to the current location.

4.4 Route planning

The navigation model described here is designed around the hierarchical nature of the cognitive map it uses. Route planning is the process that is most affected by the cognitive map, requiring more complicated algorithms that can properly utilize such hierarchical structure. The algorithm described in this section is based on the route planning algorithm described by Schmajuk and Voicu [SV06], which is designed for a hierarchical cognitive map with fixed depth of two, without a top-level region. Removing the depth restriction requires an additional algorithm, here termed the *deepest common region algorithm*, which determines the starting point of the route planning algorithm. Additionally, since the original route planning algorithm performs a single descent only, it needs to modified to perform the hierarchy descent iteratively until the bottommost layer is reached, regardless of the depth of the cognitive map or the starting point. The resulting algorithms are described in this section.

4.4.1 Deepest common region algorithm

Real world application of this navigation model are expected to have extremely large cognitive maps created through continuous exploration. Deepest common region algorithm aims to reduce the impact of the size of the map by limiting the problem to a smaller portion of the map. This is done by utilizing the structure of the map in order to find the smallest region containing both endpoints of the route.

This is done by first extracting the containment chain of both locations. A *containment chain* of a given location is an ordered list of regions starting from the universe to the parent region of that location, each region directly containing the following one in the list (see figure). The deepest common region is then the region at the largest depth contained in both of those chains. Due to properties of the hierarchical cognitive map, the deepest common region algorithm will always produce a solution, as all map units are ultimately contained by the universe region.

Additionally, the containment chain of the current location can be retained for reuse during the main navigation algorithm.

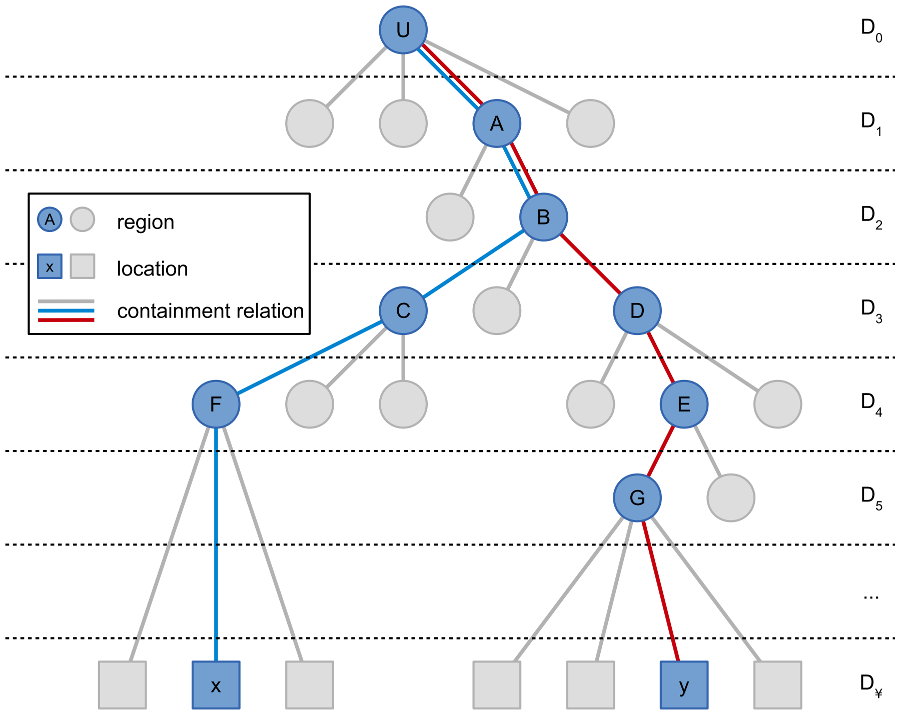


Figure 4.6: Visual depiction of a part of the map hierarchy. Containment chains of locations x and y are highlighted with blue and red color, respectively. The orientation of containment associations is always downwards.

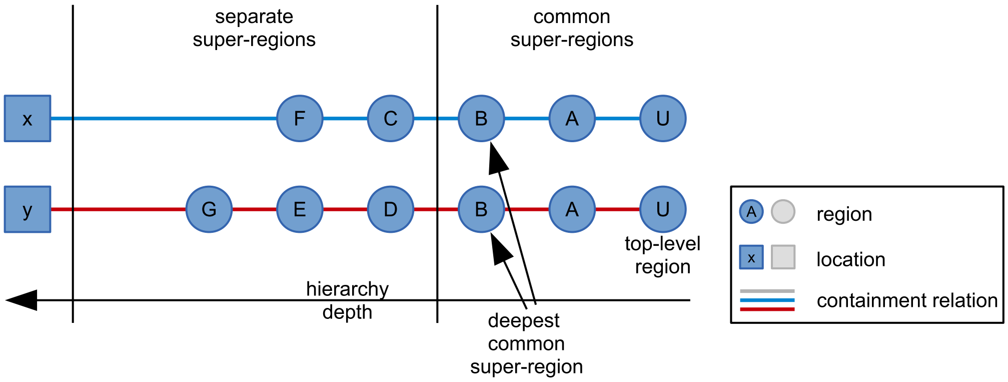


Figure 4.7: Comparison of containment chains of locations x and y. When these are represented as lists, it becomes trivial to determine the deepest common super-region.

Algorithm 4.3: Deepest common region algorithm

```
Input: current location $L_s$, current destination $L_d$
Output: deepest common region $R_{common}$
Create a region lists $RL_s$ and $RL_d$ for $L_s$ and $L_d$ respectively;
foreach $L_s$, $L_d$ as $L_x$ do
    Set the parent region of $L_x$ as the current region $R_{current}$;
    Add $R_{current}$ to the beginning of $RL_x$;
    while $R_{current}$ is not $R_{Universe}$ do
        Set the parent region of $R_{current}$ as new $R_{current}$;
        Add $R_{current}$ to the beginning of $RL_x$;
    end
end
Let $i := 0$;
while $RL_s[i] = RL_d[i]$ do
    Let $R_{common} := RL_s[i]$;
    $i++$;
end
Return $R_{common}$;
```

4.4.2 Route-planning algorithm

After having found the deepest common region the main phase of the navigation algorithm can begin. This algorithm recursively descends the map hierarchy while executing an arbitrary[3] routing algorithm on each depth level. The crux of this algorithm is determining the source and the destination nodes for the current depth level. This process will be explained in details along with examples below.

Note that a "map unit", represented by the symbol U means that a variable can represent both a region and a location. This is important as both regions and locations can represent nodes for the routing algorithm on their respective level. Locations are always and exclusively on the lowest level of the hierarchy.

Additionally, calculating a path within a set of map units means considering those map units as vertices and the explicit and implicit connection relations between them as edges of a simple directed graph upon which a routing algorithm can be performed.

Step 6 of the algorithm checks whether the agent is about to transition into another region. This will occur once the agent is located in any location directly connected to a location from the region that lies next along the path to the destination. In this case running the routing algorithm is meaningless; the algorithm instead returns one of the locations from the next region that the current location is directly connected to.

[3] This may be *any* routing algorithm that finds a path to the destination, even one that finds a suboptimal solution.

Algorithm 4.4: Route-planning algorithm

```
Input: current location $L_s$, current destination $L_d$, deepest common region $R_{common}$
Output: next step $L_{next}$ on the path from $L_s$ to $L_d$
Let $R_{current} := R_{common}$;
Let $S'_d$ be a set containing the map unit in $S_{level}$ that contains $L_d$;
do
    Let $S_{level}$ be the set of all children map units of $R_{current}$;
    Let $U'_s$ be the map unit in the $S_{level}$ that contains $L_s$;
    if $U'_s$ is in $S'_d$ then
        Let $S_{next}$ be the set of all map units contained by $U_{next}$ that $U'_s$ is connected to;
        Let $U_{next}$ be any one of the map units in $S_{next}$;
        Return $U_{next}$ as $L_{next}$;
    else
        Find a path $P(s, d)$ between $U'_s$ and any of $U'_d$ within $S_{level}$;
        Let $U_{next}$ be the next vertex along the path $P$;
        if $U_{next}$ is a location then
            Return $U_{next}$ as $L_{next}$;
        else
            Let $R_{current} := U'_s$;
            Let $S'_d$ be a set containing all map units in the $R_{current}$ that are connected to
            any map unit in $U_{next}$;
        end
    end
loop
```

The result of this algorithm is a location directly connected to the agent's current and represents the next step along the path to the destination. Using the pair (current location, next step) the decision unit can query the scenarios stored in the long-term memory to obtain the action that performs the transition from one to the other. Once the action starts being performed the navigation algorithm is suppressed until the localization routine returns a different current location, upon which the navigation algorithm is needed to obtain the new action.

4.5 Actions

In any navigation model, all calculated routes must ultimately be translated into sequences of actions whose execution facilitates traversal of the environment. In any navigation model actions describing transitions between locations must be based on environmental features that are perceived by that model; furthermore, the agent need to be able to perceive relations between itself and the features. Specifically, in a landmark-based model, such as the one being described here, actions have to be defined with respect to landmarks.

4.5.1 Landmark-based actions

In order to properly model landmark-based actions several assumptions regarding agent's abilities must be made:

- Agent is able to move in a straight line
- Agent is able to execute arbitrary turns
- Agent is able to focus any perceived landmark
- Agent is able to face area between any two visible landmarks
- Agent is able to avoid local obstacles

The first and the second assumptions are considered to be always satisfied, as rotation and translation are motions essential to traversal of two-dimensional environments, regardless of the actual method of executing these motions. These two assumptions allow two atomic actions, *GO FORWARD* and *TURN LEFT/RIGHT*, to be executed. It should be noted that the latter action has a certain delta value which is small, may or may not be fixed, and innately unknown to the agent.

The third assumption is made about the perception of the agent and stipulates the agent can detect when it's facing a certain object and react upon it. This allows for complex action *TURN TOWARDS* to be executed by defining the stopping condition of the *TURN LEFT/RIGHT* action.

The fourth assumption is important as is serves as a foundation for the replacement to the concepts of turning by an exact amount of degrees. As shown in the paper by Waller et al. [WLGB00], humans estimate most angles rather poorly, which makes encoding instructions using exact angles very unlikely. In addition, one cannot rely on any kind of fixed degree turns as doing so would make proper execution of the turn depend on the initial angle of approach to the point of turn, which cannot be guaranteed always to be the same, and consequently lead to potentially breaking navigation errors. In real world it is impossible to define such turns without a *third reference point*, the other two being the referred landmark and the agent itself. The proposed solution is to face a virtual mid

This problem is somewhat mediated if *paths* are recognized as environmental features by the model. The landmark-based model is compatible with path-based models, and this possibility is further expanded upon in section 6.6.4.

This and the previous assumption imply that a navigation action can successfully be executed only if the landmarks it is based on are visible. This is an interesting implication that places further limitations on the types of environment this approach can work in. Such approach is nonetheless consistent with human behavior.

The final assumption allows the model to forgo obstacle avoidance and instead solely focus on global navigation. Obstacle avoidance doesn't require neither localization and navigation and can therefore be implemented independently. Additionally, since the goal deliberation is performed at every execution step, actions provided by the navigation routine can be suspended in favor of those that describe obstacle avoidance. Once the obstacle has been overcome, the last action can be resumed provided it still has the highest priority.

4.6 Model integration

Figure 4.8 shows the part of the decision-making strand of the current ARS model that contains the navigation model. The processes of localization and route planning are implemented separately: localization implemented as a module, and route planning as a subroutine. Localization and route planning are usually coupled together for the reasons stated in chapter 2. Despite that they can be implemented uncoupled and/or separated as long as they use the same cognitive map and the output of the localization process is ultimately passed to route-planning process. The separate approach has been taken, as the result of the localization process can be used by the decision unit *before* the decision on whether to run navigation process or not is made, as it can potentially influence the goal selection.

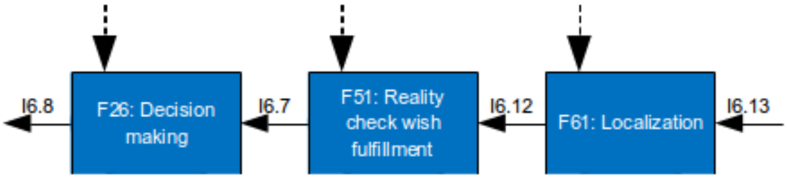


Figure 4.8: Section of the ARS functional model depicted in figure 3.6.

4.6.1 Representation of concepts

In order to successfully integrate the navigation model into the ARS model, an appropriate representation of used concepts and data structures must be defined.

Map units, locations and regions, can be viewed as atomic that form the cognitive map using connections between them. This makes the word presentation an appropriate representation of these concepts in the system and, consequently, word presentation associations the appropriate representation of relations between them.

Additionally, the model requires some means of representing the agent with respect to the environment, both locally, allowing expressing its relationship with environmental features in the perceived scene; and globally, by placing itself in relation to the cognitive map. It is therefore assumed here that there exists a way of representing the agent as a word presentation mesh (WPM).

Concept	Represented by
Location Region Landmark Action (Act) Route (Scenario) Region contents Containment chain Scene	Word presentation mesh
Connectedness relation Containment relation Landmark-Scene relation Scene-Location relation Entity-Location relation Agent-Location relation	Association

Figure 4.9: Overview of representations of localization and navigation concepts in ARS

Relation	Predicate	Content types	Interpretation
Connectedness	IS_CONNECTED_TO	Location, Location	Location is connected to Location
Containment	CONTAINS	Region, Map unit	Region contains Map unit
Landmark-Scene	IS_IN	Landmark, Scene	Landmark is in Scene
Scene-Location	DESCRIBES	Scene, Location	Scene describes Location
Entity-Location Agent-Location	IS_AT	Entity/Agent, Location	Entity/Agent is at Location

Figure 4.10: Overview of association predicates introduced into ARS

Following are tables that summarize the WPM content types and association predicated required by the navigation model.

4.6.2 Localization module

Self-localization routine is contained within is own module F61 Localization. The module receives symbolized perceptual information as thing and word presentations from the primary process via interface *I6.13*. In addition the module has direct access to the data storage containing the cognitive map from which it obtains current location candidates. If successfully localized, the resulting current location represented by its word presentation is passed back into the decision-making strand along with all other input information via interface *I6.12*.

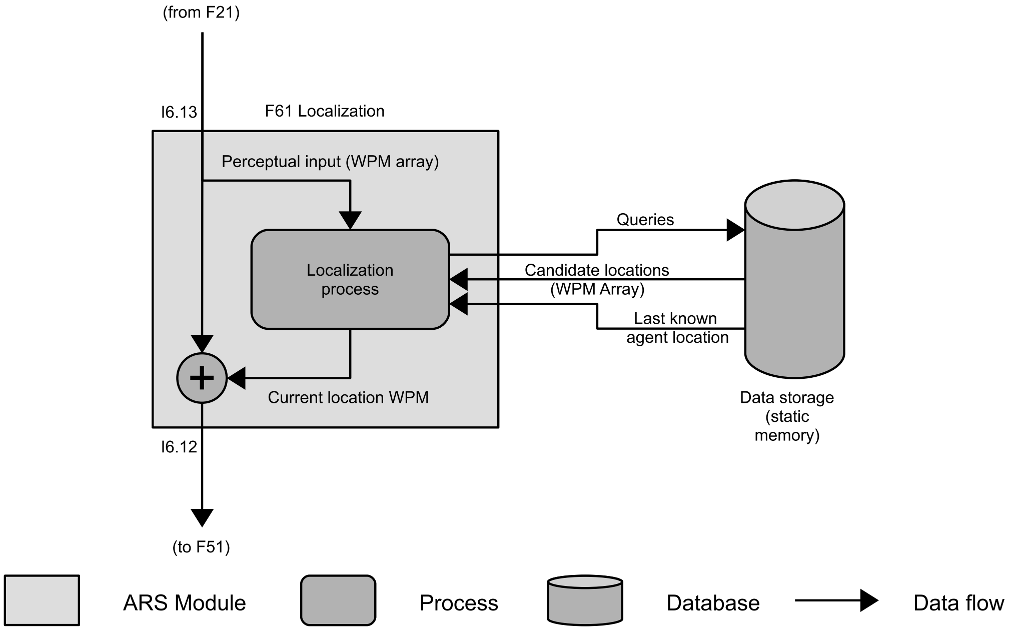


Figure 4.11: Diagram and data flow of the F61 Localization module

4.6.3 Route planning subroutine

Decision unit module (F26) is the central component of the decision-making strand of the ARS architecture. This module contains the algorithm that deliberates the current goal based on perceived input, internal drives and emotions, passed from the primary process and subsequently further processed by preceding modules of the secondary process. If this goal is out of reach, that is, not at the current location, the module has to fist determine one or more locations the goal is expected to be at, according to the knowledge at hand, and then retrieve routes to those locations. As a result of that, route planning is integrated directly into the decision unit module (F26) as a subroutine. Route planning is triggered every time a route to a distant goal is required. The decision-making unit then runs the route planning subroutine to obtain stored scenarios that result in agent reaching the set goal. Once the goal has been reached the route planning subroutine is skipped and instead scenarios are chosen that are required to interact with the goal in the intended manner.

4.6.4 Action representation

The cognitive map of the navigation model uses actions as connections between locations. At the first sight this might suggest that the proper representation of an action in the ARS would be an association, as they are analogously used to connect WPMs. This is not possible, however, as the definition and implementation of associations in the ARS does not allow for any additional data or data structures to be attached to them beside the predicate and the weight.

One possible solution to this issue is to represent actions using a WPM and two associations, which connect two WPMs representing locations to the WPM representing the action. This emulates the structure of a single association whilst allowing for a data structure to be contained within it. This approach does have some downsides, as it makes the structure more complex and also harder to retrieve from the memory. To retrieve an action connecting two locations, one must thus retrieve all word presentations directly associated with both location word presentations, and then identify which one of the resulting word presentations represents the desired action.

The better alternative is to rely on existing WPM types to achieve the desired representation. The ARS model provides acts and scenarios as means of representing remembered interaction with the environment. As any other WPM, they can be associated with a location WPM. Since explicit associations between locations and their scenes already exists in the model, it is reasonable to assume the scenarios activated by the current perceptual image would correspond to the current location. Scenarios consist of a sequence of actions, termed *acts*, which in turn can also be associated with locations. Scenarios are thus able to span multiple locations, which makes them equivalent to routes from a causal standpoint. This means that a scenario whose first act is associated with the current location, and some following act to the location of the goal, can be used to reach that goal by recreating that scenario in real world.

CHAPTER 5

Implementation

The implementation of the navigation model described in the previous chapter will be described here. The process of implementation was done in three steps. Firstly, a new simulation environment was created to create platform in which the implementation can be developed, tested, and debugged. Secondly, a proof-of-concept implementation was developed which was used to validate the generalized versions of the algorithms used in by localization and path planning processes using flexible open-source graph libraries. Thirdly, algorithms and data structures implemented in the second step were modified in order to make them analogous to the data types and interfaces of the ARS. It must be noted that the complete integration could not be achieved due to the state of the ARS implementation at the time.

5.1 Simulation

In order to to test the validity of the navigation model, a proper environment must be provided where various scenarios can be created, executed, and where the functioning of the model can be observed and evaluated. To this end a simulated environment has been created with following properties:

- the environment is a continuous two-dimensional space
- the environment is populated with environmental features - landmarks - represented by static entities
- the agent employing the navigation model is represented by a mobile entity within that space with sufficient degree of freedom of movement
- the agent has to be able to perceive environmental features and their configuration in accordance with the principles of the ARS model

- additionally, the environment contains obstacles - environmental features necessary for more proper modeling of agent's perception through vision occlusion

The tool chosen for implementation of such simulated environment is the Multi-Agent Simulator of Neighborhoods (or Networks) (MASON) library [LBP$^+$03]. It was chosen in part because of the current implementation of the ARS was developed in the same library. By using the same simulation library one can hopefully simplify the future integration into the ARS model. Additionally, MASON library has following features that were particularly suited for this purpose:

- sharp learning curve facilitated by in-depth documentation and a large number of examples
- a flexible and simple out-of-the-box solution for representing continuous two-dimensional environments
- an array of customizable portrayal classes that allow separate rendering based on the type of the displayable entity, allowing the visualization of the environment and the agent, and other important information in a clear and intuitive manner
- a user interaction API that allows mouse interaction with the visual representation of the simulated environment, necessary for moving the agent entity about without the need to reset the simulation
- ability to inspect and modify the inner state of the agent via the simulator control panel allowing for on-the-fly modification of simulation parameters

5.2 Environment

For proper testing of the navigation model a two-dimensional continuous representation of the environment is required. MASON library provides `Continuous2D` object for this purpose, which allows mapping of any object onto a two-dimensional plain. Additionally, it provides an interface that allows various kinds of inspection and geometrical analysis of the state of the environment, such as getting distance between objects or determining angles defined by three entities.

Objects are placed into the environment using `0setObjectPosition` method, which accepts an object and its intended position, represented by a pair of real numbers. This method is mostly used during the initialization process of the simulation, which is responsible for populating the environment with entities according to the specifications given in the scenario file. It is also used for updating the pose of the agent entity.

5.2.1 Simulation entities

Abstraction of reduces the number of necessary entities in the simulation to the agent, landmarks, and obstacles. Landmark and Agent objects are extensions of the Entity object, which contains a unique identifier that allows for easier instantiation and retrieval, as well as a label, which is used for visualization purposes only. Testing requires only one agent to be present in the simulation, but the possibility of having more agents in one simulation is thus also possible. Obstacles were implemented as a separate object class as they do not need to be distinguished, and therefore need no unique identifier.

The agent entity represents the agent within the simulated environment. Apart from the identifier and the label, this object additionally contains information about the current pose of the agent with respect to the coordinate system of the environment. The agent entity is portrayed in the visualization as a green dot, with three concentric rings representing segments of its field of view and a straight line representing its orientation.

Landmarks are portrayed in the simulation visualization as grey dots with labels. Labels add semantics to landmarks, which facilitates easier comprehension of the visualization of the simulation.

Despite not actually being a part of the simulated environment, locations are also present in the visualization in order to give an overview of the locations encoded in the agent's cognitive map. These are portrayed by blue dots with labels. Their only purpose is to serve as map reference to the user controlling the simulated agent, so that they would know which location they want to move the agent toward. Their position in the environment is purely arbitrary and of no consequence to the agent's behavior. The position is defined in the scenario editor described in section 5.4. Similar to landmarks, they can also have labels to help distinguish them from one another and allow the user to select destinations by their label, instead of by their unique identifier. As the locations are parts of the cognitive map, they don't share the same parent class as the entities.

5.3 Agent simulation

The implementation of the agent consists of two parts, the body and the mind. The *body* is the abstraction of the actual embodiment of the agent and is responsible for communication between agent's mind and the simulated environment which is facilitated by the simulation interface. Agent's body can further be divided into sensors, which are responsible for receiving input from the environment, and motility control, which control the pose of the agent entity within the environment by translating the actions passed from the mind.

Sensors are responsible for retrieving raw data from the simulated environment via the simulation interface, converting it into qualitative perceptual input and passing to the decision unit. The raw information should not directly accessible by the agent's intelligence as that would violate the principles of the navigation model.

Sensors perform what is an equivalent to the visual detection of landmarks by employing the visibility and distance calculations. Visibility calculation is performed for each landmark inside the vision range of the agent (see section 5.3.3 for more details), while distance estimation is performed by comparing the relative distance of the landmark to distance of visibility ranges, assigning the qualitative distance information accordingly.

Collision detection sensors are not modeled, as this capability is a part of the local navigation process, which is here treated as distinct from the global navigation model.

The *actuators* are responsible for translating the actions provided by the decision unit into movement. This is done by analyzing the action data and executing either translation or rotation of the agent entity by modifying its position and orientation data through the simulation interface. Translation is done in small increments in the direction the agent entity is oriented towards the time. This increment is constant, as the velocity change is not a relevant factor when testing the navigation model. This increment does put an implicit constraint on the environment itself, forcing the size of all locations to be such that the agent cannot skip over any of locations in one simulation step.

According to the definition of agent's actions given in section 4.5, rotation is performed relative to currently visible landmarks, by either facing a landmark directly, or facing an imaginary midpoint on a line between two landmarks. This action is performed instantly, as gradual rotation is only relevant in cases where an agent is capable of simultaneously performing translation and rotation. Updates of the environment are performed through the interface provided by the AgentEntity object, which updates the inner pose as well as the position in the simulated environment.

The ARS architecture was thus abstracted to include visual sensors, motility control and the part of the decision unit that was appropriated for the localization and navigation functionality. The decision-making process has been abstracted away in this simulator, delegating it entirely to the user by providing a simple navigation control interface through which one can pass desired destination directly to the navigation process.

Although this removes most of the defining features of the ARS model, such as most of the decision making and the implementation of emotions, it still allows for and easier subsequent integration into the model by making sure the functionality is properly decoupled from the model through proper definition of intermodular interfaces.

Agent's *mind* consists of the implementation of the navigation model. It is further split into *decision unit*, which runs the algorithms of the model, and the cognitive map, which contains the preset knowledge loaded from the scenario file.

5.3.1 Cognitive map implementation

The hierarchical cognitive map as described in this thesis can be satisfactorily modeled by a simple directed graph. The simplest method of implementing a cognitive map is to use a readily available solution for handling graphs in the language of choice. This solution must provide simple and intuitive methods for efficiently retrieving parts of a graph and testing various properties of its elements and relations between them. The chosen solution is the Java Universal Network/-Graph Framework) (JUNG) library, which is "designed to support a variety of representations of entities and their relations, such as directed and undirected graphs, multi-modal graphs, graphs with parallel edges, and hypergraphs" [OFWB03].

A particularly useful feature of the library is the ability to use any kind of objects as both vertices and edges of graphs, as well as allowing them to belong to multiple graphs simultaneously. The structure of the graph is maintained by various graph classes, which provide interfaces for graph modification and queries of vertices, edges, and various graph properties and substructures.

The ability to define multiple graphs on the same set of vertices allows for a very efficient execution of the navigation algorithm. The cognitive map is observed as a hierarchy, that is, a directed tree graph when algorithm traverses levels and as a different simple directed graph when finding a path on a certain level. This is ensured by separating the edges representing different kinds of associations into different graphs.

The decision to use this library for the implementation of the cognitive map was made in order to simplify the implementation of the agent, as the cognitive map is the most complex data structure of the model, so using a reliable library to perform graph queries was deemed the most practical solution. Since the compatibility of this approach with the ARS data structures is not immediately clear, additional conversion methods were designed in order to demonstrate this compatibility (see section 5.3.2).

5.3.2 Memory retrieval

Even though it would be possible, and also significantly simpler, to implement a localization and navigation routine as separate modules that use their own internal knowledge base, as opposed to using the shared long-term knowledge base of the ARS, this approach carries several disadvantages. By separating the cognitive map from the rest of the knowledge base one loses the benefits offered by data structures of the ARS. A unified knowledge base allows emotions and drives to be associated with regions and locations, enabling a true emotion-based approach to navigation that would avoid unpleasant parts of the environment based on the associations formed in the memory. Moreover, associating a potential goal with its location in the cognitive map makes determining the agent's destination trivially simple, that is, as simple as following an association.

Route planning algorithm requires complex data structures: region children graph and the hierarchy chain. However, the only method of memory access currently available returns a single WPM along with all of its internal and external associations. Retrieved associations are hereby assumed to be both the associations that are starting from this WPM as well as those ending in it. This section will explain how complex structures can be retrieved from the memory using the above mentioned method only.

It should be noted that individual required WPMs can be retrieved from memory *ad hoc* during the execution of the route planning algorithm using the `searchMesh` method, if the implementation allows it. The proof-of-concept implementation of the navigation model, however, utilizes graph libraries for realization of the cognitive map for efficiency and simplicity. The simple algorithms listed below therefore serve as a proof of compatibility of with the ARS only and are not designed as a substitute any efficient retrieval methods that may come in the future.

Region children graph consists of all children subunits of a certain region, which can be either locations or other regions, along with all connectedness associations that exist between them. The algorithm that can be used to obtain this structure is shown in listing 5.1. The last step of this algorithm is optional and can be skipped if the implementation of the route finding algorithm is capable of working with ARS data structures. The algorithm for obtaining the hierarchy chain is shown in listing 5.2. This algorithm can also be used to determine if a subunit belongs to a neighboring region, which is required in the path finding algorithm.

Algorithm 5.1: Children connectedness graph retrieval algorithm

```
Input: parent region WPM $R$
Output: directed graph $G$
Let $S$ be an empty set of map units;
Search memory for complete WPM $R'$ using $R$ as pattern;
foreach external association $A$ of $R'$ do
    if $A$ is of type $CONTAINS$ with $R'$ as source WPM then
        Let $U$ be the destination WPM of $A$;
        Search memory for destination WPM $U$ and add it to $S$;
    end
end
Construct directed graph $G$ WPMs in $S$ to nodes and associations of type
$CONNECTED\_TO$ as arcs;
```

5.3.3 Simulation of visual perception

As the model is designed for navigation in two dimensions, the simulated environment is two-dimensional, and so the vertical aspect of the field of view has been accordingly ignored. The most important feature of the simulated vision is the 360 degrees field of view. This represents the most apparent deviation of the simulated vision from human vision, which has a total field of view of about 200 degrees [Hen93, p. 2].

The main factor is the ability of a human agent to track the position of several recently seen objects that are no longer visible, even while moving [HBF$^+$06]. By being aware of the existence and the approximate location of recently seen features the agent can integrate them into the overall perception, inferring their configuration with respect to visible features to a given extent, making them effectively visible for purposes of localization.

Another helpful factor is the movement of the eyes in humans. When accounting for rotation of the eyeballs around their vertical axis, the sum total human field of view becomes larger. If

Algorithm 5.2: Hierarchy chain retrieval algorithm

```
Input: map unit WPM $U$
Output: region list $L$
Let $L$ be a list of map units;
Search memory for complete WPM $U'$ using $U$ as a pattern;
do
    Let $A'$ be null;
    foreach external association $A$ of $U'$ do
        if $A$ is of type $CONTAINS$ with $U'$ as destination WPM then
            Set $A' := A$;
        end
    end
    if $A'$ is not null then
        Set $U'$ := destination WPM of $A'$;
    else
        Exit loop;
    end
loop
```

the head and the body rotation are considered as well, this field expands to cover a full circle, or 360 degrees angle. These movements can be executed quite quickly, and in conjunction with the feature tracking above, it can be assumed that the perceptual field can be refreshed in time roughly corresponding to the length of a step in the simulation time, or, at worst, the time between two executions of localization, effectively providing the agent with a 360 degrees field of view for navigation purposes.

The field of view is divided into three ranges, separated by concentric rings centered in the agent entity: near, medium and far. These ranges emulate the perception of distance by assigning a qualitative distance value to any perceived landmark. The distance is measured using the positions of two given entities within the simulations space.

5.3.4 Obstacles and vision occlusion

Occlusion is a common phenomenon in real world environments, where environmental features are hidden by other environmental features. Since landmarks in this simulation are represented by points, and therefore have cannot cause occlusion, linear *obstacles* are introduced into the simulated environment as a simple way to simulate occlusion. They are defined by a pair of landmarks, and can be either an occluding or non-occluding. Since non-occluding obstacles do not obscure vision they are a purely cosmetic addition to enrich the visualization of the environment.

This allows a landmark visibility check to be performed by determining whether the line segment defined by the landmark and the agent intersects any of the line segments defined by occluding obstacles. This is done using implementation of the openly available Python algorithm translated into Java [Boe]. The details of the algorithm are provided at the same source and will not be mentioned here for the sake of brevity.

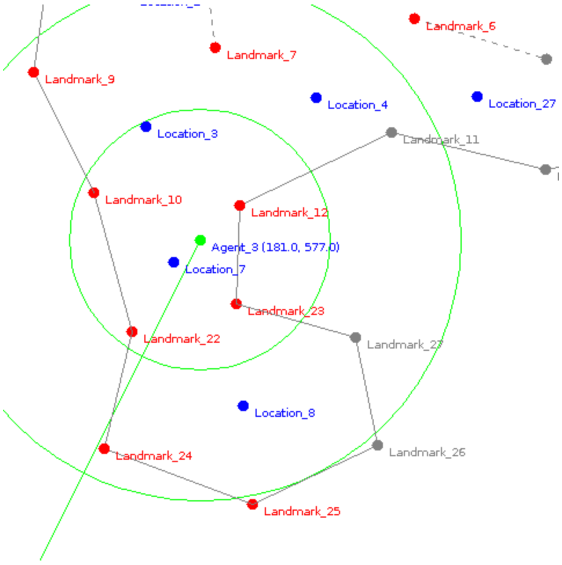


Figure 5.1: A run-time example of occlusion. Visible landmarks are highlighted red.

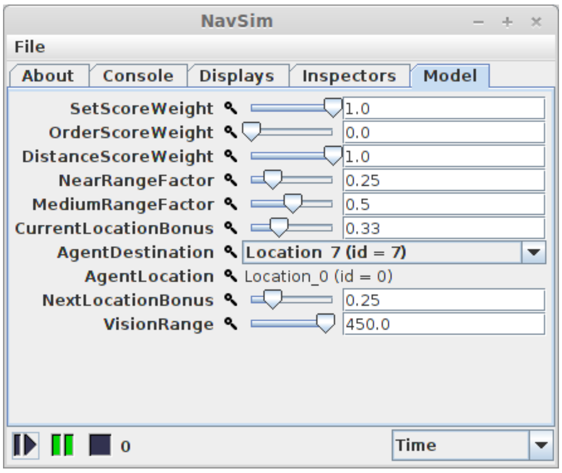
The visibility check is performed only on landmarks that are within the vision range of the agent. Similarly, only those obstacles that are defined by at least one of the visible landmarks are taken into account during the visibility check. Visible landmarks are highlighted in red in the simulation visualization.

5.3.5 Simulator interface

This interface that facilitates simulation control for the user is the part of the simulator control panel. It is created by allowing simulator inspectors to read or modify various public variables provided by the agent class. The appearance of the control panel is shown in figure 5.2.

The panels consists of the simulation parameter control, the navigation control, and the simulation control. Simulation parameter control features sliders that allow various parameters of the agent's mind and body to be modified on the fly. Available are: controls for heuristic score weights, vision range controls, and the current and next location bonus heuristic weights.

Navigation control consists of a dropdown box containing a list of all locations and regions stored within the agent's cognitive map. Selecting a location or a region triggers the navigation process using selected map unit as the current destination, which is abstracts the decision-making process from the standpoint of navigation. Additionally, there is a display showing the agent's current location as determined by the localization process.



SetScoreWeight	1.0
OrderScoreWeight	0.0
DistanceScoreWeight	1.0
NearRangeFactor	0.25
MediumRangeFactor	0.5
CurrentLocationBonus	0.33
AgentDestination	Location 7 (id = 7)
AgentLocation	Location_0 (id = 0)
NextLocationBonus	0.25
VisionRange	450.0

Figure 5.2: MASON agent control panel showing agent parameter controls.

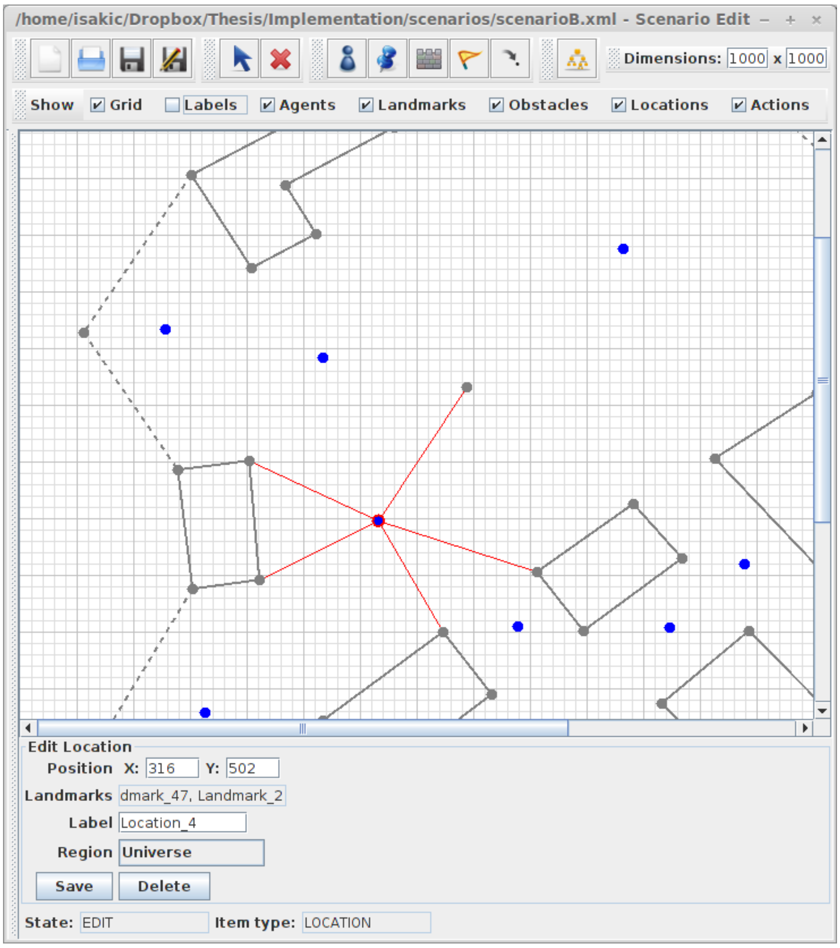
The control panel also provides the controls for starting, pausing, stopping, and resetting the simulation run, which are available by default and present in every control panel.

5.4 Scenario editor

Testing self-localization and navigation necessitates testing scenarios. Creating scenarios in this particular case involves positioning landmarks and obstacles in two-dimensional environment. In addition, due to lack of learning ability agent needs a predefined cognitive map that matches that environment. In order to avoid having to encode scenarios manually a *scenario editor* was developed. This editor allows creation of arbitrarily large testing environments through a simple graphical interface inspired by graphical processors, featuring a point-and-click mouse interface, toolbars and edit dialogs, as well as a separate view for editing the hierarchical structure of cognitive maps.

The editor supports addition, editing and removal of five types of objects: the agent, landmarks, obstacles, locations, and actions. The default selection mode allows the user can click on any object and edit its properties through an edit dialog that appears upon selection. Objects such as the agent, landmarks, and locations can also be freely moved within the environment by dragging them with the mouse.

Obstacles, locations, and actions are defined by landmarks, and their creation necessitates selecting defining landmarks. Objects can be removed by selecting the removal action and then clicking on an object in the editor. Any other objects that depend on the deleted object, such as obstacles depending on a landmark, will be deleted along with the selected object.



/home/isakic/Dropbox/Thesis/Implementation/scenarios/scenarioB.xml - Scenario Edit

Dimensions: 1000 x 1000

Show ☑ Grid □ Labels ☑ Agents ☑ Landmarks ☑ Obstacles ☑ Locations ☑ Actions

Edit Location

Position X: 316 Y: 502

Landmarks dmark_47, Landmark_2

Label Location_4

Region Universe

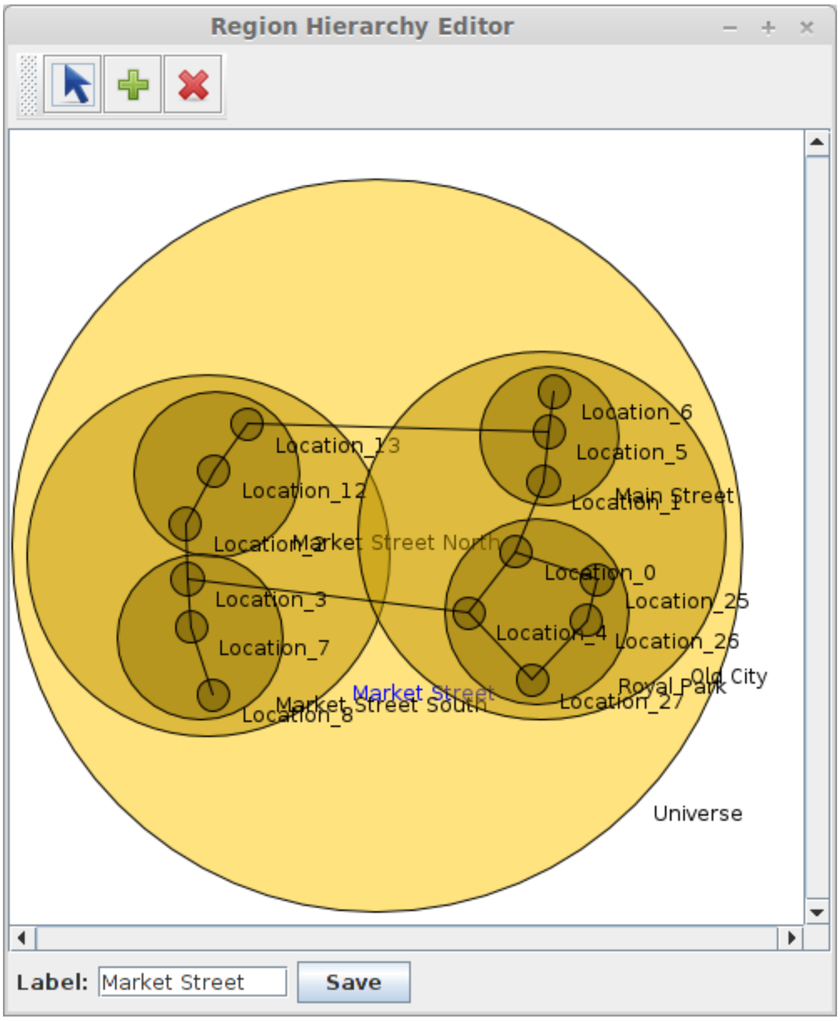
State: EDIT Item type: LOCATION

Figure 5.3: Main view of the scenario editor.

5.4.1 Region hierarchy editor

Region hierarchy editor is a separate view of the scenario editor used to define hierarchy of the cognitive map associated with the scenario. It takes all locations and their interconnections defined in the scenario editor and displays them as circles connected by lines. Regions are represented as circles as well, but they are distinguished from locations by being able to contain other region and locations. When editing the hierarchy for the first time there is only one region defined, the “Universe” region, which contains all locations defined in the scenario editor.

Creation and removal of regions is facilitated through the mouse interface and activated by selecting the appropriate action in the toolbar. Removing a region will place the contents of the deleted region within its parent region. This behavior maintains the consistency of the hierarchy, making it impossible to create an invalid cognitive map by removing regions. This also means that the “Universe” region cannot be removed.



Label: Market Street

Figure 5.4: Hierarchy view of the scenario editor.

Locations and regions can also be moved and arranged freely within the editor using mouse. The graphic representation of regions will automatically resize to maintain the proper representation of containment relations. This allows creation of clearer graphical representations of hierarchies, but has no impact on the actual locations within the scenario editor. Positions of all elements within this editor are saved in the file along with the rest of the data. As such, they are completely optional and can be removed from the file without breaking the cognitive map.

5.4.2 Scenario files

Scenarios files store all data necessary for the simulation. Scenario files consists of two main parts: environment description and the cognitive map description. Environment description contains information about the dimensions of the environment, and the description of all entities within the environment: agent, landmarks, and obstacles. Cognitive map description consists of location, region, and action descriptions. These are defined using landmarks defined in the environment description. Entities, locations, regions, and actions all possess unique identifiers that are set automatically by the editor and are invisible to user. They are also used for landmark identification by the perception and the localization algorithms. Additionally, each location and region description is supplemented by their position in the hierarchy editor view, which allows preservation of the layout in that view.

Scenario files are read and written using a custom-made loader class, which is used both by the editor and the simulator. XML format was chosen as it is easy to generate and parse using standard Java libraries for XML. The downside to this approach is the large size of created files caused by expressive syntax of the language. A listing of a scenario file schema is listed in Appendix B.

CHAPTER 6

Results and Discussion

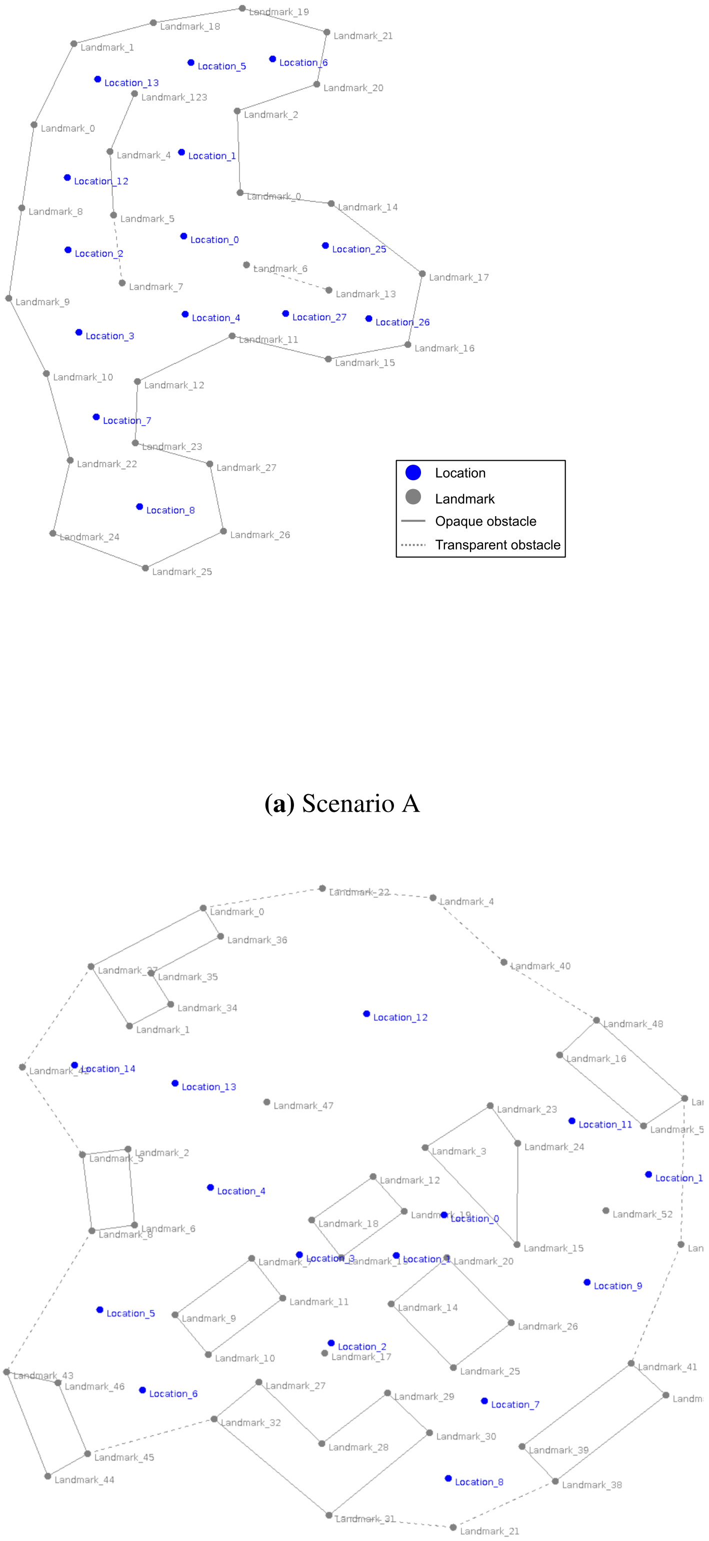
Having described the implementation of the simulation in the previous chapter, a description and discussion of test scenarios and results obtained will be provided in this chapter. Tests have been performed separately for the localization and route planning processes and are provided here separately. Finally, concepts and functionality that are necessary or beneficial to the navigation model will be mentioned in the future work section, followed by the summary and the conclusion of this thesis.

6.1 Test results generation

The implementation of the model and the simulation offered proper environment for testing, but lacked means of representing results in an intuitive way, such that would be appropriate for static post-run analysis and presentation. In order to solve this problem the implementation code was augmented with two additional features for test result generation of localization and route planning processes, respectively.

6.1.1 Localization result visualization

For purpose of generating test results for the localization process a separate visualization method was developed that operates outside of the simulation run time, that is, uses the existing inner simulator functionality statically. After the simulation environment has been initialized, the method directly controls the position of the agent, moving it in a scanning fashion across the two-dimensional space of the simulation in small increments. At each position the agent is instructed to perform localization. The results are mapped to a fixed RGB color palette, assigning a unique color to each recognized location. The same position is then marked by a rectangle of the same color, with dimensions corresponding to the increment so that in the end the whole simulation space is covered.



(a) Scenario A

(b) Scenario B

Figure 6.1: Test scenarios visualized in MASON. Legend applies to both figures.

Resulting images are essentially a mapping of internal environmental representation onto a two-dimensional grid via the localization process, which allow *post-hoc* visual analysis of the performance of the localization algorithms. By decreasing the displaced increment one achieves greater resolution at the expense of the increased execution time, which also depends on the area of the simulation space.

6.1.2 Route planning result visualization

The idea behind the visualization of calculated routes is to allow the agent to leave "footprints", or "breadcrumbs", as it traverses the environment. To achieve this, a simple placeholder object was added to the simulation. This object has a portrayal class attached to it which renders each object as a purple dot. These objects are generated each time the agent moves, thus leaving a visible trail behind it. The generation of the footprint is not triggered when the agent is moved using mouse, so that the agent's position may be reset freely without ruining the visualization. Traversed paths are then documented by taking snapshot of the simulation visualization.

6.2 Test scenarios

During the development of the implementation several smaller test scenarios were developed for testing, debugging, and calibration of the simulation environment. These will not be covered in this section, as they are either too simple or not realistic enough to produce usable results.

The goal behind selection of test scenarios was to test the impact of different configuration of landmarks in the environment on the overall accuracy and performance of the model. This was made primarily with the intent to test and verify the localization process, as the route planning process was expected to work regardless of the layout, so long as the cognitive map was constructed properly and the localization process completed successfully. The selection was ultimately narrowed down to two scenarios, which will be referred to in the rest of the chapter as scenario A and scenario B.

Since the execution time of the visualization method for localization results mentioned above grows quickly with the size of the environment, the dimensions of test scenarios were kept relatively small, 1000x1000 pixels[1], in order to allow for generation of larger sets. Additionally, the increment was was set at 5 pixels, which allowed for creation of images of acceptable resolution while keeping the rendering time under two minutes.

Scenario A depicted in figure 6.1 features simulates a small section of an urban-like environment similar to an old city center, and is populated with evenly distributed landmarks with obstacles placed in a manner that simulates streets. It should be noted that the density of landmarks in a scenario is not a differentiating parameter in the true sense, as a scenario with sparser landmark distribution can be achieved by simply reducing the vision range of the agent without modifying the actual environment.

[1] The simulation was designed so that the one pixel of the visualization corresponds to one unit of length in the simulation.

Scenario B, also depicted in figure 6.1, was made specifically to test the effect of an uneven distribution of landmarks across a single scenario on the localization process. It simulates a more open-space environment akin to a village, with small streets and open areas like fields. Scenario B features several areas where landmarks are significantly more spaced out than in the other parts, which are significantly smaller and feature closely placed landmarks.

6.3 Localization results

Colored surfaces representing locations depend entirely on the remembered scene associated with it. One of more important questions regarding localization is how accurately a location can be represented using scenes, which are qualitative in nature. This can be evaluated by observing how well the surfaces that are mapped to a location correspond to the expectations.

Test results made with scenario A show good performance of the localization process in environments with evenly and closely distributed landmarks, where definition of locations is simpler and less prone to errors. This implies that the navigation model in its current state could perform well in urban environments, as long as the landmark detection and filtering performs according to the assumptions.

Tests run on scenario B demonstrate some deficiencies of the approach to localization. The results show that the larger locations, which are thus encoded with spread out landmarks, tend to be encroached upon by nearby locations that are encoded using more tightly grouped landmarks in areas where they come in contact with one another, causing the jagged borders visible in the visualizations. The cause of this behavior has been traced back to the design of the distance heuristic method that penalizes distant defining landmarks. This further underlines the necessity of improving current heuristics and reinforce them with additional heuristics in order to overcome this phenomenon and allow locations of any realistic size to be recognized and encoded properly.

6.3.1 Impact of parameters

Simulation parameters that were considered for testing were the score weights for the three available heuristics, and the near and far-mid vision range ratios. In addition, the effects of the current location heuristic was observed separately by presetting a chosen location, then generating test results to observe the differences in size of locations.

The results show that the variation of the heuristic score weights only had a significant effect on the overall score when the scores were set to zero as opposed to one. This section will thus only discuss results where one or more heuristics were turned off, that is, having its score weight set to zero. This however does not necessarily mean that having continuous range of score weights is useless, as the weight management could be automated and dynamically modified by the navigation model itself. This possibility might be reserved for future work.

The test results showed noticeably worse performance when either the landmark set heuristic or the landmark distance heuristics were turned off, while turning off the landmark order heuristic had a negligible effect on the localization; on the contrary, having landmark order heuristic turned on produced noticeably worse results in case of scenario B, where one of the locations has been encoded using a non-enclosing landmarks, that is, those that are not on the border of the location. This produced a parallax effect in the perceived scene, causing reduction in landmark order heuristic score due to the change in the perceived order. The resulting area of the location was very irregular, encroached upon by neighboring locations in parts where the order did not match the order specified by the scene of that location. This leads to the conclusion that the landmark order heuristic is a poor choice of a localization heuristic and should be eliminated from the future versions of the model.

This also implies that neither the landmark set heuristic not the landmark distance heuristics produce good results when employed alone. In case of the landmark set heuristic the problem arises from the inability to estimate the agent's position based on perceiving landmarks alone, as they are in this case all perceived to be equidistant from the agent. This means that many candidate locations received perfect scores as long as all the defining landmarks were visible. A significant reduction in localization accuracy was also shown when using landmark distance heuristic only. The result images show extremely irregular and unintuitive location borders, but the reason for this phenomenon is not yet clear, but it might have something to do with the modeling of distance, as the serrated edges of locations correspond to distance range borders. This suggests that the distance should be modeled differently, either by improving the distance resolution or by modeling distance as a relative configuration.

6.3.2 Results in unmapped environment

The environments described by test scenarios is not entirely mapped out, which leaves parts parts of the environment next to the edge of the simulated space not mapped to any location. In the test result images provided in this chapter unmapped parts are blended out to allow for easier viewing. As the visualization code does not recognize the bounds of the simulation as specified by the user, it runs the localization algorithm outside these bounds as well, which creates some interesting artifacts in the unmasked images which will be discussed here.

The color distribution outside the bounds shows how the algorithm will continue to assume the last known location as it ventures into the unknown part of the environment, as shown in figure 6.2. This is important for the future mapping algorithm, as it will allow it to connect newly discovered and defined locations to the existing parts of the map. This should occur once the similarity of the current scene to the scene of the last known location falls below certain threshold, at which point a scene will be constructed and stored along with the new location.

Another part of test result images worth noting are the parts colored black. This represents parts of the environment where the localization algorithm gave no valid result because no landmarks are visible from there. This as evident by the shape of the nearest two location surfaces, which have a circular border to the portion colored black, centered around the closest landmark. These borders represent the farthest the agent can go into featureless space and still be able to recover

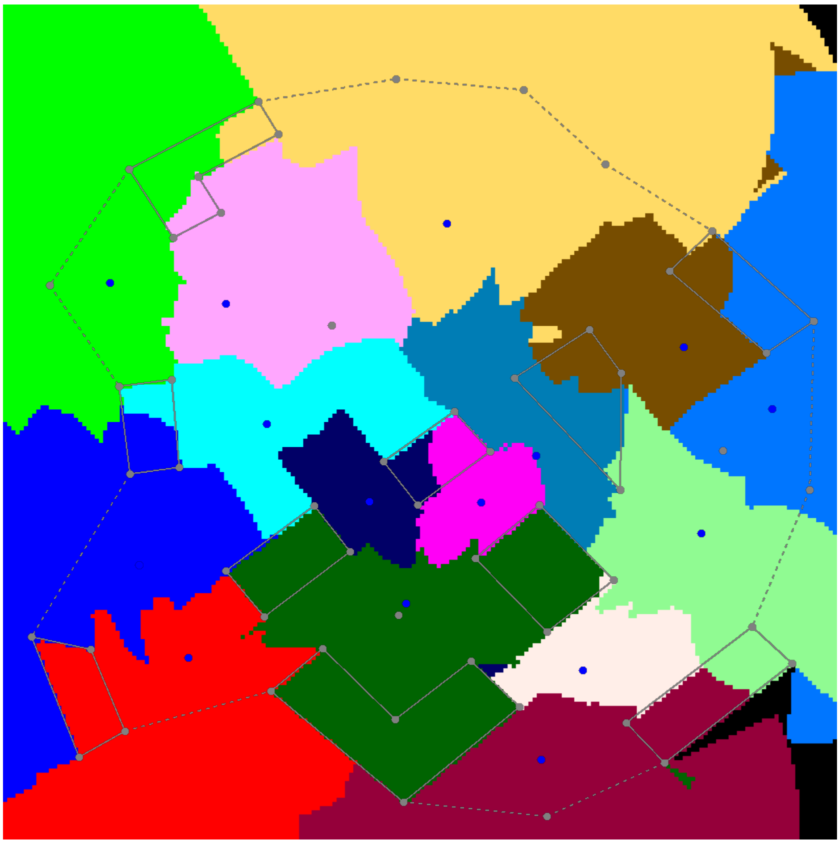


Figure 6.2: Unmasked test result image of scenario B. Black areas are present in upper and lower right corners of the image.

and return to the known environment. If the score of the last known location gets too low, and no new locations are created and added, agent should stop moving in that direction and return. The return maneuver could be something as simple as execution of a "go to landmark" action.

6.3.3 Final remarks

Overall the localization routine performed satisfactorily for the given set of test scenarios. However, the current set of implicit and explicit limitations under which it can do so shows its deficiencies and points towards possible improvement, as stated above. This leads to the conclusion that while the landmarks are essential to successful human self-localization and navigation, they are by no means sufficient.

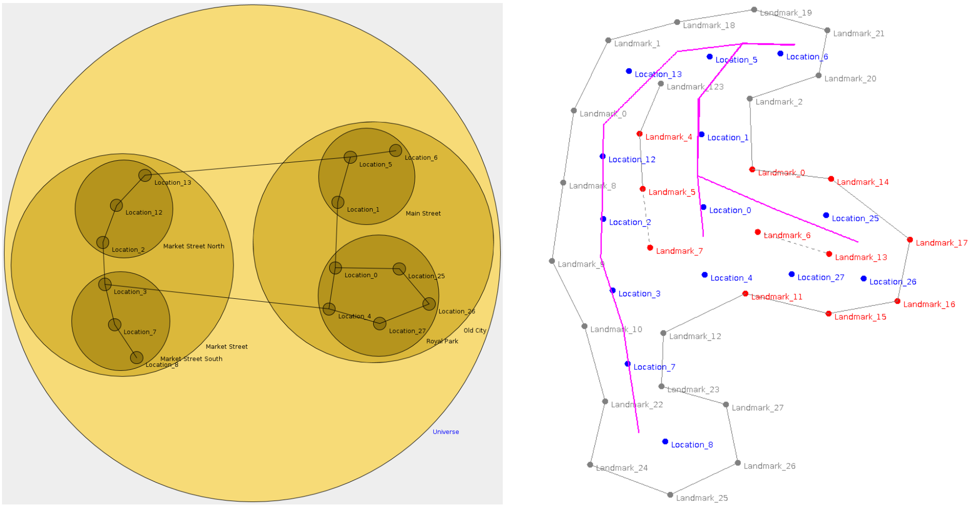
While there are no explicit rules governing the shape of these surfaces, there are some properties that are more beneficial to the localization than the others and should be striven towards. Firstly, the shapes should be *continuous*. This is fairly obvious, as a location is defined as an atomic part of the environment, and thus theoretically cannot be fragmented any further. Secondly, the shapes should be convex. Since navigation depends on the fact the agent must be able execute a transitional action from any point in the location, having locations of concave shapes would risk the agent unexpectedly crossing into a location that is not along the path and thus disturbing the navigation process.

Both test scenarios were inspired by human-influenced environment, with buildings, streets, town squares, and so on. The implicit borders did not match the human-made ones to the degree expected. This implies that humans may use some addition environment features to define limits of such locations in the environment, such as changes of the surface type, various obstacles, and generally using a more complex geometry relationships between them that were not covered by this thesis. For example, understanding that a town square is generally a wide, flat surface in a city surrounded by buildings, or that a forest is an area populated with trees would go great ways to assist an agent in determining its exact location, instead of having to rely solely on landmarks and its relative position to them.

The main problem when discussing localization based on locations such as they are defined in this thesis is the question of borders between locations, that is, where one ends and another begins, and what defines them. The images showing the localization results shown only colors of highest-scoring location, regardless of the difference in score between the best and the second-best location. In reality, such borders are hardly ever defined.

6.4 Route planning results

Route planning was tested on scenario A using the cognitive map tailored to that scenario. Additional test scenarios were deemed unnecessary, as the goal of the testing was to demonstrate the proper functionality of the algorithms involved. In order to ensure the test is complete, the map was designed so that all the novel features of the algorithm can be tested somewhere within the map, such as finding the common region, the hierarchy descent, and the region transition. A visualization of the used map is shown in figure 6.3(a), as it is seen in the scenario editor. Smallest circles represent locations, other larger circles represent regions, and lines represent connectedness relation between locations, with directions abstracted away. Additionally, containment relation is visualized literally; that is, a region containing a location will be shown as a circle containing another circle.



(a) Cognitive map for scenario A.

(b) Routes calculated in scenario A.

Figure 6.3: Route navigation test.

The test results conclusively show that navigation process performs correctly for the given scenario and map. Also shown in figure 6.3(b) are several routes, represented by purple lines, traversed by the agent whose calculation involved all of the steps mentioned above, specifically routes from location 6 to locations 4, 8, and 26. Note that the scenario was tested exhaustively; figure shows only some of the calculated routes for the sake of clarity. Results of performed tests prove the validity of the algorithm for maps of arbitrary sizes by induction.

Testing the performance of the algorithm was performed. As the algorithm was intended to function on very large cognitive maps, maps containing thousands of locations would need to be created, along with matching hierarchies, in order to be able to compare it to other route planning algorithms using maps that correspond to the intended purpose. In any event, the bottom-up calculation of routes is not desirable in the first place due to its being non-bionic in nature. Further details are provided in section 6.5.

6.4.1 Route planning failure

The process is known to fail only in situations when the cognitive map does not properly match the environment described by the scenario. This can only occur in cases when one or more locations are not convex, which may cause landmarks used for definition of the action selected by the route planning process not to be visible at the critical moment. This might pose a problem since the convex shape of locations cannot be guaranteed.

Fortunately, this situation can be avoided in several non-exclusive ways. When manually creating cognitive maps this can be achieved through location definition by ensuring visibility of action defining landmarks from all points within a location. Additionally, since actions that define transition between locations are allowed be complex in this model, it is entirely possible to add a pre-transition action that brings the agent from any given point within that location into position where necessary defining landmarks are clearly visible. Lastly, locations that have a very irregular shape should be abstracted into a region, and then split into preferably convex-shaped locations.

6.5 Discussion on route planning

By observing the inner workings of the route planning algorithm it is clear that it makes the whole the route planning process significantly less human-like, and therefore less bionic in nature than the localization process. The reason for this lies in the creation of the cognitive map.

Currently, the only way to test the validity of the route planning algorithms is to equip the agent with manually defined knowledge of a particular simulated environment. From navigation standpoint one can distinguish three general setups of agent's knowledge about the environment:

- no *a priori* spatial knowledge
- structual *a priori* spatial knowledge
- structural and routing *a priori* spatial knowledge

In the first approach one can consider the cognitive map to be empty, and is to be filled as the agent explores the environment. During exploration new locations are discerned and encoded along with the routes connecting them. This also allows for immediate storage of routes taken during exploration and their association with newly encoded locations and regions.

The second approach provides knowledge about locations and regions in the environment and their hierarchy to the agent. It also provides knowledge about connectedness of locations. This information can be understood as a set of essentially atomic routes with minimum span, that is, connecting two locations near each other. This knowledge is sufficient for the agent to employ route planning algorithms successfully, but at a significant computation cost, as no complex routes are known and need to be chained together whenever a destination has to be reached. This means that the path calculation algorithm of the route planning process is employed most extensively in this approach.

This leads to the conclusion that is far more beneficial to equip agents with knowledge of the environment that has been assembled through exploration, rather than manual encoding by operators, as it facilitates a human-like navigation behavior better than the current approach.

In reality, humans are born with no such knowledge, and instead gather their spatial knowledge by traversing the environment. Such traversal is almost always directed when the actual position of the destination is unknown. Humans obtain knowledge about the destination by being led to the destination, following verbal directions to the destination, or deriving directions and routes from external representations of the environment such as maps, city plans, public transportation diagrams, etc. New routes can also be discovered independently through exploration. If there is no destination, newly visited locations are incorporated into the spatial knowledge along with the routes taken to them. The article by Foo et al. [FWDT05] shows that this process is tied to landmark cognition. This also implies two distant locations can be integrated into the cognitive map without learning about the part of the environment between them, apart from the route itself.

This makes the second approach a “necessary evil”, employed to circumvent the deficiencies of the current simulation methods, and should be phased out once agent’s learning capabilities are sophisticated enough to correctly perform mapping processes starting with an empty map. It is therefore justified to use an algorithm based on *a priori* knowledge as a sort of a “stepping stone” towards more bionic implementations. As the learning capabilities of the ARS are further developed, so will this algorithm be phased out in favor of more cognitive-based approaches such as route learning, route integration, and route combination.

All of characteristics of human navigation mentioned place heavy emphasis on the learning capability of an intelligent agent. At the time of the writing the ARS implementation possessed no such capability, which precluded further research into this topic. As such, it is therefore justified to use an algorithm based on *a priori* knowledge of the environment to discover novel routes, such as the one developed in this thesis. As the learning capabilities of the ARS are further developed, so will this algorithm be phased out in favor of more cognitive-based approaches such as route learning, route integration, and route combination.

6.6 Future work

As the work on the model of self-localization and navigation progressed, several features lying outside of the scope of the thesis were identified as being potentially beneficial to agent’s ability to navigate the environment. These features will be described and discussed in this section.

6.6.1 Path length

The purpose of most route planning algorithms is to find the shortest path within a given graph. The length of a route is a sum of weights of all edges within that route. These weights are commonly interpreted as the length of the path between two vertices.

The navigation algorithm in this thesis does not take into account the length of routes. There are several reasons that make inclusion of length or distance into the model impossible at the moment:

- ARS does not collect odometry data and does not perceive the exact passage of time
- Locations do not have uniform size

- Locations completely cover the known environment

If one assumes the locations are *similar* in size, one could use locations as units of distance, thereby assessing the length of the route according to the number of “steps”, or transitions, it took to reach the destination. This assumption was the reason a greedy route planning algorithm was used for navigation.

Another possibility related to the route learning is the concept of length that might be associated with a route. A good example is an agent that is given route information from a knowledge source, along with some approximate of its length, such as “a couple of hundred meters” or “about a kilometer”. This approximation can be stored and then retrieved during decision making to obtain an estimate of a total length of a path being considered. Another, closely related idea is to allow the agent to store comparisons of routes with same source and destination, which can be either learned or deducted. This would allow the agent to choose the shorter path when presented with multiple paths to a destination without any additional calculation.

6.6.2 Location, region, and route desirability

In everyday situations human decision on which path to take is based on many other factors than just location connectivity and the length of the route. Desirability of a place is influenced by factors such as accessibility, safety, emotional response, familiarity, and recent use. These and possibly other factors, in conjunction with the internal state of the traveler, their feelings, desires and urges, exert a significant influence on the decision which route to take toward the current goal.

Locations are regions that trigger certain non-negligible emotional response from the agent are assumed to be accessible by the decision unit. These can then be factored in in the navigation procedure by placing weights on incoming edges of respective vertices in the graph. Implementing such weights in the current navigation model is trivial, as it involves modifying data structures to accommodate for an array of different types of responses to them by the agent. The route planning algorithm then becomes a more general case of shortest-path algorithm.

The challenge lies with proper integration of this concept into the ARS model. Every place and region stored in the memory is represented by a word presentation mesh. This allows for associations to be extracted from the mesh for purposes of evaluating the desirability of a certain place in the environment. The navigation module does not calculate these values on its own; instead, the decision unit would have to retrieve those responses from the storage and aggregate them into a single value that would be then used by the route planning algorithm. This would necessitate adapting the route planing algorithm to return quality scores along with the calculated routes, in order to allow the decision unit to select the most fitting route available.

6.6.3 Spatial configurations of locations and regions

Semantic Spatial Hierarchy [Kui00] allows for spatial relations to be defined between locations and regions on the topological level. These are particularly useful for the exploration process, as it allows the agent to expand the spatial knowledge by traveling between locations using previously unknown routes, guided by known of inferred relations between known parts or the environment.

Another relevant use for such relations is the resolution of the problem of false negatives in localization, also known as loop closing problem. If a known location isn't recognized upon revisit, due to unfavorable positioning or temporary occlusion during localization, it might be encoded as a new separate location, introducing inconsistency into the map. If the agent has knowledge about spatial relations, however, it could potentially deduce that it should be located at or near a particular location, which would thus resolve the problem.

Adding this feature to the model requires research of the following problems: how to represent the configurations with respect to the current model, how to adapt the ARS model to enable reasoning upon these configurations, and finally, how to recognize and learn these configurations from the environment.

6.6.4 Paths as environmental features

Paths as environmental features are quite commonly selected to facilitate self-localization and navigation, as they are quite ubiquitous in real world environments and often created for the sole purpose of aiding human navigation, especially in landmark-poor parts of the environment.

Unfortunately, ARS architecture currently does not posses the capability to recognize, encode and utilize paths in any way. This was the main reason they were not included in the model presented here. However, since the model was designed with extensibility in mind, a decision influenced by the ARS still largely being a work in progress, it is imaginable paths could be added to the model with next to no alteration.

Assuming interaction with paths would be realized as actions similar to actions presented in this thesis, one would not need to change the navigation routine at all, as its only interaction with actions is their retrieval.

Regarding scenes, one would have to introduce a concept of a *feature*, which would represent an abstraction of both landmarks and paths, along with any other feature that might be utilized in the future. Features would then replace landmarks as contents of the scenes. The set of relations would have to be augmented to accommodate the new type of feature, such as *"landmark A is on path B"*.

6.6.5 Relative distance as a feature configuration

A paper by Waller et al. [WLGB00] is a source of information regarding feature configurations that are easily recognized and extracted by humans. Amongst other things it states: "Our results can be summarized fairly succinctly: in landmark-based place learning in these environments, people tended to rely primarily on information about relative distances except when: 1.) it created nonmetric distortions of target-to-landmark enclosure relationship, or 2.) angular information was very salient during learning (e.g., containing all right angles) [WLGB00, p. 350]."

This further stresses the need to add relative distance to the set of configurations used in the scene, and possibly thereby replace the absolute distance configuration entirely. The deficiencies of the landmark distance heuristic were documented in this thesis, and this

6.6.6 Enclosure as a feature configuration

In the same paper by Waller et al. [WLGB00], one configuration of particular interest was the *enclosure* configuration, defined in the paper as "the state of being surrounded or contained within a number of landmarks". This corresponds to the problem of determining whether a point in plain lies within a simple polygon, which is defined using landmarks as polygon points [Hai94].

Utilizing enclosure would greatly simplify definition of locations in landmark-rich environments by allowing what seems to be natural tendency towards segmentation of environment into convex polygons. This would in turn make navigation less error-prone and make the current location heuristic less important, as it was implemented as a kind of *ad hoc* solution to the problem of transition between locations.

Enclosure as a feature configuration was not included in this this model as there was no indication of how it would be handled by the perception model of the ARS. Therefore, further research is recommended in order to determine how such a configuration could be recognized, stored, and utilized within the ARS.

6.6.7 Composite Landmarks

In this model a landmark is abstracted to a point in the environment. As such, it has no dimensions. On the contrary, real world landmarks have dimensionality and this property is often critical to proper localization and navigation.

For example, a landmark building can be observed from different standpoints. Assuming that building is not entirely featureless, as a ball or a cylinder would be, different standpoints yield different images of the same observed object.

These images can be abstracted into landmarks themselves, allowing, for example, the northern face of a museum to viewed as a separate landmark from the western face of the same museum.

Additionally, when the orientation is ultimately irrelevant, both of those faces can be subsumed under one single landmark that represents the museum as a whole.

Adding this functionality would allow multiple distinct encodings of the same landmark, representing views from different standpoints, which would be semantically connected into a single overarching landmark, termed *composite landmark*, thereby solving the problem of landmark recognition with respect to different viewpoints.

6.6.8 Integration of localization into memory retrieval system

As the localization algorithm started to take its present shape, its similarity to the retrieval algorithm of the information management system of the ARS became more and more apparent. Both of the algorithms compare data structures against a set of candidates stored within a larger data structure, returning the best possible match. However, in order for the localization routine to gain access to the cognitive map the memory has to be queried, adding a significant performance penalty. It is hypothesized that the localization functionality could be eventually integrated into the memory retrieval algorithm.

Such integration would make the F61 localization module as it currently stands obsolete, thereby further simplifying the ARS model, as well as accelerating the localization process by eliminating unnecessary memory access.

6.6.9 Fuzzy location borders

The images of localization test results show many instances of awkwardly-shaped borders between neighboring locations, such that are unlikely to be defined by humans. This is due to several facts:

- algorithm only takes into account the best-matching location for any given point in the environment, regardless of how close the scores of other locations were to the best one
- the model of agent's vision field features strict borders between vision ranges, which translates into abrupt changes in landmark distance heuristic method score
- heuristics generally deal with small sets of features, where significant changes in scores are often caused by relatively small changes in the perception of the environment

Such clearly defined limits exist almost exclusively in graphical representation of the environment, and such borders seem to have no influence on human navigation. By making locations fuzzy one would improve the problematic behavior during transitions between locations, which was the reason behind the introduction of the current location heuristic. Further research into fuzzy logic as a part of a psychoanalytically-inspired is required in order to show whether implementing such a model is feasible.

6.6.10 Ambiguous localization

One of the possible solutions to the problem of localization uncertainty, particularly when running the localization routine for the first time is to develop and track several possible current locations simultaneously. This would include a variable number of candidate locations that satisfy a certain similarity criterion with the perceived location. This would be initial set of hypothesized current locations that would get smaller as more data is obtained during navigation, until the set is eventually narrowed down to a single candidate location. The hypothesis set is modified as the agent navigates the environment, which adds or eliminates possibilities based on the connectedness of locations in the cognitive map. After each localization a new set of candidate locations is obtained and compared to the last set. A location from the last set is either retained, if it is found in the new set, replaced, if a directly connected locations is found, or otherwise removed. This can sometimes lead to a temporary increase in number of hypotheses, especially in the ambiguous parts of the environment around the location borders, but should gradually get narrowed down to a single location. This functionality might allow the agent to traverse mazes as well, which are by definition almost devoid of landmarks.

The biggest benefit to implementing such a functionality is the increased robustness of the process; while it is unlikely for uncertainty to occur in landmark-rich environment where landmarks are mostly distinct form one another, this property of the environment cannot be always guaranteed, even more so since the impact of yet to be implemented learning algorithms on the localization process is currently unforeseeable.

6.7 Conclusion

Research of the topic of the navigation in autonomous agent provided a small but very valuable insight into the possibilities available to models such as the ARS. Results of the tests show confirm the validity of the approach described in this thesis, but more importantly help pave the way towards a robust and reliable solution to the human-like navigation problem. The most important conclusions reached can thus be summarized in several points:

- Feature-based navigation is demonstrated to be compatible with the ARS and its concepts, even in the limited scope as shown in this thesis. It is also corroborated by significant amount of research into human navigation behavior, which makes it a suitable approach to the problem in this particular case. Further research into additional types of environmental features and their configurations, their proper recognition, filtering, and encoding, is expected to further improve the power of the navigation model, as well as to bring it closer to exhibiting human-like behavior as seen in the real world. The ability of the ARS to provide properly processed data is absolutely vital to the proper functioning of the navigation model and thus represents the biggest challenge pertaining to any real-world implementation of the model.
- As demonstrated by tests done in this thesis, and corroborated by published research, distance as a feature configuration is *critical* to successful localization. Despite there

existing methods such as feature triangulation methods that do not rely on distance as a feature configuration, they use geometrical methods of localization that are too analytic and exact to be viable option when striving toward a human-like model. It is recommended that any further research into feature recognition should also encompass feature-agent (absolute) and feature-feature (relative) distance recognition.

- Map building through learning and exploration is one of the most important factors in facilitating human-like navigation. The cognitive map is the fundamental part of the the agent's ability to navigate and all behavior in this context is directly related to its structure and contents. Manually created maps will not break the functionality, but will preclude the emergence of true human-like behavior by providing an unrealistic, however accurate, initial knowledge of the environment.

- The route planning algorithm serves as a valid basis for further development of this part of the model. It was shown to work on both flat maps as well as on hierarchical maps of various depths, and, due to its recursive nature, is expected work on maps of any depth of the hierarchy. Its ability to function on manually constructed maps makes it a working temporary solution to the problem, until more natural maps described above become possible. Further research in this aspect should be directed towards methods of analysis and combination of known routes, which should be able to construct routes from currently known paths in a simple and intuitive manner.

In conclusion, this thesis is just the first step into the area of research that has long been avoided due to perceived intractability of symbolic high-level reasoning. While the test results demonstrated deficiencies of the navigation model in its present state, they also showed how these might be remedied in the future. As the ARS model continues to develop, more and more opportunities to improve the navigation model will present themselves, eventually resulting in a bionically inspired navigation model supported by high-level reasoning that will be on par with the current state of the art.

Bibliography

[BDM$^+$13] Dietmar Bruckner, Dietmar Dietrich, Clemens Muchitsch, Alexander Wendt, and Samer Schaat. *Naturwissenschaftliches, psychoanalytisches Modell der Psyche für Simulation und Emulation*, 2013.

[Boe] Daniel Boe. Line Segment Intersection Algorithm. `http://bryceboe.com/2006/10/23/line-segment-intersection-algorithm/`. Accessed: 26.08.2014.

[Bor09] Robert Borer. Episodic Memory Based Self Localization and Orientation for Autonomous Agents. Master's thesis, Vienna University of Technology, 2009.

[Bwa] Bwag/Commons. Wien - stephansdom. URL: `http://de.wikipedia.org/wiki/Stephansdom_(Wien)#mediaviewer/Datei:Wien_-_Stephansdom.JPG`. Licensed under Creative Commons Attribution-ShareAlike 4.0 International licence. Accessed: 26.08.2014.

[CN01] H. Choset and Keiji Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2):125–137, Apr 2001.

[DFZB09] Dietmar Dietrich, Georg Fodor, Gerhard Zucker, and Dietmar Bruckner, editors. *Simulating the Mind - A Technical Neuropsychoanalytical Approach*. Springer, Wien, 2009.

[DS74] Roger M Downs and David Stea. *Image and environment: Cognitive mapping and spatial behavior*. Transaction Publishers, 1974.

[FG97] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In JörgP. Müller, MichaelJ. Wooldridge, and NicholasR. Jennings, editors, *Intelligent Agents III Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Computer Science*, pages 21–35. Springer Berlin Heidelberg, 1997.

[FG02] Juan A. Fernandez and Javier Gonzalez. *Multi-Hierarchical Representation of Large-Scale Space: Applications to Mobile Robots*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[FWDT05] Patrick Foo, William H. Warren, Andrew Duchon, and Michael J. Tarr. Do humans integrate routes into a cognitive map? map- versus landmark-based navigation of novel shortcuts. *Journal of Experimental Psychology: Learning, Memory and Cognition*, pages 195–215, 2005.

[Gru07] Andreas Gruber. Neuro-psychoanalytically inspired episodic memory for autonomous agents. Master's thesis, Technische Universität Wien, Institut für Computertechnik, 2007.

[Hai94] Eric Haines. Point in polygon strategies. *Graphics gems IV*, 994:24–26, 1994.

[HBF$^+$06] ToddS. Horowitz, RandallS. Birnkrant, DavidE. Fencsik, Linda Tran, and JeremyM. Wolfe. How do we track invisible objects? *Psychonomic Bulletin & Review*, 13(3):516–523, 2006.

[Hen93] David B Henson. *Visual fields*. Oxford University Press New York, 1993.

[Ken38] M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1/2):81–93, June 1938.

[Kit94] Robert M. Kitchin. Cognitive maps: What are they and why study them? *Journal of Environmental Psychology, Vol 14(1), Mar 1994, 1-19.*, 1994.

[Kui00] Benjamin Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1–2):191 – 233, 2000.

[LBP$^+$03] Sean Luke, Gabriel Catalin Balan, Liviu Panait, Claudio Cioffi-Revilla, and Sean Paus. Mason: A java multi-agent simulation library. In *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*, volume 9, 2003.

[LDW91] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, Jun 1991.

[Lu] Daniel Lu. "Particle2dmotion". URL: `http://commons.wikimedia.org/wiki/File:Particle2dmotion.svg#mediaviewer/File:Particle2dmotion.svg`. Licensed under Creative Commons Attribution-Share Alike 3.0 licence. Accessed: 26.08.2014.

[MB13] Marina Magnabosco and Toby P. Breckon. Cross-spectral visual simultaneous localization and mapping (SLAM) with sensor handover. *Robotics and Autonomous Systems*, 61(2):195 – 208, 2013.

[OFWB03] Joshua O'Madadhain, Danyel Fisher, Scott White, and Y Boey. The jung (java universal network/graph) framework. *University of California, Irvine, California*, 2003.

[ON78] John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Oxford: Claredon Press, 1978.

[SGB05] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, volume 2, 2005.

[SH99] Molly E. Sorrows and Stephen C. Hirtle. The Nature of Landmarks for Real and Electronic Spaces. In Christian Freksa and David M. Mark, editors, *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*, volume 1661 of *Lecture Notes in Computer Science*, pages 37–50. Springer Berlin Heidelberg, 1999.

[SSC87] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 850–850, Mar 1987.

[SV06] Nestor Schmajuk and Horatiu Voicu. *Exploration and navigation using hierarchical cognitive maps*, chapter 9. Comparative Cognition Press, 2006.

[TFBD01] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99 – 141, 2001.

[Tol48] Edward C. Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189–208, July 1948.

[VB08] R. Velik and D. Bruckner. Neuro-symbolic networks: introduction to a new information processing principle. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, pages 1042–1047, July 2008.

[Vel08] Rosemarie Velik. *A Bionic Model for Human-like Machine Perception*. PhD thesis, Vienna University of Technology, Institute of Computer Technology, 2008.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina, Chapel Hill, NC, USA, 1995.

[WFJL08] Diedrich Wolter, Christian Freksa, and Longin Jan Latecki. Towards a Generalization of Self-localization. In Margaret E. Jefferies and Wai-Kiang Yeap, editors, *Robotics and Cognitive Approaches to Spatial Mapping*, volume 38 of *Springer Tracts in Advanced Robotics*, pages 105–134. Springer Berlin Heidelberg, 2008.

[WKBM$^+$97] Steffen Werner, Bernd Krieg-Brückner, Hanspeter A. Mallot, Karin Schweizer, and Christian Freksa. Spatial Cognition: The Role of Landmark, Route, and Survey Knowledge in Human and Robot Navigation. In *Ph.D. in Computer Science in*, pages 41–50. Springer, 1997.

[WLGB00] David Waller, JackM. Loomis, ReginaldG. Golledge, and AndrewC. Beall. Place learning in humans: The role of distance and direction information. *Spatial Cognition and Computation*, 2(4):333–354, 2000.

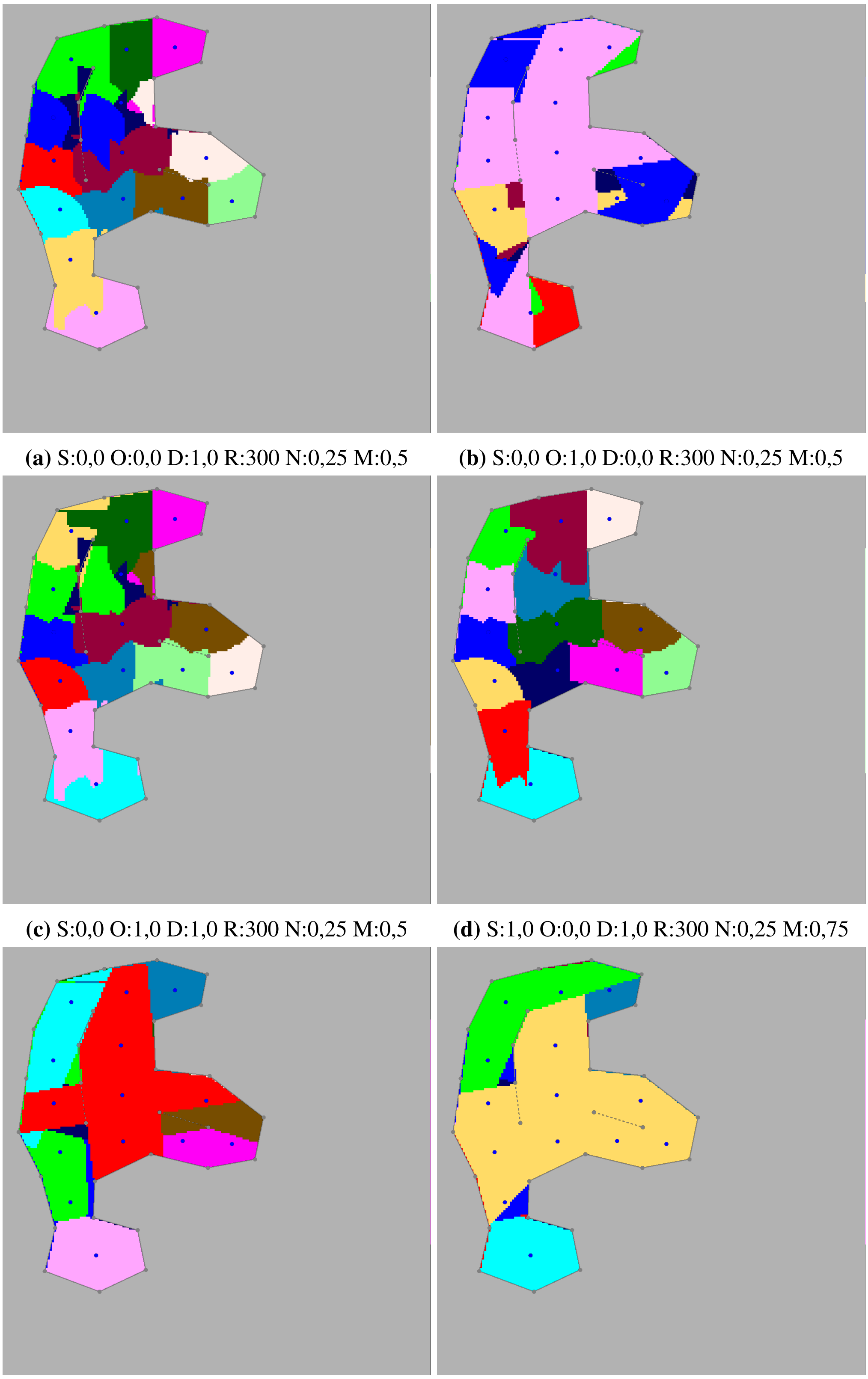
[WRN04] Stephan Winter, Martin Raubal, and Clemens Nothegger. Focalizing Measures of Salience for Wayfinding. In *In L. Meng, A. Zipf & T. Reichenbacher (Eds.), Map-based Mobile Services - Theories, Methods and Implementations*, pages 127–142. Springer Geosciences, 2004.

[Zei10] Heimo Zeilinger. *Bionically Inspired Information Representation for Embodied Software Agents*. PhD thesis, Vienna University of Technology, 2010.

APPENDIX A

Localization test results

All images shown here are masked to improve visibility of locations and the shape of the environment. Each test results has the parameters specified in the caption. Parameter key: S – landmark set heuristic score weight, O – landmark order heuristic score weight, D – landmark distance heuristic score weight, R – vision range, N – near distance range ratio, M – mid-far distance range ratio. See section 6.3 for the interpretation of these results.



(a) S:0,0 O:0,0 D:1,0 R:300 N:0,25 M:0,5

(b) S:0,0 O:1,0 D:0,0 R:300 N:0,25 M:0,5

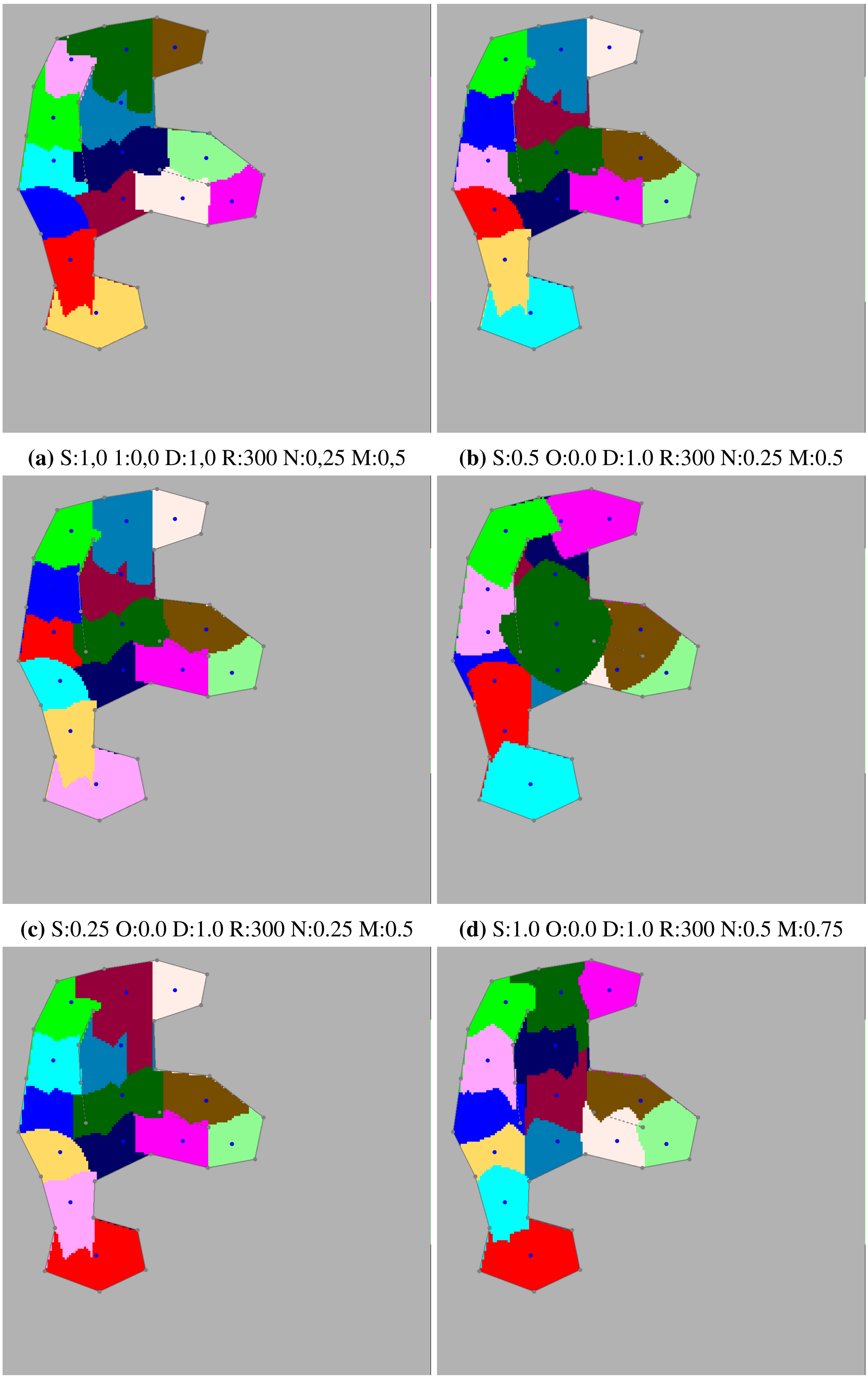
(c) S:0,0 O:1,0 D:1,0 R:300 N:0,25 M:0,5

(d) S:1,0 O:0,0 D:1,0 R:300 N:0,25 M:0,75

(e) S:1,0 O:1,0 D:0,0 R:300 N:0,25 M:0,75

(f) S:1,0 O:0,0 D:0,0 R:300 N:0,25 M:0,5

Figure A.1: Scenario A, first set



(a) S:1,0 1:0,0 D:1,0 R:300 N:0,25 M:0,5

(b) S:0.5 O:0.0 D:1.0 R:300 N:0.25 M:0.5

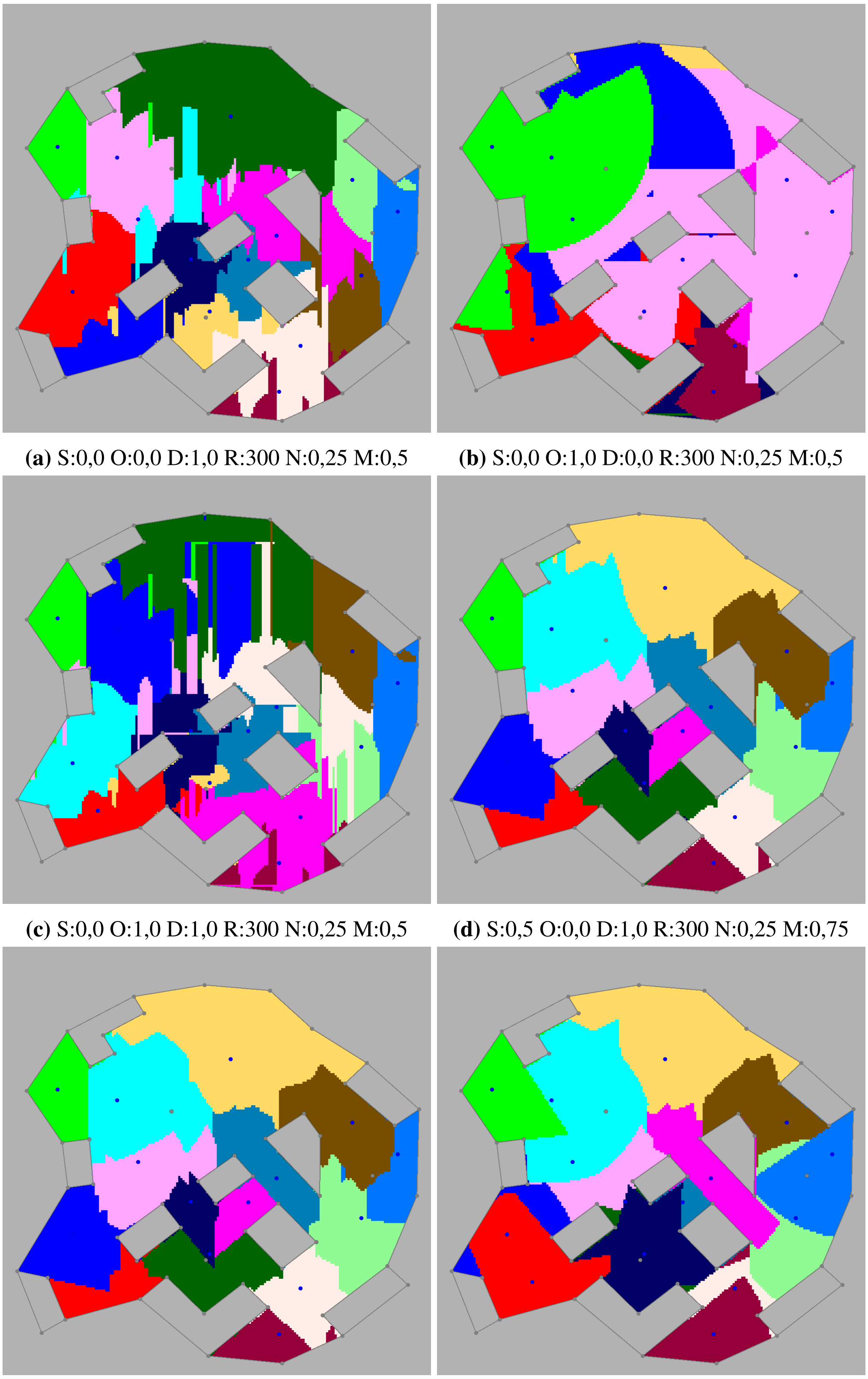
(c) S:0.25 O:0.0 D:1.0 R:300 N:0.25 M:0.5

(d) S:1.0 O:0.0 D:1.0 R:300 N:0.5 M:0.75

(e) S:1.0 O:0.0 D:1.0 R:300 N:0.25 M:0.75

(f) S:1.0 O:0.0 D:1.0 R:300 N:0.33 M:0.66

Figure A.2: Scenario A, second set



(a) S:0,0 O:0,0 D:1,0 R:300 N:0,25 M:0,5

(b) S:0,0 O:1,0 D:0,0 R:300 N:0,25 M:0,5

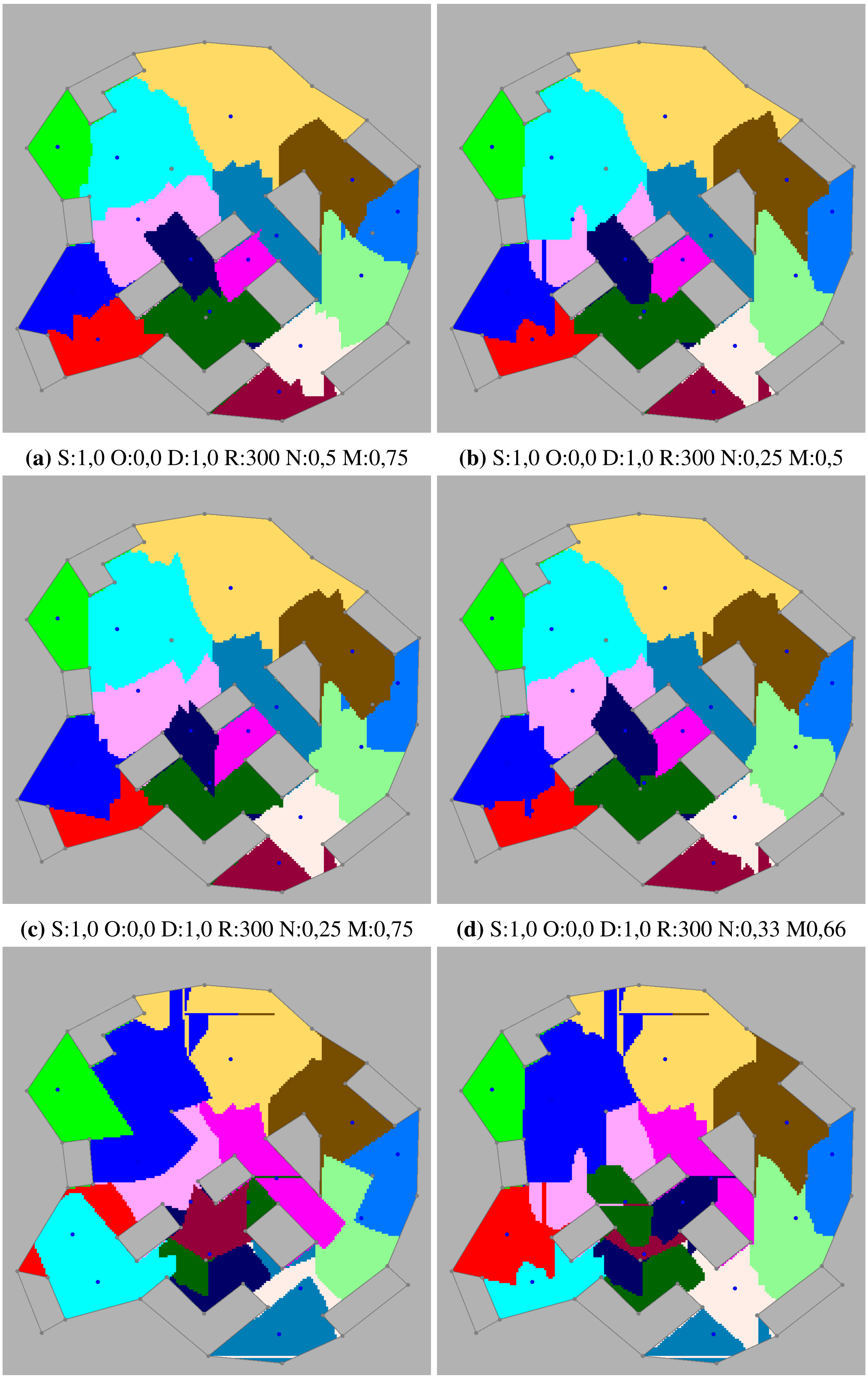
(c) S:0,0 O:1,0 D:1,0 R:300 N:0,25 M:0,5

(d) S:0,5 O:0,0 D:1,0 R:300 N:0,25 M:0,75

(e) S:0,25 O:0,0 D:1,0 R:300 N:0,25 M:0,75

(f) S:0,0 O:1,0 D:0,0 R:300 N:0,25 M:0,5

Figure A.3: Scenario B, first set



(a) S:1,0 O:0,0 D:1,0 R:300 N:0,5 M:0,75

(b) S:1,0 O:0,0 D:1,0 R:300 N:0,25 M:0,5

(c) S:1,0 O:0,0 D:1,0 R:300 N:0,25 M:0,75

(d) S:1,0 O:0,0 D:1,0 R:300 N:0,33 M0,66

(e) S:1,0 O:1,0 D:0,0 R:300 N:0,25 M:0,5

(f) S:1,0 O:1,0 D:1,0 R:300 N:0,25 M:0,5

Figure A.4: Scenario B, second set

APPENDIX B

Listing of the XML Schema for Scenario Files

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="
    unqualified" elementFormDefault="qualified">
  <xs:element name="scenario" type="scenarioType"/>
  <xs:complexType name="dimensionsType">
    <xs:sequence>
      <xs:element type="xs:short" name="width"/>
      <xs:element type="xs:short" name="height"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="positionType">
    <xs:sequence>
      <xs:element type="xs:float" name="x"/>
      <xs:element type="xs:float" name="y"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="landmarkType" mixed="true">
    <xs:sequence>
      <xs:element type="positionType" name="position" minOccurs="0"/>
      <xs:element type="xs:string" name="label" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:byte" name="id" use="optional"/>
    <xs:attribute type="xs:byte" name="ref" use="optional"/>
  </xs:complexType>
  <xs:complexType name="visionType">
    <xs:sequence>
      <xs:element type="xs:float" name="range"/>
      <xs:element type="xs:float" name="angle"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="agentType">
```

```
    <xs:sequence>
      <xs:element type="positionType" name="position"/>
      <xs:element type="xs:string" name="label"/>
      <xs:element type="xs:float" name="orientation"/>
      <xs:element type="visionType" name="vision"/>
    </xs:sequence>
    <xs:attribute type="xs:byte" name="id"/>
  </xs:complexType>
  <xs:complexType name="entitiesType">
    <xs:sequence>
      <xs:element type="landmarkType" name="landmark" maxOccurs="unbounded"
          minOccurs="0"/>
      <xs:element type="agentType" name="agent"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="obstacleType">
    <xs:sequence>
      <xs:element type="landmarkType" name="landmark" maxOccurs="unbounded"
          minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="occluding" use="optional"/>
  </xs:complexType>
  <xs:complexType name="obstaclesType">
    <xs:sequence>
      <xs:element type="obstacleType" name="obstacle" maxOccurs="unbounded"
          minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="environmentType">
    <xs:sequence>
      <xs:element type="dimensionsType" name="dimensions"/>
      <xs:element type="entitiesType" name="entities"/>
      <xs:element type="obstaclesType" name="obstacles"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="rpositionType">
    <xs:sequence>
      <xs:element type="xs:float" name="x"/>
      <xs:element type="xs:float" name="y"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="locationType" mixed="true">
    <xs:sequence>
      <xs:element type="positionType" name="position" minOccurs="0"/>
      <xs:element type="rpositionType" name="rposition" minOccurs="0"/>
      <xs:element type="xs:string" name="label" minOccurs="0"/>
      <xs:element type="landmarkType" name="landmark" maxOccurs="unbounded"
          minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:byte" name="id" use="optional"/>
    <xs:attribute type="xs:byte" name="ref" use="optional"/>
  </xs:complexType>
  <xs:complexType name="locationsType">
  <xs:sequence>
    <xs:element type="locationType" name="location" maxOccurs="unbounded"
        minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="fromType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute type="xs:byte" name="ref" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="toType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute type="xs:byte" name="ref" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:complexType name="actionType">
  <xs:sequence>
    <xs:element type="fromType" name="from"/>
    <xs:element type="toType" name="to"/>
    <xs:element type="xs:string" name="type"/>
    <xs:element type="landmarkType" name="landmark" maxOccurs="unbounded"
        minOccurs="0"/>
  </xs:sequence>
  <xs:attribute type="xs:short" name="id" use="optional"/>
</xs:complexType>
<xs:complexType name="actionsType">
  <xs:sequence>
    <xs:element type="actionType" name="action" maxOccurs="unbounded"
        minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="regionType">
  <xs:sequence>
    <xs:element type="xs:string" name="label"/>
    <xs:element type="locationType" name="location" maxOccurs="unbounded"
        minOccurs="0"/>
    <xs:element type="regionType" name="region" maxOccurs="unbounded"
        minOccurs="0"/>
  </xs:sequence>
  <xs:attribute type="xs:byte" name="id" use="optional"/>
</xs:complexType>
<xs:complexType name="regionsType">
  <xs:sequence>
    <xs:element type="regionType" name="region"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="mapType">
  <xs:sequence>
    <xs:element type="locationsType" name="locations"/>
```

```
      <xs:element type="actionsType" name="actions"/>
      <xs:element type="regionsType" name="regions"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="scenarioType">
    <xs:sequence>
      <xs:element type="environmentType" name="environment"/>
      <xs:element type="mapType" name="map"/>
    </xs:sequence>
    <xs:attribute type="xs:float" name="version"/>
  </xs:complexType>
</xs:schema>
```