Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
Gerstner Laboratory

ROBOT MAPPING FOR LARGE ENVIRONMENTS

Doctoral Thesis

Karel Košnar

Prague, May, 2011

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Artificial Intelligence and Biocybernetics

Supervisor: Ing. Libor Přeučil, CSc.
Supervisor-specialist: RNDr. Petr Štěpán, Ph.D.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

USED MATHEMATICAL NOTATION

$\mu_A(x)$	Membership function
$\perp, \top$	Contradiction and tautology
Ω	Set of possible worlds
$p(x)$	Probability function
$N(x)$	Necessity function
$\Pi(x)$	Possibility function
$m_\Theta(x)$	Belief mass assignment on frame of discernment Θ
$\mathcal{O}(x)$	Opinion
$E(x)$	Probability expectation
$\oplus$	Fusion operator for opinions
$\otimes$	Discounting operator for opinions
$\|X\|$	Size of the set or absolute value
$\\|X\\|, \\|x-y\\|$	L^2 norm, Euclidean distance
$< X, Y >$	Inner product of vector X and Y
$f \star g$	Cross-correlation of functions f and g
$NCC(f, g)$	Normalized cross-correlation function
ϑ	Threshold
$[a, b]$	Interval of real numbers from a to b
$G(V, E)$	Graph with set of vertices V and edges E
v	Graph vertex
$e, e = (u, v)$	Graph edge, edge with explicit start and target vertex

आज्ञानतिमिरान्धस्य ज्ञानाञ्जनशलाकया।
चक्षुरुन्मीलितं येन तस्मै श्री गुरवे नमः॥
गुरुर्ब्रह्मा गुरुर्विष्णु गुरुर्देवो महेश्वरः।
गुरु सक्षत् परब्रह्म तस्मै श्री गुरवे नमः॥

Guru stotram, shri Adi Shankaracarya

Salutation to the noble Teacher, who has opened the eyes blinded by darkness of ignorance with the eye-drops of knowledge.

ACKNOWLEDGEMENT

I would like to express my thanks to the Libor Přeučil and Petr Štěpán, my supervisors. Moreover I would like to express my thanks to to all colleagues at Intelligent and Mobile Robotics group, in particular to Tomáš Krajník and Vojtěch Vonásek for their help with implementation of the jockeys, to Jan Chudoba for help with the hardware issues, and to Miroslav Kulich and Jan Faigl for their valuable comments.

I am also grateful to my family for the support during my research, and specially to my wife for her endless patience.

The work presented in this thesis was supported by Czech Technical University grants CTU0614813 and CTU0505513 and by the Ministry of Education of the Czech Republic by the projects 2C06005 and 1M0567.

1 INTRODUCTION

The whole humankind dreams about a creation of an artificial creature which will be their servant, helper or companion. This dream is a part of the human nature. It is possible to find it in the Jewish folklore in a form of the Golem, an animated anthropomorphic being, created entirely from inanimate matter. In the Talmud, it is written that Adam was initially created as a Golem. The most famous Golem was reportedly created by rabbi Jehuda Loew ben Bezalel in Prague under emperor Rudolph's II. rule.

One of the goals in alchemy was to create a homunculus. The first record of an artificial being in alchemical literature appeared in the book *Visions of Zosimos*, written in the third century AD. The artificial being is denoted as an *anthroparion* in this book, and it is something similar to the Golem but with a higher level of autonomy and intelligence. The Arabic word *takwin* refers to the artificial creation of life in the laboratory, including human life.

In a modern time, the robotics has the ambition to fulfill this ancient dream - to build an artificial man. The first humanoid robot, a flute player, was built by Jacques de Vaucanson in 1738 [85]. This robot was able to play 12 different melodies. Vaucanson had arrived at those sounds by mimicking every muscle by which a man would make them. Bellows inside the robot produces a varying flow of air. The mechanical lips could open, close and move backwards or forwards. A moveable metal tongue located inside the mouth governs the air-flow and creates pauses. Because the robot was made from wood, and wooden fingers was not soft enough to play a metal flute correctly, Vaucanson adds the glows which work as an artificial skin.

Robot gakutensoku was built in Japan, Osaka in 1929. Gakutensoku could change its facial expression, and move its head and hands via an air pressure mechanism. In 1937, the robot Electro was built by the Westinghouse Electric Corporation in USA,Ohio. Two meters tall, weighing 120 kilograms, humanoid could walk by voice command, speak about 700 words (using a 78-rpm record player), smoke cigarettes, blow up balloons, and move his head and arms. Leonardo da Vinci's mechanical robot hidden in medieval armor was designed in 1495, but it is not known whether or not the device was build during Leonardo's lifetime. Since the discovery of the sketchbook, the robot has been built in 1950s faithfully based on Leonardo's design. This proved it was fully functional, as Leonardo had planned.

On the other hand, there exists the fear that the robots will become intelligent and autonomous too much and rise against their creators - humans. Such a fear can be found in many movies and books, e.g. the novel Frankenstein or Modern Prometheus written by Mary Shelley. Even the famous three Asimov's robotic laws express the necessity of the humans' control over the robots. Also the drama Rossum's Universal Robots (RUR) written by Karel Čapek, which is cited in almost every robotic literature for the first usage of the world *robot*[1], warns of possible robots' rebellions.

As the robots were used more and more as tools, the robots lost their anthropomorphic form and get the form best suitable to perform a specific task. Maybe, the fear from rebellion was also some non-conscious motivation, because people did not dread the manipulator even the intelligence of such a robot can be on very high level. In case of mobile robotics, robots obtain the shapes of different vehicles, boats, submarines, airplanes and helicopters. The wheels and the caterpillars usually overcame the artificial legs on the ground.

The anthropomorphous robots keep their place mainly in an entertainment, e.g. toys like Nao humanoid robot. The robots like Honda's Asimo, Sony's Qrio, or Toyota's robotic trumpet player are going in the footprints of early robots, and their purpose is mostly in showing the technical maturity of their builder. The human-like shape can be advantageous in the field, where the robots are in a tight contact with humans or co-work with them. One example for all, the NASA in collaboration with the General Motors is building the robonaut, the humanoid robot to work side by side with human astronauts. Notable is the fact, the robot is planned to use two types of lower bodies, legged and wheeled according to the environment to move through.

The field, where the robotics is still taking the inspiration from the humans and animals is the cognition and intelligence. Cognitive robotics views animal and human cognition as a starting point for the developing of robotic information processing. Target robotic cognitive capabilities include perception processing, attention allocation, anticipation, planning, complex motor coordination, reasoning about the environment, actions and the goals. Cognitive robotics is concerned with endowing the robot with an intelligent behavior by providing the robot with a processing architecture that will allow it to learn and reason about how to behave in response to complex goals in a complex world. Perception and action, and the notion of symbolic representation are therefore core issues to be addressed in cognitive robotics.

1.1 MOTIVATION

Nowadays, robots are quickly moving from the laboratories and factories into households, offices, and streets. Mobile robots have to explore and navigate large-scale every-day environments with only general assumptions about their properties and structure. To do this, they have to sense their environment, construct a representation of the environment, reason over that representation, and perform the

1 To satisfy the tradition, the word *robot* was invented by the Karel's Capek brother Josef and refers to the old Czech world *robota* which means a mandatory work

actions to fulfill their goals. They also have to interact and communicate with humans in an understandable form. In other words, the robot must possess cognitive capabilities in order to operate in an every-day environment. The cognitive capabilities are:

- Informational attitudes such as knowledge and beliefs,
- motivational attitudes such as preferences and goals,
- cognitive capabilities such as reasoning, decision making, planning, observing and communicating and
- physical capabilities to move in the physical world, and to interact safely with objects in that world.

The aim of the thesis is to allow a mobile robot to operate autonomously in large every-day environment. Therefore the thesis focuses on the finding suitable representation of the spatial knowledge about the surrounding environment. Two sources of inspiration are used: the cognitive theories, how the humans and animals represent the spatial knowledge, and existing implementations of robotic maps, which are the robotic representation of the spatial knowledge.

The robot needs the mechanism, how to build the map (inner representation of the knowledge about the environment) autonomously. Therefore, the algorithm for autonomous exploration of the unknown environment will be proposed in the thesis.

As the information gathered during the exploration are subject to errors, the mechanism for handling the uncertain information is requested. The reasoning algorithm will be proposed in the thesis. It has to combine the uncertain information with the aim to diminish the uncertainty and to deduce the new information for the actual knowledge.

1.2 THESIS OUTLINE

The thesis is organized into two parts. The first part is an introduction into the problem, and consists from this introduction, Chapter 2 which defines the problem of robotic mapping, Chapter 3 introduces the goals of the thesis. The following chapters present the current state of the art in the areas of cognitive theories of the humans spatial representation (Chapter 4), robotic mapping (Chapter 5), exploration (Chapter 6) and reasoning with uncertainty (Chapter 7).

The second part of this thesis is a description of the thesis contribution. Chapter 8 compares the existing representation of spatial knowledge and proposes the novel representation of this knowledge. Then Chapter 9 follows, with the description the method for reasoning about the spatial knowledge. Chapter 10 consists of detailed description of a localizing and navigating algorithms used for proposed spatial representation. Chapter 11 focuses on exploration algorithms for this representation. The experimental verification of the localization, navigation, exploration and mapping is described in Chapter 12. The contributions of the work and the future research are summarized in Chapter 13. The thesis concludes in Chapter 14.

2

PROBLEM DEFINITION

This thesis aims to solve the problem of a mobile robot operation in a large environment. The autonomous movement is a crucial ability for the mobile robot to operate in any environment and fulfill given tasks. The robot needs a model of the environment to be able robustly, autonomously and repeatably move in the environment. Therefore, the thesis deals primarily with the problem of mapping.

The problem of *mapping* is defined as a process of learning and maintaining an inner spatial model of an initially unknown environment. This inner spatial model is called *a map*. A mobile robot moves through an unknown environment and integrates local spatial information, called observations, gathered over time into coherent overall model which correctly reflects the spatial properties of a section of the external world.

The environment is called *large-scale* as a space can not be perceived at once: the sensor limitations make it necessary for a mobile robot to navigate through the environment when building a map. In this case, the map must be integrated from local observations gathered from different places over time. Note, the property of "being a large-scale" is defined by perceptual mechanisms and capabilities rather than by physical size of the space.

The integration process during the mapping is supposed to be either incremental (on-line), new information is incorporated into the internal model an then discarded or off-line, when all the input data are available at once and it is possible to process the data even repeatedly.

The mapping problem is challenging because the available information is typically erroneous, imprecise, and ambiguous. A large part of the problem is the perceptual aliasing. Different places in the environment look similar, and it is not possible to distinguish these places using only the local observation. The problem is to establish the correct correspondences between the currently observed entities which represent the robot's current local observation and the memorized entities in the robot's spatial model - the map.

2.1 FORMAL DEFINITION

A classical formal definition of mapping problem uses a probabilistic terminology. The robot moves through the environment and observes its' local surrounding. Let o_t be a local observation at time t, a vector of readings from robot sensors. Let

$O_T = \{o_1, o_2, o_3, \ldots, o_T\}$ be a sequence of observation from the beginning till the time T. It is assumed, without loss of generality, that the robot takes exactly one measurement in each time t.

The robot is moving actively using its' actuators (wheels, legs, propellers etc.). Let u_t be the control action between time $t-1$ and t, a vector of control values for the actuators. Let $U_T = \{u_1, u_2, u_3, \ldots, u_T\}$ be a robot's actions from the beginning till the time T.

Let m denote the map of the environment. As the map can be incrementally updated with every incoming observation, the map in time t is denoted as m_t.

The on-line mapping is defined as determining the map m_t that maximizes the conditional probability

$$p(m_t|o_t, u_t, m_{t-1}),$$

where the o_t and u_t is directly accessible for the robot, m_{t-1} is a map computed in previous step and m_t is actual map. It is easy to see, that the knowledge is build incrementally, only the actual observation and control action is used. The final map m is a map m_T computed in the last step T of mapping process.

The off-line mapping is defined similar to on-line mapping as maximization of conditional probability

$$p(m|O_T, U_T),$$

where the final map m is computed from sequence of all observations and control actions.

The map is always gathered for a specific reason. Two main purposes for mapping in robotics are the localization and navigation of a mobile robot.

Let a map m be :

LOCALIZABLE for a robot r if there is a function $loc(t)$, that estimates the position of r in m at any position in time t.

TRAVERSABLE for a robot r if there is a predicate $conn(x, y)$ that for any elements $x, y \in m$ tells if there is a sequence of actions for r to go from x to y. The $conn(x, y)$ often computes also the sequence of action to follow the connecting path.

Being localizable and traversable are not properties of a map itself, but rather the properties of the pair a map and a robot. Localizable map contains the information to allow a given robot to answer the question "Where am I?", and a traversable map contains the information to answer the question "How can I go from A to B?".

There exists numbers of approaches solving the mapping problem together with localization. These are denoted as simultaneously localization and mapping (SLAM) or concurrent localization and mapping (CLM) approaches.

On-line slam seeks, besides the map, the present robot position x_t, a vector describing the position of the robot according to the conditional probability

$$p(x_t, m_t|o_t, u_t, m_{t-1}).$$

The position for a wheeled mobile robot on a flat ground is usually defined by three-dimensional vector (two-dimensional coordinate and heading). As the robot is in a raw terrain or is moving freely in the space, then the position is a six-dimensional vector (tree-dimensional coordinate and three rotational angles). The position can be described also with higher-dimensional space, if the robot is more complex and it is not possible to deal the robot as a rigid object.

Full SLAM (off-line) is the problem of calculating the join posterior probability

$$p(X_T, m|O_T, U_T),$$

over the entire robot path X_T and the map m, where $X_T = \{x_1, x_2, x_3, \dots, x_T\}$ is a robot path.

The localization is consider as crucial for operation of a mobile robot by many researchers. The navigation and traversability of the map is often pushed into background and the navigation is often solved through localization and finding a sequence of way-points.

This is not the only possible approach. Humans and animals are able to navigate without explicit localization. As the goal of this thesis is to allow the robot to operate and move in the large environment, the navigation is the crucial ability. The localization is required only in particular situation and can be viewed as a support for navigation but not the main goal. Therefore the thesis focuses on building the traversable map of the large environment for the purposes of navigation of the mobile robot.

2.2 CONDITIONS

To solve the mapping problem in full generality is hard, therefore the following limitation of the environment is assumed. First, the properties of environment is defined as follows.

Let the environment be *structured environment* if it consists of mutually disjoint regions with strictly defined borders. The structure of the environment is given by regions itself and their connectivity and adjacency.

Let the structured environment be *stable*, if all the changes of environment does not influence the structure of the environment.

For example, indoor environments with rooms and corridors, urban environment with streets and city blocks or suburban environment with roads, gardens and houses are typical structured environments. Similarly, a typical park-like environment has a structure given by pathways and lawn regions. Typical urban environment is highly dynamic environment with number of moving objects but is stable as the roads and buildings defining the regions and its structure are not changing.

The environments are supposed to be structured and stable in the rest of the thesis.

3 THESIS GOALS

The aim of this thesis is to allow a mobile robot to operate autonomously in a large-scale every-day environment. The robot has to build a map of an unknown environment and store it in an appropriate data structure. Then the robot uses the map for localization and navigation while fulfills tasks (like a pick up and delivery).

A model of the environment should also be understandable for people, as the robot interacts often with people. Usually, human operator specifies the tasks or targets of the robot movement. The interaction (or cooperation) is sometimes necessary even during the map building.

Therefore, the main goals of this thesis have been determined as :

1. To perform a study of the currently used representations of the spatial knowledge. The study will elaborate the properties and weakness of the currently used representation of the spatial knowledge in natural and artificial systems.
2. To propose a scalable probabilistic representation of the space - a map which is able to represent diverse types of environments, indoor as well as outdoor and deal with the uncertainty. Expected size of the operational spaces lies in order of kilometers. An important expected property is possibility of incremental building of the map as novel information is discovered. The proposed map must be reusable.
3. To propose a method of an autonomous exploration without necessity of the environment modification. As the robot has to operate in an every-day type of environment, it is necessary to be able to learn the environment without any modification made to it.
4. To propose a method for reasoning about uncertain spatial knowledge. All the data about the environment are uncertain, due to the fact that these are gathered using real sensors, which are subject to errors and noise. The reasoning method should take these uncertain data and combine them with a view to minimize the uncertainty in the build of knowledge. The reasoning then allows to gain the information which is not directly observable by the sensors.
5. To implement and integrate the proposed methods into a unified mapping framework. The implementation will work with the real sensors and robots

in every-day environments. As the robot must be able to operate in an environment with real sensors, the set of localization and navigation methods must be integrated with the proposed mapping methods.

6. To verify the proposed methods in realistic environments and conditions. The methods will be experimentally verified and evaluated in the diverse types of the environment and with diverse sensors. The verification will aim not only to obtaining of the map but also to the usage map in real navigation and localization tasks later on.

4 COGNITIVE MAPS

This chapter describes, how the animals and humans deal with the representation of the space. Such a representation can be source of inspiration for a robotics.

The mental representation of the environment is called cognitive map. This term was created by E. Tolman in [81]. During his experiments with rats in a maze, he observed that rat's behavior is not a matter of mere simple stimulus-response connections. Rats rather use an inner representation of the surroundings which Tolman calls the cognitive map. The cognitive map was used in the case the learned path was closed. The rats were able to choose the correct alternative route even they never went these. The cognitive map is learned by rats even if they are not rewarded for it, and they also remembered what they learned previously.

Downs and Stea in [16] define cognitive mapping as "a process composed of a series of psychological transformations by which an individual acquires, codes, stores, recalls, and decodes information about the relative locations and attributes of phenomena in his everyday spatial environment."

Many researches from different branches of the science are interested how the human beings and animals cope with spatial knowledge. The process of acquisition, manipulation and utilization of the cognitive maps is studied from different points of view and for different reasons. Psychology and pedagogy is interested in a description how this mental representation is being developed during growing of the children and how the people learn the environment. Neuropsychology still tries to find how the hippocampus is related to spatial reasoning and learning. Biology studies how the animals (also the humans) are finding their way. Also architects are interested in how people sense the surroundings and navigate in the city.

4.1 DEVELOPMENTAL THEORY

The developmental theory is based on children's cognitive maps development. The researchers mainly build on Piaget theory [63] and recognize three stages of the development: (1) Landmarks, (2) Route map and (3) Survey map.

4.1.1 *Landmarks stage*

The landmarks level is the first stage and can be also called object level. Children start to recognize objects and are able to distinguish them. Landmarks are objects in the environment vital in determining orientation and current location. Landmarks function as a kind of environmental index. The fundamental property of the landmarks is that they must be uniquely identifiable. In reality, this requirement can hardly be fulfilled.

4.1.2 *Route map stage*

At the route map stage, children connect the landmarks by the route. These routes reflect the children direct experiences rather than abstract connections. The route map is egocentric due to reliance on the direct and personal experience. As the child is moving through his neighbourhood their cognitive map reflects it. The routes connect the landmarks and have a topological character.

4.1.3 *Survey map stage*

The survey map stage is defined by a qualitative change of the space knowledge representation. This change can be described as a movement from egocentric to allocentric view. It means the objective frame of reference appears. One stream of existing theories is modelling this objective frame of reference as a precise Euclidean spatial information. But there are little experimental data to support this notion in people's mind. Also it appears the ability to determine spatial relationships of the objects that are not close in space in this stage.

The survey map provides and overview of the large-scale space which is usually too large to seen at once. Once the survey map is developed, the global information is available for solving different types of tasks, e.g. way-finding.

4.2 NEUROLOGICAL THEORY

Neurological theory describe the cognitive mapping as a neurological process in the brain. O'Keefe and Nadel in [62] formulate a hypothesis that the primary function of hippocampus is to form a cognitive map. This theory is based on the existence of place-coded neurones in the hippocampus of rodents and specific rhythmic electrical activity recorded from the hippocampus during certain behaviours. The activity of different place cells in hippocampus corresponds to sensory stimulus from different places. Place cells do not form a Euclidean representation of the environment nor are strictly topological as the rodents are able to find shortcuts.

Later experiments with primates and humans described hippocampal cells that fire in relation to the place the subject is looking at, rather than the place its body is located. It is a frequent observation that without a fully functional hippocampus, humans may not remember where they have been and how to get where they are going.

4.3 computational models of cognitive maps

This section presents the models of cognitive mapping made by scientists from the field of computer sciences and artificial intelligence. They build their models on the psychological and neurological theories, but use formalism common in computer sciences and artificial intelligence. This formalism allows to run simulations on the computers. These simulated results were reused by psychologists and neurologists and bring novel approach into the field of cognitive sciences.

4.3.1 *TOUR model*

The first comprehensive computational model of the human cognitive map is a TOUR model introduced by Kuipers in his PhD thesis [42]. The TOUR model [43] was strongly inspired by Kevin Lynch's seminar book, The Image of the City [53], and by studies of the development of children's spatial knowledge [63]. All evidence suggests that the human cognitive map represents a space quite differently from a printed map, which has a single global frame of reference. Unlike previous attempts to model spatial knowledge, the TOUR model included several distinct representations for large-scale space, such as procedures for following a route from one place to another, a topological map containing places connected by paths, and localized metrical maps with separate frames of reference.

The TOUR model divides spatial knowledge into five categories and each category has its' own representation.

routes are represented as a sequences of actions.

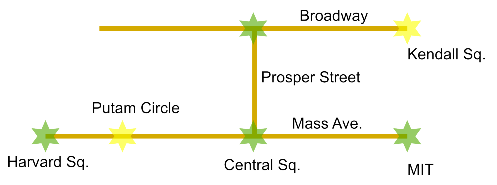
a topological structure represents local topological properties of route networks, including the ordering of places on a route and local geometry of intersections.

frame of reference defines relative positions of objects with respect to only local referential point. Different frames of reference may not be comparable.

dividing boundaries provide a qualitative partial knowledge of position.

regions provide useful levels of abstraction for stating properties of their elements.

There are three classes of representation in the TOUR model: (1) representation of the particular environment , (2) description of the current position and (3) inference rules to manipulate the knowledge of two previous classes. Both the description of current position (see Fig. 1b) and the environmental descriptions (see Fig. 1c) may be incompletely specified. In this case, the TOUR model will function properly, although with degraded performance.



(a) Map

You are here:

Place: [Place2: Central Square]

Path: [Path3: prospect Street]

Direction: +1

(b) Current position

N

Path1:

Name: Mass Ave

Row: (Place1 Place2 Place3)

Path2:

Name: Broadway

Row: (Place4)

Path3:

Name: Prospect Street

Row: (Place4)

Place1:

Name: Harvard Square

On: [Path1: Mass Ave]

Star: nil

Place2:

Name: Central Square

On: [Path1: Mass Ave]
[Path3: Prospect Street]

Star: (0. Path1 -1)
(90. Path3 -1)
(180. Path1 +1)

Place3:

Name: MIT

On: [Path1: Mass Ave]

Star: nil

Place4:

Name: nil

On: [Path2: Broadway]
[Path3: Prospect Street]

Star: nil

(c) Environment representation

Figure 1: TOUR computational model of human cognitive map.

The inference rules that manipulate knowledge take place mostly through an interaction between the environmental description and description of the current position. Furthermore, the only accessed environmental descriptions are the ones referred by the description of the current position. Therefore the most operations are quite efficient due to a lack of search for environmental elements.

The inference rules fall into categories that correspond roughly to the kinds of representation.

- Rules which work with the route instruction and can complete the missing parts in each representation with information from the others.
- Rules maintaining the current heading or 2-D orientation with respect to current frame.
- Rules detecting structural features such as dividing boundaries.
- Rules which use the hierarchy of regions, boundaries and referential frames to solve the route-finding and position-finding problems.

The TOUR maintains a topological model of learned environments, but stores route knowledge separately, in production rules. The TOUR model has been criticized for the separation of route knowledge and topological model of the learned environment. Kuipers admits [51] that this route knowledge is insufficient to find novel routes or shortcuts. However, the TOUR has the capability to deal with these shortcomings to some degree by the use of its stored topological information.

4.3.2 *PLAN model*

PLAN (Prototypes, Locations and Associative Networks) [12] is cognitive mapping theory, build on the top of the developmental theory. Basic functions of human cognitive map during the way-finding defined in PLAN are:

LANDMARK IDENTIFICATION is the most basic component of way-finding. The landmark identification problem in way-finding is to separate out distinctive objects in the environment, called *landmarks*, which can later be used in route planning and can be recognised while traversing the chosen route. The landmark identification problem primarily concerns the object recognition system.

PATH SELECTION involves choosing a route to the goal. In this case a path is not a direction, but is more algorithmic, for example, a series of places that will lead to the goal. In many models of cognitive mappings, paths are conceptualised as sequences of landmarks. To follow a path one goes from landmark to landmark in the sequence. The path selection problem is cognitive, often requiring the selection of one path among a number of alternatives.

DIRECTION SELECTION involves choosing a direction in which to travel. The direction selection problem, while generally visual, is more locational than the landmark identification problem If the goal is in sight, a reasonable direction to pick would be towards the goal. For goals that are not in sight the direction selection problem is more difficult; beyond the fact that the goal cannot be seen, sometimes a journey will require a series of turns and shifts in direction. Thus, direction selection at the starting point is rarely sufficient to guide an entire route.

ABSTRACT ENVIRONMENTAL OVERVIEWS are further generalizations of the route concept. If it is required to travel extensively in a particular environment, it would be useful to have a coherent overview of the entire environment. Rather than dealing with routes individually, such a structure would allow them to be extracted from a common abstraction. In addition, this overview would make large-scale reasoning about the environmental simpler. However, these overviews do serve to increase the efficiency of way-finding approaches without these afore mentioned capabilities. The problem of creating and abstract environmental overview requires a hierarchical synthesis of each of the other three solutions.

The PLAN builds on the three basic concepts: prototypes, locations and associative networks. The short description of them follows.

Prototypes

In the PLAN model, the landmarks are treated as a category called *prototypes* rather than a single, unique object. These categories are a generalization derived from a range of experience. In the case of a landmark, the landmark can be experienced from different angles, distances, orientations etc. The most typical features of the landmark will be part of the prototype. The prototypes are organized into hierarchies, where individual examples will be at the bottom and higher levels bring an increase of the generality.

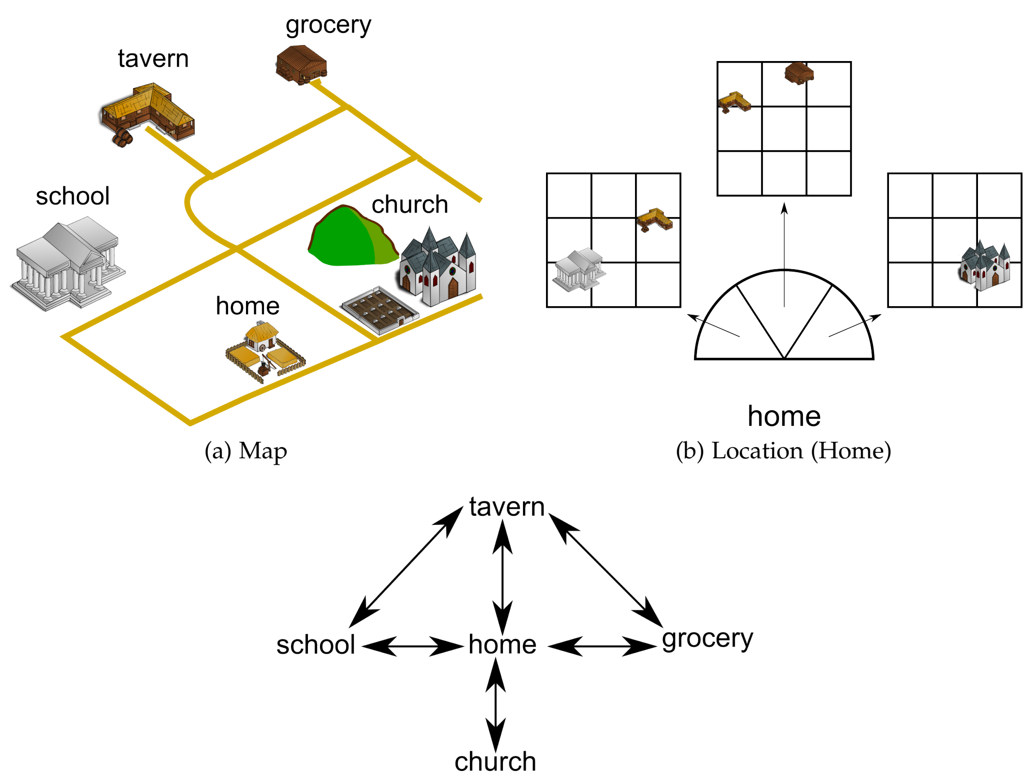
The PLAN model imposes following requirements on the landmarks:

1. Landmarks must be recognizable. This in turn requires that (a) a landmark must be recognizable from a variety of views and orientations, and (b) that in many cases only a partial view of the landmark should be sufficient to activate the entire representation.
2. The number of landmarks which can be active, or processed, at one time is limited to 5 ± 2, the number of objects that a person can think of at one time.
3. Landmarks are intimately linked to the context. A good landmark in one environment may be a poor one in another environment.

Therefore, not every object can be treated as a landmark because the fundamental property of landmarks is that they are uniquely identifiable.

Associative Network

A topological system is encoded into a *associative network* where the nodes represent landmarks, and directed links between them represent spatial proximity. In such a network paths could be extracted by following the links from one



(a) Map (b) Location (Home)

(c) Associative Network

Figure 2: PLAN computational model of human cognitive map.

landmark to the next. This associative network is called Network Activity Passing System (NAPS). All of the information is stored locally; the only other landmarks that are connected to a landmark x are those that can be seen from x.

The path searching problem is solved in connectionist manner by spreading activation from both the starting point and the goal location. The activity waves will coalesce at some intermediate node which is treated as a subgoal and the process is repeated until the complete path is extracted.

Location

A location is a local directional representation of the landmarks. An example can be seen on Figure 2b. It is able to provide a relative change in orientation for any neighboring target landmark. One of the hypotheses of the PLAN system is that the directional structures used in cognitive mapping directly reflect how humans process information (mainly visual information). Visual information is treated as a fixed 2-dimensional picture. The orientation of the picture is related to the position of the eyes, the head and the body.

The locations are stored in associative network called R-Net similar to NAPS used for storing the landmarks (in the form of prototypes). R-Net has no attempt to construct a single global spatial representation integrated over a large number of scenes. Only local information is taken instead.

The regional maps represent the survey map functionality. It is an abstraction of the R-Net and network of landmarks. Regional maps reduce the amount of information and not only provide another organization of it. More information can be extracted from lower level representation if needed.

4.3.3 *Yeap's computational model*

Yeap in [87] focuses on a problem how the information perceived directly from sensors are used to compute a cognitive map. Yeap builds his computational model on the Marr's theory [56] of vision. In short, Marr suggests the process of vision should be studied in three steps:

1. Primal sketch makes explicit the types of intensity changes present in the image.
2. $2\frac{1}{2}D$ sketch makes explicit the shape and disposition of surface relative to the perceiver.
3. 3D model representation makes explicit the three-dimensional shape of the surfaces perceived.

The cognitive map is studied as two loosely coupled modules in this theory:

RAW COGNITIVE MAP produces a map of the environment from information gathered through sensory modalities.

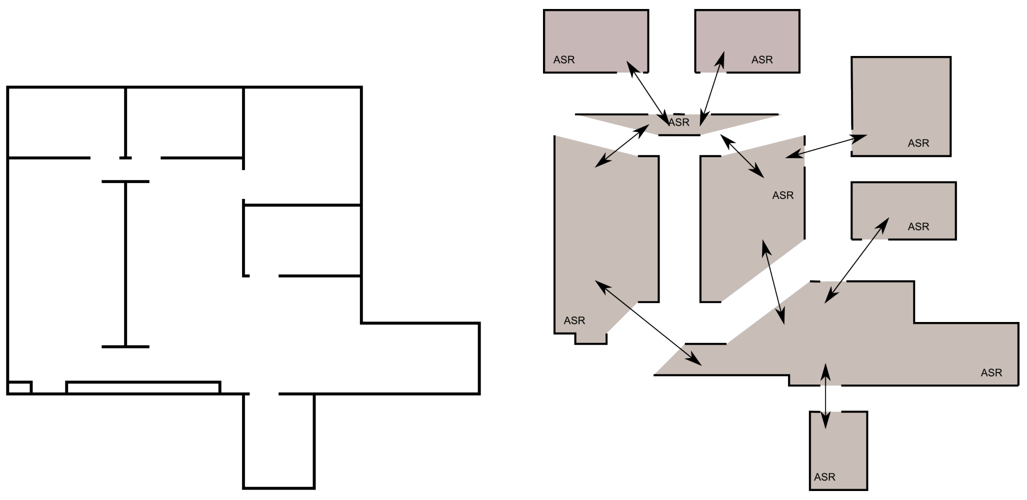
FULL COGNITIVE MAP takes the map as input and produces different spatial tasks.

Raw Cognitive map

The raw cognitive map (see Fig. 3) is represented as a non- egocentric and structured *relative-absolute model*. The important properties of the relative-absolute model are: its parts are computed locally, allows incremental building and represents the structure of the experience rather than the structure of the world.

Each local space is described with *absolute space representation - ASR*. The Absolute space representation is computed in two steps: the first identifies the extent of the space and the second computes a description in form of a volumetric descriptor computed from the $2\frac{1}{2}D$ sketch.

The global space is referred as *relative space representation*. The main problem is recognize the parts (in form of ASR) of an environment which have been visited before. The following assumptions simplify the implementation: The ASR for each part of environment is complete and ASR contains only its boundary description. The relative space representation is computed simply by relating the ASRs via their common exits.



(a) Map (b) Raw Cognitive Map

Figure 3: Yeap's computational model of human cognitive map.

Full Cognitive Map

The full cognitive map consists of different place representations which are formed by grouping the ASRs. The following types of information are made explicit:

- The inter-level connection is the linkage for building hierarchical representation.
- The intra-level connection represents the structural relationship between members of groups.
- The exit connection expresses the possible exits in terms of the actual ASRs and allow to get knowledge where one can move from current place without searching down the hierarchy.

Unlike the TOUR model, this model represents paths implicitly by connecting ASRs. Also there are no explicit landmarks in this model.

The description of spatial models used in robotics follows in next chapter.

5 ROBOTIC MAPS

Robotic mapping addresses the problem of acquiring spatial models of physical environments through the mobile robots. To acquire a map, robots must be equipped with sensors that enable it to perceive the surrounding world, e.g., cameras, range finders like sonar, laser and infrared technology, radar, tactile sensors, compasses and GPS. All these sensors are subject to errors, measurement noise, and have limited resolution and range. At the present, robust methods exist for mapping environments that are static, structured, and of limited size. Mapping unstructured, dynamic, or large-scale environments remains largely an open research problem.

Two paradigms for map representation have been mainly pursued in robotics, metric and topological maps.

5.1 METRIC MAPS

A metric map represents the environment by collecting positions of relevant landmarks, features and object with respect to a single metric frame of reference i.e. a global coordinate system [77], [49], [64]. During the process of building a metric map is necessary to determine the position of a robot in a global frame of reference. When the environment is a large-scale space, then a localization becomes an important issue. Therefore the localization and mapping is tightly coupled. Simultaneous localisation and mapping (SLAM) or concurrent mapping and localisation (CLM) [70], [75], [58], [11] solve simultaneously the mapping problem and the induced problem of locating the robot relative to its growing map.

In connection with SLAM techniques, three categories of metric map can be recognized: (1) feature-based, (2) location-based and (3) view-based.

The feature-based approaches define map as a list of parametrized objects with specified poses. Features like points, lines, corners etc., need to be extracted from the raw sensor picture of the environment and therefore maps are limited to environments for which the features are designed.

Location-based maps divide the environment into a number of regions. A special case of this concept is *occupancy grid* [24], [60], which represents free and occupied space in a rectangular grid at a fixed resolution and does not require a-priori knowledge about the environment, objects or landmarks. Every grid cell is associated with one or more values, which correspond to physical properties, whereas one of them can be the presence of obstacles. The main disadvantages of location-based map are resolution-dependency, memory consumption and computational intensity of many manipulations done over its content.

View-based maps [38] are composed of collection of full sensor readings associated with pose of the readings. These approaches can be considered as a subtype of feature-based approach where the entire scan is considered for a feature. This approach does not limit usability to any environment type and there is also no information loss. Since the sensor readings usually contain large amounts of data, the view-based maps are more demanding in terms of memory and computational resources. Moreover the view-based maps are dedicated solely for localization and navigation purposes. Easy visualisation and their use as a human readable output belongs to advantages of this approach.

Metric maps excel in solving of some low-level problems encountered in robotics [46, 48]. Metric maps reduce a pose error in small-scale space but this error dramatically grows over large-scale spaces. As a consequence, metric maps cannot easily handle large cyclical environments once the position has drifted excessively. Perceptual aliasing makes it difficult to handle loop-closure event.

Mapping and planning in very large metric maps can be time consuming. Metric maps also suffer from the lack of a good interface to higher-level symbolic problem solvers.

5.2 TOPOLOGICAL MAPS

A topological map [42], [69], [57], [11] generally represents spatial knowledge as a graph, describing locations and object of interest as nodes and their spatial relations as edges. The concepts mostly used in topological mapping are described in following sections.

5.2.1 *Spatial Semantic Hierarchy*

The basic concept outlined in the TOUR model (see Sec. 4.3.1) has been refined into the Spatial Semantic Hierarchy (SSH)[44]. The SSH treats observations gathered during movement through environment as the fundamental source of experience for building a cognitive map of large-scale space. The SSH approach uses a natural five-level semantic hierarchical description of large-space.

SENSORIMOTOR LEVEL describes the sensorimotor system of the robot which provides the sensors and effectors.

CONTROL LEVEL. Control strategies and sensory measures are used to define distinctive states and produce control trajectories which move the robot from one distinctive state to the neighbourhood of another. A distinctive state is defined as a local maximum found by a hill-climbing control strategy to maximise a selected sensory feature or distinctiveness measure.

causal level works with views, actions and schemes. The views are sensory readings at a distinctive states. Actions represent the control law trajectories through which the robot moves from one view to another. The trajectories are a result of an application of a sequence of more than one control strategy. Schemes provide the causal relations among the views and actions.

topological level works with places, directed paths and paths abstracting relationships among the distinctive states and trajectories defined at the control level and relationships among views and actions at the causal level. The places and paths are nodes and edges in a topological map represents environment. This map can be used for solving problem like finding route.

geometrical level includes geometrical properties as shapes, relative distance and orientation, absolute distance etc. This geometrical properties are represented as annotation on the nodes and edges of the topological graph.

The logical dependencies among the levels are depicted on Figure 4. Different parts of the cognitive map may represent knowledge at different SSH levels, but each part of the map must respect the dependency structure.

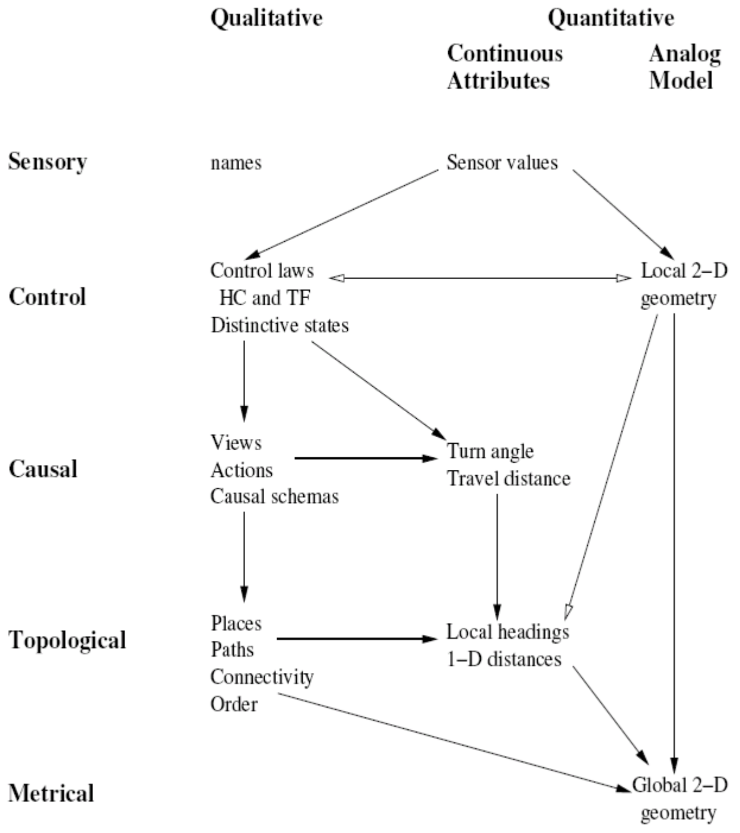


Figure 4: Spatial Semantic Hierarchy scheme according to [44]

To verify the proposed model works, Kuipers and Byun implement exploration and mapping algorithm based on SSH for a simulated robot NX [45]. NX robot has sixteen sonar distance sensors covering 360 degrees and an absolute compass for global orientation. Another implementation of exploration and mapping algorithm based on SSH is [50]. Lee implements his algorithm for the real robot Spot. Spot has three-wheeled base and ring of twelve sonar sensors.

The formal axiomatization of the causal and topological levels is provided in [69]. As the topological map is the result of an abduction process, finding the best consistent explanation of the available observations, the formalization requires a non-monotonic logic. This non-monotonic logical inference is implemented as an algorithm that creates a tree of all possible topological maps with a preference order of the leafs. The leafs represent the topological maps consistent with experience so far.

Savelli [71] subsequently augmented the existing inference system with the test for the planarity of the topological maps. This planarity test could be applied either as a consistency requirement of as a preference criterion.

5.2.2 *RPLAN*

Kortenkamp in [39] implements the cognitive map theory PLAN (Prototypes, Location and Associative Networks) [12] for a mobile robot. He calls his implementation RPLAN - Robot PLAN.

RPLAN uses the integration of sonar and vision through two theoretical concepts, gateways and scenes. The gateway is a place that is a choice point and new landmarks can be seen. In a building, these are typically doorways, outside they occur where the visual narrowing is followed by visual opening. The gateways are detected using the sonar sensors and correspond to big changes in the spatial surroundings of the robot. The scenes are visual images taken and stored by the robot at gateways and provide information that allows the robot to discriminate among gateways.

Routes connect adjacent gateways. At each gateway, the route informs the robot what direction to turn in order to move toward the next gateway along the way to the goal. Routes are stored in a network (topological map). To attach a direction to each connection in the network, Spreading activation network was used. The spreading activation network is composed of nodes, which represent places, and subnodes of nodes, which represent directions. These subnodes are connected together with links of varying strengths.

RPLAN creates an additional level of representation on top of regional networks, called regional maps. Regional maps correspond to what are called *survey maps* and support the fourth function of cognitive maps: Environmental Abstraction. Regional maps allow the robot to reason about its environment from a global perspective.

5.3 HYBRID MAPS

A hybrid map combines metric and topological approaches to support advantages and suppress disadvantages of both. The hybrid map, according the [9], can be defined as a pair $H =< \mathcal{M}, \mathcal{C} >$ where $\mathcal{M} = \{M_1, \dots, M_n\}$ is a set of maps called components and $\mathcal{C} = \{c_1, \dots, c_p\}$ is a set of links. Each link is a pair $< c_i, c_j >$, where m_i is an object of M_i and m_j is an object of M_j, with $i \neq j$.

This definition is wider than what is usually means by term "hybrid map". Therefore, in rest of the section, the hybrid map means a heterogeneous hybrid map, where at least two of its components are of essentially different types, hereafter the metric and topological.

Synergies, which can increase performance to a level that would be hard or impossible to achieve using only single component, are gained by exchanging and combining information between the components. Without this interaction, the hybrid map would be simply a storage of independent maps and most of the advantages would be lost.

Typical form of the hybrid map is a $\mathcal{M} = \{T, M_1, M_2, \dots, M_N\}$, where links $\mathcal{C}$ connect the nodes of the topological component map with the local metric maps describing in details the places represented by the nodes.

Typical example of this form is hybrid spatial semantic hierarchy [41]. The SSH approach, where the topological map is a primary structure, was extended by *local perceptual map* metrically accurate representation of the distinctive places. Where the basic SSH treats views as atomic symbols, the hybrid SSH treats the local perceptual map as the observable manifestation of a topological place. The SLAM methods are used for creating local perceptual map. The problem of closing loops is avoided while whole structure of the local map lies within the sensory observation horizon.

Local SLAM method continually maintains robot's localization in the frame of reference of the local map. Accurate incremental localization supports incorporation of observations into the local map, and accurate local motion planning. In the basic SSH, hill-climbing provides the localization but at the cost of physical motion to the distinctive place. As long the robot has enough knowledge to maintain its localization within the local perceptual map, it no longer requires physical motion.

Also Tomatis at al. in [82], [83] and [84] describe environment by a global topological map which permits moving in the whole environment, and local metric maps which the robot can use as soon as it required improved localisation precision. Local metric maps are represented by infinite lines.

Thrun et al. [80], [79] divide mapping on two phases. The topological mapping phase solves a global position alignment problem between potentially indistinguishable, significant places and the subsequent metric mapping phase produces a fine-grained metric map of the environment.

Simhon and Dudek [73] use occupancy grid as local metric descriptor of distinctive places. The global map is formed from a set of local maps organised in a topological structure. Local maps are denoted as *islands of reliability* because they provide accurate metric information.

The approach entitled Atlas [6] has same form, but metric maps are the primary structures and the topological component provides an additional information. Atlas is a SLAM framework, comprising multiple small-scale mapping algorithms, can be used to achieve real-time performance in large-scale, cyclic environments. The map representation herein consists of a graph of multiple local maps of limited size. Each vertex represents a local coordinate frame of reference and each edge denotes the transformation between local frames as a Gaussian random variable.

Jefferies at al. [29], [30] use a hybrid map in form of $\mathcal{M} = \{T, M_1, \ldots, M_N, M_G\}$ There are also local metric maps to describe places but the hybrid map has the global metric map M_G in addition. Combination of information from the topological map and absolute global metric map use for detection of cycles. The main purpose of this approach is to close cycles in the topological map. However with the cycle closed there is the opportunity to realign the global metric map, correcting the error backwards through the map.

Duckett [19] uses the same hybrid map form with the global metric map as a requested output. The topological map is used for ensuring the consistency of the global map using the relaxation technique to deal with the errors in position estimation.

FABmap [13], [14] is an approach to topological SLAM based on visual appearance. FABmap is focused on localization and recognition of previously visited places being very efficient for loop closing. This framework is focused on the vision-only localization.

The hybrid maps are likely to become the dominant paradigm for representing spatial information in autonomous robots. The main advantages of the hybrid map are that it covers a larger extent of space and with better resolution, than would by possible by any of it component maps.

The majority of maps used in robotics are focused on the localization of the robot. This thesis focuses on the proposition of the map representation more focused on the navigation of the robot, as the navigation is a crucial ability to operate the robot in a large environment. The detailed comparison of widely used spatial representation is provided in Chapter 8.

6 EXPLORATION

The process of exploration can be understood as a process of autonomous navigation of a mobile robot in an unknown environment in order to build a model of the environment. An exploration algorithm can be defined as an iterative procedure consisting of a selection of a new goal and a navigation to this goal. Such an algorithm is terminated whenever the defined condition (mission objective) is fulfilled. Besides, the usage of resources (e.g. the exploration time, the length of the trajectory) is optimized. While optimal robot motion is relatively well-understood in fully known environments, exploring robots have to cope with partial and incomplete knowledge.

The exploration strategy determines the next robot goal in each exploration iteration (one exploration step) with respect to the actual robot position, the current knowledge about the environment, and a selected optimization criterion. Any exploration strategy has to be able to adapt to any unexpected situations during map acquisition.

The gready algorithms is often used as an exploration strategy. Robot always moves from its current location to the closest location that it has not been visited (or observed) yet, until the environment is mapped.

The exploration algorithm depends on the type of the created map. Following sections give overview of the approaches for the metric and topological maps.

6.1 METRIC EXPLORATION

For a metric maps, exploration algorithms based on the Yamauchi frontier-based exploration [86] are widely used. Only a sort description of these algorithms is presented here, as the metric exploration is not a goal of the thesis.

The main idea behind frontier-based exploration is:"To gain the most new information about the world, move to the boundary between open space and uncharted territory." Frontiers are regions on the boundary between open space and unexplored space. By moving to successive frontiers, the robot can constantly increase its knowledge of the world. A greedy exploration strategy is used in Yamauchi's original paper .

The other approaches modify exploration strategies of the frontier-base exploration. The strategies differ in the way how candidates for the next goal are generated and in the criterion how the best candidate is selected. The authors of [28] discussed two simple heuristics improving Yamauchi's approach. The first one uses Voronoi diagrams to prefer exploration of the whole room in office-like environments before leaving it, while the second one repetitively re-checks whether the currently approached goal is still a frontier. A strategy selecting the leftest candidate according to a robot position and orientation with a defined distance to obstacles is described in [59].

The work[47] proposes the strategy using the distance cost that reflects traveling through all goal candidates. The cost is determined as a solution of the Traveling Salesman Problem.

6.2 TOPOLOGICAL EXPLORATION

The topological exploration is an exploration where the topological map is a model to build. This problem can be seen as a graph search, where the graph is not known in advance.

In [37] bounds of worst-case travel distance of greedy exploration is discussed. The worst-case travel distance of greedy mapping is $O(|V|^{3/2})$ on strongly connected undirected graphs $G = (V, E)$.

Bender et al. [3] solve the problem of exploration of the graph with directed edges. The vertices are completely unlabeled, but the outgoing edges of each vertex have global ordering. The robot has ability to place a marker at the vertex and recognize this marker later and to recollect it. It is shown in the paper, if the upper bound $\hat{n}$ on the number of vertices n is known and a single marker is used, the running time is $O(\hat{n}^2 n^6 d^2)$, where d is vertex degree. In addition, if the upper bound on the number of vertices is not known, it is needed $\Omega(\log(\log(n)))$ markers at least.

Dudek in [23] introduces an algorithm based on the maintenance and validation of an explored graph. As new vertices are encountered, they are added to the explored graph, which is a subgraph of the full graph, and their outgoing edges are added to the set of edges that lead to unknown places and therefore must be explored. This new vertex must be validated. Validating a vertex means making sure that it is not identical to any other vertex in the explored subgraph. This is carried out by placing a marker and visiting all vertices of the known subgraph S along edges $e \in S$, looking for the marker. If marker is not found, then vertex must be added to subgraph. To improve the performance multiple markers are used.

Rekleitis [68] proposes the exploration technique that uses the marker for undirected planar graph where the vertices are unlabeled and edges have local ordering related to the one it entered by. The main idea of the exploration is the exploration of closed path called an *ear*. Ear can be defined as a closed cycle obtained by leaving a vertex on a specific edge and selecting for traversal, at the following vertex, the edge that is next to the entry edge in a consistent orientation, until return to original vertex. Any planar graph can be decomposed into a union of ears.

The ear searching strategy is used also in [55] without using a marker. The robot keeps the exploration tree, similar to previous works [20],[21]. The exploration tree refers to the collection of possible hypotheses about the world the robot is exploring, consistent with the data accumulated so far in the exploration. The root of the tree is the initial location from which the exploration began. A level in the tree corresponds to the traversal of a previously unexplored edge. The nodes in a given level represent possible partial models of connectivity in the world, according to the locations visited so far. The hypotheses here are pruned on the principle of Occam's Razor.

Map verification is discussed in [15]. The task is to find out whatever the given map M is correct for the world G. In this work a strategy is shown, that verifies a map in $O(V_M)$ edge traversals, using a single edge marker, when M is a plane embedded graph, even though G may not be planar. Note that here is a *edge* marker used in contrary to previous works where the *vertex* marker is used.

6.3 multi-robot exploration

Usage of multiple robots is a natural extension of the exploration task. Exploration and mapping, according the Dudek's taxonomy [22], are the tasks which can group of robots perform more effectively than a single robot. Bender and Slonim [4] show that two cooperative robots can learn directed graph with unlabeled vertices using the homing sequence. They show that, the robot with single marker cannot learn a unknown directed graph in polynomial time but the two cooperative robots can.

The usage of multiple robots brings new problems to solve. To achieve the improvements over the single robot exploration, the robots in group must be coordinated and cooperative. Coordination mechanism ensure coherent behaviour of the group. Without coordination, all robots might follow the same exploration path, so that the group requires the same amount of time as a single robot would need. Therefore, coordination mechanism must choose different actions for the individual robots so that they simultaneously explore different areas of their environment.

In [8] authors used cooperative point selection instead of greedy navigation. Target point selection is based on the trade-off between the costs of reaching the target point and its utility. To determine the cost of reaching the current frontier cells, they compute the optimal path from the current position of the robot to all frontier cells using dynamic programming algorithm.

Cost of moving to neighbourhood is equivalent to the probability that the cell is occupied multiplied the distance to the cell. Utility $U_{x,y}$ is based on expected visibility range. Initially, the utility is set to 1. Whenever a target point is selected for a robot, we reduce the utility of the adjacent point in distance d according to their visibility probability $P(d)$. Probability $P(d)$ is updated during exploration and represent probability that the robot's sensors cover object at distance d.

Utility function represents the coordination mechanism. Whereas uncoordinated robots would choose the same target position, the coordinated robots select different frontier cell with the best overall evaluation.

Rekleitis et al. [65], [66] and [67] focus on the problem of reducing the odometry error during exploration. The robots explore the environment in teams of two; each robot is equipped with a robot tracker sensor that observes the other robots and reports its relative pose. The observing robot uses the position of its partner in order to update the estimate of its position.

Robot tracker sensor is a camera in [67], that allows to observe its partner. The robots are marked with a special pattern for pose estimation. The first part of the pattern is a series of horizontal cylinders. This allows the robot to be easily discriminated from background objects. The second component of the pattern is a helix that wraps once around the robot. The elevation of the center of the helix allows the relative orientation of the robot to be inferred.

In [66] a laser range-finder is used as robot's tracker sensor. That allows to cooperate in bigger groups. The motion planning strategy is such that at any time, one of the robots is stationary while the other robot is moving and acts as an artificial landmarks. On the observed robot, a target is mounted which consists of a set of three vertical planes extending from the center of the target at three distinct angles (approximately 100°, 120°, 140°). The intersection of the two planes defines a unique point and the angle between the two planes combined with their orientations provides an estimate for the orientation of the robot.

The coordination algorithm for multi-robot exploration in [76] applies a unsupervised clustering algorithm (K-Means). Unknown cells are partitioned into as many clusters as available robots by applying K-Means and each robot is assigned the region with the closest centroid. Every robot then moves to the frontier cell with the lowest cost, senses the environment from it and proceeds with the next lower cost frontier. The cost of a frontier receives a very significant, constant penalisation when the frontier does not belong to the robot's assigned region, as well as a variable penalisation that consists of the distance between the frontier and the center of the assigned region.

The proposed spatial representation should support the spatial knowledge sharing among multiple robots, to allow multi-robot exploration. It will be an advantage, if the spatial knowledge could be exchanged between robots with diverse sensors and capabilities.

7

REASONING UNDER UNCERTAINTY

This chapter describes the current state of the art in the area of the reasoning and inference under uncertainty. As the logic is the formal systematic study of the principles of valid inference and correct reasoning, this chapter focuses on the logic as a tool for reasoning. The reasoning is understood here as the process of drawing new facts from set of observations. It remains irrelevant if it is a deductive reasoning, where from preconditions and rules are inferred a conclusion, inductive reasoning discovering the rules from set of examples or abductive reasoning determining the preconditions for observed conclusions.

In context of robotic mapping, the reasoning brings the ability to infer new facts, which are not directly accessible trough the sensors. It is the process, how to build the spatial knowledge from the spatial information.

The formal logic is tightly connected with the reasoning and inference from ancient times, but the classical formal logic works with the facts or evidences which are certainly true or false. In the real world, there is nothing totally certain, especially in context of the robotics. The world is perceived using the sensors, which are subjected to errors and noise. Therefore this chapter introduces the different extensions of formal symbolic logic, which allow to incorporate the uncertainty in a quantitative way into the reasoning process. At first, the concept of uncertainty is introduced.

7.1 UNCERTAINTY

According to Smets [74] uncertainty concerns the state of knowledge of an agent (a robot) about relations between the world and the statements about the world. The statement is either true or false, but agent's knowledge about the world does not allow agent to decide if the statement stands really true or false. The uncertainty is a partial knowledge of the true value of the data and results in ignorance (having the meaning a "do not know") in opposite to certainty, which is the full knowledge of the true value of the data. The major cause of the uncertainty is imprecision in the data.

The uncertainty can be expressed and measured by Sugeno's fuzzy measures [78]. Even though has been called fuzzy measure, it should not be confused with the fuzzy set theory. The Sugeno's measure express the uncertainty associated with a statement *"x belongs to S"* where S is a crisp set and x is a particular element of X which is not a-priori located in any of the subset of X. The Sugeno's measure g satisfies the following properties:

$$g(\emptyset) = 0 \tag{7.1a}$$
$$g(X) = 1 \tag{7.1b}$$
$$\forall A, B \subseteq X, A \subseteq B, \quad g(A) \leq g(B) \tag{7.1c}$$
$$\forall A_i \subseteq X, i \in \mathbb{N}, \quad \lim_{i \to \infty} g(A_i) = g(\lim_{i \to \infty} A_i) \tag{7.1d}$$
$$\textit{where} \quad A_1 \subseteq A_2 \subseteq A_3 \ldots \quad \textit{or} \quad A_1 \supseteq A_2 \supseteq A_3 \ldots$$

The property (7.1c) is called monotony, and property (7.1d) is called Sugeno's convergence. The Sugeno measure is a normalized measure, monotonous for inclusion for finite X. Note, that the additivity property is not needed. It fits with probability measures, possibility measures, necessity measures, belief functions, plausibility measures.

There exists different approaches to model uncertainty and work with it. The most used approaches based on symbolic logics are described in the following sections. The most suitable approach will be chosen from them at the and of this chapter.

7.2 FUZZY LOGIC

The fuzzy logic [5] is based on fuzzy sets introduced by Zadeh in [88]. A fuzzy set A is defined by a membership function μ_A that assigns each element x a degree of membership to $A : \mu_A(x) \in [0, 1]$. In contrast to classical "crisp" set, where an element can either belong to a set or completely outside of this set. Classical sets allow only values 1 and 0 of the membership function, whereas fuzzy set theory deals with the whole interval between 0 and 1.

In the fuzzy logic, the logic operators are defined as follows.

$$\begin{array}{rrr}
\textit{Complement} & \mu_{\neg A}(x) = & 1 - \mu_A(x) \\
\textit{Conjunction} & \mu_{A \wedge B}(x) = & \min\{\mu_A(x), \mu_B(x)\} \\
\textit{Disjunction} & \mu_{A \vee B}(x) = & \max\{\mu_A(x), \mu_B(x)\} \\
\textit{Implication} & \mu_{A \to B}(x) = & \max\{1 - \mu_A(x), min\{\mu_A(x), \mu_B(x)\}\}
\end{array}$$

Besides, there exists alternative definitions of the operators. The following definitions are common and used.

$$\begin{array}{rrr}
\textit{Conjunction} & \mu_{A \wedge B}(x) = & \mu_A(x) \cdot \mu_B(x) \\
\textit{Disjunction} & \mu_{A \vee B}(x) = & \min\{\mu_A(x) + \mu_B(x), 1\} \\
\textit{Implication} & \mu_{A \to B}(x) = & \min\{1, 1 - \mu_A(x) + \mu_B(x)\}
\end{array}$$

However, using any of these definitions not all tautologies from classical logic are valid for fuzzy logic, e.g., $A \wedge \neg A \neq \bot$ and $A \vee \neg A \neq \top$.

The inference mechanism of the fuzzy logic is the generalized Modus Ponens, describing how from $\mu_{A'}(x)$ and $A \to B$ infer $\mu_{B'}(y)$. It is worth to mention that membership functions of A' and A can be different.

Min-max norms are used for determination of $\mu_{B'}(y)$ as follows equation:

$$\mu_{B'}(y) = \sup_x \{\min\{\mu_{A'}(x), \min\{\mu_A(x), \mu_B y\}\}$$

Alternatively the Lukasiewicz's definition follows:

$$\mu_{B'}(y) = \sup_x \{\min\{\mu_{A'}, 1 - \mu_A(x) + \mu_B(y)\}\}$$

Both definitions lead to different interpretation of the fuzzy implication, but also other interpretations exist.

7.3 probabilistic logic

The term *probabilistic logic* was firstly used in [61] by Nils Nilsson. The probabilistic logic is a semantic generalization of a logic in which the truth values of sentences are probability values. This generalization applies to any logical system, where the consistency of a finite set of sentences $\mathcal{S}$ can be established.

Let a set of possible world be Ω corresponding to a different set of consistent truth values for the sentences in $\mathcal{S}$. The probability distribution $p : \Omega \mapsto [0,1]$ is defined on this set Ω which defines probability p_i that actual world equals ω_i. The $\sum_{\omega \in \Omega} p(\omega) = 1$ holds as the possible worlds are mutually exclusive and exhaustive.

Suppose there are K possible worlds for L sentences in $\mathcal{S}$. The K-dimensional column vector P represent probabilities of the possible worlds. Let the L-dimensional column vectors $V_1, V_2, \dots, V_K$ correspond to all consistent truth valuations of the sentences in $\mathcal{S}$ (in an arbitrary order). The matrix $\mathbb{V}$ consists from vectors $V_1, V_2, \dots, V_K$, and the elements are 0 or 1 as equals to false or true. The probability of each sentence $S_i \in \mathcal{S}$ is a component of the L-dimensional column vector $\mathcal{P}$. The relation between them is expressed as equation:

$$\mathcal{P} = \mathbb{V}P = \begin{pmatrix} V_{1,1} & V_{2,1} & \cdots & V_{K,1} \\ V_{1,1} & V_{2,2} & \cdots & V_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{1,L} & V_{2,L} & \cdots & V_{K,L} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_K \end{pmatrix}$$

Probabilistic entailment is a problem when the base set of sentences (beliefs) $\mathcal{B}$ with associated probabilities is given. New belief S with associated probability is deduced from the $\mathcal{B}$. The set $\mathcal{S} = \mathcal{B} \cup \{S\}$. First, the matrix equation is solved for P using given probabilities of $\mathcal{B}$ it is used again to compute the probability of S.

Probabilistic entailment is a probabilistic generalization of the modus ponens but only for a logically consistent set of the sentences. The inconsistent set can be handled by moving the inconsistent valuation $\mathcal{P}$ to a "nearby" valuation in the consistent region. This ad-hoc solution is not suitable for fully automatic reasoning. On other hand, as probabilistic entailment presented here is a monotonic reasoning, an additional information never increases the uncertainty.

7.4 POSSIBILISTIC LOGIC

The possibilistic logic [17], [18] handles propositional or first-order logic sentences weighted by a real number, which is a lower bound of a necessity or possibility measure as used by Zadeh [89]. Possibility of the sentence x is denoted as $\Pi(x)$, and necessity as $N(x)$. Although any bounded linear order set can be used as a range for the possibility and necessity, the interval $[0, 1]$ is used for convenience. Functions N and Π are the special kind of belief and plausibility function in the sense of Shafer [72], called consonant belief and consonant plausibility function.

Possibilistic logic is well adapted to the representation of the state of incomplete knowledge. The different levels of the incomplete knowledge can be represented as follows:

- $N(x) = 1$ means that x is certainly true.
- $\Pi(x) = 0$ means that x is certainly false.
- $N(x) = 1 - \Pi(\neg x)$ the sentence x is more certainly true as its negation is less possible
- The total ignorance can be expressed as $\Pi(x) = \Pi(\neg x) = 1$ or equally $N(x) = N(\neg x) = 0$ since x and its negation are both possible true but none of them are certain at all.

The basic axiom of possibility theory

$$\forall p, \forall q, \Pi(p \wedge q) = max(\Pi(q), \Pi(q))$$

or equally

$$\forall p, \forall q, N(p \vee q) = min(N(q), N(q))$$

$N(x) > 0$ implies $\Pi(x) > 1$ A sentence must be a totally possible before becomes a somehow certain.

Important aspect of the possibilistic logic is that it provides a resolution principle. The resolution principle can be used to infer new informations from the knowledge base.

7.5 SUBJECTIVE LOGIC

The subjective logic [31] operates on the subjective beliefs about the world, which are denoted opinion. The opinion can be interpreted as a probability measure containing secondary uncertainty. The subjective logic contains standard logical operators extended with some non-standard operators [32], [36], [35], [33] which specifically depend on belief ownership. The subjective logic can be seen as an extension of both probability calculus and binary logic. An example of the usage of the subjective logic in real situations can be found in [34], which describes the application of the subjective logic to legal reasoning.

Because the subjective logic is used later in this work, some important definitions are outlined here.

The subjective logic builds on Dempster-Shafer belief theory [72]. This theory defines a set of possible situations, called the *frame of discernment*, denoted as Θ. The powerset of Θ (denotes as 2^{Θ}) contains all possible unions of the states in Θ including Θ itself. Elementary states in the frame of discernment are called atomic states because they do not contains any substates. It is assumed that only one atomic state is true at a time. If a atomic state is assumed to be true, then all states containing this state are considered true as well.

The belief mass can be assigned to one or more states in the powerset 2^{Θ}. Belief mass on an atomic state x is interpreted as the belief that the state in question is true. Belief mass on a non-atomic state is interpreted as the belief that one of the atomic state it contains is true, but it is uncertain which of the state is true.

Definition 1 (Belief Mass Assignment)**.** Let Θ be a frame of discernment. With each substate $x \in^{\Theta}$ a number $m_{\Theta}(x)$ is associated such that:

1. $m_{\Theta}(x) \geq 0$,
2. $m_{\Theta}(\emptyset) = 0$, and
3.

$$\sum_{x \in 2^{\Theta}} m_{\Theta}(x) = 1; \tag{7.2}$$

Then, m_{Θ} is called *Belief Mass Assignment* on Θ. For each substate $x \in 2^{\Theta}$

To show an application of the belief mass assignment, assume the following situation:

> *I forgot my keys somewhere, when I was moving between my home, office, classroom and laboratory. The frame of discernment holds four states $\Theta = \{x_1, x_2, x_3, x_4\}$ represents the I forgot the keys at home, in the office, in the classroom or in the laboratory respectively. Surely, only one of these states can be true, because I can not forgot the keys in two different places. I am sure, I did not forget the keys at home, so it must be somewhere in the office, classroom or the laboratory. Therefore I assign the whole belief mass to superstate $x = \{x_2, x_3, x_4\}$. Every atomic state $x_1, x_2, x_3,$and x_4 has zero belief mass because I do not prefer any of the particular place.*

The belief in a state is interpreted as a total belief that a particular state is true. The belief in state x does not depend only on belief mass assigned to x, but also on the belief mass assigned to substates of x.

Definition 2 (Belief Function)**.** Let Θ be a frame of discernment, and let m_Θ be a belief mass assignment on Θ. Then, the *belief function* is the function $b : 2^\Theta \mapsto [0,1]$ defined by:

$$b(x) = \sum_{y \subset x} m_\Theta(y), \quad x, y \in 2^\Theta.$$

Similarly is defined the disbelief as a total belief that a state is not true. The disbelief of x corresponds to the *doubt* in Shafer's notation, but in the subjective logic, the term disbelief is used as opposite to the belief.

Definition 3 (Disbelief Function)**.** Let Θ be a frame of discernment, and let m_Θ be a belief mass assignment on Θ. Then the *disbelief function* is the function $d : 2^\Theta \mapsto [0,1]$ defined by:

$$d(x) = \sum_{y \cap x = \emptyset} m_\Theta(y), \quad x, y \in 2^\Theta.$$

Uncertainty stands for ignorance or lack of evidence for the given proposition. It is something that fills void in the absence of both belief and disbelief.

Definition 4 (Uncertainty Function)**.** Let Θ be a frame of discernment, and let m_Θ be a belief mass assignment on Θ. Then the *uncertainty function* is the function $u : 2^\Theta \mapsto [0,1]$ defined by:

$$u(x) = \sum_{\substack{y \cap x \neq \emptyset \\ y \not\subseteq x}} m_\Theta(y), \quad x, y \in 2^\Theta.$$

The following equality

$$b(x) + d(x) + u(x) = 1, \forall x \neq \emptyset \tag{7.3}$$

is direct consequent of the equation (7.2) in definition 1 and the fact that the sums of the belief, disbelief and uncertainty runs over the whole powerset 2^Θ.

For the purpose of deriving probability expectation values, the relative number of atomic states is also needed in addition to belief masses. For a particular state x, the *atomicity* of x is the number of atomic states it contains, and is denotes as $|x|$. The atomicity of Θ is equal to the total number of atomic states it contains.

Definition 5 (Relative atomicity)**.** Let Θ be a frame of discernment, and let $x, y \in 2^\Theta$. Then the relative atomicity of x to y is the function $a : 2^\Theta \mapsto [0,1]$ defined by:

$$a(x/y) = \frac{|x \cup y|}{|y|}$$

Definition 6 (Probability Expectation)**.** Let Θ be a frame of discernment, and let m_Θ be a belief mass assignment on Θ. Then the probability expectation function is the function $E : 2^\Theta \mapsto [0,1]$ defined by:

$$E(x) = \sum_{y} m_\Theta(y) a(x/y), \quad x, y \in 2^\Theta.$$

Definition 7 (Opinion)**.** Let Θ be a frame of discernment with two atomic states x and $\neg x$ and let m_Θ be a belief mass assignment and b_x, d_x, u_x, a_x are the belief, disbelief, uncertainty and atomicity functions on $x \in 2^\Theta$ respectively. Then the opinion about x, denoted $\mathcal{O}(x)$ is a tuple

$$\mathcal{O}(x) = (b_x, d_x, u_x, a_x).$$

The probability expectation

$$E(\mathcal{O}_x) = b_x + a_x u_x. \tag{7.4}$$

Because the three parameters of opinion are dependent trough the equation (7.3), they represent nothing more than the pair (belief, plausibility) of the Dempster-Shafer belief theory. Keeping all three parameters simplifies the operator expressions.

The opinions can be ordered according to probability expectation value. The additional criteria are used in case of equal probability expectation values.

Definition 8 (Ordering of Opinions)**.** Let $\mathcal{O}_x$ and $\mathcal{O}_y$ be two opinions. these can be ordered according the following priorities:

1. if $E(\mathcal{O}_x) > E(\mathcal{O}_y)$ then $\mathcal{O}_x > \mathcal{O}_y$.
2. if $E(\mathcal{O}_x) = E(\mathcal{O}_y)$ and $u(\mathcal{O}_x) < u(\mathcal{O}_y)$ then $\mathcal{O}_x > \mathcal{O}_y$.
3. if $E(\mathcal{O}_x) = E(\mathcal{O}_y)$ and $u(\mathcal{O}_x) = u(\mathcal{O}_y)$ and $a(\mathcal{O}_x) < a(\mathcal{O}_y)$ then $\mathcal{O}_x > \mathcal{O}_y$.

Beside, operators from classic logic (AND, OR, NOT), there are defined the operators specific for the subjective logic like discounting and consensus. This two operators provides the mechanism for combination of two or more opinions.

The discounting operator provides mechanism for the transfer of the knowledge from one agent to another. Assume the agent A has an opinion $\mathcal{O}_A(B)$ about correctness of an agent B. The discounting operator computes the resulting agent's A opinion about the proposition p, given the opinion of the agent B about the proposition p with given opinion about the agent B.

$$\mathcal{O}A : B(x) = \mathcal{O}_A(B) \otimes \mathcal{O}_B(x)$$

Consensus operator allows to combine two opinions about same proposition given by two different agents as:

$$\mathcal{O}_{A \diamond B}(x) = \mathcal{O}_A(x) \oplus \mathcal{O}_B(x).$$

The subjective logic has also mechanism for deductive and abductive reasoning. The abduction allows inferring the precondition a as an explanation of the observed consequence b. It is a method of reasoning which finds the hypothesis that would best explain the relevant evidence.

7.6 METHOD SELECTION

In this section, the applicability of the aforementioned logic approaches are compared with respect to the application in the robotic mapping. The propositions about the world are typically evaluated as true or false and the uncertainty arises from the errors of the measurements and sensing. The advantage of the *fuzzy logic*, the ability to express vague proposition, is not crucial for this work.

It is assumed, the knowledge about the real world can be logically inconsistent, as the knowledge is build from the uncertain data. Therefore, this assumption excludes the *probabilistic logic* from the selection, as this logic needs consistent knowledge base.

The choice can be done between the *possibilistic logic* and the *subjective logic*. As the *subjective logic* provides richer representation of uncertainty and new operators, the *subjective logic* is used in the thesis.

8

SPATIAL KNOWLEDGE REPRESENTATION

The aim of this chapter is to propose a novel suitable representation for the spatial knowledge of a large environment. First the existing approaches and their properties are discussed and confronted with the requirements resulting from the properties of the large real-world environment.

After that, the spatial representation will be proposed. At the end of this chapter the advantages of proposed representation are summarized.

8.1 COMPARISON OF THE SPATIAL REPRESENTATIONS

The previous chapters (Chapter 4 and Chapter 5) describe the current state of the art in the representation of the spatial knowledge by humans and robots. Here, the analysis of these representations is provided.

Robot and humans have very different capabilities. Human cognitive mapping relies on a highly developed visual system that recognize object and landmarks with ease. These objects and landmarks are ordered into route maps and survey maps in a hierarchical structure. This structure provides a global overview of the whole environment.

The computer vision, in comparison to human visual system, provides inaccurate and often inefficient perception for the robots. Therefore, robots frequently rely on other sensors including laser range-finders and sonars. These sensors have the abilities that human's senses lack or are weak in.This primarily incorporates e.g.: The precise range estimation and dead-reckoning. Therefore the metric maps are often used to represent the environment in one global frame of reference.

Approaches using purely metric maps are vulnerable to inaccuracies in both map-making and odometry abilities of the robot. Even by taking into account all relationship between features and the robot itself, the drift in the odometry makes the global consistency of the map difficult to maintain, in large environments.

The landmark-based approaches try to handle this problem with employing the topology of the environment. The topological relationship can better handle the incremental drift of the position estimation. However, these approaches becomes a computably intractable for a large environments when a high precision is required.

Despite many differences between human cognitive mapping and robotic mapping, there are success approaches utilizing the cognitive theories. The Spatial Semantic Hierarchy (SSH) employs a Tour cognitive theory pioneered by Kuipers, subsequently was improved by many of his colleagues. The RPLAN builds on the theory called PLAN (prototypes, locations and associative networks). The common background of all these approaches is representation of the environment as a topological map.

Property	Metric maps	Topological maps
robot position	position and orientation	states or places
map building	easy (for small spaces)	complex
	require precise robot's position	doesn't require precise position
global consistency	difficult to maintain (for large space)	easy to maintain
path planning	computational intensive	efficient planning on graph
planning result	shortest paths	may yield suboptimal paths
symbolic interfacing	poor	convenient
Annotating the map	hard	easy
place recognition	based on geometry	based on properties

Table 1: Comparison of metric and topological maps.

A comparison of the metric and topological maps is shown in Table 1. The hybrid approaches combines the metric and topological representations, and their properties are depending on the partial implementation. Generally, they are better in some aspects, and worse in others. For a particular aspect, the approaches are between the metrical and topological map.

8.2 confrontation with environment properties

This work focuses on the large structured and stable environments as defined in Section 2.2. Making the map (in any form) of the environment without structure tends to be almost useless (imagine a desert or similar open-space environments).

It is required the environment is stable. Nevertheless, it does not denote a static environment, where no changes are allowed. The stable environment can contain movable objects, and dynamic obstacles. The example of moving objects are walking people or moving cars. The changes, which are not allowed are, e.g., closing and opening the doors, closing the roads, building new houses etc.

A review of the properties of this type of the environment follows. In that aspect, the environment may be:

- Large, while it contains large number of landmarks, features, objects, obstacles etc.,

- Contains areas which are inaccessible,
- Contains moving and movable objects,
- Number of places, where a robot has to interact with the environment is limited,
- Contains areas which serve exclusively to traverse to certain destinations,
- Variation of sensing conditions.

As the environment is large, it is not easy to build and keep a single consistent metric map, while intensive or intractable computations can be required. The inaccessible areas can increase the space complexity of some metric approaches. In contrast to that, the topological representation is scalable without any substantial problems and does not represent the inaccessible areas at all.

Moreover, as it is requested to use the map also to guide the operation of the robot later, it is suitable to have a possibility to annotate the map with additional information. For example, the annotation of the place with some sort of address or property, is expected. It can be done with a topological map but not with the metric one (or it may become rather complicated).

A human or an animal, does not need to know its' precise position with respect to the environment when traveling. Therefore, the robot shall also navigate without knowing the precise position. Of course, it has to avoid obstacles during the motion. The only moment, in which it needs to be aware of the precise position with respect to the environment, appears when it has to interact with the environment (e.g. docking, manipulation with the objects etc.). Therefore, it is not necessary to keep the metric map of the paths through the environment. However, it is necessary to have a precise metric information on that places where the robot should interact with the environment.

The current topological mapping approaches rely mainly on a reactive navigation - changing of the sensing condition can easily cause a failure of the reactive system. The metric approaches rely on the localization of the robot during navigation, and therefore,they appears more robust.

Besides, the topological approaches are not ready to treat efficiently uncertain information. Again, the metric map approach can handle the uncertain sensory information better, but on the other hand it may become not efficient in handling movable objects.

The known hybrid approaches combines the metric and topological in three ways:

- Single global metrical map is augmented with the topological structures.
- Vertices of the global topological map are augmented with local metric maps.
- Global metric map is divided into parts and this parts are connected in graph-like structure.

Regarding the approaches, it seems suitable to have a hybrid approach based on a global topological map for the representation of the environment. This global topological map needs to be augmented with the metric information in the places, where interaction with environment is expected. Moreover, this approach requires reliable performance of the navigation capabilities of the robot together with improved ways of handling the uncertainty.

8.3 PROPOSED SPATIAL REPRESENTATION

Novel approach to robotic mapping proposed in this section is based on the cognitive theories of the human spatial knowledge regarding the previously defined requirements. The main structure is a topological map, a directed graph $G = (V, E)$ which consists of vertices V and edges E, likewise existing robotic implementation of topological maps. The vertices stand for places in the environment whereas the edges represents navigation paths.

This classical representation is extended by explicit association of the procedural knowledge. The procedural knowledge represent the knowledge how to do something in contrast to declarative knowledge describing what it is. It is the information which is necessary to know how to get from one place to another or how to distinguish one place from another.

The metric maps can be attached to a vertex and/or an edge, enriching these by certain additional information. Each vertex or edge can carry further additional information as area, length, curvature, relative position, and so on.

All the knowledge can be stored with the measure representing the uncertainty of the knowledge. The reasoning module, described in Chapter 9, takes uncertain knowledge stored in the map and infers the new information based on predefined set of rules. It is also capable to generate the most consistent hypothesis explaining given set of observation and in cooperation with the planner, it recommend actions to verify, reject or adopt this hypothesis. Stored information is transformable in human understandable form and can be used in communication with human.

The detailed description of the proposed representation is presented in following paragraphs. The elements of the map, the vertices and the edges, are described at first.

8.3.1 *Vertex*

The vertex $v_i \in V$ represent the point of interest or the significant place in the environment. Typically vertices are placed to the points, where robot can take a decision. They are placed where are the salient points of the control behaviors. The robots' interestingness of point can radically differ from humans' interestingness.

Each vertex holds a set of descriptors D_v. The descriptors are extracted from different sensors by different algorithms. The set of metric maps M is a subset of the descriptor set $M \subset D_v$.

The information stored in the descriptors are used to localize the robot inside the local frame of reference and to distinguish the vertices from each other. The vertex is not requested to be uniquely distinguish by its description from others. The pairwise similarity $s_A(v_i, v_j)$ of two vertices is computed by a matching algorithm A from the set of matching algorithms, which compares a specific descriptors of the involved vertices. The similarity measure is used in the process of loop-closing. The localizing algorithms provide the pose of the robot $R_v = (x, y, \phi)$ in the vertex's local frame of reference. The algorithms for extraction of the descriptor, matching and localizing are grouped into one module called *localizing jockey* according the type of the sensor and data used in the algorithms.

The vertex is internally identified by a number. One physical place should correspond to one vertex. As the environment is explored, the one physical place can be represented by multiple vertices, as is visited more than once. The loop-closing is a process of removing this inconsistency. The different places can look similar due to sensor aliasing, therefore the whole structure of the environment and the edges information are take into account.

8.3.2 *Edge*

Each edge $e_i = (v_s, v_t, D_e), e_i \in E, v_s, v_t \in V$ stores the reference to the starting and target vertex and set of descriptors D, which can be used for navigation, visualization, comparison of edges or others purposes. The edge need not to hold all the descriptors.

The set of procedural knowledge $K \subset D$, where the procedural knowledge $k \in K$ describing how to traverse from the starting place v_s to the target place v_t. The procedural knowledge is an input for one of a navigating algorithm $a \in A$ from the set of navigating algorithms. The edge can hold the procedural knowledge for more than one navigating algorithm but for each navigating algorithm can hold no more than one.

It is expected, the procedural knowledge describes a deterministic behavior i.e. robot placed in the starting place v_s and using certain procedural knowledge k navigates always to the same destination v_t. The violation of this expectation is handled as a failure which can be recovered in the navigation phase but can be fatal in the exploration phase. The probabilistic extension of traversing behavior in exploration is subjected to further research.

The procedural knowledge can be implicit in the form of the reactive navigating algorithm and store a few parameters modifying the behaviour. For example the input of the reactive navigating algorithm can be a direction of the corridor to follow. The reactive navigating algorithms are encapsulated into modules called *memory-less navigating jockeys*.

Alternatively, the procedural knowledge is an explicit description of the whole path, like record of the control command or vector of the visual targets to follow in order to navigate along the specific edge. This type of procedural knowledge is gathered along the whole path by modules called *learning jockeys*, while the robot is navigated along the path by another algorithm. For navigation, this procedural knowledge is passed to the navigating module denoted *memory-based navigating jockey*.

The edge is internally identified by a number. During the discovering the environment, one physical path can be represented by more than one edge with different internal identifier. This inconsistency is removed by the reasoning procedure described in Chapter 9.

8.3.3 *Database*

The proposed annotated graph needs to be implemented in an efficient way. Therefore, the map elements are stored in a transactional database. Every vertex and edge is identified by its number and edge also holds identifiers of source and target vertices. The descriptors of the basic elements (vertices and edges) are stored in separated tables following the snowflake scheme. Each type of the descriptor is handled by a map interface which also creates new table in the database. The database scheme is depicted in Fig. 5.

This structure of the database allows easy extension of the representation. As new descriptor of the edges or vertices is needed to store, new table is added to the database. It is possible to extend even the already existing map with the new descriptors without influence on the functionality.

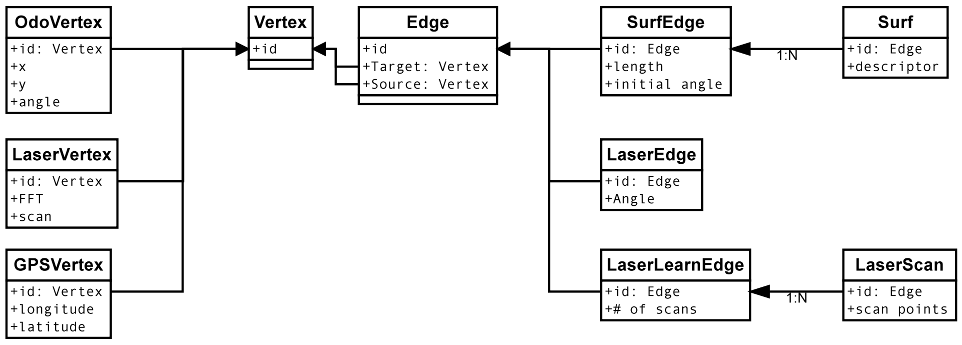


Figure 5: Database scheme of proposed representation.

8.3.4 *Map Interfaces*

A map interface is an implementation of the access module of the map stored in the database. Map interfaces form an abstract layer above the specific type of the data stored in the map. Every interface takes care about storing the specific descriptor of the vertices or edges. This information is stored in a table, that is connected to basic type using the foreign key.

The modular system of the map interfaces allows to easily extend the existing map by new type of information. If the information stored in the map is not rich enough for performing a specific task, the new algorithm and new map interface is added seamlessly. The modules are independent and do not influence each others. Since the particular stored information do not affect each other, the existing map can be extended with new type of knowledge without affecting functionality of all other algorithms.

The interface mechanism is transparent; so, a vertex can be requested for all available information as well as particular information can be specified. The information, which is not requested by the executed algorithm, is ignored and made not visible.

8.4 ADVANTAGES

The advantages of the proposed map in the form of the annotated graph are summarized in this section.

The main advantage over a current topological approaches is the rich descriptor (with the explicit procedural knowledge) of the edge in contrast to the simple link representing only the connection between the places. It allows to use a complex navigating algorithms, which needs the detailed description of the path. This description is learned during the traverse along the edge first time and then reused.

The advantages of the proposed approach over the current hybrid approaches is the possibility to have multiple metric maps related to one place. The different maps produced by different sensors or different algorithms are assigned to the same vertex, and represent the same physical location. For example, one place is represented by an occupancy grid from sonars, and a feature map generated from camera and a point map from a laser scanner, all at hand and aligned.

This type of rich representation allows to have a one map for an environment consists from parts of a different type. It is possible to have for example a map a university campus, where the areal of the campus with insides of the building is stored in one representation. It is not necessary to split it to different maps for each building and then switching between them. The robot is able to navigate fluently from indoor to outdoor and back.

Also it is possible to have a one map for different types of robots. The robot equipped with a camera shares the map with a robot equipped with a laser range-finder. The same definition of the vertices and edges, representing the same physical locations and paths, processed making-use of different algorithms for i.e. navigation and localization. This approach brings up a novel functionality, not possible for the classical metric representations.

The representation of the uncertainty increase the robustness and reliability of the representation. As the representation is augmented with the reasoning procedure, the missing information is deduce from the current uncertain information. The resulting uncertainty indicates if the current information is convenient enough to make the required conclusion, or, it is necessary to gain more information.

The reasoning procedure is described in the following chapter.

9 REASONING

The map can be understood as a knowledge base, the set of proposition about the environment, and robot-environment interaction. With the proper formal tool, it is possible to infer new facts from this knowledge base and set of rules. The facts in the map are uncertain and possibly stochastic in contrast to classical logic values (true or false). The subjective logic (see Section 7.5) is proposed as a suitable formal logic system to handle uncertain and stochastic facts proposed representation. It is suitable also for conditional reasoning which is useful in hypothesis generation and verification.

Suppose, there is a propositional language L, supplemented by the tautology $\top$ and contradiction $\bot$. The set of propositions will be finite, as the world is assumed to be limited and closed. Let Ω_L denotes the set of worlds that corresponds the interpretations of L.

For every pair of worlds in Ω_L:

$$\forall \Omega_i \Omega_j \exists \phi; \Omega_i \models \phi \wedge \Omega_j \models \neg\phi,$$

there exists a proposition in the language L that is true in one world and false in the other. This condition excludes logically equivalent worlds.

Let consider an actual world, denoted as ω_0, i.e., a world that corresponds to the actual state of environment. The robot does not know which world in a set of possible worlds is the actual world. This ignorance results from the robot's ignorance about the truth status of some propositions in the actual world. If the robot knows the truth status of every propositions of interest in ω_0, then it would know, which world is ω_0.

The robot's actual knowledge about the actual world can be encoded in set of propositions K . The robot knows the truth values of the propositions in K. It is assumed, the K is consistent and deductively closed. Then, the set Ω_K can be constructed as:

$$\Omega_K = \{\omega : \omega \in \Omega_L, \omega \models \phi, \forall \phi \in K\}.$$

The set Ω_K is set of worlds considered as possible for the robot being in the actual situation. The actual world is one of the possible worlds $\omega_0 \in \Omega_K$ if the set K is reflexive.

The situation is relatively easy, when the robot has an exact information about the environment. Then the set K contains the classical logic proposition with truth value from $\{0, 1\}$. But in reality, a robot senses the environment with sensors influenced by different types of errors. Therefore the information gathered from the environment is uncertain and cannot be expressed by classical logic propositions.

Therefore, the probabilistic extension of the subjective logic is applied at this point. The true value of the proposition p is expressed by an opinion $\mathcal{O}(p)$. The opinion $\mathcal{O}$ has four components: believe b, disbelieve d, uncertainty u and atomicity a. First three parameters define position in opinion space and are dependent according the equation (7.3) and fourth component - atomicity - is added to simplify computing of the probability expectation. Herein, the uncertainty stands for ignorance or lack of evidence for the given proposition. It fills a void in absence of both the belief and disbelief.

It is necessary to redefine the possible worlds in case of the subjective logic. Let $\mu_K(\omega)$ is compatibility of the world ω with the robot's actual knowledge

$$\mu_K : \Omega_L \mapsto [0, 1].$$

Nevertheless, the use of interval $[0, 1]$ remains purely as a convenience, and any other arbitrary interval can be used instead, but $[0, 1]$ is perfectly acceptable in this case. The understanding is that larger $\mu_K(\omega)$ expresses, that ω is more compatible with K. The extreme value of $\mu_K(\omega) = 0$ means that the world is not possible at all and $\mu_K(\omega) = 1$ means the world ω is fully possible according to the robot's actual knowledge.

Now the possible worlds cannot be expressed as the classical set because the membership function maps not to a set $\{0, 1\}$ but to an interval $[0, 1]$. If the classical set is needed, it is possible to define :

$$\Omega_K = \{\omega : \mu_K(\omega) \geq \vartheta\},$$

where ϑ is a selected threshold.

During the exploration, the robot gains the information about the actual world, and adds the propositions to K, and reduce the size of Ω_K for a given threshold. In the case when $|\Omega_K| = 1$, the robot knows the actual world and the exploration ends. There exists situations, when the robot is not able to distinguish the possible worlds even when the whole environment is explored and all available information is gathered.

In proposed approach, the whole list of possible worlds is not maintained. The actual model of the environment is one world chosen from Ω_K only.

The world nearest to the world build from direct observation is chosen. This is opposite approach to Occam's Razor principle, where the smallest number of entities is preferred. Here the larger number of entities is preferred as the possible mistake in the observation has smaller impact on the usability of the map.

9.1 observation fusion

The result of the localizing algorithm can be a value in range $[0,1]$ or $[0,\infty]$ for the proposition *"the place P_x is same as the place P_y"*. It is necessary to convert this value into the form of an opinion. The parameters for the conversion are acquired using machine learning techniques from training sets of annotated vertices.

The result can be taken as a probability expectation $E(x)$ in the case of the range $[0,1]$. The optimal discriminating threshold ϑ is determined from the receiver operating characteristic (ROC) curve (see Sec. 12.1) as a point with the maximal distance from a random classifier. As there is no evidence to prefer to belief or disbelief, the uncertainty will be maximized during the conversion procedure. An uncertainty can be derived from the equation

$$u_x = \frac{E_x - b_x}{a_x}, \tag{9.1}$$

where atomicity a_x is set according to the ROC curve. From the maximization of the equation (9.1), the opinion is derived as

$$\mathcal{O}(x) = \begin{cases} (b_x = 1 - u_x, d_x = 0, u_x = \frac{E(x)-1}{a_x-1}, a_x), & E(x) > a_x \\ (b_x = 0, d_x = 1 - u_x, u_x = \frac{E(x)}{a_x}, a_x), & E(x) \le a_x. \end{cases} \tag{9.2}$$

The results from the range $[0,\infty]$ need to be converted to the interval $[0,1]$ at first. In reality, the results of the localization are limited. This limit becomes from the properties of the input data. As the maximum sensing range of the sensor is limited and a type of the sensory data is given, the variability of descriptors is constrained. This makes possible to compute the supremum from the results and use it for the conversion. Alternatively, the guess of upper bound can be determined from the training set as a maximum value.

The second property of the localization results from $[0,\infty]$ is that 0 stands for the best match. Therefore, the values higher than the threshold ζ can be threated as mismatching. Even though the guess is not the supremum, the error appears not significant.

$$E(x) = \begin{cases} 1 - \frac{x}{\zeta}, & \frac{x}{\zeta} \le 1 \\ 0, & \frac{x}{\zeta} > 1 \end{cases} \tag{9.3}$$

As soon as the value of $E(x)$ is computed, the opinion is determined according to Equation (9.2).

The consensus operator $\oplus$ is used a for fusion of the localizing algorithms' results.

$$\mathcal{O}_A(x) \oplus \mathcal{O}_B(x)) = \begin{cases} \mathcal{O}_A(x), & \kappa = 0 \\ (\frac{b_A u_B + b_B u_A}{\kappa}, \frac{d_A u_B + d_B u_A}{\kappa}, \frac{u_A u_B}{\kappa}), & \text{otherwise}, \end{cases} \tag{9.4}$$

where κ is defined as $\kappa = u_A + u_B - u_A u_B$.

9.2 DEDUCTION

The binomial conditional deduction is defined in probability calculus as

$$p(y\|x) = p(x)p(y|x) + p(\bar{x})p(y|\bar{x}),$$

where $p(y\|x)$ is the deduced probability of the consequent y, $p(x)$ is the probability of the antecedent x, and the probability of its complement $p(\bar{x}) = 1 - p(x)$. The $p(y|x)$ and $p(y|\bar{x})$ are the conditional probability of the y for the cases the given x is true and false, respectively. Jøsang in [33] defines the conditional deduction in conformity with the probability calculus in the subjective logic. It means, both the conditional probabilities are necessary to know to deduce the consequent.

Modus Ponens says if the proposition P and proposition $P \rightarrow Q$ are true, deduction stands that Q is also true.

$$MP: \quad P, P \rightarrow Q \vdash Q.$$

Nevertheless, Modus Ponens does not resolve anything about the case if the antecedent or conditional are only partially true. Therefore, let define the Modus Ponens for the subjective logic as follows:

Let the $\mathcal{O}(P)$ be the opinion about antecedent and $\mathcal{O}(P \rightarrow Q)$ be the opinion about the rule. The question is how to determine the opinion $\mathcal{O}(Q)$. First assume $\mathcal{O}(P \rightarrow Q)$ equals to the tautology, i.e., $\mathcal{O}(\top) = (1, 0, 0, a)$. Then as long as there is a belief into truthfulness of P, there will be a belief in Q. It can be expressed as discounting the opinion about the rule by the opinion about the antecedent:

$$\mathcal{O}(Q) = \mathcal{O}(P) \otimes \mathcal{O}(P \rightarrow Q). \tag{9.5}$$

The operator $\otimes$ discounting is defined as

$$(\mathcal{O}_A \otimes \mathcal{O}_B) = (b_A b_B, b_A d_B, d_A + u_A + b_A u_B, a_B).$$

It can be easily seen from this equations, the zero belief in the antecedent produces the vacuous (totally uncertain) belief into the consequent.

9.3 REASONING IN LOOP-CLOSING

The loop closing is a process of finding the correct correspondences in a experienced set of observations gathered during the exploration. Let the map $M_E = (V_E, E_E)$ be a graph with a set of discovered vertices V_E and edges E_E. This experienced graph is in a form of chain as it represents a single run through the environment (without any loops). Alternatively, the graph may be given in a form of tree, if the exploration algorithm preserves the information about backtracking during the search.

The map M_E is converted into the list of the opinions K_E about the environment. This propositions are in a form of similarity of the vertices and edges in the task of loop-closing. The similarity of the edges and vertices is computed by the localizing algorithms. Therefore, for each pair of vertices or edges there can exist more than one opinion about their similarity.

To keep the computation complexity low, the similarity is kept in the matrix $S_a, \quad a \in A$ for each algorithm a from the set of localizing algorithms A. The elements of the matrix $S_a^V(k,l) = s_a(v_k, v_l)$, are the pairwise similarity s_a of the vertices v_k and v_l as given by algorithm a. A matrix S_a^E defined similarly for the edges $S_a^E(k,l) = s_a(e_k, e_l)$.

These similarity matrices are fused into two matrices, one global similarity matrix for opinions about vertices $\mathbb{O}^V$ and one for edges $\mathbb{O}^E$ where elements of a matrix $\mathbb{O}$ are

$$\mathbb{O}(k,l) = S_1(k,l) \oplus S_2(k,l) \oplus \ldots \oplus S_J(k,l).$$

The first step is to convert the similarity matrix S_a to opinions matrix $\mathbb{O}_a$. This is done as described in section 9.1. Then the fusion is done by the consensus operator, see Eq. (9.4).

The knowledge about the properties of the environment is converted into the form of the logical proposition with the opinions about the validity for a given environment. The following propositions were used in the rest of the paper:

$$s(v_u, v_v) \wedge s(e_k = (v_u, v_w), e_l = (v_v, v_x)) \rightarrow s(v_w, v_x) \tag{9.6}$$
$$s(v_w, v_x) \wedge s(e_k = (v_u, v_w), e_l = (v_v, v_x)) \rightarrow s(v_u, v_v) \tag{9.7}$$
$$s(v_u, v_v) \wedge \neg s(e_k = (v_u, v_w), e_l = (v_v, v_x)) \rightarrow \neg s(v_w, v_x), \tag{9.8}$$

where $s(\cdot,\cdot)$ denotes the similarity.

The implication (9.6) expresses the premise that the edges from a particular vertex are distinguishable. This premise rises from the properties of the navigating algorithms and edge description. If the algorithm is not able to distinguish the outgoing edges, these are described as one.

The implication (9.7) expresses the premise that the edges coming to the particular vertex are distinguishable. This premise comes from the assumption that the incoming edges are dual to the outgoing edges. This assumption is not valid in general for an arbitrary navigating algorithm, but is valid for the reactive navigation described later in this work.

The last implication (9.8) expresses the assumption that the environment is not a multi-graph, i.e. there are no parallel edges. This assumption is weaker than the previous two, but it remains valid for all the environments used for experiments later.

Assume, there are two matrices $\mathbb{O}^V$ and $\mathbb{O}^E$ holding the opinions about the experiences, and a set of rules in the form of implications.

All the rules have as the consequent the similarity of two vertices. Therefore, for each rule, the matrix of opinions will be computed according to equation (9.5).

Then, the algorithm fuses all matrices which are results from the previous step into one.

As the edge similarity in the antecedent of all rules exists, the computation of the resulting matrix follows Algorithm 1.

Algorithm 1: Entailment computation.

```
Input: Opinion matrices $\mathbb{O}^V, \mathbb{O}^E$, set of rules $R$
Output: $\mathbb{O}_c^V$
initialize $\mathbb{O}_c^V$ with uncertain opinion $(0, 0, 1, 0.5)$;
for $r \in R$ do
    initialize $\mathbb{O}_r^V$ with uncertain opinion $(0, 0, 1, 0.5)$;
    for $\mathcal{O}(s(e_i, e_a)) \in \mathbb{O}^E$ do
        $u, x \leftarrow e_i = (v_u, v_x)$;
        $v, y \leftarrow e_j = (v_v, v_y)$;
        $\mathbb{O}^{\mathbb{V}}(x, y) \leftarrow \mathbb{O}^{\mathbb{V}}(x, y) \oplus ((\mathbb{O}^{\mathbb{V}}(u, v) \wedge \mathcal{O}(s(e_i, e_j))) \otimes \mathcal{O}(r))$ ;
    $\mathbb{O}_c^V \leftarrow \mathbb{O}_c^V \oplus \mathbb{O}_r^V$ ;
$\mathbb{O}_c^V \leftarrow \mathbb{O}_c^V \oplus \mathbb{O}^V$ ;
```

The algorithm deduced the knowledge K_C about the loop-closing $K_E, R \vdash K_C$ from the experienced knowledge K_E and the set of rules R . The algorithm is passing through the whole edge similarity opinion matrix and for each edge and rule it computes the consequent by the Modus Ponens according equation (9.5). The indices u, v, x, y of source and target vertices are determined for each pair of the edges. The opinions about the vertices pairwise similarity is taken from the vertex similarity opinion matrix $\mathbb{O}^V$. Then, the consequent is stored into the result matrix. If there is already an opinion in the matrix, the resulting element is computed using the consensus operator.

The complexity of this algorithm is $O(|E|^2)$, where $|E|$ is the number of edges.

The map with closed loops is computed from the result of the deduction and the original map.

Algorithm 2: Loop closing algorithm.

```
Input: Map $M_E = (V, E)$, $\mathbb{O}_c^V, \mathbb{O}^E$, threshold $\vartheta$
Output: Map with closed loops $M_c = (V_c, E_c)$
for $\mathcal{O}(s(u, v)) \in \mathbb{O}_c^V$ do
    copy $M_E$ into $M_c$;
    if belief of $\mathcal{O}(s(u, v))(b) > \vartheta$ then
        for $e_j = (v, y), x \in V \cup unknown, e_j \in E$ do
            for $e_i = (u, x), x \in V \cup unknow, e_i \in E$ do
                if $\mathbb{O}^E(i, j)(b) > \vartheta \wedge$ ($x$ is unknown ) $\wedge y \in V$ then
                    set $e_i = (u, y)$ in $E_c$;
            remove $e_j$ from $E_c$;
        for $e_i = (x, v), x \in V, e_i \in E$ do
            set $e_i = (x, u)$ in $E_c$;
        remove $v$ from $V_c$
```

If there exists opinion about the similarity of the pair of vertices u and V with the belief value greater than the threshold ϑ, the vertex v is removed from the map with the closed loops. Consequently all the edges going from the removed vertex v are also removed. If there exists edge $e_i = (u, \text{unknown})$, going from the remaining vertex u, which has an unknown target and there exists a similar edge $e_j = (v, x)$, going from the removed vertex v to the known target x, the edge e_i is modified to $e_i = (u, x)$.

10

FRAMEWORK INTEGRATION

The proposed spatial knowledge representation and the reasoning procedure assume the existence of localizing and navigating algorithms. This chapter describes the proposed and implemented algorithms, which where used in the experiments later on. These algorithms work with different sensors and robotic platforms.

All of the algorithms are used with a common environment representation. The unified approach to handle and to establish communication amongst these algorithms is necessary. Therefore, all the new methods for localization, navigation along with reasoning and map representation are incorporated into one framework called Large Maps Framework (LaMa).

The LaMa Framework is build as a modular system and therefore allows to easily incorporate new methods into it. Mechanism for modules coordination and cooperation is incorporated into the LaMa framework and therefore the individual modules need no specific mechanism for these interactions.

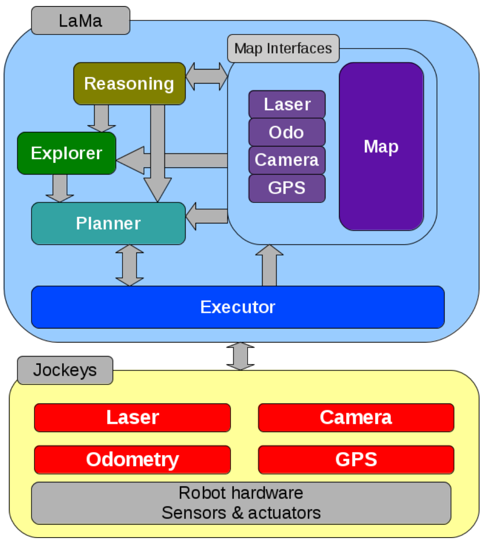


Figure 6: LaMa Architecture

The framework consists of several backbone parts: Map, Executor and Jockeys. The Map holds gathered information about the environment and provides interfaces for writing, modifying and reading data. The knowledge about the environment is stored according the spatial knowledge representation proposed in Chapter 8.

The Executor module provides abstraction for higher level planning, hides the implementation details and enables handling of all types of map elements in a unified way. Moreover, this concept enables coordination and cooperation of navigating, learning and localizing Jockey modules.

10.1 EXECUTOR

This thesis introduces the procedural knowledge as a crucial property of the proposed spatial knowledge representation. The procedural knowledge can hardly be stored in the map directly, in form of algorithms. Therefore the procedural knowledge is split into two parts, the algorithm and the data. The data are stored directly in the map, more specifically in the transactional database. These data are accessible using the map interfaces. The specification of the used algorithm is stored along with these data.

The algorithms are stored separately as libraries or executable programs. The implementation strongly depends on the used robot hardware and sensors, therefore it is not reasonable to access directly these libraries or programs, as there exists a particular implementation for each sensor or robot hardware. The access to these must be unified to achieve universal representation, otherwise the map representation differs for every possible combination of sensors and robots.

In the LaMa Framework, the Executor provides a unified access to different navigating , learning and localizing algorithms working directly with the hardware of the robot. These are are encapsulated as Jockeys. The higher levels of LaMa Framework accesses the robot hardware through the Executor and Jockeys only.

The Executor is responsible for executing the plan made by a Planner or an Explorer. The Executor calls a specific Jockey according to the requested action. If there are more Jockeys admissible for the current action at once, the one with the best expected performance is chosen. The Executor supervises the behavior of the executed Jockey and if the Jockey fails, tries substitute the requested behavior by executing another Jockey with similar function.

During the exploration phase, the Executor runs Jockeys following the best possible information gain about the environment. It tries to run all available Learning Jockeys during an edge traversal and runs Localizing Jockeys in the target vertex after the traversing ends.

The Executor coordinates the concurrently running Jockeys. Only one Jockey can control the robot actuator at once. It may occur, that one or more Jockeys needs to stop the robot and/or access actuators of the robot in a specific situation. Then the Executor checks if it possible to interrupt the Jockey currently controlling the robot and provides access to hardware for the requesting Jockey. After the reason for an interrupt vanishes, control is returned to navigation Jockey.

The Executor also performs the localization in the newly visited vertex. It tries to decide if robot enters new vertex or just revisit the already mapped one. It takes in a count the time consumption of each Jockey and optimizes gathered information against the required time. The Executor calls Jockeys in order of their efficiency.

10.2 JOCKEYS

The algorithmic part of the procedural knowledge is denoted as Jockey. Numbers of different navigating and localizing algorithms were designed and implemented during the development of herein presented framework. Even with the presence of the Executor is reasonable to have the unified access to all the algorithms. Therefore, these algorithms, representing particular functionalities, were designed as an independent software modules with defined interface.

These modules are called *Jockeys* as, in certain sense, are "riding" the robot. It means that these modules have direct access to the robot hardware and are able (and allowed) to control the robot actuators, to read the robot sensors and to process raw data gathered from the robot's sensors.

Diverse algorithms represent the different types of the procedural knowledge. Some of these are dedicated to the robot navigation, another ones are able to gather the data necessary for navigation later on, some provides both the functionalities. There are algorithms for extraction of the descriptors from data, another are matching the descriptors or localize the robot inside the place.

Generally, the functionalities of the algorithms can be grouped into three basic types: navigation, localization and learning. According these functionalities, three different core types of modules are defined :

- Navigating Jockey,
- Learning Jockey and
- Localizing Jockey.

Whereas for each type of Jockey is the specific interface in form of communication protocol defined.

Jockeys are connected to the Executor through socket and XML based protocol. Socket communication is chosen, as it allows to split execution part (the Jockeys) from deliberative part(Map and Reasoning) in logical as well as spatial manner. The spatial detachment allows to run the Jockeys on the different computation resources than the deliberative part. For example, the Localizing Jockey can run on an intelligent IP camera and be connected to the rest of the system only by Ethernet.

The XML protocol is chosen due to the ease to extend and customize, and ensure connectivity of different types of control programs not depending on the background operating system. It is necessary to have an extensibility property, as the protocol is extended for each individual Jockey, according the functions and needs of each ones. The extended part is passed into the related Map Interface where is further processed. Each Jockey is related to one or more Map Interfaces (see sec. 8.3.4) storing the data into the map.

10.2.1 *Navigating Jockey*

The navigating Jockey takes care of traversing edges with respect to the procedural knowledge of a specific type. It guides the robot along the given edge and determines the end of this edge when entering into a next vertex. If it is possible to interrupt the execution of the navigating algorithm, the Navigating Jockey is call interruptible. This property allows to restart it later from a slightly different position while the algorithm still keeps the capability to navigate to the original target vertex.

The navigating Jockey has two different subtypes: Memory-less and Memory-based.

Memory-less Navigating Jockey

A Memory-less Jockey uses a reactive navigating strategy. The procedural knowledge represented by reactive navigating strategy has minimal data part. Only the information needed to distinguish outgoing edges from each other is stored in the map.

The reactive algorithm computes the control commands directly from the sensory input and has no persistent internal state. Therefore, the Memory-less Navigating Jockeys are often interruptible.

Main advantage of reactive navigating is the ability to traverse an unknown edges which was not traversed before. It allows to discover new locations, which are the target vertices of these edges. Therefore the Memory-less Jockeys are used mainly during the exploration of the environment.

Memory-based Navigating Jockey

A Memory-based Jockey needs the data part of the procedural knowledge about the edge for the proper operation. The data for the Memory-based Jockey are typically more complex than for the Memory-less one. These data (edge descriptor) are stored in the map and must be inserted into the map by the coupled Learning Jockey in advance. Therefore, the Memory-based Navigating Jockey can guide the robot along previously visited edges only.

The Memory-based navigation can be more precise and repeatable than the reactive one. This Jockey can determine the failure of the navigation and report it to the Executor module. In addition, the Jockey can estimate the possibility of the failure according the given edge descriptor in advance.

Secondary, the Jockey can provide the measurement of similarity of the appropriate edges. This property is used in the Reasoning procedure described in Chapter 9.

10.2.2 *Learning Jockey*

The Learning Jockey is tightly coupled with particular Memory-based Navigating Jockey which utilizes the procedural knowledge of the same type. This Jockey produces the procedural knowledge by gathering the information while a robot is driven by another Navigating Jockey (typically Memory-less), as the Learning Jockey is not supposed to control the robot actuator while traversing the edge.

The gathered procedural knowledge is divided into algorithmic and data parts. The data parts is stored in the map as the edge descriptor. This edge descriptor can be utilized by Memory-based navigating Jockey later on.

It is possible, that the Learning Jockey needs to control the robot actuator occasionally, in order to increase the performance, get more data from the sensor, get more time for data processing or fix the errors. In this case, the Learning Jockey can request the Executor to interrupt the running Navigating Jockey. If the Navigating Jockey is interruptible, the Executor suspends the Navigating Jockey, and the control over the robot actuators grants to the Learning Jockey. The original Navigating Jockey is reactivated, as soon as the Learning Jockey achieves the goal and notifies the Executor, that the reason for interruption disappears.

The Learning Jockey extends the existing edge with the new procedural knowledge how to traverse the edge. This property allows to build rich but unified spatial representation for different types of robots equipped with wide range of sensors. Such a map is suitable for a heterogeneous robotic team, as it allows easy sharing of the spatial knowledge among team members.

10.2.3 *Localizing Jockey*

The Localizing Jockey is related to the specific type vertex descriptor and specific sensors. This Jockey processes the sensor data into the form of vertex descriptor. Resulting descriptor is stored in the map using the specific Map Interface. Different sensors require different Localizing Jockeys for handling specific hardware and processing data, nevertheless these Jockeys can utilize same Map Interface and therefore share the stored information. On the other hand, Localizing Jockeys with different algorithms can share the same sensors.

When the robot arrives into the vertex, the Jockey gathers information about the surrounding environment, computes and stores specific descriptors into the map and also discovers outgoing edges from the current vertex (if possible). Different Localizing Jockeys can produce different vertex descriptions and also discover different outgoing edges.

The Localizing Jockey is able to distinguish vertices of the specific type from each other and compare the actual vertex with other vertices stored already in the map. As results it gives the probabilities of being in some of the previously visited vertices. This behavior is used for global localization and loop-closing.

In addition, Localizing Jockey can estimate robot relative position in the vertex local frame of reference, if the stored descriptor has the form of the metric map. This ability is crucial, if the robot has to interact with the environment.

Further follows the detailed description of particular algorithms. They are grouped according the types defined above.

10.3 NAVIGATING MEMORY-LESS JOCKEYS

10.3.1 *Visual Reactive Navigation*

This memory-less navigating jockey called GeNav (from Gerstner navigation) is designed to use a visual system to navigate in the environment and detect certain locations. The jockey tries to navigate along the path in the environment and is expected, the path has a distinctive color. The navigation is done reactively, solely on the information from the camera. The places of interests are the crossing of the paths, where the robot can decide to follow another path. A magnetic compass as an absolute sensor of robot heading is used for the outgoing edges (paths) distinction.

The path and crossings are recognized in the color picture taken by calibrated camera aimed at a surface in front of the robot. The viewed area spans from 1 to 5 m in the direction of the robot movement and approximately 3 m to both sides. It is supposed, that color of the path is given by other method or sensor, or is known in advance. Path color can be also entered by an operator.

A hue-saturation-value (HSV) color space is utilized for path color specification, because a Cartesian product of HSV values color description offers greater invariance to the changing illumination than similar description in Red-Green-Blue color space. To prevent costly calculations of HSV description of every evaluated pixel during recognition procedures, a RGB lookup color table is first computed from the HSV color specification.

The jockey implements two behaviors: Path traversing and Crossing detection.

As the navigation starts, a command with azimuth of traversed path is received and the robot turns to the given azimuth using the magnetic compass. Afterwards, the robot moves forwards a certain finite distance and path traversing routines are activated. Leaving an actual crossing, the crossing detection routine is temporarily inhibited to prevent recognition of previously visited crossing.

In the path traversing mode, the jockey attempts to keep the robot in the middle of recognized path while driving it forwards. This procedure estimates the width of the recognized path and executes crossing recognition routines when this width changes rapidly. Once a crossing is recognized and approached, the place description is sent to executor module.

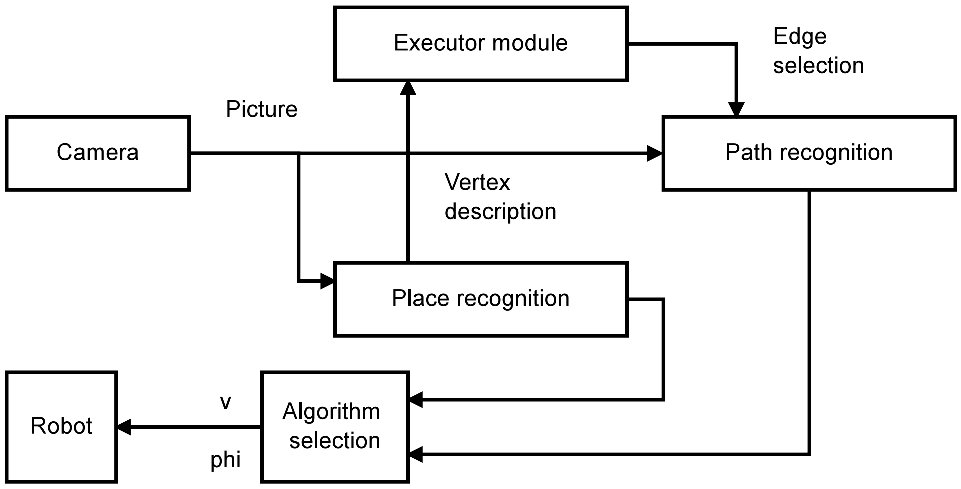


Figure 7: Block scheme of visual memory-less navigating jockey (GeNav).

Path traversing

The path traversing behaviour follows the central line of the recognized path. In the first step of the algorithm, last row of acquired image is searched for pixels of path color and a mean value of their horizontal coordinate is computed. After that, an identification of path boundaries on this row is performed, i.e. a pixel sequence of other than path color is searched in both directions from the mean position of pixels of path color. The path centre and the width are then calculated out of the detected boundaries. If the width is greater than a predefined threshold, the algorithm proceeds to a higher row with search start position given by the current path centre coordinate. The search algorithm is completed whenever the path width drops below this threshold.

Robot forward velocity v and turn speed φ vector is computed by:

$$\begin{pmatrix} v \\ \varphi \end{pmatrix} = \begin{pmatrix} \alpha(h-r) - \beta\left|\sum_{i=r}^{h}(\frac{w}{2} - m_i)\right| \\ \beta\sum_{i=r}^{h}(\frac{w}{2} - m_i) \end{pmatrix}, \tag{10.1}$$

where h and w are the image height and width respectively, m_i is detected path centre of the i^{th} row, r is the last processed row number and α, β are constants.

As noise is usually present in the image and may influence smoothness of the computed driving parameters, the centre and the width values are smoothed by second order linear adaptive filters.

Crossing recognition

The crossing recognition detects the crossing of the paths in the picture from the camera, finds the outgoing edges and computes its angles using the magnetic compass. Unlike path recognition, employability and precision of this routine requires the camera to be calibrated. If the detected path width differs from

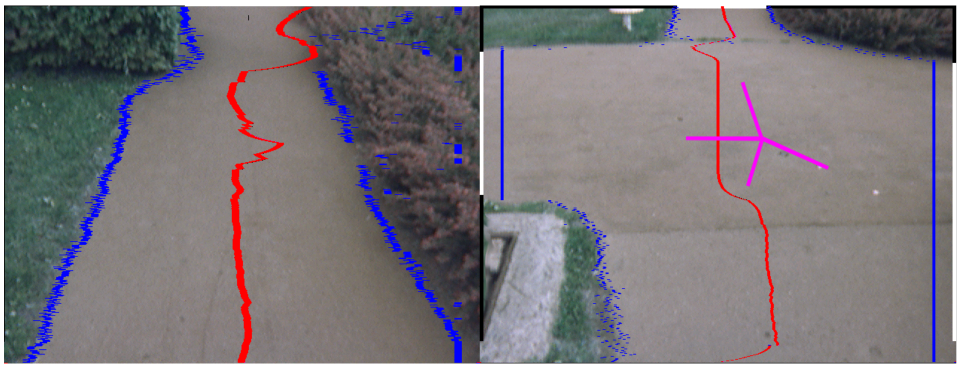


Figure 8: Detected path and crossing

the predicted one consecutively, crossing detection routines are activated. This performs search for compact regions of path color on the periphery of the sensed image. Regions not connected by a path to the center of detected crossing are removed. Image coordinates of the remaining region centers are converted to the robot coordinate system (crossing is considered to be planar and collinear with robot undercarriage). The crossing description is then calculated out of these regions, detected crossing center and compass measurements. This description consists of a set of path bearings leading out of the crossing.

Finally, the image of crossing center is searched for a large blob - a sufficiently large compact region of predefined color. If the blob is found, the crossing is designated as a "base vertex". Then the robot moves to the computed crossing center coordinates controlled in the open-loop. The description is delivered to the executor module.

10.3.2 *Laser-based Navigation Jockey*

The task of the LaNav jockey is to navigate a robot reactively through an environment using purely data from a laser range finder. It is assumed, that an omni-directional laser range-finder is applied, or two 180 degree scanners are employed.

The jockey operates in three main modes:

LEAVE MODE, where the robot leaves the current place,

NAVIGATION MODE, where the jockey navigates robot through the environment based on laser data and

GOAL MODE, where the robot has to reach (possibly) near goal.

The navigation mode is used for exploring the environment. During the navigation, robot tries to find a significant places, e.g. crossings. If such a place is detected, the Goal-mode is employed to navigate the robot towards the detected place.

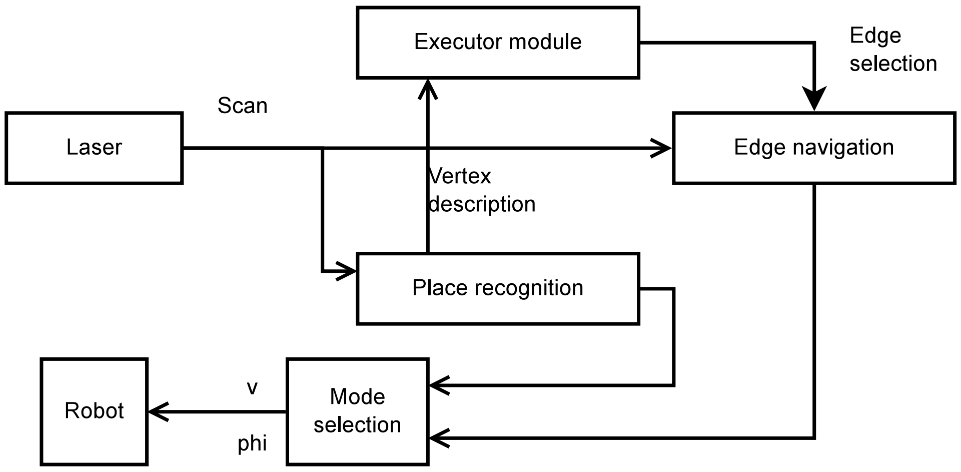


Figure 9: Block scheme of laser memory-less navigating jockey

Leave Mode

The *leave mode* takes the edge description from the Executor and starts the navigation along this edge. The *navigation mode* cannot be used for this task, while in *navigation mode* significant places are continuously detected and robots attempts to move towards their centers. If the navigation mode starts at a center of a significant place, it will possibly detect it and report to the Executor module. In such a case, robot may never escape from that place.

The edge descriptor is a relative direction, in which the robot has to leave the current place. At first, the robots turns to this direction and it computes maximum distance, which can be reached in this direction without collision. In the second phase, the robot moves along a straight line to this distance. Operating in the *leave mode*, significant places are not detected and range scan data are used only to detect new unexpected obstacles. If the robot moves to the distance at least radius of the start place, LaNav switches into the navigation mode.

If not successful, the navigation mode may also be started directly, but the start place has to be ignored, if it is detected repeatably. To ignore the start place, its position and size are stored and tracked using odometry, as the robot moves.

Navigation Mode

In *navigation mode* robot uses a reactive navigation algorithm which is based on actual scan data only. This aims to navigate the robot in collision free manner and move robot on such a trajectory, that maximizes possibility, of interesting place detection. Such a trajectory should be close to medial axis of the environment. The navigation algorithm computes control input for the robot based on distances to the nearest obstacles in the environment.

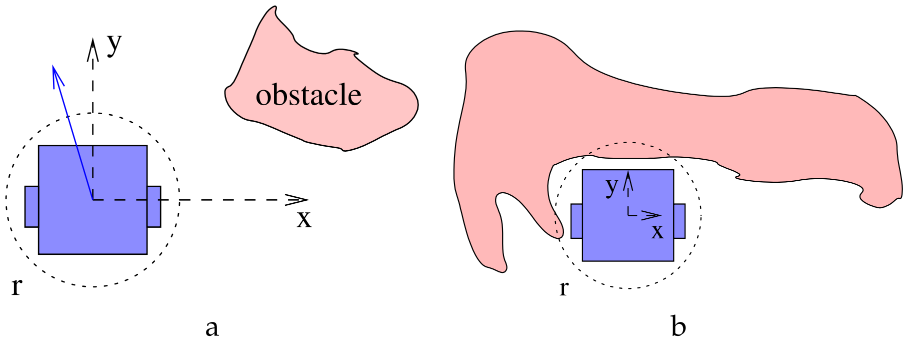


Figure 10: A coordination system of a robot, r denotes the safe radius. In this case, the robot will move in direction of blue arrow, because the nearest obstacle is on the right (a). Example of a situation near a corner, where the robot has to turn back (b).

The forward speed v and turning speed ϕ are computed as:

$$v = \begin{cases} 0 & \text{if nearest obstacle is too close} \\ v_{const} & \text{otherwise} \end{cases}$$
$$\phi = -\phi_{const} s_x / n,$$

where v_{const} and ϕ_{const} are numerical constant and $s_x = \sum_{i=0}^{n} x_i$, where x_i is x-coordinate of a obstacle detected by i-th laser beam in a robot local frame of reference as depicted in Figure 10. This controller thus turns the robot to that direction where the obstacles are more distant. The robot thus moves along a trajectory close to medial axis. To prevent collisions with closest obstacles in front of the robot, the forward velocity is set to zero (complete stop), if the distance to the closest obstacle in front of the robot is less than a preselected safety diameter of the robot. In such a case robot just may turn towards a free area.

During the navigation mode the scan data are processed in order to detect places and its centers. If a place is found and the place center is in front of the robot, the robot changes to *goal mode*.

Goal Mode

In this mode, the center of the detected place is denoted as goal $G = (x_G, y_G)$, and robots tries to reach it. However, during moving towards a place center, the position of the place center is continuously updated. If the place is not longer detected as a valid place, due to recent sensory readings, the *navigation mode* is entered. Otherwise, robot finish the *goal mode* in the nearest crossing center and reports this to the Executor module.

During the *goal mode* the forward speed v and turning speed ϕ are computed as:

$$v = \max(1, c_v x_G) \tag{10.2}$$
$$\phi = \max(1, c_\phi y_G), \tag{10.3}$$

here the goal coordinates lay in a robot local frame of reference and c_v, c_ϕ are treated as multiplicative constants to angular and radial speeds, depending on robot dynamics.

Place detection

The detection of interesting places (crossings) is based on search for frontiers in laser range scan. The scan $S = \{p_1, \dots, p_n\}$ consists of n points, where each point $p_i = (x_i, y_i)$ is a detected point in a robot local frame of reference. The points form a polygon P, where the robot's position is at location $(0, 0)$. A frontier is defined by such a two consecutive points $p_j, p_{j+1} \in P$, whose distance is larger than a predefined parameter d_t. In our case the parameter d_t is defined as the radius of the robot, thus the frontiers are defined as segments in the polygon, though that the robot can escape the polygon. For the navigation purpose, the direction α_i of each frontier is stored. This angle is used as an edge descriptor.

The frontiers can be detected in $O(n)$, where n is number of points in the polygon, the algorithm of frontier detection is shown in Alg. 3. The example of a detected frontiers is depicted in Fig. 11b.

Algorithm 3: Frontiers detection.

```
Input: $P = \{p_1, \dots, p_n\}$: laser data, $n$ is number of beams, threshold $r$
Output: $F = \{(\alpha_1, p_1, q_1), \dots, (\alpha_m, p_m, q_m)\}$: $m$ detected frontiers, each frontier
        has a direction $\alpha_i$ and its boundary points are $p_i$ and $q_i$
for $i \in 1, \dots, n$ do
    $d$ = distance between $p_i$ and $p_{i+1}$;
    if $d > d_t$ then
        $\alpha$ =direction to midpoint on line $(p_i, p_{i+1})$;
        $F = F \cup \{(\alpha, p_i, p_{i+1})\}$
```

Detection of center of an interesting place

To be able to navigate the robot towards the interesting place, we have to define the center of the place. This is determined as the center of the largest free circle in the laser data. The free circle does not contain any point from the laser polygon. Moreover, the free circle must lay inside the scan polygon.

The detection of the largest free circle is based on a Voronoi diagram constructed over the laser polygon. The algorithm first replaces all frontiers (which are segments of length larger than d_t) by a set of points. In the second step a Voronoi diagram is built from the polygon data and from the frontier points. The Voronoi diagram is a graph $VD = (V, E)$ with the set of Voronoi verices V and Voronoi edges E. The center of the largest free circle is located at the vertex $v \in V$ of the Voronoi diagram. However, first the Voronoi edges outside the polygon must by removed, otherwise the largest free circle will likely be located outside the polygon. An edge is removed from the Voronoi diagram, if at least one of its points is located outside the polygon P. The reduced Voronoi diagram

$$VD_R = (V_R, E_R), : \{e = (p, q) \in E_R, p, q \in V_R : p, q \in V \wedge p, q \text{ inside polygon area}\}$$

.

The center of the largest free circle in the polygon is then searched among all vertices $v \in V_r$:

$$c_{max} = \max_{v \in V_r} c(v), \tag{10.4}$$

where $c(v)$ is a radius of the largest free circle with a center v. The searching of the maximum can be speeded up by using the KD-tree.

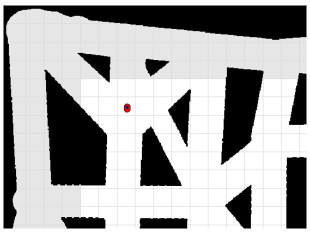
The center of the largest free circle can be found in $O(n^2)$ time, where n is number of points in P'. Herein the most time consuming operation is the filtering of outer voronoi edges, which is $O(n^2)$. The algorithm of the center detection is depicted in Alg. 4. The examples of detected center of an interesting place in a polygon is are depicted in Fig. 11.

Algorithm 4: Finding a largest free circle in a polygon.

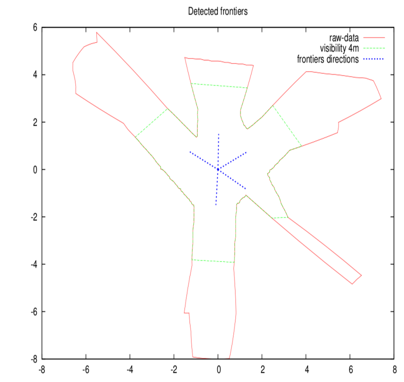
```
Input: $P = \{p_1, \dots, p_n\}$ laser scan polygon, $F = \{(\alpha_1, p_1, q_1), \dots, (\alpha_m, p_m, q_m)\}$
       set of $m$ frontiers
Output: $c_x, c_y, c_r$ center and radius of the largest empty circle
$P''$ = replace frontier $F$ by set of points;
$P' = P \cup P''$ ;
$V$ = voronoi digram of points $P'$;
$V_r$ = filter voronoi edges outside polygon $P'$ ;
$K$ = build KD-tree from $P'$;
for edge $e = (p, q) \in V_r$ do
    $(dist_1, k_1)$ = find nearest point in $P'$ to $p$ using KD-tree $K$;
    if $dist_1 > dist_m$ then
        $c_x = p.x;$
        $c_y = p.y;$
        $c_r = dist_1;$
        $dist_m = dist_1$
    $(dist_2, k_2)$ = find nearest point in $P'$ to $q$ using KD-tree $K$;
    if $dist_2 > dist_m$ then
        $c_x = q.x;$
        $c_y = q.y;$
        $c_r = dist_2;$
        $dist_m = dist_2$
```

10.3.3 *Manual Navigation*

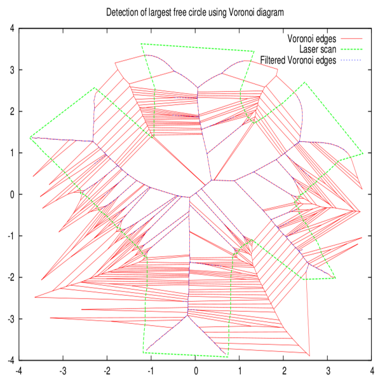
The manual navigation (MANAV) is a special case of the Memory-less navigating Jockey. The human operator manually drives the robot following the command set



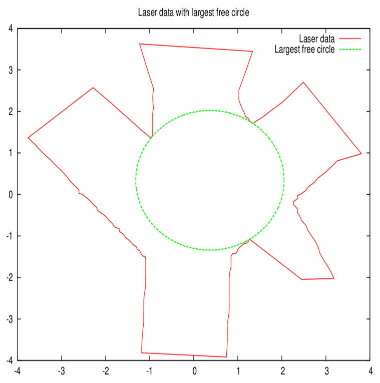
a



b



c



d

Figure 11: Example of detection of frontiers and largest empty circle in laser scan. (a) simulation environment, (b) detected frontiers, green line denotes polygon in a reduced visibility (2 m), (c) Voronoi diagram made of points on green polygon, blue line denotes filtered Voronoi edges, (d) the detected largest free circle.

requested by the executor. It is necessary to have at least one Learning Jockey to profit from the manual navigation. The edges traversed by MANAV can be mapped even in spit of absence of Learning Jockey, but these edges are traversable only in presence of the human operator.

The power of the manual navigation arises in the combination with the Learning and Memory-based navigating Jockey. In this case, the human operator can lead the robot to areas inaccessible by reactive navigation or can add the used-defined place. The trajectory is learned and represented in the map in a way, the robot is able to traverse this learned trajectory autonomously later on.

It works as follows: The executor asks the human operator to traverse an edge. Human operator drives the robot along desired edge. Simultaneously, the Learning Jockeys are learning the actual edge, if Learning Jockeys are presented. The operator denotes the end of the edge and return the control of the robot to the executor.

The intention of the manual navigation is in the exploration phase. But it is also possible to use it in the case of failure. Another possible usage of the MANAV is recovery action during the operation of the system. If the robot fails to navigate the edge and the executor runs out of all possible recovery action, the last recovery is to ask the operator to navigate the robot out of this situation, if the operator is present.

10.4 LEARNING AND MEMORY-BASED NAVIGATING JOCKEYS

As the Learning and memory-base navigation are tightly connected, the Learning and Memory-based navigating Jockeys are described together. The Memory-based navigating Jockeys are able to move only along the pre-learned edge. They are useless in absence of appropriate Learning Jockey and vice versa. The learned edge is traversable only using the appropriate Memory-based navigating Jockey.

10.4.1 *Visual Surf-based Navigating and Learning Jockey*

This algorithm [40] provides the functionality of the learning and memory-based navigating jockeys at once. It is called SURFNav according the SURF features used for image processing. For the navigation is necessary to have an edge descriptor previously acquired by SURFNav learning jockey.

The SURFNav jockey recognizes objects in the image taken by a forward looking camera and corrects direction of robot movement. Measurements from the magnetic compass and odometry are integrated as well.

A brief explanation of object extraction from the image is given in following subsection. Then the description of learning jockey function is given followed with the description of the navigating jockey.

Object recognition

SURFNav uses Speeded Up Robust Features [1] to identify robust and reliable landmarks in the image. This algorithm processes gray-scale images in two phases: At first, a local brightness extrema detector is applied to the image. In the next phase, a scale, rotation and skew invariant descriptor of detected extrema neighborhood is computed. Algorithm provides image coordinates of salient features together with their descriptor. To speed up computation time, the image is horizontally split and its parts can be processed simultaneously by multiprocessor machine. See processed image with highlighted feature positions on Figure 12.

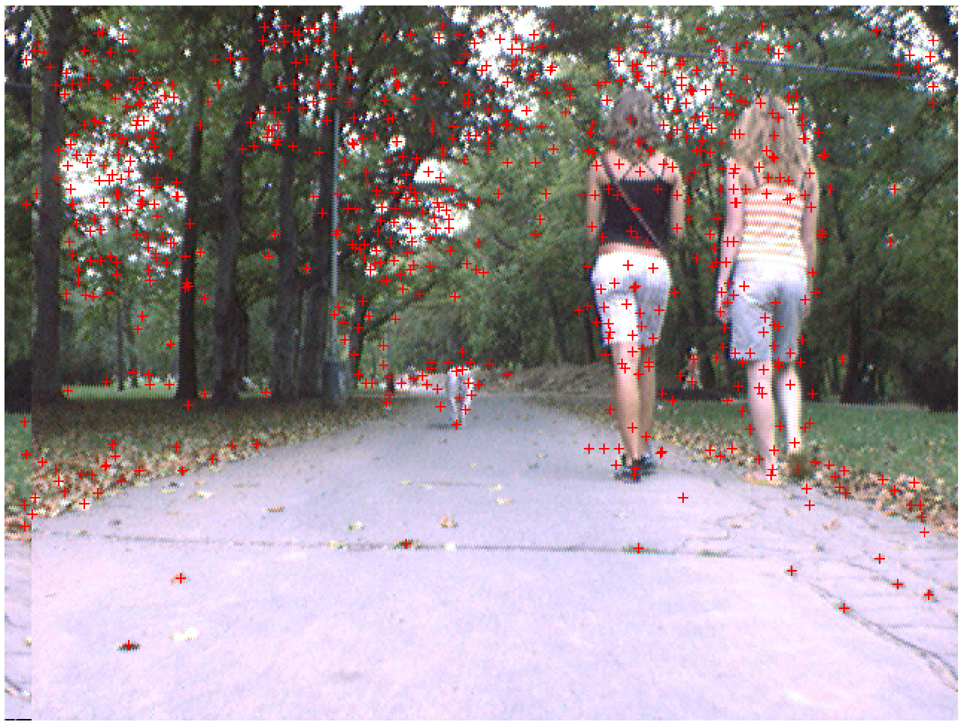


Figure 12: Image and detected features.

Learning Jockey

In the learning mode of the SURFNav, the learning jockey expects the robot is guided through the edge along a piece wise linear trajectory. At the beginning of each edge, the jockey resets robot odometry counter, reads compass data and takes a predefined number of images. Objects, which have been detected in predefined numbers of subsequent snapshots of this series are considered to be stable. Stable objects with constant positions are regarded as stationary. Positions and descriptors of stored objects are saved. Afterwards, the robot starts to move forwards, the jockey obtains and processes images and records odometric data. When an object is detected for the first time, the algorithm saves its descriptor, image coordinates and robot distance from the segment start. Saved objects are tracked over several pictures and their positions in the image are assigned to current robot position within a segment. Tracking of an object is terminated after three subsequent unsuccessful attempts to detect it in the image. Its descriptor, image coordinates and odometric data in moments of the first and the last successful recognition are inserted into the dataset describing the traversed segment. Edge learning is

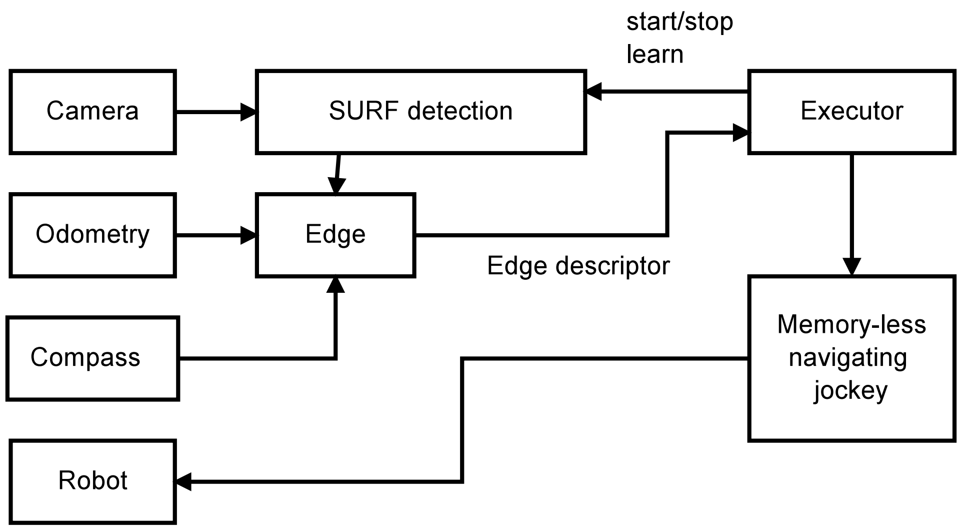


Figure 13: Block scheme of SurfNav learning jockey.

terminated by an executor, which stops the robot (edge length is saved).

Navigation Jockey

When navigation mode is started, the navigating jockey receives the edge description and turns robot to the indicated direction. After that, the odometry counter is reset, forward movement and picture scanning are initiated. Objects, which are expected to occur in the image, are selected from learned set. These are the objects with the first and the last detection distance greater, respectively lower than the current robot distance from edge start. Expected image coordinates in current camera image are calculated by linear interpolation using aforementioned distances. Selected objects are rated by a number of frames which they have been detected in and top-rated objects are chosen as suitable for navigation. For each candidate, the most similar object is searched in the set of actually detected ones. The similarity is calculated from an Euclidean distance of descriptors of both compared objects. A difference in horizontal image coordinates is computed for each such a couple. A modus estimate of those differences is then converted to a correction value of movement direction. After the robot travels distance greater or equal to the length of the given segment, jockey announces the edge has been traversed.

An important aspect of this navigation algorithm is its functionality without the need to localize the robot or to create a three-dimensional map of detected objects. Even though the camera readings are utilized only to correct the direction and the distance is measured by an imprecise odometry, it is shown, that if the robot changes direction often enough, it will keep close to the previously learned trajectory.

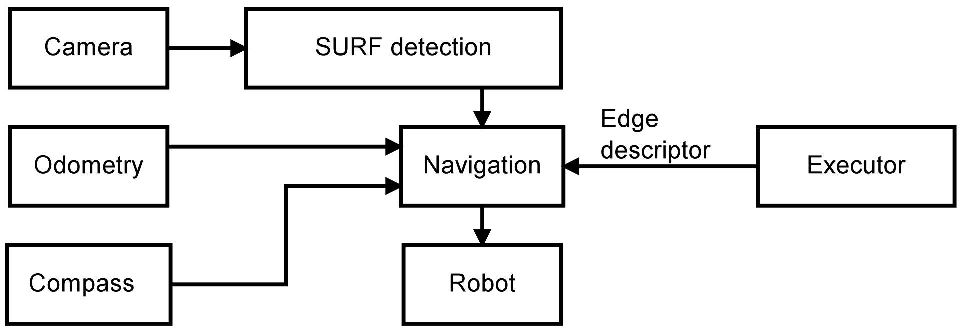


Figure 14: Block scheme of SurfNav memory-based navigating jockey.

10.4.2 *Laser Memory-based Navigation Jockey*

The laser memory-based navigating jockey relies on the "record and replay" approach.

The navigating jockey takes the first scan from the descriptor of the learned edge and localizes the actual scan within it. The localization reuses the approach of the Bosse and Zlot [7], where histograms of the angles and projections of the scan is used. While the method is not invariant to the width of the histogram bins in diverse environments, this parameter has to be determined adaptively. At normal conditions histogram bin starts at a predefined size.

Algorithm 5: Laser memory-based Navigation

```
input: edge description = vector of scans $S$
repeat
    take first scan $s \leftarrow S$;
    localize robot $c, \varepsilon \leftarrow localize(n_{bins}, s)$;
    while error of the localization $\varepsilon > \tau$ do
        modify $n_{bins}$ ;
        if all possible $n_{bins}$ tried then
            return navigation failed
        localize robot $c, \varepsilon \leftarrow localize(n_{bins}, s)$;
    if $|c, origin| > \varsigma$ then
        navigate to localized sensing location $c$ ;
    else
        remove $s$ from $S$;
until empty $S$ ;
return navigation succeed
```

According to the quality of the computed match of two current scans ε, measured by average distance between nearest points in compared scans, the size of the bin is lowered until the quality not reached required precision.

After the matching algorithm ends, the center of the scan (sensing location) c is localized within the robot's local frame of reference. In next steps robot is navigating towards this point and periodically re-localize the position of the sensing position using an actual laser scan. After the robot reaches the sensing location, the scan from next sensing location is taken and matched with the actual scan. This procedure is repeated until the robot reaches the destination vertex of the edge.

This navigation method is relatively insensitive to moving objects like passing people. Also small changes of static objects do not influence the navigation algorithm substantially.

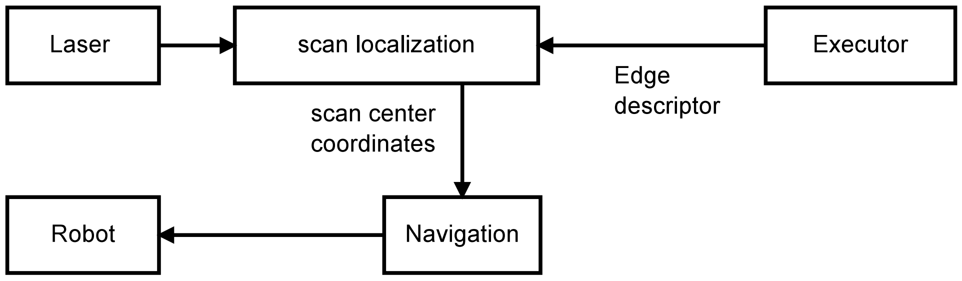


Figure 15: Block scheme of laser memory-based navigating jockey.

10.4.3 *Laser Learning Jockey*

The learning jockey simply records the traversed edge by taking a single laser range scan from each sensing location along the edge. The sensing locations are distributed along the edge to ensure the navigating jockey is able to follow this edge. The fist sensing location is placed at the beginning of the edge and the scan of this place is stored into the descriptor of the edge. The actual laser scan is taken and localized within this scan. The relative position c and error ε of the localization is taken.

The next sensing location is created when the distance of actual position to the previous sensing location is higher than the threshold ϑ. The size of the threshold determines the sampling of the edge and influences the precision of the following the learned edge.

The next sensing location is also created if the error ε is bigger than the τ. This error pointed to that the previous sensing location is not descriptive enough to navigate to them from actual location. In this case, the previous scan with lower error is used to create sensing location and store in the edge descriptor. Typical structures/situations, where the error of localization increase radically are doors and narrow passages. The environment structure on each side of the door (or the passage) can be very different and there is not sufficient information of the other

Algorithm 6: Laser memory-based Edge Learning

```
output: edge description $D$
$D \leftarrow \emptyset$;
take actual scan $s$;
create sensing location $sl_1 \leftarrow s$;
add $sl_1$ into $D$;
$i \leftarrow 1$;
repeat
    take actual scan $s$;
    localize robot $c, \varepsilon \leftarrow localize(n, sl_i)$ ;
    if $error\ of\ the\ localization\ \varepsilon > \tau\ or\ |c, origin| > \vartheta$ then
        $i++$;
        $sl_i \leftarrow s$;
        add $sl_i$ into $D$;
    take actual scan $s$;
    $sl_n \leftarrow s$;
    add $sl_n$ into $D$;
until Edge traversal ends ;
return D edge descriptor
```

side in the actual scan. This rule ensures the placement of the sensing location inside the door.

The last sensing location is placed at the end of the edge. The learned edge is then possible to traverse in both direction.

10.5 localizing jockeys

The Localizing Jockeys provide two functionalities. They determine the similarity of vertices and localize the robot within the local frame of coordinates of the vertex.

10.5.1 *Laser-based Localization Jockey*

The robots equipped with a laser rangefinder may use the laser-based localization jockey. This jockey consists of algorithm for detection of the places of interest and algorithm detecting outgoing edges.

The place of interest (intersections) can be defined as a place where more than two edges are joined. The detection algorithm searches for the outgoing edges, however these depend on robot position. The algorithm thus determines the center of the place from which the outgoing edges are searched. This center is computed as a center of the largest free circle in the polygon formed by a laser data.

During the outgoing edge detection the scan is considered as a simple polygon in which the frontiers are searched. The frontier searching algorithm works in two steps. At the first step all polygon vertices, which are distant from the current standpoint more than a given threshold r are removed. The polygon edges which length is larger than the size of a robot make a set of frontiers candidates. From this set the frontiers through which the robot cannot pass due to physical constrains are removed.

The algorithms are same as in the section 10.3.2. The algorithm for frontiers detection is algorithm 3. The algorithm for center detection is algorithm 4.

The number of found frontiers is influenced by a value of the threshold r. The values of the threshold cannot be preset while the robot is assumed to operate in both large free areas (where the r should be large) and narrow corridors (where the r should be small).

Therefore value of the threshold is dynamically adopted to structure and shape of the current environment based on processing of incoming laser data. The histogram of measured distances is build from laser data and the highest score bin determines the value of r.

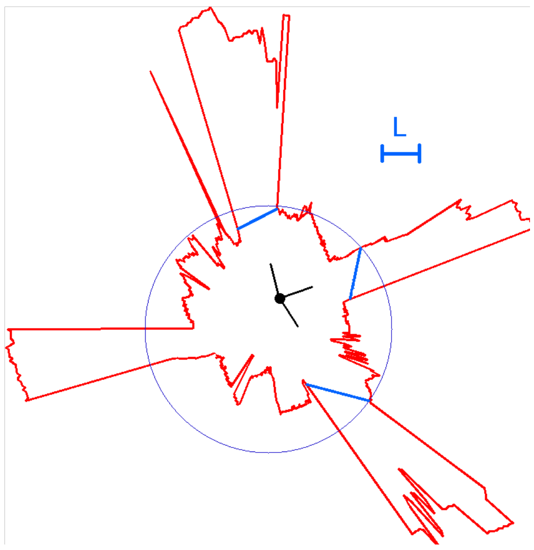


Figure 16: Raw laser data (red) with detected frontiers (blue). The threshold r is depicted by the blue circle , L denotes the size of the robot.

10.5.2 *Descriptor matching*

Each vertex in the topological map has assigned several descriptors obtained by various jockeys. The descriptor matching algorithm computes similarities between place descriptors of the same type. For each descriptor type several diverse algorithms can be used to determine the similarity.

The algorithms for similarity computing, produces a single quantitative measure for each pair of vertices descriptors being compared. In this context, any descriptors in a pair are considered to represent the same physical place if the similarity is larger than a threshold ϑ. This threshold must be set for each particular matching algorithm separately. The value of the threshold ϑ can be determined using ROC

(Receiver Operating Characteristic) [25]. For the above mentioned algorithms for similarity computation the classification threshold ϑ has been determined using the ROC see sec. 12.1.

Normalized Cross-Correlation

A way how to obtain level of similarity for a given pair of laser range data is evaluation of a cross-correlation function for the measured distances. The laser scan is a finite set of measured range data d of size N. Let d be an access function to a finite data set as

$$d(iN + k) = d(k), i \in \mathbb{N}, 0 \leq k < N$$

.

As the similarity is intended to be from the interval $[0, 1]$ in is necessary to use a normalized version of cross-correlation:

$$(f \star g)(y) = \frac{1}{N-1} \sum_{x=1}^{N} \frac{(f(x) - \bar{f})(g(x+y) - \bar{g})}{\sigma_f \sigma_g}, 0 \leq y < N,$$

where $\bar{f}$ means average of f and σ_f is a standard deviation. This function returns 1 only if the both functions are equivalent.

Alternatively, the normalized vector F can be defined as

$$F(x) = f(x) - \bar{f}$$

and the above sum is equal to

$$\langle \frac{F}{\|F\|}, \frac{G}{\|G\|} \rangle,$$

where $\langle \cdot, \cdot \rangle$ is a inner product and $\| \cdot \|$ is the L^2 norm.

The normalized cross-correlation method is called $NCC(f, g)$ for rest of the paper. The similarity of two laser scans is then computed as :

$$s_{NCC}(d_i, d_j) = \max_y NCC(d_i(x), d_j(x + y)).$$

Fourier Transformation

More sophisticated approach to evaluate the similarity may be normalized cross correlation of the Fourier coefficients calculated by corresponding discrete Fourier transformation of the data in polar coordination system:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, 1, \ldots, N-1$$

The Fourier coefficients are computed using the fast Fourier transformation implementation from [26] and therefore is denoted as FFT for rest of the paper. As

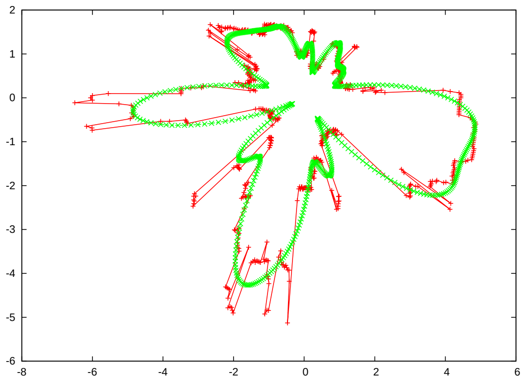


Figure 17: The raw laser data (red) and data reconstructed using first $k = 32$ harmonic functions (green)

amplitudes of the Fourier coefficients express the similarity, descriptor is invariant to rotation. It permits a robot to scan the place descriptors regardless of the its orientation.

Descriptor based on FFT is defined as

$$D_{FFT} = \{X_1, X_2, \ldots, X_k\}$$

To minimize influence of the noise in the data, the first $k < N$ harmonic functions can be considered for similarity computing only. As the first harmonics describe a coarse character of the laser data, the higher harmonics describe preferrably the noise components of the data. An example scan of real environment and the scan reconstructed from its first $k = 32$ harmonic functions are depicted in Fig. 17.

The similarity of two descriptors $s(D_i, D_j)$ is computed

$$s(D_i, D_j) = \mathrm{NCC}(D_i, D_j)$$

Polygonal Methods

If the laser data are represented by a simple polygon, the polygon matching algorithms can be used for computation of the similarity between place descriptors. The polygonal matching algorithms have been studied for many years [52] and with the assumption that the global properties of the polygons like area, perimeter, moments etc. can be used to measure the similarity. However these methods do not consider a shape of the polygon. Different places of interest can have assigned polygons, that cannot be discriminated using their global properties only. Hence shape matching algorithms are used for similarity measurement.

Integral Invariant

The integral invariant method [54] relies on measurement of similarity between two curves that represent an integral invariant of the polygon. The integral invariant for a continuous curve C is defined as computing $I(p)$ by equation:

$$I(p) = \int_C \|p - x\| \, \mathrm{d}s(x),$$

where $\|x - y\|$ is Euclidean distance in $\mathbb{R}^2$, for every point $p \in C$. The discrete version holds for a polygon P:

$$I(p) = \log \sum_{x \in P} (\|p - x\|)$$

for every point p from polygon and the descriptor then is:

$$D_{II} = \{I(p_1), I(p_2), \ldots, I(p_n)\} \quad n = |P|$$

where n is number of points in the polygon P.

The similarity $s(D_i, D_j)$ is computed as:

$$s(D_i, D_j) = \sum_{k=1}^{n} (D_{iu} - D_{jv}), \quad (u, v) = C(k),$$

where $C(k)$ is k-th correspondence. The best correspondence is defined by minimising the function:

$$\sum_{\substack{k \in 1 \ldots M \\ l \in 1 \ldots N}}^{L} |I_1(k\Delta s_1) - I_2(l\Delta s_2)|^2 + \frac{(h_1 + h_2)}{2} + \alpha \left| \frac{h_2 - h_1}{h_1 + h_2} \right|^2 \frac{h_1 + h_2}{2},$$

(k, l) are corresponding points ,$I(k\Delta s)$ is a integral invariant in k-th point and h_1, h_2 are determined by warping function $h = (h_1, h_2)$ which represents the pointwise correspondence between points of curves (as defined in [54]).

The integral invariant is more sensitive to global changes (the shape of the polygon) and less to changes imposed by a noise in depth data along the horizon.

Shape context

This approach assigns to each polygon vertex a shape context [2], which describes the near neighborhood of the vertex in question. Then, two vertices are considered similar if their shape contexts are similar. The shape context is defined as the two-dimensional histogram $H(p)$ of logarithmic polar distances from a particular vertex to other vertices in the polygon.

The distance between two shape contexts is given as a distance between two such histograms

$$C(p_i, p_j) = \frac{1}{2} \sum_{i=1}^{K} \frac{\big(H_i(k) - H_j(k)\big)^2}{H_i(k) + H_j(k)},$$

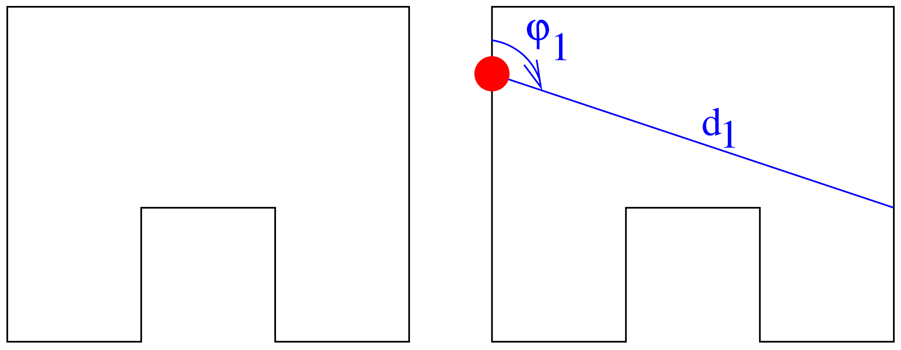


Figure 18: Shape context of one point.

where K is number of bins in the histogram.

Similarity measure between two polygons can be then computed as follows: First the shape context is computed for each vertex and the shape distance between vertices of the two polygons are computed. The correspondences between polygon vertices are resolved by application the bipartite matching problem. The distance between two polygons is obtained as a sum of distances between resulting matched vertex pairs.

Tangent Space

Traditionally, the closed polygon can be represented as a list of vertices or by giving a list of line segments. Alternatively, a polygon can be represented using a tangent space - a list of angle-length pairs, whereby the angle at a vertex is an accumulated tangent angle at this point while length is the normalized accumulated length of polygon sides up to this point. The length is normalized to be 1 for all polygons.

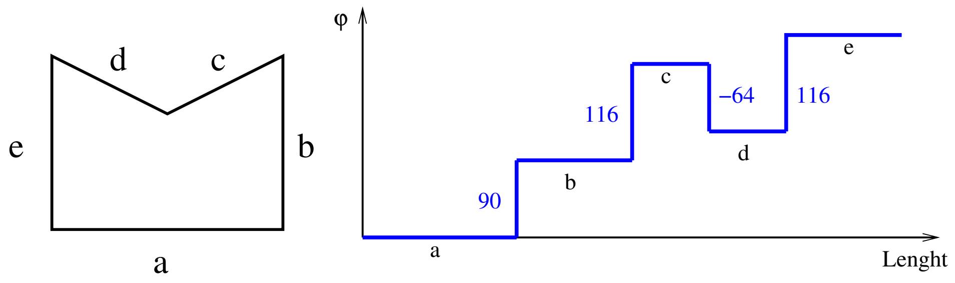


Figure 19: Closed polygon and corresponding tangent space representation.

As the tangent space representation depends on the starting vertex, lets define a tangent space representation $t_p : [0,1] \mapsto [0,2\pi]$ to be a projection from normalized length to accumulated tangent angle starting from point $p \in P$. Then the similarity of two polygons is a minimum difference between all possible variants of the tangent space representation:

$$s(P, Q) = min_{p \in P} \int_0^1 |t_q(x) - t_p(x)|dx.$$

Scan Line

The scan line matching algorithm is based on the approach introduced in [10]. The shape descriptor is computed from the intersection $l \cap P$ of the randomly placed lines $l \in L$ with the polygon P. All the intersecting points $x_1, x_2, x_3, \ldots, x_2n \in \mathbb{R}^2$ are ordered and forms n compact intervals. The intersection function is defined as

$$S_{l \cap P}(\xi) = \sum_{k=2}^{2n} \sum_{i=1}^{k-1} (-1)^{i+k+1} I\{x_k - k_i > \xi\}$$

for all interval lengths $\xi > 0$, and where

$$I\{x_k - x_i > \xi = \begin{cases} 1 & , \quad \|x_k - x_l\| > \xi \\ 0 & , \quad \text{otherwise} \end{cases} \tag{10.5}$$

If $x_k - x_i$ represents an interval strictly interior or exterior to a polygon, the sum is incremented. If the interval represents a collection of intervals both interior and exterior, the sum is decremented.

The descriptor of the polygon is then defined as

$$D_{SL}(\xi) = \frac{1}{N} \sum_L S_{l \cap P}(\xi)$$

and the similarity of two polygons

$$s(D_i, D_j) = \sum_{\xi \in \Xi} |D_i(\xi) - D_j(\xi)|,$$

where the Ξ is a set of interval lengths used in the descriptor.

11 EXPLORATION

In this chapter, the algorithms for the topological exploration are described. The topological exploration of the environment can be seen as the graph exploration. Many of the graph exploration algorithms rely on the unique labeling of the vertices or edges. These approaches are not suitable for the robotic exploration at all, or require to put the artificial landmarks into the environment to produce such a labeling.

The currently used algorithms often use an movable marker to allow robot distinguish the vertices from each other. It puts the hard requirements to the robot hardware which must be able to put the marker into the environment and collect it later back. Therefore in the first section, the algorithm with one not-movable marker is proposed. This marker is placed at the starting place, called "base vertex".

In the second section, the algorithm is further modified to avoid usage of the marker at all. The reasoning procedure is used to close the loops in the environment instead of the marker.

11.1 EXPLORATION WITH MARKER

The algorithm consists of two phases: exploration and vertices merging.

In the *exploration phase*, the robot moves through the environment and makes its own map $G_M = (V_M, E_M)$ of the world. As the robot cannot distinguish particular vertices from each other, it is also unable to close loop in exploration without visiting the "base" vertex (or interacting with some other robot). Moreover, every visited vertex must be handled as previously unvisited one unless the robot proves the contrary.

The *vertices merging phase* starts whenever the robot detects the "base" vertex. This situation allows to close the loop and merge identical vertices. In the exploration phase, one place in the environment might be represented by more vertices in the map. This inconsistency is reduced in the *vertices merging* phase.

The algorithm works properly only if the robot is able to follow all detected edges. Existence of complementary edges is also necessary. Edge $\bar{e} \in E$ is complementary to edge $e \in E$ if and only if expression (11.1) holds.

$$\forall e \exists \bar{e}, \quad e, \bar{e} \in E, e = (u,v), \bar{e} = (v,u), \quad u,v \in V \tag{11.1}$$

Moreover, the robot knows complementary edge $\bar{e}$ after passing e. It means that the robot is able to backtrack its movements.

After the robot passes from the vertex u to v, it also knows how to move from v to u even without passing this way back. As the robot knows azimuth from which it entered a vertex, it also knows the way back.

Exploration Phase

The robot moves through environment and stores vertices and edges into the map during this phase. At the beginning, the robot has no information about the environment. The robot starts to follow actual edge until a crossing is detected. This crossing is stored in the map as the first vertex. The nearest unexplored edge is used for the further movement.

The exploration phase is based on a graph depth-first search (DFS) algorithm. This algorithm is greedy because the robot follows the nearest vertex with unexplored edge. If there is more than one unexplored outgoing edge from an actual vertex, the edge for next step is chosen randomly with uniform probability. It is possible to use breadth-first search (BFS) algorithm, but the robot travels typically larger distances with BFS.

When the robot arrives to the next vertex (crossing), the edge between this and previous vertex is added into the map. The complementary edge is known from entry azimuth and is also added into the map. If the robot visits the "base" vertex, vertices merging phase is executed.

Algorithm 7: exploration phase

```
follow edge $c(e)$;
if detected place then
    add new vertex $v$ to the map $V_M$;
while exists unexplored edge in the world do
    if all edges from u was explored then
        find path to nearest vertex with unexplored edge;
        choose first edge $e$ from path;
        execute edge $e$;
    else
        choose randomly unexplored edge $e$;
        store azimuth into $t(e)$; execute $e$ to move to $v$;
        add vertex $v$ into the map $V_M$;
        add edge $e = (u, v)$ to the map $E_M$;
        add edge $\bar{e} = (v, u)$ to the map $E_M$;
```

As the robot uses only the greedy exploration strategy, exploration can take a long time. If the environment is a tree-like graph with n crossings, it is guaranteed, the exploration finishes in $2n$ steps. When cycles occur in the environment, the robot can stuck in the cycle, if the cycle does not contain the "base".

Therefore, the *metrical heuristic function* is utilized. This heuristic function estimates metric position of the vertex from robot odometry. After the robot spends certain time in unexplored space, edges directing to the base are preferred. Edges are still chosen randomly but not with uniform probability. The Roulette-wheel selection is used. The parts for each edge are allocated according to its deviation from the direction to the base. The largest part of the roulette-wheel has edge with lowest deviation.

Random choice must be kept because errors in computation of base position are affected by cumulative errors in its odometry. Also paths may not be straight but can have different shapes.

Vertices Merging Phase

When the robot arrives into the "base" vertex, the *vertices merging* phase is started. First, the actual vertex recognized as base is merged with base vertex in the map. Next, the robot makes the map consistent according Algorithm 8.

Algorithm 8: vertices merging phase

```
merge($v_{actual}, v_{base}$) ;
while $\exists u, v, w \in V_M : \quad e_1 = (u, v), e_2 = (u, w), \quad e_1, e_2 \in E_M \wedge t(u, v) \approx t(u, w)$
do
    merge(v,w);
```

It is assumed, there may exist exactly one edge of each type leading from every vertex. Type of the edge is denoted as $t(e)$ or $t(u, v)$. The same edge type means that the difference between azimuths is smaller than the azimuth recognition precision. If two or more edges of the same type lead to different vertices, these vertices necessarily represent the same place in the world and therefore are merged. This is repeated recursively.

Algorithm 9: merge(u,v)

```
while $\exists x \in V_M : e = (v, x) \in E_M$ do
    if $e = (u, x) \notin E_M$ then
        add $e = (u, x)$;
    remove $e = (v, x)$;
while $\exists x \in V_M : e = (x, v) \in E_M$ do
    if $e = (x, u) \notin E_M$ then
        add $e = (x, u)$;
    remove $e = (x, v)$;
remove $v$;
```

Terminal Condition

The whole exploration procedure terminates whenever the environment is explored completely. It means that the robot has available a complete map of the environment at this time. The map is complete if and only if no vertex of the map has unexplored edge.

11.2 MARKER-LESS EXPLORATION

The previous exploration algorithm has a drawback of usage of the marker. In this section, the modification of the previously described exploration algorithm is described. The usage of the marker is replaced with the reasoning procedure.

The exploration strategy is breadth-first search modified to minimize backtracking moves of the robot. In this approach, the exploration strategy examines all outgoing edges from the current vertex. The presence of at least one learning jockey is required to ensure the backtracking.

Algorithm 10: Exploration algorithm

```
start Memory-less navigating jockey to find the nearest vertex $u$;
start Localizing jockeys to get descriptor and outgoing edge;
add $u$ to the map $M$;
add edges at the end of $open$ list;
set $u$ as $actual$ vertex; while $open$ not empty do
    remove edge $e$ nearest to $actual$ from $open$ list;
    if $e_{start} \neq actual$ then
        find route to $e_{start}$;
        traverse route; set $e_{start}$ as $actual$;
    start Learning Jockeys;
    traverse $e$ with memory-less jockey;
    stop Learning jockeys;
    get vertex $v$;
    get learned edges $e_l = (u, v)$ and $\bar{e}_l = (v, u)$;
    if $M$ contains $v$ then
        set $v$ as $actual$;
    else
        add $v$, $e_l$, $\bar{e}_l$ into $M$;
        traverse $\bar{e}_l$;
```

The robot starts at an arbitrary location in the environment. As the first step, the first place is found using a reactive navigating algorithm. This place is put into the map as a vertex. Then, the robot starts to explore all outgoing edges from vertex. During the navigation along the edge, the learning jockey is started. The learned edge is used for backtracking. After all the edges outgoing from actual vertex are explored, and their target vertex is known, the robot moves to next vertex in the BFS manner.

When it is checked if the map M contains the newly discovered vertex v, the loop-closing procedure (Alg. 2) is called. If belief in the similarity of the new vertex v with any of the existing vertex is higher than threshold $b > \vartheta$, the method deduces that the map M contains vertex v. Along with this check, the other loops can be discovered and closed and therefore, the map M is replaced with the actual map with closed loops M_c.

11.2.1 *Knowledge-base Exploration Strategy*

Both previously described exploration strategies, breadth-first search and depth-first search, are uninformed search strategies. The reasoning process has the partial information about the environment, therefore the novel informed exploration strategy is proposed here.

The knowledge-base exploration strategy utilizes the information about the visited vertices, traversed edges as well as the unexplored edges. The new information is gained every time the new vertex is discovered or the unexplored edge is traversed and the target vertex becomes known. Therefore, the robot gets new information after traversing an unexplored edge but the amount of information differs.

The knowledge-base exploration strategy chooses as a next step such an unexplored edge which brings the largest enrichment of to the knowledge about the environment. The increase of the knowledge is equivalent to decrease of uncertainty in sense of subjective logic. Biggest decrease of uncertainty is caused by loop closing, when a numbers of unexplored edges are eliminated from the model, as each unexplored edge has the uncertainty in a target vertex.

Algorithm 11: Knowledge-base exploration strategy

```
Input: Map $M = (V, E = \{E_U \cup E_E\})$
Output: edge to explore
for $e = (u, x) \in E_U \quad u \in V, x = unknown$ do
    for $f = (v, w) \in E_E \quad v, w \in V$ do
        compute similarity $\vec{\mathcal{O}}(e) \leftarrow \mathcal{O}(S(e, f)) \wedge \mathcal{O}(s(u, v))$;
return $edge \leftarrow \arg\max_e(\vec{\mathcal{O}})$;
```

The input of the algorithm is the map M, where the set of edges is divided into two subsets E_U set of unexplored edges, and E_E set of explored edges. The algorithm finds the unexplored edge, which is the most similar to some explored one and the starting vertices of both the edges are also the most similar. The opinion from the subjective logic is used to combine and compare the similarities. This represents the hypothesis, that the most similar pair of edges represents the one physical route. The verification is made by traversing the unexplored edge and discovering its' target vertex. The result of the exploration step enters the reasoning process and supports or weaken this hypothesis.

Theoretical complexity of the knowledge-base exploration strategy is $O(|E_E||E_U|)$. In real computation, the precomputed values of the vertices and edges similarities are used and the complexity is $O(|E_E|)$ as only the inner loop of the algorithm is necessary to run.

The experimental results are summarized in next chapter.

12

EXPERIMENTS

The experimental results of the proposed methods are summarized in this chapter. The results are aggregated into the groups according the performed task.

The results of topological localization are described in the next section. The results from the simulated, indoor and outdoor environments are presented.

The experiments to verify the navigation algorithms in the real world environment is described in Section 12.2. Last Section 12.3 describes the result of the automated exploration of the diverse simulated environment as well as the real world indoor and outdoor environment.

12.1 LOCALIZATION

The localization in topological maps can be viewed as a problem of classification. A classifier is a mapping of instances into a certain class. The instances are the pair of the vertices and the two classes are positive, if the two vertices represents the same physical place, and negative otherwise. The classifier result can be in a real value (continuous output) in which the classifier boundary between classes must be determined by a threshold value ϑ. A receiver operating characteristic (ROC), or simply ROC curve firstly appears in a signal detection theory during World War II for detecting enemy objects in battle fields. Soon it was introduced in psychology to account for perceptual detection of stimuli. ROC analysis since then has been used in medicine, radiology, and other areas for many decades, and it has been introduced relatively recently in other areas like machine learning and data mining.

ROC curve, is a graphical plot of the sensitivity, or true positive rate, vs. false positive rate for a binary classifier system as its discrimination threshold ϑ is varied. The true positive rate TPR is fraction of true positive hits of classifier $c : \{P \cup N\} \mapsto \{0, 1\}$ out of all positive P cases with a given threshold ϑ

$$c_\vartheta(x) = \begin{cases} 1 & , c(x) > \vartheta \\ 0 & , c(x) \leq \vartheta \end{cases} \tag{12.1}$$

Method	**Complexity**	Computation time [s]
fft	$O(n\log(n))$	0.01
ncc	$O(n\log n)$	0.01
tangentspace	$O(n\log(n))$	0.1
integral invariant	$O(n^2)$	0.15
scanline	$O(nl)$	0.4
shape context	$O(n^3)$	2.7

Table 2: Complexity of the matching algorithms.

and false positive rate FPR is the fraction of false positive hits out of all negative N cases.

$$TPR(\vartheta) = \frac{\sum_{x\in P} c_\vartheta(x)}{|P|} \tag{12.2}$$

$$FPR(\vartheta) = \frac{\sum_{x\in N} c_\vartheta(x)}{|N|} \tag{12.3}$$

12.1.1 *Algorithms complexity*

The complexity of the localizing algorithms is shown in Tab. 2. The n is the number of measurement in the scan or number of points in the polygon. The $n = 761$ in case of the scans and can be smaller for the polygons as the scan is filtered. The l is the number of lines used in scan line algorithm and typical value used in experiments is $l = 2000$. The computation was performed on the computer with processor Pentium 4, 3GHz and 1 GB memory.

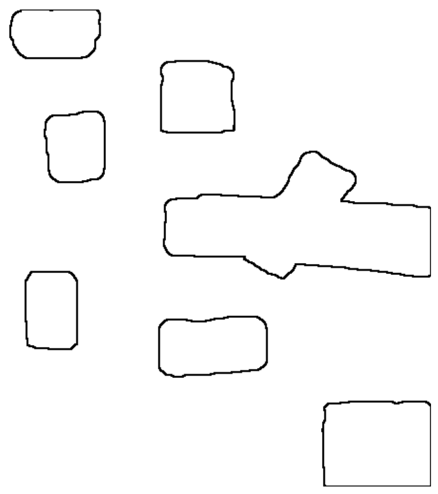
12.1.2 *Simulated Environment*

The localizing algorithms were tested at first in the simulator. The position from the simulator is used as a ground truth.

As the environments were chosen the standard simulated environment from the Player/Stage system, in which the simulation were performed. This environments are depicted in Fig. 20.

The robot performs a random walk in the environment and concurrently detects the places. Each place and its description is stored in the map as vertex. As the robot travel through the environment, the places are visited repeatedly and each time is inserted into the map. Therefore, multiple vertices represent the same place in the environment. All the vertices nearer than one meter were considered as a representing the same place.

The localization was performed for each pair of vertices. It forms a test set with 4352 negative and 799 positive examples for the *cave* environment and 4983


(a) Cave

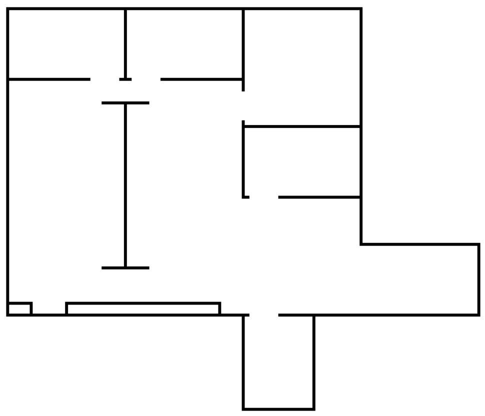
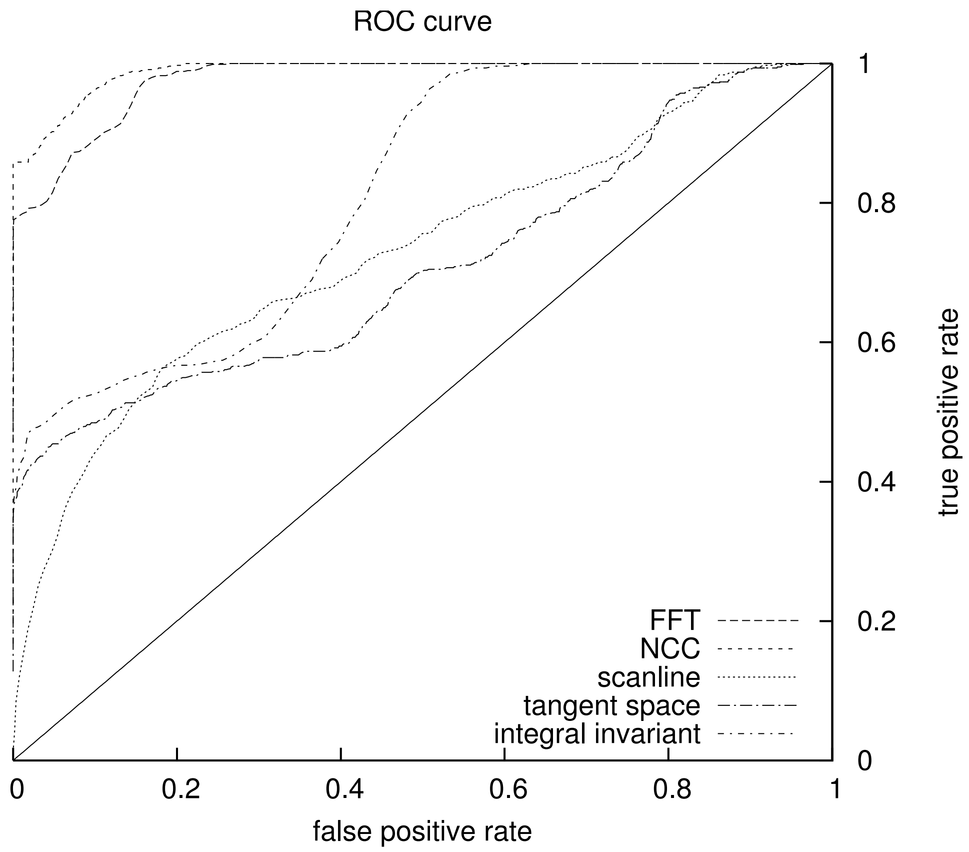

(b) Autolab

Figure 20: Simulated environments.

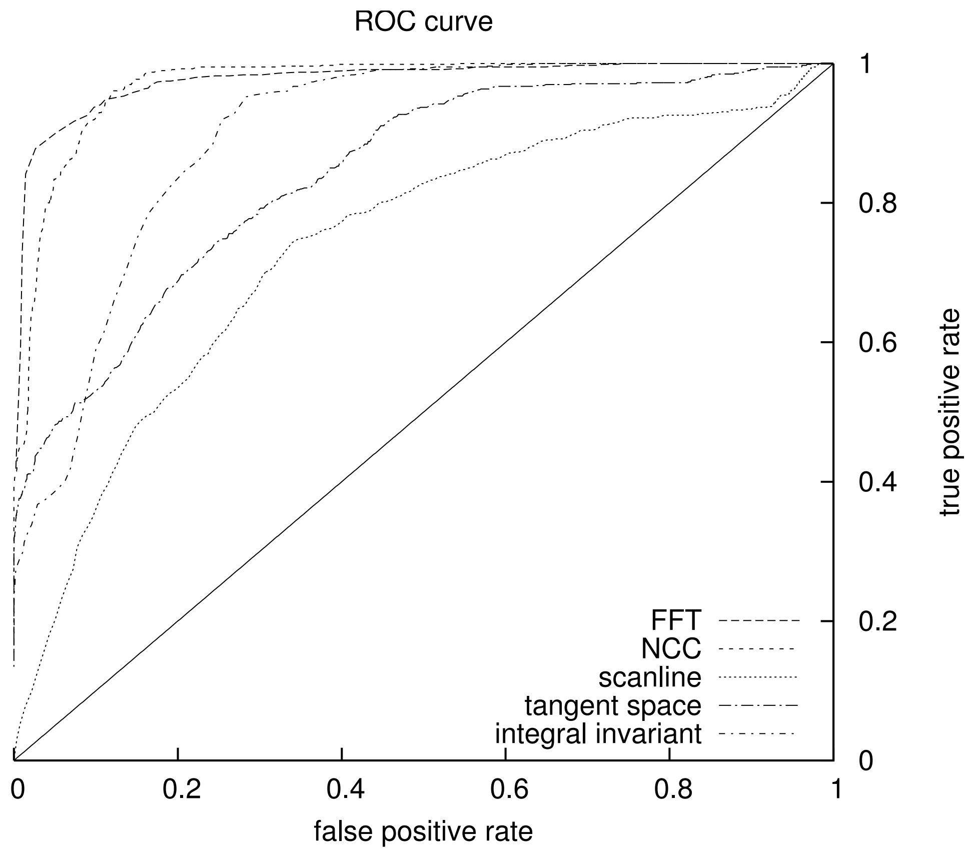
negative and 795 positive examples for the *autolab* environment. Results of the localization (classification) are presented in the form of ROC curve.

The area under curve (AUC) is related to the quality of the classifier. It is depicted in the table for the quantitative comparison of the classifier. The values of the optimal threshold and the appropriate true positive and false positive rates are depicted in the table 3 and 4.



matching algorithm	AUC	ϑ	TP[%]	FP[%]
fft	0.976	0.964	97.5	16.1
ncc	0.989	0.723	95.4	8.9
scanline	0.725	0.967	56.2	18.1
tangent space	0.673	0.841	44.9	4.3
integral invariant	0.809	0.94	96.4	50.9

Table 3: Classifier results for cave environment.



matching algorithm	AUC	ϑ	TP[%]	FP[%]
fft	0.974	0.995	87.8	2.6
ncc	0.972	0.712	96.1	12.0
scanline	0.734	0.939	74.2	33.9
tangent space	0.814	0.726	74.2	24.5
integral invariant	0.895	0.96	95.2	28.3

Table 4: Classifier results for autolab environment.

12.1.3 *Indoor Environment*

The data for indoor environment was collected using the G2 bot equipped with 2 sick laser range-finder pls-100 in our laboratory. The robot moves trough the environment and visit each place multiple times, similarly to the simulation. The odometry is taken as a ground truth. The raw composition of the scans with positions of all places is depicted in Fig. 21 The dataset produces 200 negative and 76 positive examples.

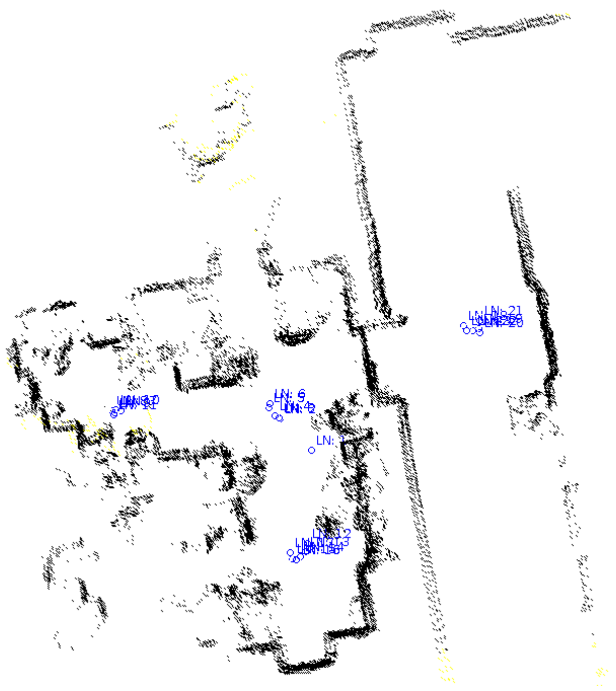
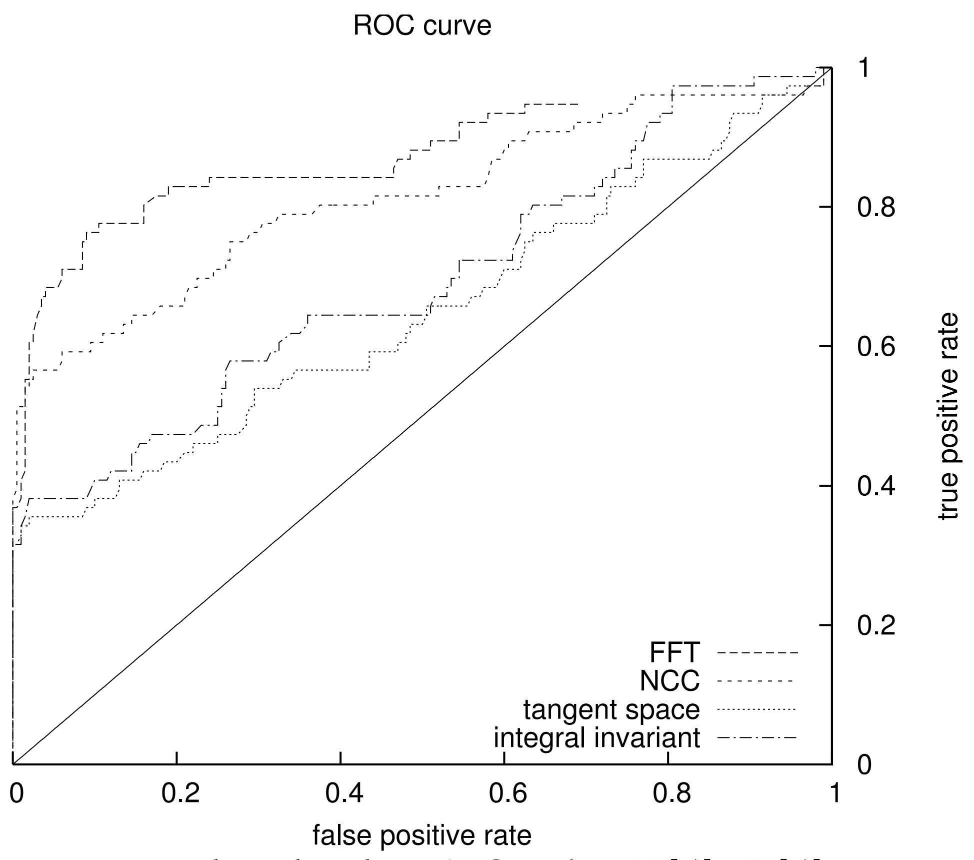


Figure 21: Map of indoor environment.

The resulting ROC and the values of optimal thresholds and values of true positive and false positive rates are shown in the table 5.



matching algorithm	AUC	ϑ	TP[%]	FP[%]
fft	0.572	0.918	76.3	9.0
ncc	0.812	0.711	56.6	2.5
tangent space	0.643	0.846	35.5	2.0
integral invarinant	0.687	0.963	38.2	2.0

Table 5: Classifier results for indoor environment.

12.1.4 *Outdoor Environment*

To verify the proposed algorithms, the data from the Radish robotics dataset repository was used. The dataset named *kenmore_pradoroof* submitted by Michael Bosse in May 2007. The dataset contains the laser range data collected from two SICK LMS lasers mounted on the roof of a Toyota Prado driving through suburban streets in Kenmore, QLD, Australia. The dataset does not contains the ground truth.

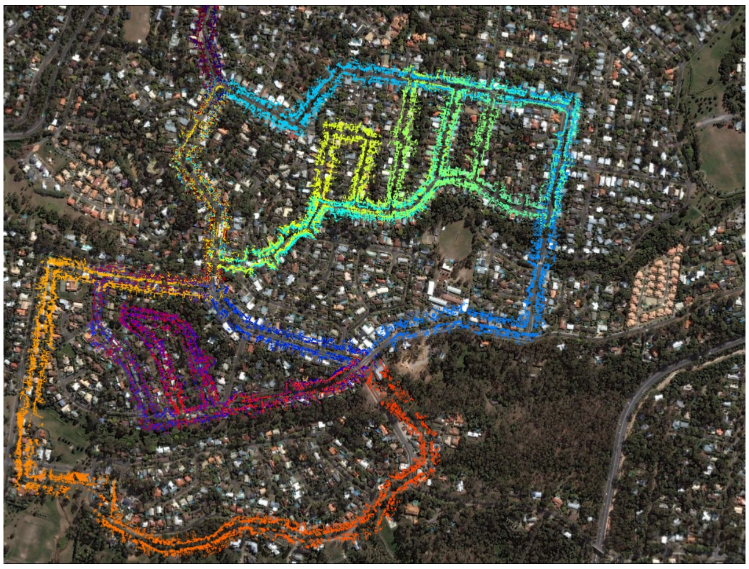
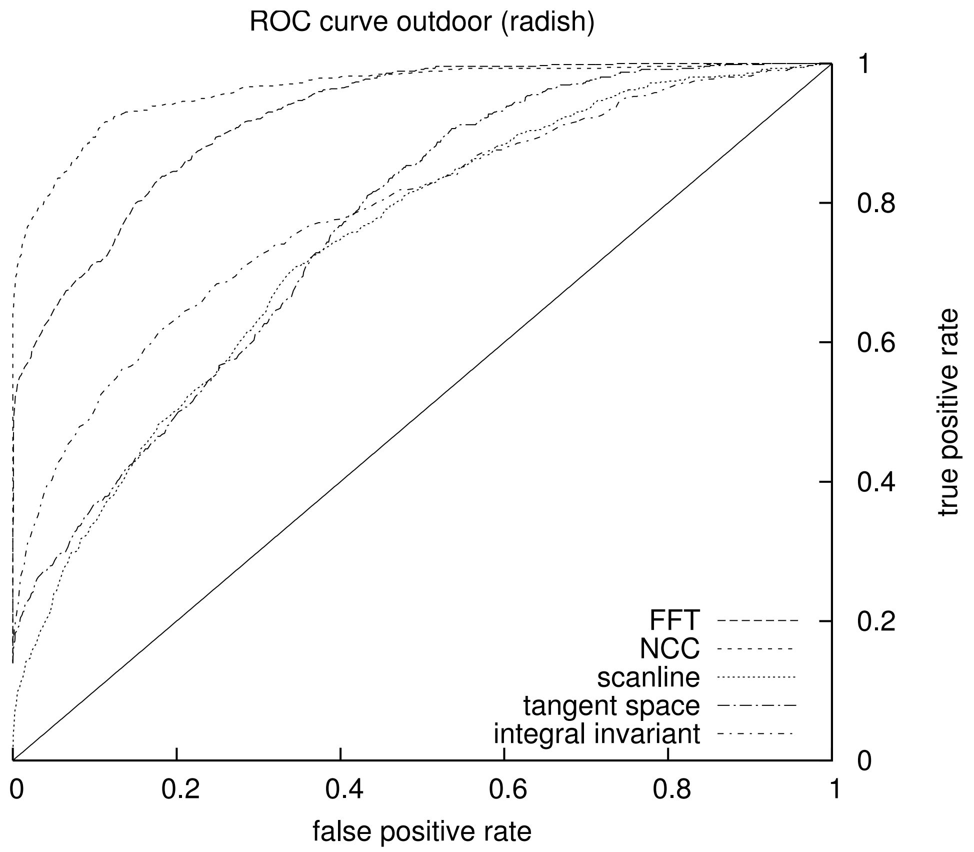


Figure 22: Map of outdoor environment.

Only first few hundreds of meters was used. At first, there were detected places using the algorithm 3. There exist multiple scans slightly displaced for each physical place, as the scans were taken in sequence. Therefore there exists multiple vertices for each physical places with slightly different descriptors. First 100 vertices was used to create the test set. Then these vertices were grouped according the timestamps in the dataset.

The test set contains 4332 negative and 718 positive examples. The ROC curve and parameters are summarized in Tab. 6.



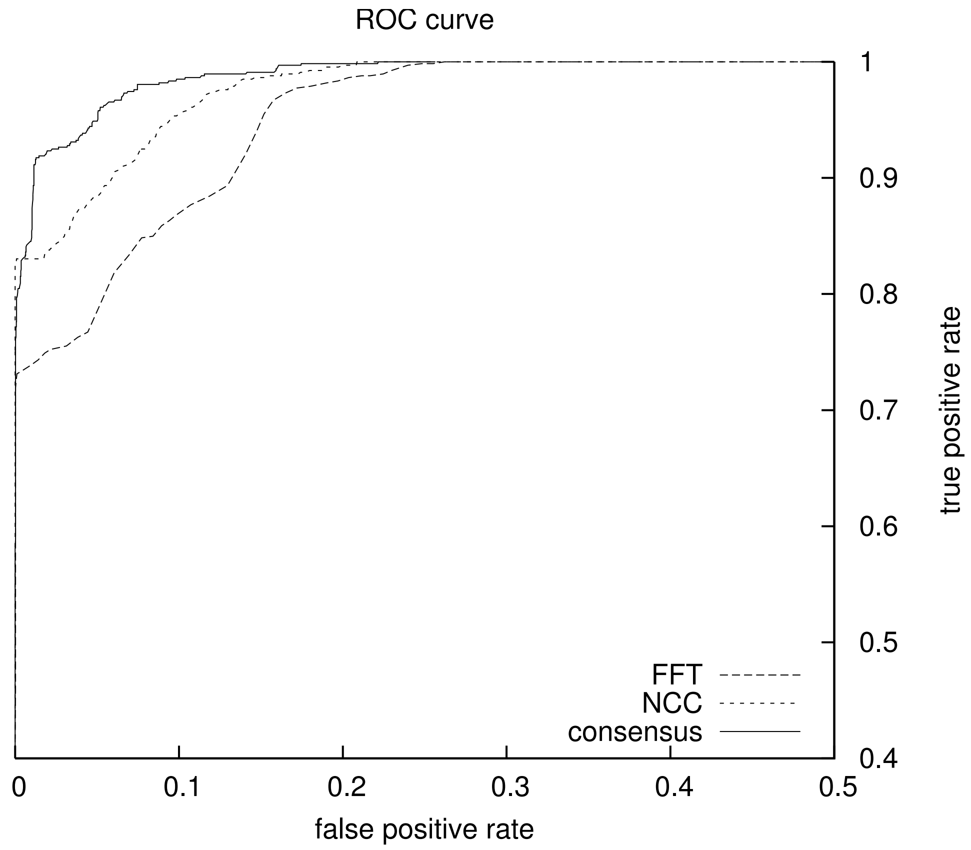
matching algorithm	AUC	ϑ	TP[%]	FP[%]
fft	0.922	0.917	83.8	18.2
ncc	0.963	0.784	91.5	11.1
scanline	0.732	0.856	70.9	34.6
tangent space	0.748	0.729	81.1	43.3
integral invariant	0.785	0.915	61.6	17.9

Table 6: Classifier results for outdoor environment.

12.1.5 *Fusing the Localizing Jockeys*

This section desrcibes the results of the fusing the localizing jockeys using the consensus operators. The results of the localizing jockeys are converted to the opinions according the equations described in sec 9.1. Then these opinions were fused with the consensus operator $\oplus$.

According the previous experiments described here, only the NCC and FFT localizing algorithms are used for the matching the places. These two algorithms have best performance and also are quicker than others. A similar improvement is achieved by fusing this two algorithms in all the tested environments. Therefore, here is presented only the result for the cave simulated environment. Only the left upper corner of the ROC diagram is displayed to improve readability. It can be seen that mainly the area under curve and false positive rate was improved in the table 7.



matching algorithm	AUC	ϑ	TP[%]	FP[%]
fft	0.976	0.964	97.5	16.1
ncc	0.989	0.723	95.4	8.9
consensus	0.993	0.554	93.6	1.2

Table 7: Classifier results for consensus operator.

12.2 NAVIGATION

To prove that the navigation jockeys works in real environments, set of tests were performed. Suitable navigating jockeys were chosen for specific environments.

12.2.1 *Visual Navigating Memory-less Jockey in Outdoor Environment*

For the verification of the performance of the reactive navigating jockey (GeNav), a rosarium at Kinsky garden[1] in Prague was chosen for having narrow and short paths and crossing abundance.

The experiment was performed with robotic platform Pioneer 3AT equipped with TCM2 compass. The robot was equipped with Fire i-400 camera providing 15 color images per second at 640x480 pixel resolution. The images were processed in real time by Intel Core 2 Duo notebook.

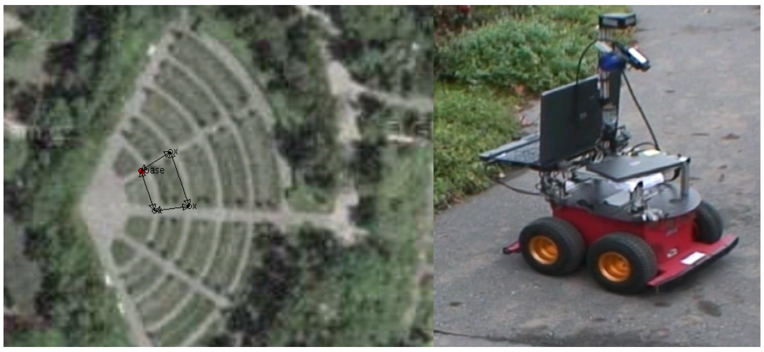


Figure 23: (a) Outdoor experiment map; (b) robotic platform

The navigating jockey was guided to the rightmost outgoing edge for every following step. The robot repeated this trajectory until the first error occurs. After the robot missed the crossing, it was placed at a starting place and restarted. The approximate success rate was 92%.

12.2.2 *Visual Navigating Memory-based Jockey in Outdoor Environment*

The experiment with the memory-based surfnav jockey was performed in the "Stromovka" park[2] in Prague. The park pathways were denoted by characters **A** to **W** (see 24). As a memory-based jockey needs to have a pre-learned map, the robot has been guided by human operator along them while LaMa *SURFNav learning jockey* was recording edges and vertices.

1 Kinsky gardens 50°4′53.579″N, 14°23′48.846″E
2 Kralovska obora, Stromovka 50°6′18.778″N, 14°25′33.395″E

The size of mapped area was approximately 400×300 m. While there was a need to guide the robot along each pathway in both directions, the total map length was ~ 5 km.

Experiments were performed by Pioneer 3AT robotic platform with TCM2 compass. Robot was equipped with Fire i-601c camera providing 30 images per second at 1024x768 pixel resolution. A wide angle objective with focus length 3.5 mm was used. Images were processed in real time by Intel Core 2 Duo notebook. Only the upper half of the picture was processed in order to use more distant objects as landmarks.

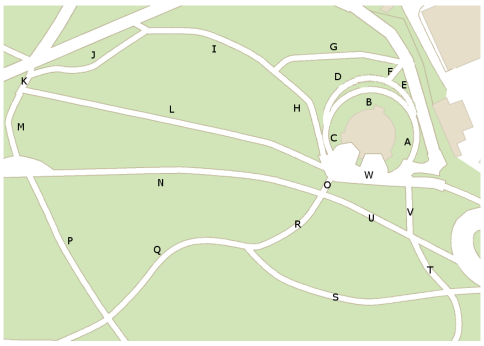


Figure 24: Park pathway map

Pathway sequence	length [m]
ABCW	250
ORQPMKJIH	850
ABCHGFDCW	500
HGFEAWOUVW	600
UTSROCDEBCW	700
Total	2 900

Table 8: Outdoor test path description

On the next day, five sequences of mapped pathways (see Table 8) forming a closed path were randomly chosen and the robot was switched to traverse them. One week later, we performed similar navigation tests on the same place with the "old" map. In all runs robot was able to finish desired path.

Precision of following the learned path was measured as distance between learned and actual robot positions and was physically measured just in "crossing" vertices. Learned positions were marked on the ground directly and the error was measured as distance between actual robot position when localized in vertex and related marker on the ground. The achieved (maximum) positioning error was less then $0.85m$ with mean error of $0.34m$.

12.3 EXPLORATION

The exploration and autonomous mapping of the previously unknown environment is a task, which checks all the components of the mapping framework. The experiments were performed in all three types of environments. First, the exploration using the marker was tested. In the next sections are described the experiments with the marker-less exploration algorithms, with the loop-closing procedure.

12.3.1 *Exploration with marker of Simulated World*

Because real-world testing is a time costly process, exploration behavior has been tested on a simulator. The robot behavior was simulated using MobileSim[3]. Synthetic camera images were automatically generated from a hand-drawn map of a part of Kinsky garden in Prague. In order to improve the realism of generated images, real-world textures were used and artificial noise was added.

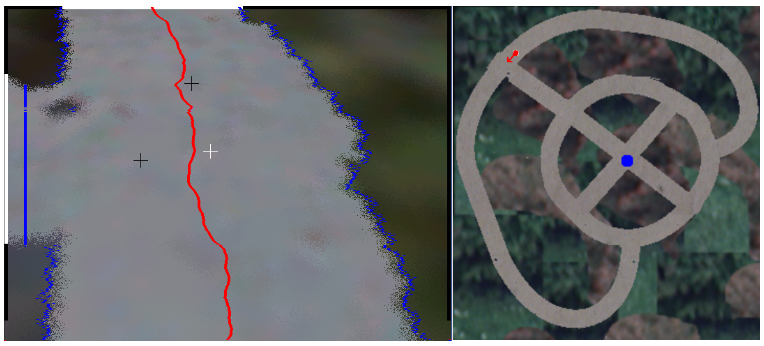


Figure 25: Generated view and textured map

The system was tested on four maps of various sizes (see figure 26). Ten test runs were performed for each map. Exploration time, number of failed exploration attempts and number of crossing passages were recorded (see table 9).

The topological exploration algorithm requires the system providing node information to be absolutely inerrant. Even that the GeNav system recognition success is approximately 98% (in simulated environment), exploration success rate drops fast with the increasing number of passed crossings.

12.3.2 *Exploration Strategies without Marker*

The experiments were performed in simulator on tree maps of various sizes. The simulated robot was equipped with two laser range-finders covering together the view of 360 degree. Five laser-based jockeys are involved in this experiment: *laloc* -

3 http://robots.mobilerobots.com/MobileSim/

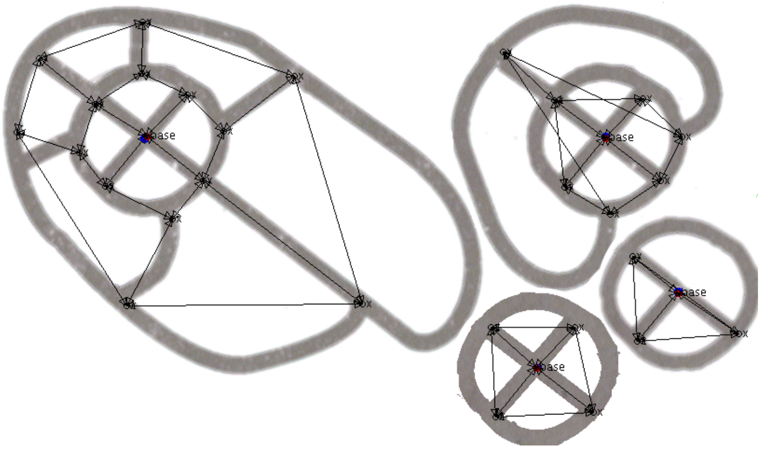


Figure 26: Explored maps

Map size (crossings)	Crossings traversed	Failures	Exploration Time (s)
Minimal (4)	11	0	365
Small (5)	14	0	418
Middle (8)	28	2	922
Large (14)	63.2	4	2283

Table 9: Simulation results

localizing jockey, *lanav* - memory-less (reactive) navigating jockey, *lalearn* - learning jockey, *lalearnav* - memory-based navigating jockey, and *odoloc* - localizing jockey using odometry.

The *lanav* jockey discovers new edges in the environment, during the navigation of these edges, the *lalearn* learns the edge for later backtracking. The *laloc* jockey is used for vertices description, localization, and loop closing. The *lalearnnav* is used for backtracking, when the robot needs to traverse to a vertex, where the path is already known.

Each map contains loops, which must be closed during the exploration. The *odoloc* jockey was not used in the exploration process directly but was used as an "oracle", which check the correctness of the loop closing process. If there was a mistake in the loop closing, this mistake was recorded and fixed according the "oracle".

Three exploration strategies were tested: depth-first search, breadth-first search and knowledge search. Depth-first search (DFS) is a greedy approach. The (topologically) nearest edge is chosen in each step. Breath-first search (BFS) searches systematically edges in oder given by order of vertex discovery. All outgoing edges from first vertex are explored, after them the outgoing edges from second vertex etc. The knowledge search (KNOW) is a sort of best-first search. The edge, which brings the most information is chosen for exploration. The reasoning process chooses such edge, which is expected to bring new knowledge of the structure, to confirm or refute the loop closure.

map	algorith	edges	total steps	backtrack	missed loops	errors
	BFS	15	29	7.3	1.3	1.7
Small (7)	DFS	14	18.3	1.7	0	0
	KNOW	16	31.6	14.3	0	0.6
	BFS	18	56	12	1.7	1.3
Middle (9)	DFS	19.6	30.3	6.3	1	1.7
	KNOW	21	47	24	0	2
	BFS	32	75	30	6	3
Large (15)	DFS	32.3	60.7	17.3	7.7	3.3
	KNOW	37.6	99	60.3	0	4.3

Table 10: Comparison of exploration strategies

The results are depicted in Table 10. For each map and exploration strategy is depicted the real number of edges in the resulting map, number of total steps needed to explore the whole map, number of steps made on already knows edges, number of missed loop closing corrected by the oracle, and number of corrections made by oracle when the loop was wrongly closed.

The DFS strategy finishes the exploration in the smallest number of steps as is expected. The BFS strategy is slightly worst than the DFS in the number of steps. On the other hand, BFS is slightly better, because the errors are mostly the missed loop closing, which are less fatal than wrongly closed loops. The KNOW strategy is the best in the number of mistakes, as the most information is gained in each step of the exploration. It is necessary to note that the KNOW strategy has reasoning procedure in background, because next step is computed by the reasoning procedure. The big number of steps (backtracking) is needed to verify all hypothesis in order, which does not take into account the distance to required edge.

Therefore, the combination of the BFS strategy with the reasoning procedure is used in following experiments.

12.3.3 *Exploration without Marker*

The experiments were performed in a simulated environment and with real robots in indoor and outdoor environments. During these experiments, robot performs an exploration task with loop-closing. Multiple jockeys were used and coordinated by the executor module.

12.3.4 *Simulation*

The laser-based localizing and navigating jockeys were tested in a simulator of Player/Stage system [27]. The used environment matches the pre-programmed description entitled the "Cave" in Player/Stage system. The simulated robot was equipped with two laser range-finders covering together the view of 360 degree. Four laser-based jockeys are involved in this experiment: *laloc* - localizing jockey, *lanav* - memory-less (reactive) navigating jockey, *lalearn* - learning jockey and *lalearnav* - memory-based navigating jockey.

The executor then coordinates afore listed jockeys. The *laloc* jockey discovers vertices and outgoing edges, also describes each vertex with actual laser scan. The *laloc* jockey also provides similarity measure of the vertices employing diverse computation methods described in Section 10.5.2. The *lanav* jockey is used to navigate the discovered unknown edges. Besides the *lanav* jockey is executed also the *lalearn* jockey. The *lalearn* jockey is learning actual traversed edge as well as the opposite one. The *lalearnav* jockey is used for traversing already navigated and learned edges. While the *lalearn* jockey learns also the edge opposite to actually traversed one, it is possible to use the *lalearn* and *lalearnav* for its' recovery in case of failure of the reactive (*lanav*) jockey.

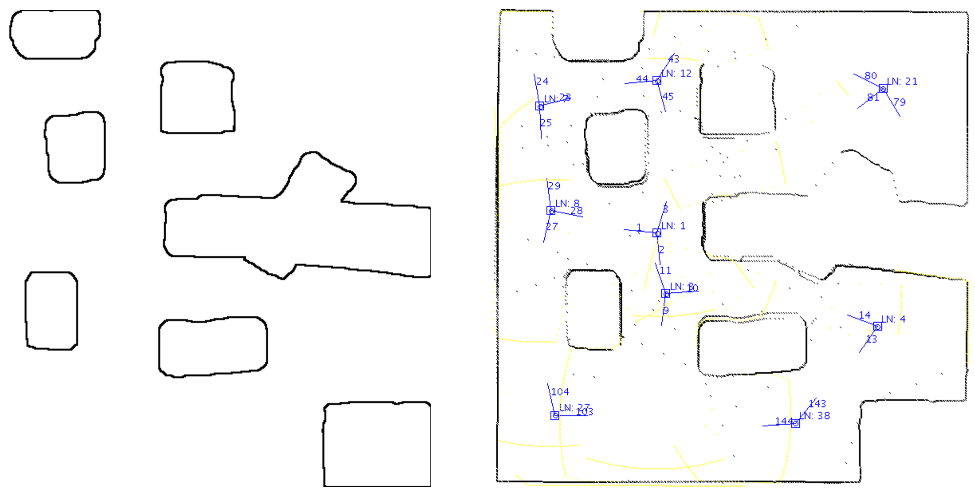
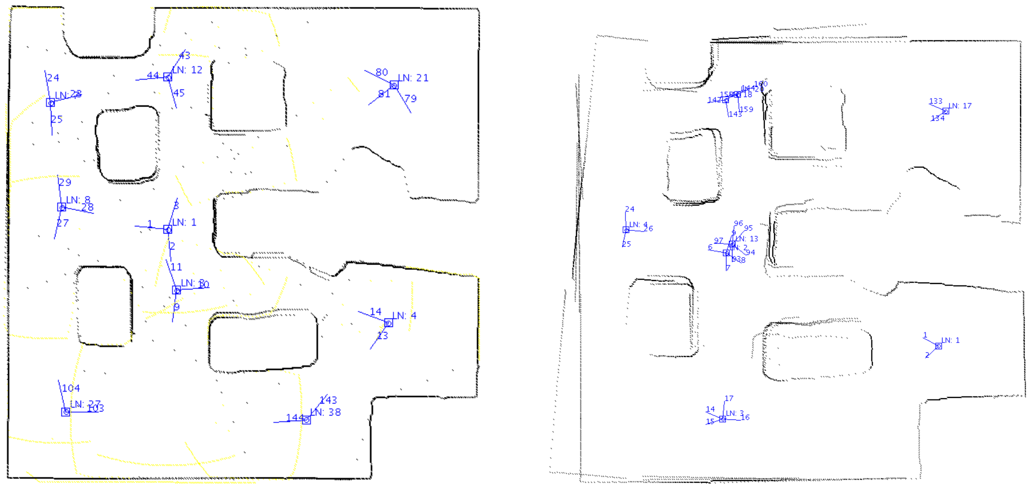

(a) Real environment (b) Resulting map

Figure 27: Simulated environment for exploration.

The reconstructed map depicted in Figure 28 has loops closed only by the similarity of the vertices. This is the best map acquired without employing the reasoning module for loop closing. There can be seen that some places are represented by multiple (two) vertices. This is caused by displacement in vertices positions and consequently smaller similarity in descriptors. One place is actually missing (see left bottom corner), what is caused by too high similarity with another place. The vertex position from the odometry is used only for the visualization purposes and is not used for exploration and loop-closing at all.



(a) with reasoning (b) without reasoning

Figure 28: Map gathered with reasoning (a) and without reasoning (a) used for loop-closing.

12.3.5 *Indoor environment*

The results of exploration of real indoor environment is described in this section. No loops are present in the experimental environment. Experiments were performed by the G2Bot robotic platform with two Sick LMS 200 laser range-finders in a configuration where the lasers provide whole 360° view.

The same setup of the jockeys as in the simulator was used for the real indoor environment. Also the same exploration algorithm were performed.

In this experiment only part of the space was mapped. The robot was able to navigate in the environment even in the narrow passages as can be seen in Figure 29a. The exploration stops after 26 steps and lasts 93 minutes at average.

The resulting map is depicted in Figure 29b. The odometry is used for displaying the proper position in the one frame of reference only.

12.3.6 *Outdoor environment*

The outdoor experiment was performed in park-like environment. In this experiment robot performs exploration task with loop-closing. The experiment was performed by Pioneer 3AT robotic platform with on-board sensors comprising magnetic TCM2 compass and Fire i-601c camera providing 1024*x*768 images at 30 FPS, a lens used with focus length 3.5 mm. Gathered images were processed in real time by Intel Core 2 Duo laptop PC.

Different jockeys were used in this experiment. Reactive memory-less navigating jockey GeNav keeps the robot on the pathways of the garden and recognizes the crossings. The GeNav localizing jockey provides the similarity measure of the

(a) Environment

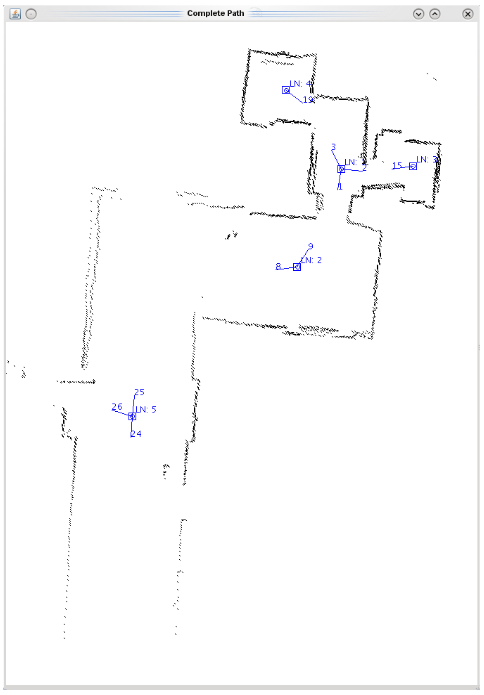

(b) Resulting map

Figure 29: Indoor environment

vertices based only on the number and angles of outgoing edges. This localization purposely causes strong sensor aliasing of the vertices to show ability of framework to utilize information stored in edges. Memory-based navigating jockey SurfNav learns the traversed edges using the SURF from pictures in learning mode. In navigating mode, SurfNav where able to traverse learned edges. There is also third mode of SurfNav module, used for comparison of the learned edge utilizing this ability in loop-closing procedure. The GPSLoc localizing jockey was used for visualization of the vertices and data were not used in the algorithm at all.

Robot traverses every outgoing edge driven by GeNav jockey while SurfNav jockey stores visual descriptors along this edge for later usage. The edge ends in next crossing discovered by GeNav. This new crossing is registered with all outgoing edges with corresponding angles. Robot then returns the same edge back to get the SURF descriptors in both directions and then continues with edge exploration.

New registered vertex is compared with all vertices existing in the map. The candidates are chosen according to the number (amount) and angles of the outgoing edges. In addition, all outgoing edges itself are compared to distinguish vertices. Then the robot starts to traverse each edge to compare this edge's descriptor with a stored one. While traversing the whole edge can be very time consuming, robot traverse only a specified section of the edge. Even the comparison is done on only partial description of the edge with stored complete edge, the method provides an estimation of the similarity of edge pair. In the case the edge for comparison is not known, still not explored, the loop closing is postponed until new relevant information is gained.

It has to be pointed out, that the used environment contains nearly all vertices almost undistinguishable and whole loop-closing algorithm must rely only on similarity of edges only. The experiment shows, that the edges gathered by SurfNav jockey are information-rich, and it was sufficient to compare first 10 cm of the edge to distinguish edge from each other.

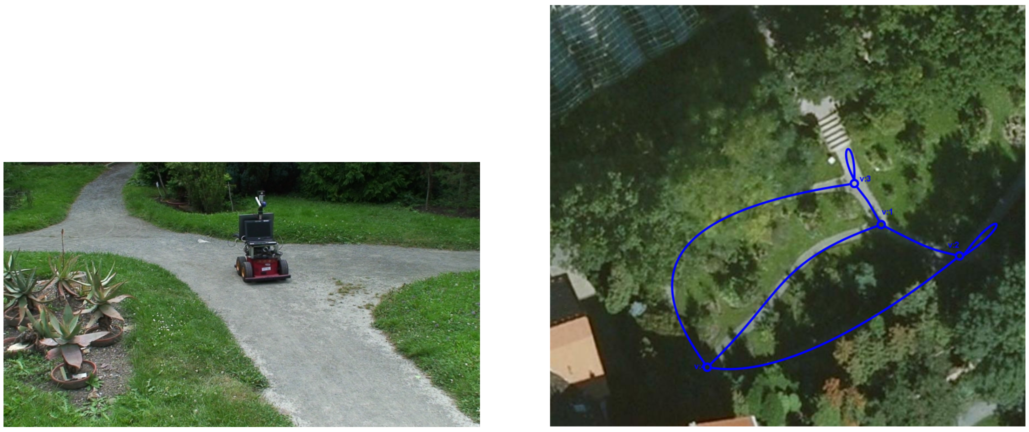

(a) Real environment (b) Resulting map

Figure 30: Outdoor environment - botanical garden, Albertov, Prague

The map of the botanical garden Albertov can be seen on Figure 30b. Position of the vertices where taken from GPS receiver, nevertheless GPS data was not used by the algorithm at all. The size of the working environment of the robot was reduced by an on-road obstacle. The robot considers these blocked paths as dead-ends and in the map are represents as self-loop.

The exploration stops after 36 steps and lasts for 103 minutes on average. Note, that all existing loops were correctly closed.

The created map was applied for navigation again one month later. The robot traversed about 500 meters using the given map and no significant decrease of navigating precision and robustness was observed although the environment did proceed with minor changes due to seasonal changes.

13 CONTRIBUTION AND FUTURE WORK

13.1 CONTRIBUTION

The following section recapitulates the contribution of the thesis. In comparison with existing topological maps, the proposed representation brings the following achievemnts:

- Paths in an environment are represented as edges in a map. This representation contains a procedural knowledge (how to traverse the edge) in contrast to previously used simple edge representation in the sense of connection or existence of route.

  The procedural knowledge representation is necessary to utilize various navigational algorithms not only reactive ones, but also the algorithms relaying on the previously learned information. The each edge can hold more than one procedure to navigate from the starting point to the destination. It allows to select the most reliable navigational algorithm or the most suitable for current conditions, state of the environment and the robot itself.
- The proposed modular concept of jockeys - algorithms for navigation, learning and localization allows to utilize multiple senors and algorithms concurrently. This approach allows to use a single map for representing heterogeneous environment consisting from a wide range of vertex and edge types.
- The concept of learning and memory-base navigating jockey brings a qualitatively improvement into the area of topological mapping. The flexibility and reactivity of the navigation method is required during the exploration of the environment and the map learning. The robot needs to move autonomously through the unknown environment; thus, the reactive navigation approaches are used.

  On the other hand, the repeatability and rigidity is requested during the navigation using the map. It is not possible to fulfill this antagonistic requirements in nowadays topological mapping system. To solve this problem in the proposed framework, the pairs of learning and memory-based navigating jockeys are introduced. The classical reactive navigation is used during the exploration of an unknown environment, but simultaneously one or more learning jockeys are running. These jockeys memorize the trajectory, and store the data necessary for traverse of the edge - the procedural knowledge.

Whenever there is a request to traverse already learned edge, the appropriate memory-based jockey is called. If there exists more than one procedural knowledge, the best navigating algorithm can be chosen according to a-priori computed or observed performance for a particular edge. The behavior of memory-based jockeys is deterministic, also they are able to recognize and report the failure in contrast to reactive memory-less navigating jockeys.

- The uncertainty representation using the subjective logic brings the power of the symbolic reasoning with the uncertain propositions. The advantage of the subjective logic is shown in the combining the information from different localizing modules into one opinion about the position of the robot. Also the proposed symbolic description of the environment properties and usage of this feature is shown in the loop-closing during the exploration. The used representation of uncertainty allows the extension of the representation by the non-deterministic behavior of navigation algorithms.

13.2 GOALS FULFILLMENT

This section compares the achievements of the work with the goals defined in Chapter 3.

1. *The goal was to perform a study of the currently used representations of the spatial knowledge.* The current state of the art in the human spatial knowledge representation is described in Chapter 4. The currently used representations in robotics are described in Chapter 5. Then follows (Chapter 6) the description of the exploration algorithms used in metric and topological maps. Chapter 7, the last chapter of the study, presents the overview of the formal logical systems, which are able to work with the uncertainty. The findings, arising from the study, are used in the proposition of the novel spatial representation, exploration and reasoning method.
2. *The goal was to propose as scalable probabiilistic representation of the space.* The proposed spatial representation is described in Chapter 8. Rich descriptions of the environmental elements - places and routes are mapped to the graph-like structure where places are stored as vertices and routes as edges. This structure is easy scalable from its nature. The richness of the environmental elements' description ensures that the map can describe wide variety of the environmental types concurrently. The modular structure of the map interfaces allows straightforward extension of the map representation. As experimental results in Chapter 12 shows, the same map representation is able to describe the indoor as well as outdoor environment depending on the choice of navigating and localizing algorithms.
3. *The goal was to propose a method of an autonomous exploration wothout necessity of the environment modification.* Proposed methods for autonomous exploration are described in Chapter 11. The first proposed exploration method does not meet the requirement not to modify environment, because it requires marking of the "base". However, the marking can be easily realized (any

object of distinctive color) and therefore it is not an issue. The second of the proposed exploration algorithms is fully marker-free and it is able to explore an unknown environment. Experimental results are shown in Section 12.3.

4. *The goal was to propose a method for a reasoning about uncertain spatial knowledge.* The method for working with uncertainty is described in Section 9. The reasoning with uncertainty is employed by fusing the results of the localizing jockeys and the loop-closing procedure. The results of the localizing jockeys fusion is depicted in Section 12.1.5. The loop-closing was used in the exploration experiments. The advantage of the loop-closing procedure is illustrated in Fig. 28.

5. *The goal was to implement and integrate the proposed methods into a unified mapping framework.* The implementation and integration is described in Chapter 10. The spatial knowledge representation, the reasoning algorithm, the navigating and localizing algorithms are integrated into the unified framework called Large Maps (LaMa). The proposal modular framework LaMa allows the integration of different algorithms working with different sensors and robotic platforms.

6. *The goal was to verify the proposed methods in realistic environments and conditions.* The verification of the proposed map representation, reasoning and exploration are described in Chapter 12. The verification were performed in the simulated, indoor and outdoor environments under the realistic condition. The localizing and navigating algorithms were firstly tested separately. Afterwards, the performance of the integrated framework was tested by the autonomous exploration task.

Beside these main goals, the localization algorithms are described in the section 10.5. From the experimental results (described in Section 12.1), it can be easy to see, that proposed proposed localizing algorithms based on FFT and cross-correlation outperforms the existing shape-matching algorithms working on polygons. The proposed methods exploit the properties of the range laser-scans and therefore are better for this application.

13.3 FUTURE WORK

The proposed representation may be further extended in following areas:

The proposed exploration algorithm based of the knowledge maximization will be further investigated. The experiment shows, that the choice of the next goal guided only by the expected knowledge gain is not sufficient. It is necessary to include the distance into the objective function as well.

Representation of the uncertainty may be used for the non-deterministic navigation strategies. According to this, the planning algorithm will be proposed, which will be able to take the non-deterministic behaviour into account. Such a representation is a first step for the next extension.

Mapping of the dynamic environment is a most challenging problem in the context of topological maps. The edges in the graph may be replaced with the multivalued opinion about the expected result of the navigation strategy. Also the vertices may be extended with the description of the possible changes in their descriptors. Moreover, the forgetting mechanism can be employed to deal with the situation, when some places or routes physically disappear from the environment.

Multiple robots may be used for the exploration of the environment. The usage of multiple robots improves the exploration not only quantitatively but also qualitatively. the verification of the hypotheses may be done with cooperation of two or more robots.

Also there is a possible usage of the group of heterogeneous robot during the navigation. There can exist parts of the environment, where is possible to navigate only using a specific sensors or abilities. These parts are inaccessible for the robot without these sensors or abilities. The robots may form the formation and combine their abilities and sensors to reach the previously inaccessible parts of environment.

14 CONCLUSION

The robotics is a wide multi-disciplinary field of science. This work addresses the problem of robotic mapping of an unknown large-scale environment. The proposed map representation is based on the cognitive theories of the human spatial knowledge. The methods for localization, navigation, reasoning and exploration together with the map proposed representation are incorporated into modular framework called Large Maps Framework (LaMa). This LaMa framework is considered as a knowledge base allowing to handle and utilize spatial knowledge of various environment in a unified way.

The experimental verification of the proposed methods shows, that the framework is able to handle indoor as well as outdoor environments in the scale of hundreds of meters and perfectly operate in them later on. The automated exploration of unknown environment can be seamlessly extend with a human assisted exploration. The performance of navigation is improved using the pairs of learning and memory-based navigating jockeys.

The established principles proposed in the thesis are successfully exploited in the European projects from *Symbiotic Evolutionary Robot Organisms* (Symbrion) founded by FET Proactive Intiative: pervasive adaptation and *Robotic Evolutionary Self-Programming and Self-Assembling Organisms* (Replicator) founded by Cognitive Systems, Interaction and Robotics. The partial implementation of the LaMa framework was previously used in the Robotour competition. Our team wins this competition in years 2008 and 2009.

BIBLIOGRAPHY

[1] H. Bay, T. Tuytelaars, and L. Gool. Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*, 2006. (Cited on page 83.)

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002. (Cited on page 91.)

[3] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *STOC*, pages 269–278, 1998. (Cited on page 42.)

[4] M. A. Bender and D. K. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proceedings of the 35th Symposium on Foundations of Computer Science (FOCS'94)*, pages 75–85, 1994. (Cited on page 43.)

[5] M. Berthold. Fuzzy logic. In M. Berthold and D. J. Hand, editors, *Intelligent Data Analysis*, pages 321–350. Springer Berlin Heidelberg, 2003. (Cited on page 46.)

[6] M. Bosse, P. Newman, J. Leonard, and S. Teller. An atlas framework for scalable mapping. In *in IEEE International Conference on Robotics and Automation*, pages 1899–1906, 2003. (Cited on page 40.)

[7] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *Int. J. Rob. Res.*, 27(6):667–691, 2008. (Cited on page 85.)

[8] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000. (Cited on page 43.)

[9] P. Buschka and R. Saffiotti. Some notes on the use of hybrid maps for mobile robots. In *In Proc. of the 8th Int. Conf. on Intelligent Autonomous Systems (IAS)*, pages 547–556, 2004. (Cited on page 39.)

[10] M. Cannon and T. Warnock. A shape descriptor based on the line scan transform, 2004. (Cited on page 93.)

[11] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125 – 137, April 2001. (Cited on pages 35 and 36.)

[12] E. Chown, S. Kaplan, and D. Kortenkamp. Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1):1–51, 1995. (Cited on pages 29 and 38.)

[13] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. (Cited on page 40.)

[14] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 2010. (Cited on page 40.)

[15] X. Deng, E. E. Milios, and A. Mirzaian. Robot map verification of a graph world. *J. Comb. Optim.*, 5(4):383–395, 2001. (Cited on page 43.)

[16] R. M. Downs and D. Stea. *Cognitive maps and spatial behavior: Process and products*, pages 8–26. 1973. (Cited on page 25.)

[17] D. Dubois, J. Lang, and H. Prade. Automated reasoning using possibilistic logic: Semantics, belief revision, and variable certainty weights. *IEEE Transactions on Knowledge and Data Engineering*, 6:64–71, 1994. (Cited on page 48.)

[18] D. Dubois and H. Prade. Necessity measures and the resolution principle. *Systems, Man and Cybernetics, IEEE Transactions on*, 17(3):474 –478, May 1987. (Cited on page 48.)

[19] T. Duckett and A. Saffiotti. Building globally consistent gridmaps from topologies. In *Proc. SYROCO'00, 6th Int. IFAC Symposium on Robot Control*. Elsevier, 2000. (Cited on page 40.)

[20] G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *IJCAI*, pages 1639–1647, 1993. (Cited on page 43.)

[21] G. Dudek, P. Freedman, and S. Hadjres. Mapping in unknown graph-like worlds. *Journal of Robotic Systems*, 13(8):539–559, 1996. (Cited on page 43.)

[22] G. Dudek, M. Jenkin, and E. Milios. A taxonomy of multirobot systems. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, 2002. (Cited on page 43.)

[23] G. Dudek, M. Jenkins, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotic and Automation*, 7(6):859–865, 1991. (Cited on page 42.)

[24] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In S. S. Iyengar and A. Elfes, editors, *Autonomous Mobile Robots: Perception, Mapping, and Navigation (Vol. 1)*, pages 60–70. IEEE Computer Society Press, Los Alamitos, CA, 1991. (Cited on page 35.)

[25] T. Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006. (Cited on page 89.)

[26] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *Gnu Scientific Library: Reference Manual*. Network Theory Ltd., Feb. 2003. (Cited on page 89.)

[27] B. Gerkey, R. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *11th International Conference*

on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 2003. (Cited on page 116.)

[28] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. Evaluating the Efficiency of Frontier-Based Exploration Strategies. In *Proc. of the Joint Conf. of the 41st Int. Symposium on Robotics and the 6th German Conference on Robotics*, pages 36–43, Munich, Germany, Jun. 2010. (Cited on page 42.)

[29] M. E. Jefferies, M. C. Cosgrove, J. T. Baker, and W.-K. Yeap. The correspondence problem in topological metric mapping - using absolute metric maps to close cycles. In *KES*, pages 232–239, 2004. (Cited on page 40.)

[30] M. E. Jefferies and W. K. Yeap. The utility of global representations in a cognitive map. In *Conference on Spatial Information Theory (COSIT)*, volume 2205, pages 233–246. Springer, 2001. (Cited on page 40.)

[31] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–212, 2001. (Cited on page 48.)

[32] A. Jøsang. The consensus operator for combining beliefs. *Artif. Intell.*, 141(1/2):157–170, 2002. (Cited on page 48.)

[33] A. Jøsang. Conditional reasoning with subjective logic. *Journal of Multiple-valued Logic and Soft Computing*, 15(1):5–38, 2008. (Cited on pages 48 and 64.)

[34] A. Jøsang and V. A. Bondi. Legal reasoning with subjective logic. *Artif. Intell. Law*, 8(4):289–315, 2000. (Cited on page 49.)

[35] A. Jøsang, J. Diaz, and M. Rifqi. Cumulative and averaging fusion of beliefs. *Information Fusion*, 11(2):192–200, 2010. (Cited on page 48.)

[36] A. Jøsang and D. McAnally. Multiplication and comultiplication of beliefs. *Int. J. Approx. Reasoning*, 38(1):19–51, 2005. (Cited on page 48.)

[37] S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Proceedings of the International Conference on Robotics and Automation, IEEE, 2001.*, pages 3594–3599, 2001. (Cited on page 42.)

[38] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009. (Cited on page 36.)

[39] D. Kortenkamp. *Cognitive maps for mobile robots: A representation for mapping and navigation*. PhD thesis, University of Michigan, 1993. (Cited on page 38.)

[40] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil. Simple Yet Stable Bearing-Only Navigation. *Journal of Field Robotics*, 27(5):511–533, September 2010. (Cited on page 82.)

[41] B. Kuipers, J. Modayil, P. Beeson, M. Macmahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *in IEEE Int. Conf. on Robotics & Automation*, pages 4845–4851, 2004. (Cited on page 39.)

[42] B. J. Kuipers. *Representig Knowledge of Large-Scale Space.* PhD thesis, MIT, Cambridge, Massachusetts, 1977. (Cited on pages 27 and 36.)

[43] B. J. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978. (Cited on page 27.)

[44] B. J. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000. (Cited on pages 7, 36, and 37.)

[45] B. J. Kuipers and Y. T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991. (Cited on page 37.)

[46] M. Kulich. *Lokalizace a tvorba modelu prostředí v inteligentní robotice (in Czech).* PhD thesis, CTU, Prague, 2003. (Cited on page 36.)

[47] M. Kulich, J. Faigl, and L. Přeučil. On distance utility in the exploration task. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2011. (Cited on page 42.)

[48] M. Kulich, R. Mázl, and L. Přeučil. Self-localization methods for autonomous mobile robots based on range scan-matching. In *Proceedings of Field and Service Robotics Conference*, pages 373–378, Helsinky, 2001. FSA-Finnish Society of Automation. (Cited on page 36.)

[49] M. Kulich, P. Štěpán, and L. Přeučil. Feature detection and map building using ranging sensors. In *Proceedings of the 1999/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, pages 201–206, Tokyo, Japan, 2001. The Institute of Electrical Engineers of Japan. (Cited on page 35.)

[50] W. Y. Lee. *Spatial Semantic Hierarchy for a Physical Mobile Robot.* PhD thesis, Department of Computer Sciences, The University of Texas, 1996. (Cited on page 38.)

[51] T. S. Levitt and B. Kuipers. Navigation and mapping in large scale space. *AI Magazine*, 9(2):25–43, 1988. (Cited on page 29.)

[52] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31:983–1001, 1998. (Cited on page 90.)

[53] K. Lynch. *The Image of the City*. The MIT Press, June 1960. (Cited on page 27.)

[54] S. Manay and B.-W. Hong. Integral invariants for shape matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1602–1618, 2006. Member-Cremers,, Daniel and Member-Yezzi,, Anthony J. and Member-Soatto,, Stefano. (Cited on page 91.)

[55] D. Marinakis and G. Dudek. Pure topological mapping in mobile robotics. *Robotics, IEEE Transactions on*, 26(6):1051 –1064, 2010. (Cited on page 43.)

[56] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* Henry Holt & Company, June 1983. (Cited on page 32.)

[57] M. J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transaction on Robotic and Automation*, 8(3):304–312, 1992. (Cited on page 36.)

[58] R. Mázl and L. Přeučil. Simultaneous localization and map building for intelligent mobile robotics. In *Proceedings of Workshop 2003*, pages 204–205. CTU, Prague, 2003. (Cited on page 35.)

[59] Y. Mei, Y. hsiang Lu, C. S. G. Lee, and Y. C. Hu. Energy-efficient mobile robot exploration. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2006. (Cited on page 42.)

[60] H. Moravec and A. Elfes. High resolution map from wide-angle sonar. In *Proceedings of the IEEE International Coference on Robotics and Automation*, pages 116–121, 1985. (Cited on page 35.)

[61] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71 – 87, 1986. (Cited on page 47.)

[62] J. O'Keefe and L. Nadel. *The hippocampus as a cognitive map / John O'Keefe and Lynn Nadel*. Clarendon Press ; Oxford University Press, Oxford : New York :, 1978. (Cited on page 26.)

[63] J. Piaget and B. Inhelder. *The child's conception of space*. W. W. Norton, New York, 1967. (Cited on pages 25 and 27.)

[64] L. Přeučil, P. Štěpán, M. Kulich, and R. Mázl. Towards environment modeling by autonomous mobile system. In *Knowledge and Technology Integration in Production and Services.*, pages 509–516, New York, 2002. Kluwer Academic / Plenum Publishers. (Cited on page 35.)

[65] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1340–1345, Nagoya, Japan, August 1997. Morgan Kaufmann. (Cited on page 44.)

[66] I. Rekleitis, R. Sim, G. Dudek, and E. Milios. Collaborative exploration for map construction. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2001. (Cited on page 44.)

[67] I. M. Rekleitis, G. Dudek, and E. E. Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, 2001. (Cited on page 44.)

[68] I. M. Rekleitis, V. Dujmovic, and G. Dudek. Efficient topological exploration. In *ICRA*, pages 676–681, 1999. (Cited on page 42.)

[69] E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152:47–104, 2004. (Cited on pages 36 and 38.)

[70] W. Rencken. Concurent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ International Coference on Intelligent Robots and Systems*, 1993. (Cited on page 35.)

[71] F. Savelli. Loop-closing and planarity in topological map-building. In *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1511–1517, 2004. (Cited on page 38.)

[72] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976. (Cited on pages 48 and 49.)

[73] S. Simhon and G. Dudek. A global topological map formed by local metric maps. In *IROS*, volume 3, pages 1708–1714, Victoria, Canada, October 1998. (Cited on page 39.)

[74] P. Smets. Imperfect information: Imprecision and uncertainty. In *Uncertainty Management in Information Systems*, pages 225–254. 1996. (Cited on page 45.)

[75] C. M. Smith and J. Leonard. A multiple hypothesis approach to concurrent mapping and localization for autonomous underwater vehicles. In *Proceedings of International Coference on Field and Sevice Robotics*. Springer-Verlag, 1997. (Cited on page 35.)

[76] A. Solanas and M. A. Garcia. coordianted multi-robot exploration through unsupervised clustering of unknown space. In *Proceedings of 2004 IEEE/RSJ International conference on Intelligent Robots and Systems*, pages 717–721, 2004. (Cited on page 44.)

[77] P. Štěpán. *Vnitřní reprezentace prostředí pro autonomní mobilní roboty (in Czech).* PhD thesis, CTU, Prague, 2001. (Cited on page 35.)

[78] M. Sugeno. Fuzzy measures and fuzzy integrals: a survey. In M. Gupta, G. Saridis, and B. Gains, editors, *Fuzzy automata and decision processes*, pages 89–102. North Holland, Amsterdam, 1977. (Cited on page 45.)

[79] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998. (Cited on page 39.)

[80] S. Thrun, J.-S. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *AAAI/IAAI*, pages 989–995, 1998. (Cited on page 39.)

[81] E. C. Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189–208, July 1948. (Cited on page 25.)

[82] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Combining topological and metric: A natural integration for simultaneous localization and map building. In *Proceedings of the Fourth European Workshop on Advanced Mobile Robots (Eurobot 2001)*, September 2001. (Cited on page 39.)

[83] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building:closing the loop with multi-hypotheses. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2002. (Cited on page 39.)

[84] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 2003(44):3–14, 2003. (Cited on page 39.)

[85] G. Wood. *Living dolls : a magical history of the quest for mechanical life*. Faber, London, 2002. (Cited on page 15.)

[86] B. Yamauchi. Frontier-based exploration using multiple robots. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 47–53, Minneapolis, Minnesota, United States, 1998. ACM Press. (Cited on page 41.)

[87] W. K. Yeap. Towards a computational theory of cognitive maps. *Artificial Intelligence*, 34(3):297 – 360, 1988. (Cited on page 32.)

[88] L. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965. (Cited on page 46.)

[89] L. Zadeh. Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, 1:3—28, 1978. (Cited on page 48.)

AUTHOR'S PUBLICATION RELATED TO THE THESIS

BOOK CHAPTER

- L. Přeučil, P. Štěpán, T. Krajník, K. Košnar, A.V. Rossum, and A.H. Salden. Cognitive World Modeling. In *Symbiotic Multi-Robot Organisms*, pages 165–183. Springer, Berlin, 2010. (17%).

JOURNAL

- T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil. Simple, Yet Stable Bearing-Only Navigation. *Journal of Field Robotics*, 27(5):511–533, September 2010. (5%).

CONFERENCES

- K. Košnar and L. Přeučil. Mapping Unknown Environment via Exploration by Group of Reactive Robots. In *Proceedings of Workshop 2006*, volume A, pages 154–155, Prague, 2006. CTU. (90%).
- K. Košnar, L. Přeučil, and P. Štěpán. Topological Multi-Robot Exploration. In *Proceedings of the IEEE Systems, Man and Cybernetics Society United Kingdom & Republic of Ireland Chapter 5th Conference on Advances in Cybernetic System*, pages 137–141, New York, 2006. IEEE - Systems, Man, and Cybernetics Society. (60%).
- K. Košnar, T. Krajník, and L. Přeučil. Visual Topological Mapping. In *European Robotics Symposium 2008*, pages 333–342, Heidelberg, 2008. Springer. (50%).
- K. Košnar, T. Krajník, V. Vonásek, and L. Přeučil. LaMa - Large Maps Framework. In *Proceedings of Workshop on Field Robotics, Civilian-European Robot Trial 2009*, pages 9–16, Oulu, 2009. University of Oulu. (40%).