Lehrstuhl für Steuerungs- und Regelungstechnik
Technische Universität München
Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss

Probabilistic Cognition for Autonomous Systems: Abstraction, Semantics and Knowledge

Ziyuan Liu

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 06.05.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 10.09.2014 angenommen.

Foreword

This dissertation summarizes my work as a research assistant at the Institute of Automatic Control Engineering (LSR), Technische Universität München and Corporate Technology (CT) of Siemens AG. It was financially supported by the Institute for Advanced Study (IAS), Technische Universität München, and CT of Siemens AG. I gratefully acknowledge this generous support.

First of all, I would like to express my gratitude to my supervisor, Dr. Georg von Wichert, for the opportunity and trust he gave me to start my research under his supervision. I thank him for his patient guidance and insightful advices which have greatly helped me during the last years. I thank him especially for the freedom that he has given me to let me grow and find my own way. I sincerely thank Prof. Dr.-Ing./Univ. Tokio Martin Buss for giving me the opportunity to conduct my research at LSR, which is undoubtedly one of the best established robotics institutes in the world. Additionally, I would like to thank Dr. Gisbert Lawitzky for giving me the chance to work with the "Robotic and Autonomous Systems" (RAS) team at the Siemens AG. This experience has enriched my life with great pleasure and will definitely have a long-lasting constructive influence on my career.

I am very grateful to the administrative support I received from the institute and my LSR-colleagues. Special thanks go to Larissa Schmid, Domenik Weilbach, Waltraud Werner, Wolfgang Jaschik and Dr. Dirk Wollherr.

Furthermore, I would like to give my grateful thanks to my colleagues at the RAS team of Siemens AG. I thank them all for providing me an enjoyable and open working environment and for the support that they have given me in various ways. I also thank them for the great time we spent together, for the fun we had, and for the ideas we shared. All these have enlightened my way of thinking and my perspective towards life. Special thanks go to Dong Chen, Michael Fiegert, Dr. Wendelin Feiten, Dr. Axel Rottmann and Dr. Kai Wurm for their technical supports.

The last and most important thanks go to my beloved family. Their unconditional love and support have been encouraging me through my whole life. No words can fully express my gratefulness to them. I wish them only the very best.

Munich, May 2014 Ziyuan Liu

This work is accomplished with the support of the Technische Universität München - Institute for Advanced Study, funded by the German Excellence Initiative.

IAS TUM Institute for Advanced Study

to my family

...

Abstract

In recent years, the performance of autonomous systems, such as intelligent robots, has been greatly improved. Powerful CPUs, bigger RAMs, new sensors and faster data transmission have made many applications possible which seemed to be unrealistic in the past. However, the performance of such systems tends to become quite limited, as soon as they leave their carefully engineered operating environments. Here, the primary challenge is to manage the complexity and uncertainty of the real world, which typically come from the following sources: the limited measurement accuracy and other limitations of the sensors, modelling errors and purposefully made simplifications in the system's internal representations, unobserved environment dynamics and random effects. On the other hand, people may ask, why humans can handle highly complex problems with relatively limited computational power (compared with computers). It is obvious that abstraction, semantics and knowledge together play an important role. Humans understand the world in abstract terms which are assigned with semantic meanings. Moreover, humans have necessary knowledge, based on which they can make inference given only partially available data.

In this dissertation, a knowledge-supervised MCMC (KSMCMC) sampling technique is developed to provide autonomous systems the ability to abstract and to infer based on given knowledge and data. KSMCMC is realized by combining Markov logic and data driven MCMC sampling. Based on Markov logic, task-specific context knowledge can be formulated as descriptive logic rules which help to better explain the data. Using data driven MCMC, samples can be efficiently drawn from unknown complex distributions. As a whole, KSMCMC is a new method of finding compact abstract model for input data by combining high-level knowledge processing with low-level data processing in a systematic way. We demonstrate the effectiveness of the proposed KSMCMC sampling technique in two typical tasks in the robotic domain: semantic mapping and scene analysis.

Using KSMCMC, a new system for the automatic generation of semantic maps from preprocessed sensor data is proposed. This system is realized in the form of a probabilistic generative model using a Bayesian formulation, in which the generated semantic maps are aligned with the preprocessed sensor observations that a robot made during an environment exploration and mapping stage. This introduces a bottom-up path into the approach by using data driven discriminative environment feature detectors to analyze the continuous noisy sensor observations. The proposed system differs from previous semantic mapping approaches that mostly use various classification methods in a pure bottom-up fashion to label either spatial regions or places based on context or that assign semantic labels directly to portions of the observations. Instead, the proposed system constructs a parametric, abstract, semantic and top-down representation of the domain under the consideration: a classical indoor environment containing several units of different types connected by doorways. The generated parametric abstract model of the perceived environment not only accurately represents the environment geometry, it also provides valuable abstract information. These maps are structured similarly to a scene graph and are well suited for higher level reasoning and communication purposes.

In addition, KSMCMC is used to build a system for modelling table-top scenes. The proposed system demonstrates a probabilistic approach to generate abstract scene graphs for table-top scenes using object pose estimation as input. This system explicitly makes use of task-specific context knowledge by defining this knowledge as descriptive logic rules in Markov logic. Integrating these with a probabilistic sensor model, maximum posterior estimation of the scene parameters is performed using KSMCMC. The proposed system is evaluated using real world scenes. Experimental results confirm that this system generates correct scene graphs which represent the perceived table-top scenes well. By reasoning in the defined Markov logic network, false estimates of the object poses and hidden objects of the perceived scenes are correctly inferred.

Zusammenfassung

In den letzten Jahren hat sich die Leistung von autonomen Systemen, beispielsweise von intelligenten Robotern, stark verbessert. Leistungsstarke Prozessoren, ausreichende Arbeitsspeicher, neue Sensoren und schnelle Datenübertragung haben viele Anwendungen ermöglicht, die in der Vergangenheit unrealistisch zu sein schienen. Die Flexibilität und Leistungsfähigkeit solcher Systeme wird dennoch ziemlich begrenzt sein, sobald sie ihre sorgfältig konstruierten Betriebsumgebungen verlassen. Hier liegt die primäre Herausforderung in der Komplexität und Unsicherheit der realen Welt, die in der Regel folgende Ursachen haben: die begrenzte Messgenauigkeit und andere Beschränkungen der Sensoren, Modellierungsfehler und Vereinfachungen der Systemrepresentation, unbeobachtete Umgebungsdynamik und zufällige Effekte. Auf der anderen Seite stellt sich die Frage, warum Menschen hochkomplexe Probleme mit relativ begrenzter Rechenleistung (verglichen mit Rechnern) lösen können. Es ist offensichtlich, dass Abstraktion, Semantik und Wissen gemeinsam eine wichtige Rolle spielen. Die Menschen verstehen die Welt in abstrakten Begriffen, denen semantische Bedeutungen zugeordnet sind. Darüber hinaus verfügen Menschen über das notwendige Hintergrundwissen, auf dessen Grundlage sie auch bei teilweise unvollständiger Datenlage Schlußfolgerungen machen können.

In dieser Arbeit wird eine Knowledge-Supervised-MCMC (KSMCMC) Samplingtechnik entwickelt, um autonomen Systemen die Fähigkeit des Abstrahierens und Schließens zu verleihen. KSMCMC ist auf der Basis der Kombination von Markov-Logic und datengetriebenem MCMC-Sampling realisiert. Mithilfe von Markov-Logic kann aufgabenspezifisches Kontextwissen in der Form deklarativer logischer Regeln formuliert werden, mit deren Hilfe die einlaufenden Sensordaten besser erklärt werden können. Mit dem datengetriebenen MCMC-Sampling können effizient Stichproben aus unbekannten komplexen Verteilungen gezogen werden. Als Ganzes ist KSMCMC eine neue Methode, die systematisch die High-Level-Wissensverarbeitung mit der Low-Level-Datenverarbeitung kombiniert, um ein kompaktes abstraktes Modell für die Eingangsdaten zu finden. Wir demonstrieren die entwickelte Methodik und die damit zu erzielenden Ergebnisse anhand zweier klassischer Robotikaufgabenstellungen: semantisches Mapping und Szenenanalyse.

Mithilfe von KSMCMC schlagen wir ein neues System zur automatischen Generierung der semantischen Karten aus vorverarbeiteten Sensordaten vor. Dieses System ist in der Form eines probabilistischen generativen Modells mithilfe einer Bayes-Formulierung realisiert, in der die erzeugten semantischen Karten mit den vorverarbeiteten Sensorbeobachtungen evaluiert werden. Das System unterscheidet sich von früheren Ansätzen, die das Problem des semantischen Mappings meist in einem reinen Bottom-Up-Stil durch semantische Annotation räumlicher Strukturen mithilfe der verschiedenensten Klassifikationsmethoden lösen. Stattdessen baut das vorgeschlagene System eine abstrakte, semantische und Top-Down-Darstellung der Umgebung auf, die explizit das verfügbare Kontextwissen berücksichtigt: es handelt sich um eine klassische Innenraumumgebung, die sich aus verschiedenen Typen von Räumen zusammensetzt, für deren Komposition bestimmte Regeln gelten. Das erzeugte parametrische und abstrakte Modell der wahrgenommenen Umgebung repräsentiert die Umgebungsgeometrie sehr genau. Außerdem bietet es auch wertvolle abstrakte Informationen über die Umgebung. Diese erzeugten semantischen Karten werden ähnlich wie ein Szenengraph aufgebaut und sind hervorragend für eine Weiterverarbeitung auf höherer Ebene geeignet.

Außerdem verwenden wir KSMCMC um ein System für die Modellierung von Table-Top-Szenen zu bauen. Das vorgeschlagene System verwendet einen probabilistischen Ansatz zur Modellierung von Table-Top-Szenen und erzeugt abstrakte Szenengraphen mit der Objektlageschätzung als Eingabe. Dieses System verwendet ebenso explizit formuliertes, aufgabenspezifisches Vorwissen, das deklarativ über logische Regeln in Markov-Logic definiert wird. Mithilfe eines probabilistischen Sensormodells wird eine Maximum-A-Posteriori Schätzung der Szenenparameter unter Verwendung von KSMCMC durchgeführt. Das vorgeschlagene System wird anhand von Experimenten mit realen Szenen evaluiert. Die eperimentellen Ergebnisse bestätigen, dass dieses System sinnvolle Szenengraphen erzeugt, die die sensorisch wahrgenommenen Table-Top-Szenen korrekt erklären. Mithilfe des resultierenden Markov-Logic-Netzwerks werden fehlerhafte Lagehypothesen der Objekte verworfen und die Existenz versteckter Objekte in der wahrgenommenen Szenen richtig erschlossen.

Contents

Abbreviations

AI	Artificial intelligence
BI	Bayesian inference
BLN	Bayesian logic network
CPR	Cell prediction rate
CPU	Central processing unit
DDMCMC	Data driven Markov chain Monte Carlo
DESIRE	Deutsche Servicerobotik Initiative
DOF	Degree of freedom
DL	Description logic
FOL	First order logic
HRI	Human robot interaction
KP	Knowledge processing
KSMCMC	Knowledge-supervised Markov chain Monte Carlo
LUT	Look-up table
MCMC	Markov chain Monte Carlo
MH	Metropolis-Hastings
MLN	Markov logic network
MN	Markov network
OBB	Oriented bounding box
OGM	Occupancy grid map
OWL	Web ontology language
PCL	Point cloud library
PF	Particle filtering
PR2	Personal robot 2
PW	Possible world
RAM	Random access memory
RANSAC	Random sample consensus
RGBD	Red green blue depth
ROS	Robot operating system
SGVG	Saturated generalized Voronoi graph
SIFT	Scale-invariant feature transform
SL	Semantic labelling
SLAM	Simultaneous localization and mapping

1 Introduction

In recent years, the performance of autonomous systems, such as intelligent robots, has been greatly improved. Powerful CPUs, bigger RAMs, new sensors and faster data transmission have made many applications possible which seemed to be unrealistic in the past. However, the performance of such systems tends to become quite limited, as soon as they leave their carefully engineered operating environments. Here, the primary challenge is to manage the complexity and uncertainty of the real world, which typically come from the following sources: the limited measurement accuracy and other limitations of the sensors, modelling errors and purposefully made simplifications in the system's internal representations, unobserved environment dynamics and random effects.

On the other hand, people may ask, why humans can handle highly complex problems with relatively limited computational power (compared with computers). It is obvious that abstraction, semantics and knowledge together play an important role. Humans understand the world in abstract terms which are assigned with semantic meanings. Moreover, humans have necessary knowledge, based on which they can make inference given only partially available data. For instance, if a person sees a desk in an office room, instead of memorizing the world coordinates of all the surface points of the desk, this person will only notice that there is an object "desk" at a certain position, and even this position is probably described in abstract terms like "beside the window" or "near to the door". According to experience (learned knowledge), this person can make some reasonable assumptions, such as there could be some "books" in the "drawer" of the desk, instead of some "shoes" being inside, without opening the drawer. An example of human level inference is depicted in Fig. 1.1.

The use of abstraction, semantics and knowledge allows to define the system behaviour on higher levels and independently of the exact setting of the environment and the exact sensor readings. This abstractly defined behaviour is applicable to a wider range of situations and thus increases the overall robustness of the system. In our work, we aim to provide autonomous systems the ability to abstract and to infer based on techniques of knowledge processing and data processing so that they can better handle the complexity and uncertainty of the real world.

1.1 Knowledge Processing

Knowledge processing is mainly concerned with representing information about the world in a form that a computer system can utilize to solve complex tasks. The origin of knowledge processing (also known as knowledge representation and reasoning) dates back to

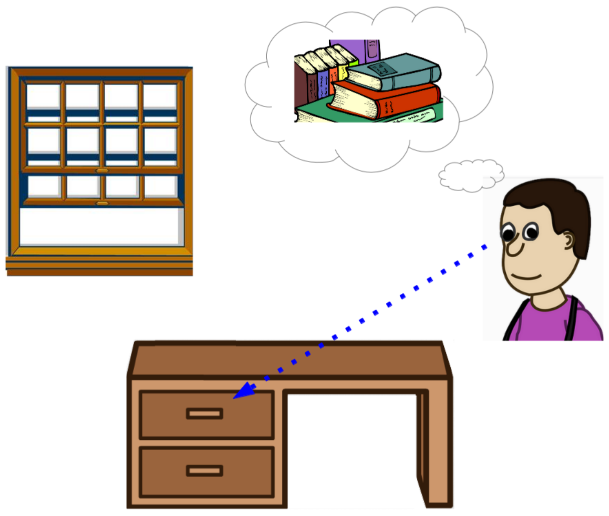


Fig. 1.1: An example of human level inference.

the 1950s, as several AI pioneers, such as Newell and Simon, attempted to make general purpose problem solvers that could in theory solve any problem. The system proposed by Simon et al. was called General Problem Solver (GPS) [99]. As its name, this system was created as a universal solver machine with the hope that it should solve all formalized symbolic problems. GPS was implemented in the information processing language (IPL) [100] and was the first of its kind which separated the knowledge of problems from the solving strategy. To solve a specific problem, GPS would decompose this problem into several sub-problems and construct strategies to solve each of these sub-problems. GPS could indeed solve several simple problems, but it failed to tackle more complex ones due to the explosion of state space. Despite the limited applicability, GPS was undoubtedly an important step towards advanced knowledge processing systems.

Through the successful development over the last decades, knowledge processing has found its applications in more fields: social network analysis [149, 18], expert system [32, 55], data mining [112, 95], business process management [62, 54], search engine [94, 159] and so on. In recent years, knowledge processing has drawn tremendous interests in the robotics community as well. Several knowledge representation systems have been proposed to solve robotic tasks. Tenorth [136] proposed a knowledge representation and inference system for robot manipulation tasks. This system provides robots necessary information about the objects and environments that they interact with. In principle, this system combines different information sources into a big knowledge base, and these sources include: encyclopedic and common-sense knowledge, such as instructions from the internet; knowledge about objects and environment models, such as types of objects and semantic representation of the environment; knowledge about robot actions and processes, such as properties and effects of the actions that robots can perform; and knowledge about robots own capabilities, such as dependencies and inter-dependencies on capabilities and components that are essential for certain robot actions. Robot control decisions can be made through inference in the defined knowledge base.

1.2 Data Processing

According to [31], data processing refers to the collection and manipulation of items of data to produce meaningful information. Only after the data has been processed and assigned certain meanings, it becomes useful information. The form of data is very diverse, and it could be numbers, text, images, sound, and so forth. Other than knowledge processing, data processing handles data directly in the continuous domain.

One of the earliest computerized data processing systems was used by the Census Bureau of Unites States in the 1950s. This system made limited use of electronic computers and was adopted for the 1950 United States Census [141]. Nowadays, data processing has found its successful applications throughout all areas of our society, and it involves many aspects. Some notable examples of these aspects include:

- Data Validation: this process checks for correctness, meaningfulness, and security of the input data using standardised validation methods. Examples of this process are code validation [63] and spelling check [82].
- Data Sorting: this process has two fundamental functions. It either arranges items in a certain sequence, or it groups items with similar properties together. An example of this process is the RANSAC algorithm [28].
- Data Retrieval: this process extracts the desired data from a database which typically contains a very large amount of data. An example of this process can be found in [24].
- Data Aggregation and Fusion: this process combines multiple pieces of data together which could come at different time steps. An example of this process is the SLAM process [50] which aggregates laser data that are obtained over time to generate a consistent representation of the perceived environment.
- Data Modelling: this process defines and analyses data features, and finally creates a model for the input data. Examples of this process can be found in [123].

1.3 Challenges

Although it would be very helpful to combine knowledge processing with data processing, this is yet not easy to realize, because they are principally in two different domains which are the symbolic domain and the continuous domain.

Knowledge processing is one of the core research areas in the field of artificial intelligence (AI). In principle, knowledge processing belongs to the symbolic domain, where entities, attributes, formalisms and other components are defined based on meaningful symbols which do not necessarily have direct associations with data in the continuous domain. This fact is well reflected in the formalisms that are typically used to build a knowledge base. These formalisms include “semantic networks” and several “logic-based formalisms”.

Semantic networks [128] use a directed or undirected graph to represent knowledge in patterns of interconnected nodes and edges. In the graph, nodes and edges indicate the involved concepts and their relations. In general, semantic networks provide a declarative graphic formulation for knowledge representation that supports reasoning about the represented knowledge using automated systems. Semantic networks have several common versions which share similar graphic structure but have their own emphasis and features, such as, definitional networks [155], executable networks [92] and learning networks [80]. An example of knowledge representation using semantic networks is WordNet [93] which is a lexical database for the English language.

Logic-based formalisms include description logic [7], propositional logic [69] and predicate logic [146]. These formalisms share a similar syntax and are formal systems for knowledge representation. In general, they model logical statements as "axioms", and in this way, the underlying symbols and relations are joined together. Description logic is usually used to model ontologies [48] describing knowledge as a hierarchy of concepts within a domain, using a shared vocabulary to denote the types, properties and interrelationships of those concepts [43]. Predicate logic has several variants: first order logic [126], higher order logic [78] and their extensions, such as Markov logic [117] and Bayesian logic [68]. In predicate logic, "predicates" are used to model features of entities that are represented by symbols. Logical connectives and quantifiers link the defined predicates together to formulate knowledge as descriptive rules in the form of logic formulas.

According to the formalisms of knowledge representation, it is obvious that knowledge processing happens on the symbolic and abstract level and does not necessarily associate with data sets in the first place. This is totally different from data processing which intends to extract valuable information directly from certain amount of data that could be very noisy or even chaotic. Although both of knowledge processing and data processing have many successful applications in their own domain, the combination of both in a systematic and theoretically sound manner is rarely seen. The challenge lies in the lack of a well-defined interface that connects the two domains in such a way, that knowledge processing and data processing can fully deploy their own strength and work together as a consistent unity.

1.4 Contributions

In this dissertation, a knowledge-supervised MCMC (KSMCMC) sampling technique is developed to provide autonomous systems the ability to abstract and to infer based on given knowledge and data. KSMCMC is a combination of Markov logic [117] and data-driven MCMC sampling [161]. Based on Markov logic, task-specific context knowledge can be formulated as descriptive logic rules. These rules define the system behaviour on higher levels and can be processed by modern knowledge reasoning techniques. Using data-driven MCMC, samples can be efficiently drawn from unknown complex distributions. As a whole, KSMCMC is a new method of fitting abstract semantic models to input data by

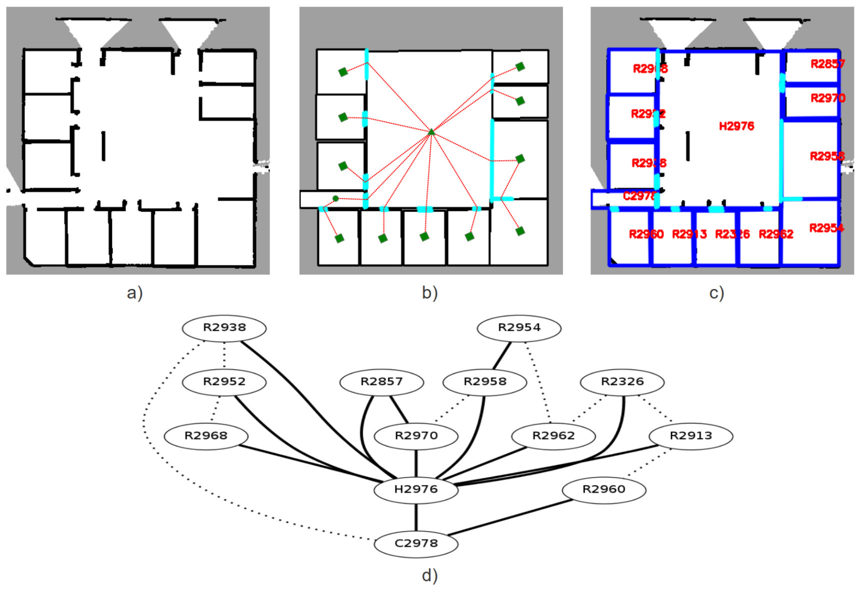
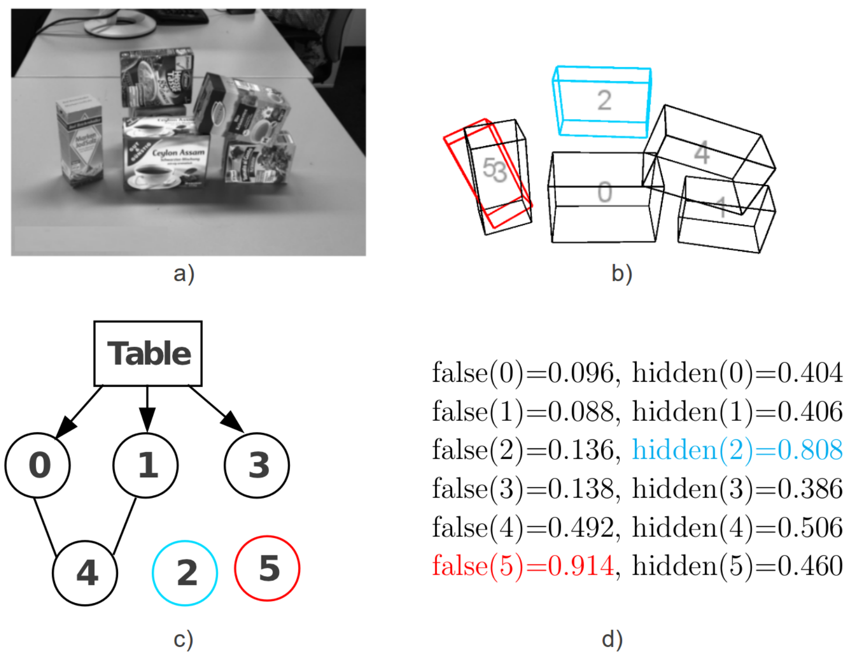


Fig. 1.2: a) The occupancy grid for the perceived environment generated by a SLAM process. b) The parametric semantic map generated by the proposed system. Small triangles, circles and rectangles show the geometric centres of halls, corridors and rooms. The environment topology is shown by red dashed lines. c) A direct comparison between the occupancy grid and the generated semantic map. d) The abstract representation of the semantic map. Nodes represent space units, with "R", "C" and "H" indicating rooms, corridors and halls. Solid edges indicate connection by doors (drawn in cyan in figure c), and dashed edges indicate connection by walls (drawn in blue in figure c).

combining high-level knowledge processing with low-level data processing in a probabilistic and systematic way. We demonstrate the effectiveness of the proposed KSMCMC sampling technique in two typical tasks in the robotic domain: semantic mapping and scene analysis.

Using KSMCMC, a new system for the automatic generation of semantic maps from preprocessed sensor data is proposed. This system is realized in the form of a probabilistic generative model using a Bayesian formulation, in which the generated semantic maps are aligned with the preprocessed sensor observations that a robot made during an environment exploration and mapping stage. By defining context knowledge in Markov logic and reasoning in the defined knowledge base, the whole sampling process is guided to converge towards the environment representations that comply with the defined knowledge and match the data well at the same time. Using discriminative feature detectors, the noisy sensor observations are analysed to provide proposals for the sampling process. This introduces a bottom-up path into the sampling process and increases the convergence speed. The proposed system differs from previous semantic mapping approaches that mostly use various classification methods in a pure bottom-up fashion to label either spatial regions or places based on context or that assign semantic labels directly to portions of the observations. Instead, the proposed system constructs a parametric, abstract, semantic and



false(0)=0.096, hidden(0)=0.404
false(1)=0.088, hidden(1)=0.406
false(2)=0.136, hidden(2)=0.808
false(3)=0.138, hidden(3)=0.386
false(4)=0.492, hidden(4)=0.506
false(5)=0.914, hidden(5)=0.460

Fig. 1.3: a) The input stereo image (only left image is shown). b) Estimated 6D poses. c) Scene graph generated by the proposed system. Arrows indicate the relation of stable support (existence of support surface). Undirected lines indicate the relation of contact. False estimates and objects implying hidden objects are highlighted in red and cyan respectively. d) Query probabilities on false estimates and hidden objects.

top-down representation of the domain under the consideration: a classical indoor environment containing several units of different types connected by doorways. The generated parametric abstract model of the perceived environment not only accurately represents the environment geometry, it also provides valuable abstract information. These maps are structured similarly to a scene graph and are well suited for higher level reasoning and communication purposes. An example of the proposed system is illustrated in Fig. 1.2.

In addition, the KSMCMC sampling technique is used to build a system for modelling table-top scenes. The proposed system employs a probabilistic approach to generate abstract scene graphs for table-top scenes using 6D object pose estimation as input. This system explicitly makes use of context knowledge that describes how such table-top scenes could be constructed. This knowledge is defined as descriptive logic rules in Markov logic and is used to calculate the probability of scene graphs. Combining the probability of scene graphs with a probabilistic sensor model, maximum posterior estimation of the scene parameters is performed using KSMCMC. The proposed system is evaluated using real world scenes. Experimental results confirm that this system generates correct scene graphs which well represent the perceived table-top scenes. These scene graphs explain the composition of the observed scenes correctly and provide valuable semantic information on the inter-object relations which is very useful for robot manipulation tasks. By reasoning in the defined Markov logic network, false estimates of the object poses and hidden objects of the perceived scenes are correctly inferred. An example of this system is shown in Fig. 1.3.

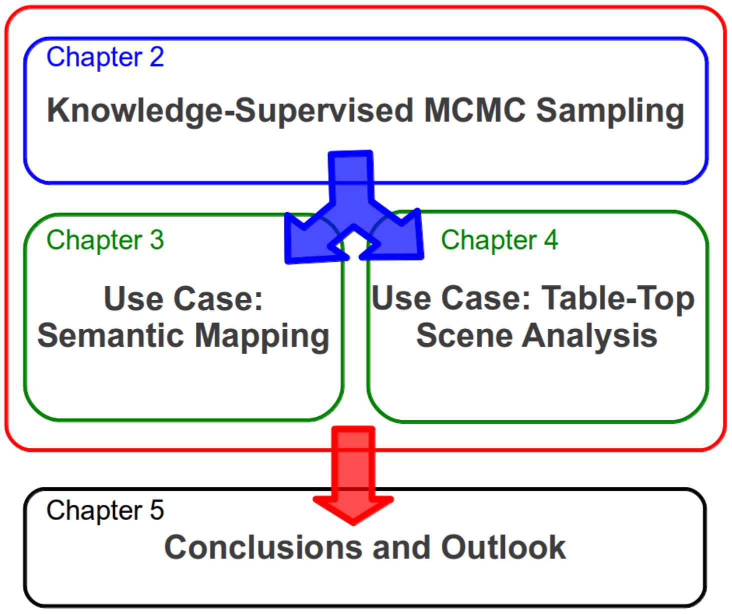

Chapter 2
Knowledge-Supervised MCMC Sampling
Chapter 3
Use Case: Semantic Mapping
Chapter 4
Use Case: Table-Top Scene Analysis
Chapter 5
Conclusions and Outlook

Fig. 1.4: Outline of the dissertation.

1.5 Outline of the Dissertation

As depicted in Fig. 1.4, the remainder of this dissertation is structured as follows: in chapter 2, the theory of the proposed knowledge-supervised MCMC sampling technique is explained. We introduce the mathematical background of this method and discuss the role of abstraction, semantics and knowledge. In chapter 3, the KSMCMC sampling technique is used to solve the problem of semantic mapping. Based on KSMCMC, a new system for generating semantic maps from occupancy grids is proposed. We elaborate on the realization of this system and evaluate its performance through experiments. In chapter 4, KSMCMC is employed to tackle another typical challenge in the robotic domain, which is table-top scene analysis. Using KSMCMC, abstract scene graphs are generated for table-top scenes. The proposed system is evaluated in various experiments. In chapter 5, we provide conclusions and give an outlook.

2 Knowledge-Supervised MCMC Sampling

In this chapter, we introduce a new sampling technique called knowledge-supervised MCMC (KSMCMC). We propose to realize this KSMCMC sampling technique by combining Markov logic and data-driven MCMC sampling, because the former is a powerful tool for modelling uncertain knowledge and the latter provides an efficient way of drawing samples from unknown complex distributions. Based on Markov logic, task-specific context knowledge can be formulated as descriptive logic rules. These rules define the system behaviour on higher levels, regardless of the exact setting of the environment and the exact sensor readings. This abstractly defined behaviour is applicable to a wider range of situations and thus increases the overall robustness of the system. As a whole, KSMCMC is a new method of fitting abstract semantic models to input data by combining high-level knowledge processing with low-level data processing in a probabilistic and systematic way. Within the framework of KSMCMC, knowledge processing and data processing can fully deploy their own strength and work together as a consistent unity.

The remainder of this chapter is structured as follows: in section 2.1, an overview is provided on commonly used context knowledge. We focus mainly on the approaches that use context knowledge to analyse and process data. In section 2.2, we introduce some background knowledge which is essential for understanding the concept of Markov logic networks. In section 2.3, we elaborate on the theory of KSMCMC and explain how to apply this technique to solve problems.

2.1 An Overview on Context Knowledge

So far, many approaches have attempted to adopt context knowledge (information) for data processing. Most of them code such knowledge directly into the continuous domain and do not take the symbolic domain into account. A common way to use context knowledge is to formulate it as certain constraints or boundary conditions so as to restrict the system states to a reasonable small set. This way of utilizing context knowledge can not make the best use of knowledge processing, because the power and the flexibility of knowledge representation and reasoning that exist in the symbolic domain are abandoned in this case. In the following, we provide a systematic overview on the commonly used context knowledge.

Galleguillos and Belongie [34] provided a survey on context-based object categorization. They addressed the problem of incorporating different types of contextual information for robust object categorization in computer vision. Considering the most common levels of extraction of context, they reviewed different ways of using contextual information for object categorization. They first grouped the contextual features into three categories, which are semantic context, scale context and spatial context. These contextual features can be any information that is not directly produced by the appearance of the involved objects and can be obtained from several sources, such as, image tags and presence or locations of other objects. The five different classes of relations between an object and its surroundings proposed by Biederman et al. [12] (*interposition, support, probability, position, familiar size*) are considered a classical example of contextual features.

The semantic context of an object is defined in terms of its co-occurrence with other objects and in terms of its occurrence in scenes. It generally indicates what other objects to expect in a scene given a specific object. Normally, Semantic context can be obtained from the following sources: expert knowledge [29, 130], annotated databases [157], external knowledge-bases [114] and learning methods [157, 114]. In [104], it was shown that the identification of briefly presented drawings of real world objects is influenced by prior presentation of visual scenes. Experiments indicate that the observers accuracy for object categorization is improved if the target object is shown to the observer in an appropriate scene. By contrast, if the target object is shown to the observer in an inappropriate scene, the accuracy of object categorization declines. Other examples using semantic context can be found in [138], [147] and [114].

The scale context indicates that objects have a limited set of size relations with other objects in the scene. An example of scale context is Biederman's *familiar size* [12] which is a contextual relation based on the scales of an object with respect to others. To establish the scale context, the identification of other objects in the scene must be given. In addition, some other prerequisites should also be fulfilled before scale context can be used, such as the specific spatial and depth relations among objects. In [138], a simple framework is proposed to model the relationship between context and object properties. Object properties are learned as contextual features based on the correlation between the statistics of low level features across the entire scene. Scale context is then used for scale selection in the detection of high level structures as objects. Other examples using scale context can be found in [85], [140], and [139].

Spatial context can be defined by the likelihood of finding an object in some position and not others with respect to other objects in the scene. In general, spatial context represents the co-occurrence of other objects in the scene. Thus it provides qualitative information about the scene configuration, i.e., where those objects are usually located. In [9], the consequences of pairwise spatial relations on human performance in recognition tasks between objects that typically co-occur in the same scene are assessed. This work indicates that proper spatial relations among objects decreases error rates in the recognition of individual objects. Other examples using this type of context can be found in [53], [97], [107] and [118].

In fact, it is very rare to use only a certain kind of these context types in an application. More commonly, they are combined together to make the best use of context information. An example of using the three context types is provided in [34] and depicted in Fig.

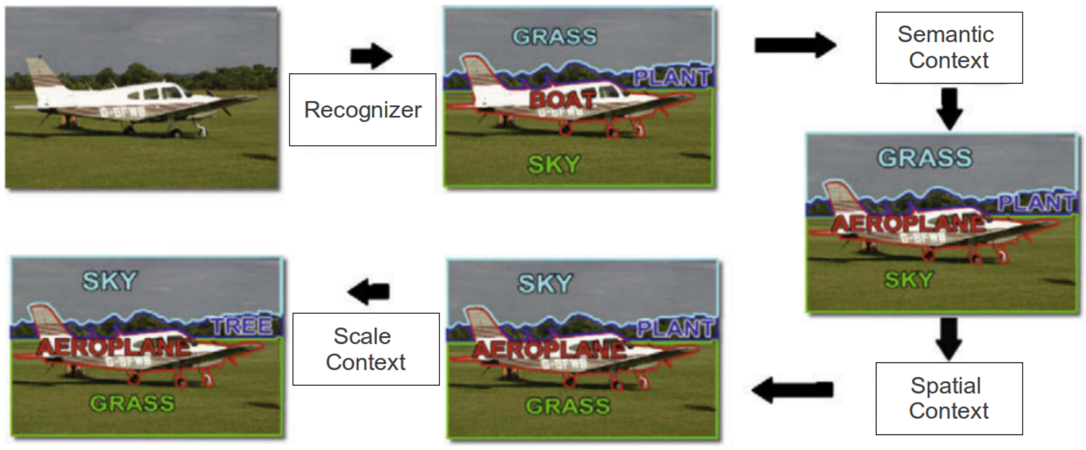


Fig. 2.1: An example of using the three context types provided in [34]. The input image is first labelled by the recognizer. Then the labelling performance is improved using the three context types.

2.1. Here, an idealized object categorization system incorporating the semantic context, spatial context and scale context is illustrated. First, the input image is segmented, and each segment is labelled by the recognizer. Next, different types of context information are adopted to refine the labelling performance.

2.2 Markov Logic Networks

People use knowledge to express their belief on a certain topic, which is learned from their daily life. Such knowledge holds for the most cases, nevertheless, there still exit scenarios where it fails. Thus, it is reasonable to model and use knowledge in the form of soft rules, which allow the existence of contradiction and retain the flexibility by defining knowledge as descriptive rules. For this purpose, Markov logic networks (MLNs) [117] are a good fit, because they combine first-order logic [10] and probabilistic graphical models [75]. First-order logic is able to compactly present knowledge in formulas (hard rules), and probabilistic graphical models are good at handling uncertainty. The combination of both makes it possible to express knowledge as soft rules (formulas attached with weight indicating uncertainty) in a systematic way.

Although Markov logic networks are a quite new method, invented in 2006, several MLNs-based approaches have been proposed so far, such as [71], [81], [150] and [101]. Before explaining the theory of Markov logic networks, we first briefly introduce the two fundamental ingredients of MLNs, which are Markov networks and first-order logic.

2.2.1 Markov Networks

According to [109], a Markov network is a model for representing the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathbb{X}$, which constructs an undirected Graph G, with each variable represented by a node of the graph. In addition, the model has one potential function ϕ_k for each clique in the graph, which is a non-negative real-valued function of the state of that clique. Then the joint distribution represented by a Markov network is calculated as

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}), \tag{2.1}$$

with $x_{\{k\}}$ representing the state of the variables in the kth clique. The partition function Z is calculated as

$$Z = \sum_{x \in \mathbb{X}} \prod_k \phi_k(x_{\{k\}}). \tag{2.2}$$

By replacing each clique potential function with an exponentiated weighted sum of features of the state, Markov networks are usually used as log-linear models:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j \omega_j f_j(X) \right), \tag{2.3}$$

where $f_j(x)$ is the feature of the state and it can be any real-valued function. For each possible state $x_{\{k\}}$ of each clique, a feature is needed with its weight $\omega_j = \log \phi_k(x_{\{k\}})$. Note that for the use of MLNs only binary features are adopted, $f_j(x) \in \{0, 1\}$. For more details on Markov networks, we refer to [109].

2.2.2 First-Order Logic

Here we briefly introduce some definitions in first-order logic, which are needed to understand the concept of Markov logic networks, for more details on first-order logic, we refer to [37].

- *Constant* symbols: these symbols represent objects of the interest domain.
- *Variable* symbols: the value of these symbols are the objects represented by the constant symbols.
- *Predicate* symbols: these symbols normally describe relations or attributes of objects.
- *Function* symbols: these symbols map tuples of objects to other objects.
- An *atom* or *atomic formula* is a predicate symbol used for a tuple of objects.
- A *ground atom* is an atom containing no variables.
- A *possible world* assigns a truth value to each possible ground atom.
- Together with logical connectives and quantifiers, a set of logical formulas can be constructed based on atoms to build a *first-order knowledge base.*

2.2.3 MLNs

Unlike first-order knowledge bases, which are represented by a set of hard formulas (constraints), Markov logic networks soften the underlying constraints, so that violating a formula only makes a world less probable, but not impossible. The fewer formulas a world violates, the more probable it is. In MLNs, each formula is assigned a weight representing how strong this formula is. According to [117], the formal definition of a MLN is:

A Markov logic network L is a set of pairs (F_i, ω_i), where F_i is a formula in first-order logic and ω_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \dots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (equations (2.1) and (2.3)) as follows:

1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground atom is true, and 0 otherwise.*

2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the ω_i associated with F_i in L.*

The probability over possible worlds x specified by the ground Markov network $M_{L,C}$ is calculated as

$$\begin{aligned} P(X = x) &= \frac{1}{Z} \exp\left(\sum_i \omega_i n_i(x)\right) \\ &= \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)}, \end{aligned} \tag{2.4}$$

where $n_i(x)$ is the number of true groundings of F_i in x, $x_{\{i\}}$ is the state (truth values) of the atoms appearing in F_i, and $\phi_i(x_{\{i\}}) = e^{\omega_i}$. Z is a normalization factor which is the same for all possible worlds with the same number of constants C. For more details on Markov logic networks, we refer to [117].

2.3 Knowledge-Supervised MCMC

Knowledge-supervised MCMC sampling (KSMCMC) is a probabilistic framework that systematically combines data processing in the continuous domain with knowledge processing in the symbolic domain. By establishing this framework, we aim to set up a procedure that systematically makes use of context knowledge in the form of descriptive logic rules to fit abstract semantic models to data sets acquired in the continuous domain. In this way, these data sets are represented by compact abstract models with certain semantic meanings which are well suited for high-level reasoning and communication purposes.

A main criterion for evaluating how well the abstract semantic model matches with the data is the posterior probability of the model M conditioned on the data D: $p(M|D)$. According to Bayes' theorem, the following equations are derived:

$$\begin{aligned} p(M, D) &= p(D|M) \cdot p(M), \\ &= p(M|D) \cdot p(D). \end{aligned} \tag{2.5}$$

Then the posterior probability of the model conditioned on the data, $p(M|D)$, is calculated as:

$$p(M|D) = \frac{p(D|M) \cdot p(M)}{p(D)}. \tag{2.6}$$

By ignoring the term $p(D)$ which is the same for all models, $p(M|D)$ is finally calculated as:

$$p(M|D) \propto p(D|M) \cdot p(M). \tag{2.7}$$

Here, the term $p(D|M)$ is usually called likelihood and indicates how probable the observed data set is for different settings of the model. Note that the likelihood is not a probability distribution over the model, and its integral with respect to the model does not (necessarily) equal one [13]. The term $p(M)$ is called the prior probability and describes what kind of models are possible at all. Our goal is then to find the model M^* that best explains the data and meanwhile has a high prior probability, which leads to the maximum of the posterior probability:

$$M^* = \underset{M \in \Omega}{\operatorname{argmax}}\, p(M|D), \tag{2.8}$$

where Ω indicates the entire solution space.

In the framework of KSMCMC, M^* is obtained using a data-driven MCMC sampling process that is guided by context knowledge defined as descriptive logic rules in Markov logic networks. Using the defined context knowledge, the prior distribution is shaped in such a way, that the models that comply with the underlying knowledge base are assigned a high prior probability, and models that contradict or not fully comply with the knowledge base are given a low prior probability. Multiplying the prior probability of a model with its corresponding likelihood, which indicates how well data matches with this model, the posterior probability of this model conditioned on data is calculated. This posterior probability is then used to evaluate models in the data-driven MCMC sampling.

The general idea of shaping prior distribution is explained in Fig. 2.2, using a one-dimensional example. The likelihood for different settings of models, which contains three local optima, is depicted in Fig. 2.2-a. The prior distributions represented by context knowledge defined in MLNs are shown in Fig. 2.2-b. Different context knowledge bases (set of rules) represent different prior distributions (green and red). If no context knowledge is used, it is the same as implementing the context knowledge that represents a uniform distribution which does not influence the posterior, i.e. posterior is only proportional to likelihood in this case. The corresponding posterior distributions obtained using the two prior distributions are demonstrated in Fig. 2.2-c. By shaping prior distribution using context knowledge, the posterior distribution is influenced so that the number of local optima decreases, which means, the models that comply with the defined context knowledge and match the data well tend to have a high posterior probability. In the following, the entire modelling and sampling procedures of KSMCMC are explained in detail.

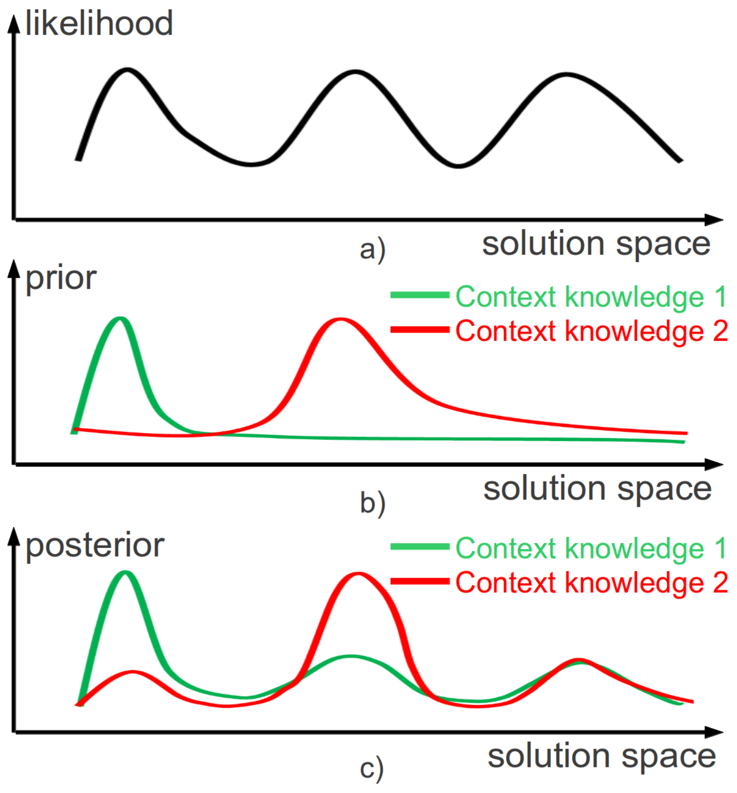


Fig. 2.2: The general idea of shaping prior distribution illustrated using a one-dimensional example. **a)** The likelihood for different settings of models, which contains three local optima. **b)** The prior distribution represented by context knowledge defined in MLNs. Different context knowledge bases (set of rules) represent different prior distributions (green and red). **c)** Corresponding posterior distributions obtained using the two prior distributions shown in figure b.

2.3.1 Model Definition

In general, the semantic abstract model M that is used to represent the data D should be defined in the following way:

$$M := \{V_c, V_a\}, \tag{2.9}$$

where V_c and V_a denote the set of the continuous and abstract variables respectively.

Continuous Variables

As shown in Fig. 2.3, the model definition spans over the continuous and the symbolic domain. V_c contains necessary variables that are needed to represent the data in the continuous domain. Based on the variables in V_c, instances of the defined model can be generated. The model can be evaluated by comparing the generated instances with the data. This is also a possible way of calculating the likelihood in equation (2.7). In principle, the set of continuous variables V_c serves as the interface to data processing in the continuous domain. For example, in the scenario of fitting a line model to point clouds in the 3D Cartesian space, V_c would be the variables that are needed to formulate the 3D line equation.

Abstract Variables

Based on the semantics that needs to be encoded into the model, a set of abstract variables V_a can be defined. These abstract variables are highly related to the continuous variables and describe the semantic features of the data in the symbolic domain. Here semantic features refer to the task-specific and meaningful definitions that are required to represent the continuous data on a higher level, i.e. the semantic level. These semantic features do not come along with the data in the first place, thus they must be manually defined. The definition of these abstract variables depends mainly on the scenario and the users' needs. For instance, abstract variables could be "room", "corridor" and "hall" for semantic indoor mapping, or "road", "traffic flow" and "building" for traffic scene understanding.

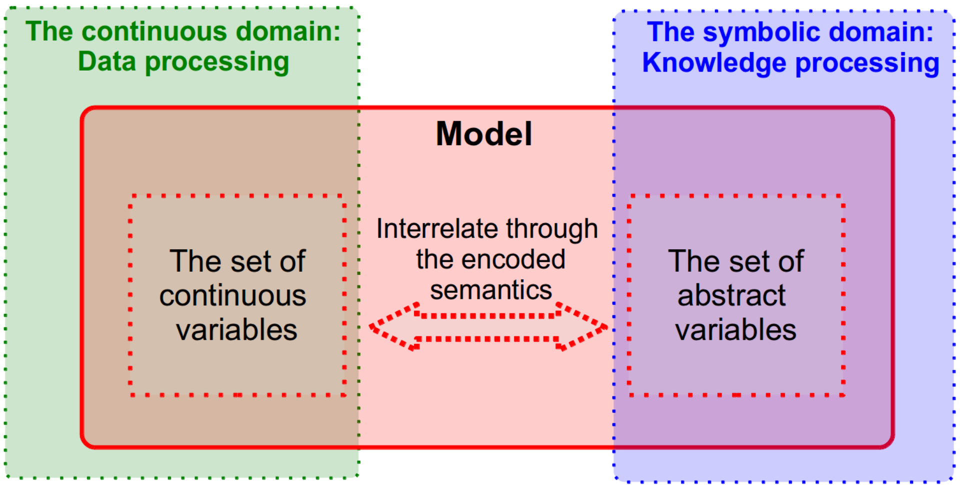

The continuous domain: Data processing
The symbolic domain: Knowledge processing
Model
The set of continuous variables
Interrelate through the encoded semantics
The set of abstract variables

Fig. 2.3: The general definition of semantic abstract models. The model contains a set of continuous and a set of abstract variables. These variables serve as interfaces to the continuous and symbolic domain. These two sets of variables interrelate with each other through the encoded semantics, so that the two domains are systematically joined together.

Interrelation through Semantics

In principle, the set of abstract variables V_a serves as the interface to knowledge processing in the symbolic domain. V_a provides the fundamental elements for expressing knowledge bases in Markov logic networks. V_a interrelates with the set of continuous variables V_c through the underlying semantics. V_a contains semantic features that are abstracted directly from V_c. These semantic features define the underlying model on a higher level and enable the use of context knowledge. Through the interrelation between V_a and V_c, the continuous domain and the symbolic domain are systematically joined together. In this way, data processing and knowledge processing can fully deploy their own strength and meanwhile act together as a consistent unity.

2.3.2 Knowledge Representation and Reasoning

Having defined the model, the next step is to represent task-specific context knowledge using Markov logic networks. This involves several intermediate steps which are explained in the following.

Knowledge in Natural Language

The first step of knowledge representation and reasoning is always to think about what kind of knowledge needs to be encoded into the knowledge base which is comprised of several rules. This begins by writing down the rules in the form of natural language. This is a iterative process: think and rethink which rules are really needed to formulate the desired knowledge base. To keep the efficiency of knowledge reasoning on a acceptable level, minimalism should be taken into account: keep only the minimum set of rules that are needed.

Definition of Logic Rules

After the rules are written down in natural language, we need to transform them into descriptive logic rules using Markov logic networks (MLNs). Since MLNs use the semantics of first-order logic (FOL) for logical formulation, we refer to previous literature on FOL [37] for defining logic rules. In addition to rule definition, another issue is to consider what kind of uncertainty should be assigned to each rule. In MLNs, the uncertainty of rules is indicated by the weights of logical formulas. The greater a weight is, the more probably the corresponding rule holds. The weights can either be learned or manually designed. Examples can be found in [87], [64], [65] and [66].

Definition of Predicates

In predicate logic, such as FOL, predicates are the basic elements used to formulate logic rules. As shown in Fig. 2.4, all these predicates are defined in the set of abstract variables V_a of the model. Among the predicates, there exist two main groups which differ from each other with respect to their functionalities in logical formulations. These two groups are evidence predicates and query predicates, and they are explained in the following:

- Evidence predicates: these predicates introduce features of the model (that are originally defined in the continuous domain) into the symbolic domain and enables the knowledge reasoning process in general. Truth values of evidence predicates can be directly extracted from the continuous variables V_c of the model. With respect to the knowledge reasoning process, evidence predicates are known elements and serve as inputs.

- Query predicates: truth values of query predicates are unknown in the first place. In general, they are the answers that the knowledge reasoning process should deliver given the truth values of evidence predicates. Within model definition, query predicates play a key role in interrelating the set of abstract variables with the set of continuous variables. They bring the results of knowledge reasoning in the symbolic domain back into the continuous domain, where these results are used to guide the processing of continuous data and variables.

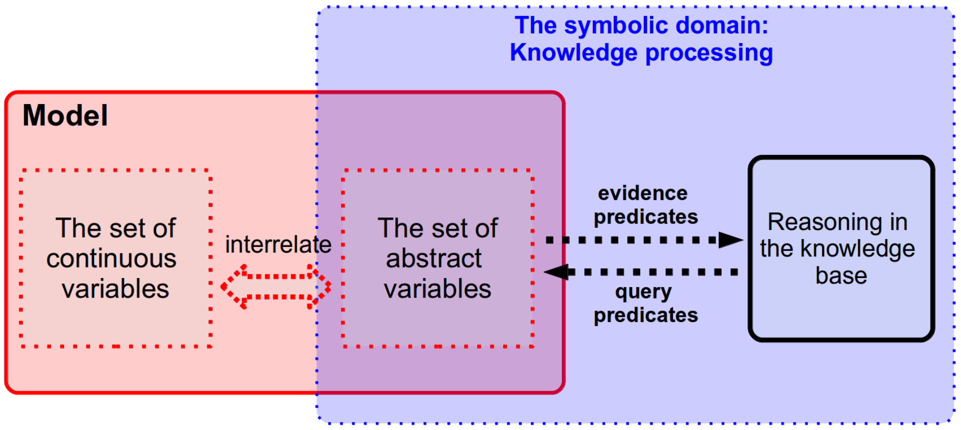

The symbolic domain: Knowledge processing
Model
The set of continuous variables
interrelate
The set of abstract variables
evidence predicates
query predicates
Reasoning in the knowledge base

Fig. 2.4: The functionality of evidence and query predicates. Truth values of evidence predicates can be directly extracted from the continuous variables of the model. Query predicates are the answers that the reasoning process should deliver given the truth values of evidence predicates.

2.3.3 Likelihood Definition

The goal of likelihood definition is to define a function for data-based model evaluation. In general, likelihood $p(D|M)$ measures how well data D matches to the model M. While data is typically a large set of values, a model is usually described by a comparably smaller set of values (variables). If a model is defined in an analytical closed form, such as a mathematical equation, likelihood could be simply calculated as the deviation between data and the model.

However, in many circumstances, the model itself is much more complex and does not have an analytical formulation. In such cases, it is very difficult to directly compare data with the model. To realize likelihood $p(D|M)$, a common method is to generate synthetic data D_M using the model M that needs to be evaluated. It should be noted that the synthetic data D_M must be generated in a form that is compatible with the data D that is used to evaluate the model M. Finally, likelihood is calculated as the result of how well D matches to D_M.

2.3.4 Prior Definition

Principally, prior probability $p(M)$ describes the uncertainty of a model M before any evidence is taken into account. In other words, it indicates how probable a model M is, regardless of data D. The definition of prior is often purely subjective. In the case that no useful information is available about the underlying model, the prior is defined by assigning equal probabilities to all model settings, i.e. using a uniform distribution as prior. However, in this way, prior loses its influence on posterior.

Within the framework of KSMCMC, prior is defined with the help of knowledge processing in Markov logic networks. Useful information about the underlying model is specified by the knowledge base that is defined as descriptive logic rules. Reasoning results of the defined knowledge base are used to define prior distribution. By doing so, prior distribution is shaped in such a way that models that comply with the defined knowledge base are assigned a high prior probability, and accordingly, models that contradict with the defined knowledge base are given a low prior probability.

Given the model definition

$$M := \{V_c, V_a\},$$

prior $p(M)$ is calculated in the following form:

$$\begin{aligned} p(M) &= p(V_c, V_a), \\ &= p(V_c|V_a) \cdot p(V_a), \end{aligned} \tag{2.11}$$

with V_c being the set of continuous variables, and V_a being the set of abstract variables. $p(V_a)$ is the prior probability of the set of abstract variables. $p(V_c|V_a)$ is the conditional probability of the set of continuous variables given the set of abstract variables. Depending on the viewing angle, $p(V_c|V_a)$ and $p(V_a)$ can be realized in different ways. The fundamental idea is to use the results of the knowledge reasoning process to design functions that express reasonable semantic interrelations between these variables.

2.3.5 Design of Data-Driven MCMC

After likelihood and prior are defined, the posterior probability $p(M|D)$ is calculated as their multiplication. The next step is to find the model M^* that has the maximum posterior probability (equation (2.8)). Given the high-dimensional and complexly structured solution space, this is indeed a big challenge. Here, the critical question is how to efficiently search for M^* in such a complex solution space. To tackle this challenge, we adopt the data-driven Markov chain Monte Carlo (DDMCMC) technique [161], which is a modern variant of Markov chain Monte Carlo (MCMC) methods [98, 13].

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from probability distributions. In MCMC methods, a Markov chain is constructed, whose equilibrium distribution is identical with the desired distribution. Here, a Markov chain is a stochastic process and contains a sequence of random variables which follows the Markov property. The Markov property states that the next state of the chain depends only on the current state and not on the sequence of states.

In MCMC methods, the Markov chain is constructed by sequentially executing state transitions according to a proposal distribution. The state of the chain after certain burn-in time is then used as a sample of the desired distribution. Burn-in refers to the practise of throwing away some iterations at the beginning of the MCMC run. States of the underlying Markov chain during the burn-in time can not yet represent samples of the desired distribution. According to [144], three criteria should be taken into account while designing a Markov chain:

- The Markov chain should be ergodic, i.e. from an arbitrary initial state, the Markov chain should be able to visit any other states in finite time.
- The Markov chain should be aperiodic. A Markov chain is considered to be aperiodic if every state of this Markov chain is aperiodic.
- The Markov chain should fulfil the condition of detailed balance equations [40]. This condition requires that every state transition of the Markov chain is reversible.

Data-Driven MCMC

Data-driven MCMC (DDMCMC) [161] is a modern extension of MCMC methods and is originally proposed for image segmentation purposes. Like traditional MCMC methods, the DDMCMC method also constructs well-balanced Markov chain dynamics to explore complex solution spaces. As the term “data-driven” already tells, in DDMCMC, the extension lies in that DDMCMC adopts data-driven (bottom-up) techniques to propose state transitions for the Markov chain. By doing so, the Markov chain dynamics is greatly accelerated, and the burn-in time is largely reduced, compared to traditional MCMC methods.

In DDMCMC, state transitions are proposed based on features that are extracted from data using bottom-up detectors. As shown in Fig. 2.5, values of the continuous variables of the underlying model are influenced by these state transitions. In general, the definition of state transitions is quite subjective. Depending on the use case, state transitions can have different functionalities. Some typical functionalities include birth/death of certain components, value changes of certain parameters, merge/split of certain entities and so on. In principle, it is helpful to arrange these state transitions as reversible pairs, so as to ensure that the Markov chain fulfils the condition of detailed balance equations.

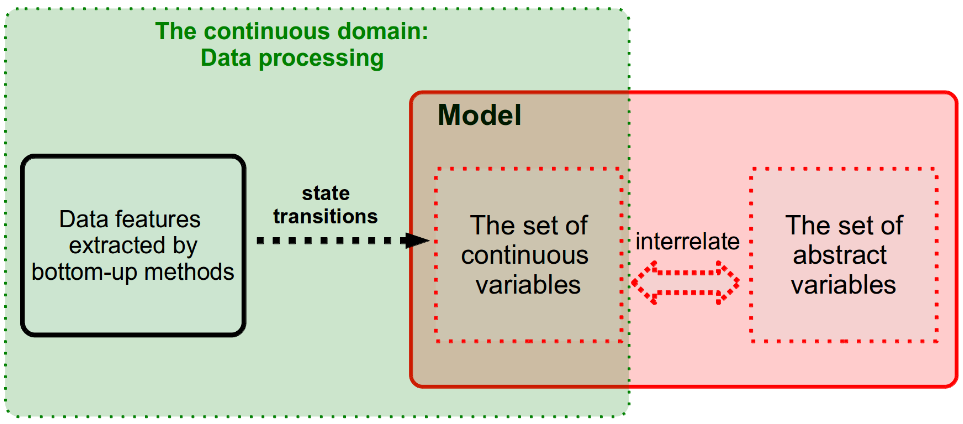

The continuous domain:
Data processing
Model
Data features extracted by bottom-up methods
state transitions
The set of continuous variables
interrelate
The set of abstract variables

Fig. 2.5: State transitions are proposed based on extracted data features in DDMCMC. Values of the continuous variables of the underlying model are influenced by state transitions.

Metropolis-Hastings Algorithm

Among the existing MCMC methods, the Metropolis-Hastings algorithm is a popular method for obtaining a sequence of random samples from a complex probability distribution. Here we only introduce some basic ideas about the Metropolis-Hastings Algorithm to help better explain the KSMCMC Framework. For more details on the theory of the Metropolis-Hastings Algorithm, we refer to [13] and [21].

In the Metropolis-Hastings algorithm, samples are iteratively generated in a random manner. These samples follow the Markov property, i.e. the next sample is only dependent on the current sample. In each iteration, a new sample is proposed by a state transition which follows certain proposal distribution. Then the new sample is either accepted or rejected. In case of acceptance, the new sample is used in the next iteration. Otherwise, the current sample is reused in the next iteration, and the new sample is abandoned. As more and more samples are randomly generated, the distribution of the underlying Markov chain approximates the desired distribution more accurately.

In the Metropolis-Hastings algorithm, the newly proposed sample in each iteration is accepted by the following acceptance probability $\lambda(M, M')$:

$$\lambda(M, M') = \min\left(1, \frac{p(M'|D) \cdot Q(M|M')}{p(M|D) \cdot Q(M'|M)}\right), \tag{2.12}$$

where M is the current sample, and M' is the new sample. $p(M|D)$ is the posterior probability defined in equation (2.7). $Q(M'|M)$ is the proposal probability of generating M' from M. Accordingly, $Q(M|M')$ is the corresponding reverse proposal probability. In a general problem, it is unclear which proposal distribution should be used to calculate $Q(M'|M)$ and $Q(M|M')$. The choice of the proposal distribution is a task-specific engineering problem. Within the framework of DDMCMC, the proposal probability can be calculated with the help of bottom-up feature detectors that are used to propose state transitions.

The acceptance probability $\lambda(M, M')$ should be interpreted as follows:

- With $\lambda(M, M') \geq 1$, the new sample is definitely accepted.
- With $\lambda(M, M') < 1$, the new sample is accepted only by chance. In this case, the acceptance probability for the new sample is $\lambda(M, M')$. This is realized by drawing a random number θ according to the uniform distribution $\mathcal{U}(0, 1)$, and comparing θ with $\lambda(M, M')$:
  - With $\theta \leq \lambda(M, M')$, the new sample is accepted.
  - With $\theta > \lambda(M, M')$, the new sample is rejected.

3 Use Case: Semantic Indoor Mapping

The primary challenge for any autonomous system operating in realistic, rather unconstrained scenarios is to manage the complexity and uncertainty of the real world. In robotics this holds, as soon as the robots leave the carefully engineered production environments in which they have been so successful in the past decades.

The typically high degree of uncertainty in real-world environments, that makes a robot's life so hard, comes from the following sources: the limited measurement accuracy and other limitations of the system sensors, modelling errors and purposefully made simplifications in the system internal representations, unobserved environment dynamics and random effects in action execution. While it is unclear how exactly humans and other higher animals master these problems, it seems evident that abstraction, semantics and knowledge together play an important role. The use of abstract concepts allows to define the system behaviour on higher levels and independently of the exact setting of the environment and the exact sensor readings.

In this chapter we address the first two of the problems mentioned above, in that we provide the system with a limited capability of abstraction allowing for a higher-level understanding of its environment. In addition, we directly address the uncertainty related issues by strictly following a probabilistic approach that explicitly models and keeps track of the uncertainty associated with any variables of the problem. As a by-product, the system capability to use predefined concepts will ease cooperation in mixed human-robot tasks, since a common language used by both the human and the robot is a precondition for efficient exchange of information between both parties.

To illustrate the general idea, we use an example from an indoor navigation scenario, namely semantic indoor mapping. The objective of the presented method is to provide an abstract, semantically annotated but still probabilistic map of the indoor environment. For this purpose, we first use a robot – equipped with a 2D laser scanner – to build an occupancy grid of the environment using a simultaneous localization and mapping (SLAM) method [41, 26, 50, 96] and then employ the procedure described in the remainder of this chapter to extract the semantic information. To do this, we use our knowledge-supervised Markov chain Monte Carlo (KSMCMC) sampling technique to generate samples from the probability density function capturing the distribution of probable worlds the robot could encounter. The maximum posterior solution could then be used as an estimate of what the world semantically looks like.

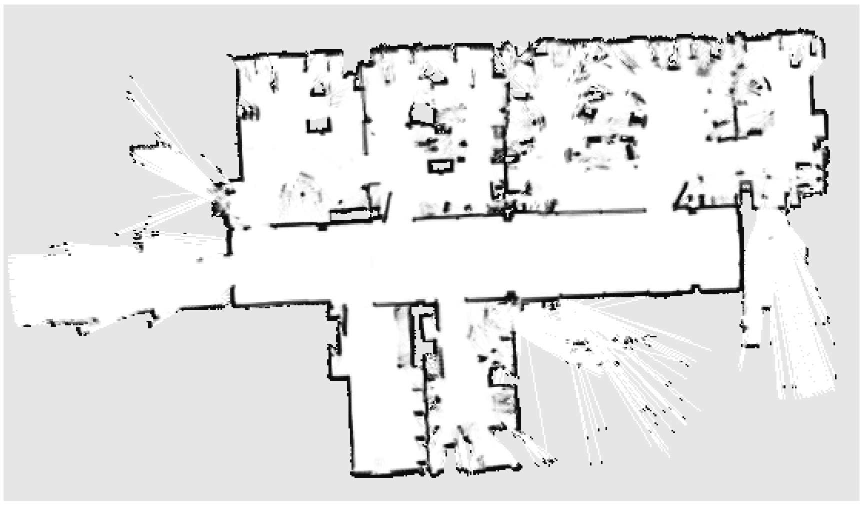


Fig. 3.1: A typical occupancy grid map of an indoor environment, obtained from the Robotics Data Set Repository (Radish) [58].

3.1 Introduction

Most of todays' mapping approaches aim to construct a globally consistent, metric map of the robot's operating environments. See Fig. 3.1 for a typical result. Such maps enable the robot to localize itself with respect to the environment and thus determine its global pose in an assumed flat world with an accuracy of typically a few centimetres in translation and below one degree in rotation. Based on this capability, the robot can also plan a path and navigate towards a goal which is also specified by its metric position in the global map reference frame. However, the robot does not understand its environment in terms of typical semantic concepts like rooms, corridors or functionally enriched concepts like a kitchen or living room. Furthermore, the robot does not understand relations like adjacency, connectivity via doors, or properties like rectangularity that – if known to be relevant to the given environment – could help to build the maps in the first place.

Our work aims at extracting such semantic models of the environment from the more or less raw sensor data. In this context, we assume, that a map, like the one depicted in Fig. 3.1, was already constructed using one of the proven methods available for this purpose [41].

Different from metric mapping, semantic mapping aims to construct a semantic map for the environments that the robots work in. The focus of semantic mapping is to describe the environments on the semantic/abstract level, so as to provide valuable semantic information for higher level applications, such as Human Robot Interaction (HRI) [76, 39, 129]. An example on the comparison between metric and semantic mapping is illustrated in Fig. 3.2.

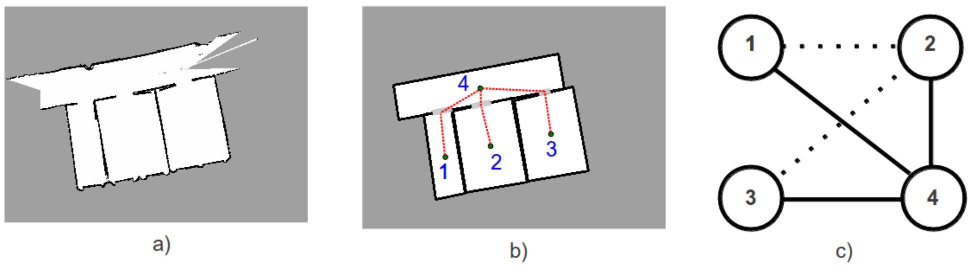


Fig. 3.2: Metric mapping vs. semantic mapping: a) A 2D metric map (occupancy grid map) obtained using [41]. Such maps are a big matrix of occupancy values and do not provide semantic level information. b) The corresponding semantic map generated by [86]. Basic semantic type "unit" is introduced, which is represented by a rectangle. Four units are found in the environment. Circles indicate the unit center. Light-gray shows the door. The topology of the map is demonstrated in dashed lines. c) The semantic map represented in pure abstract level. Solid lines indicate that two units are connected by a door, whereas dashed lines connect two neighbour units.

3.2 Related Work

In recent years, semantic mapping has drawn great interest in the academia. In the state of the art there exist many proposals which can be categorized according to different criteria, such as 2D/3D, indoor/outdoor and so on. In the following we review these related work with respect to their output form.

A very big body of literature focuses on semantic place labelling which divides the environment into several regions and attaches each region a semantic label like "office room" or "corridor". Park and Song [108] proposed a hybrid semantic mapping system for home environments, using explicitly the door information as a key feature. Combining image segmentation and object recognition, Jebari et al. [70] extended semantic place labelling with integrated object position. Based on human augmented mapping [137], rooms and hallways are represented as Gaussians to help robot navigate in [102]. As shown in Fig. 3.3, Pronobis and Jensfelt [113] integrated multi-modal sensory information, human intervention and some common-sense knowledge to classify places with semantic types. Other examples on semantic place labelling can be found in [38], [77] and [125].

Different from place labelling, another big group of work concentrates on labelling different constituents of the perceived environments with semantic tags, such as walls, floors, ceilings of indoor environments, or buildings, roads, vegetations of outdoor environments. In [103], a logic-based constraint network describing the relations between different constituents is used for the labelling process in indoor environments. An example of this work is shown in Fig. 3.4. Persson and Duckett [110] combined range data and omni-directional images to detect outlines of buildings and nature objects in an outdoor setting. Zhu et al. [162] implemented a semantic labelling system on a vehicle to classify urban scenes, based on range image. [111] and [131] show the application of semantic constituent labelling in

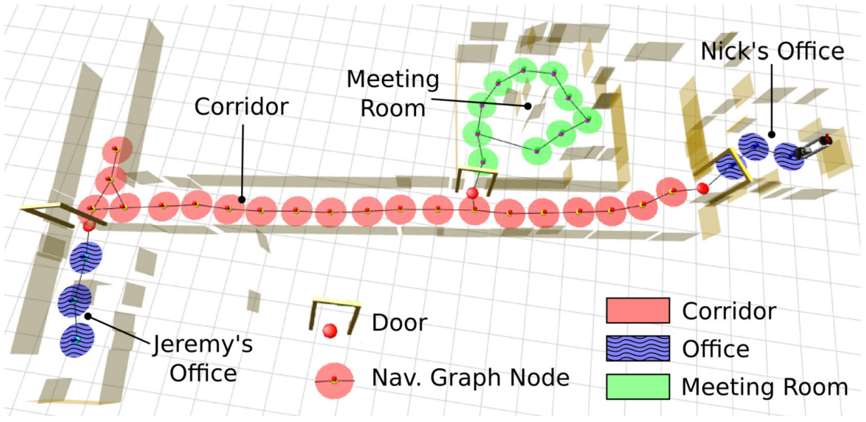


Fig. 3.3: An example of semantic place labelling [113].

underwater scenarios. Other examples in this category can be found in [122], [121], [3] and [156].

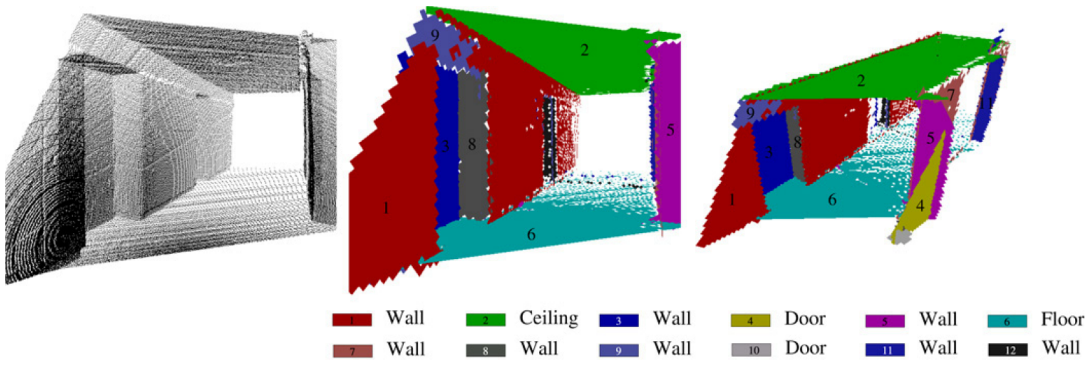


Fig. 3.4: An example of semantic labelling of environment constituents [103].

Another group of related work concentrated on generating topological maps of the environments traversed by a robot. Blanco et al. [14] proposed a hybrid metric-topological approach based on Bayesian inference. In addition to traditional metric maps, their approach was able to reconstruct the path of the robot as a topological map. As shown in Fig. 3.5, Tao et al. [134] demonstrated a topological SLAM process which used the saturated generalized Voronoi graph (S-GVG) to represent the topology of an office-like environment. Based on particle filtering, Werner et al. [153] proposed to build topological maps for an indoor environment by disambiguating places which appear indistinguishable using neighbourhood information extracted from a sequence of observations. Other examples in this group can be found in [5], [27] and [4]. Approaches in this category were mainly interested in building globally consistent metric maps while providing topology along the robot path as a by-product. In this sense, they represent the early tries toward semantic mapping.

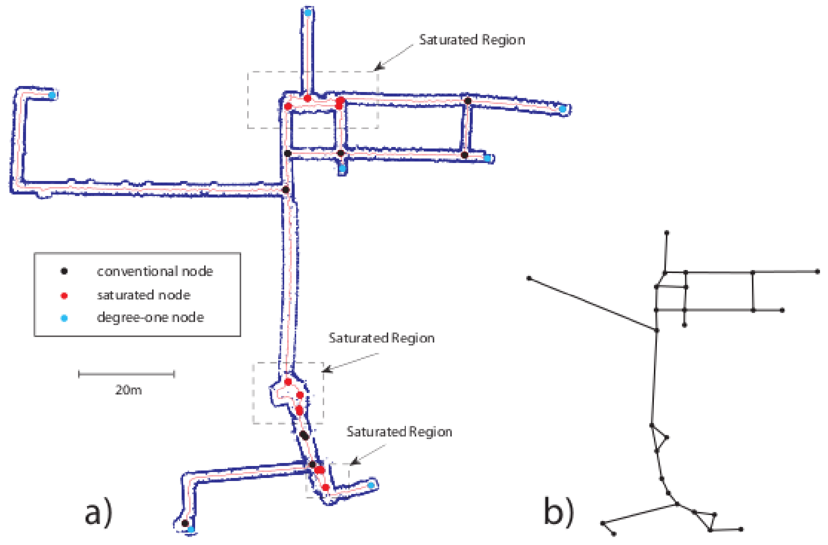


Fig. 3.5: An example of topological mapping [134].

A fourth category of literature consists of object-based semantic mapping systems which use object as basic representation unit of the perceived environment. Such systems usually adopt point cloud processing (e.g. Point Cloud Library [119]) and image processing (e.g. OpenCV [16]) techniques to model or detect objects, and object features like appearance, shape and 3D locations are often used to represent the objects. Rusu et al. [120] proposed a hybrid semantic object mapping system for household environments (mainly kitchens), based on 3D point cloud data. Objects modelled in this work are those which perform utilitarian functions in the kitchen such as kitchen appliances, cupboards, tables, and drawers. An early example on object-based semantic mapping is shown in [84], where a relational object map is proposed for laser-based mobile robot 2D mapping, by modelling geometric primitives like line segments as objects. More examples on object-based semantic mapping can be found in [115], [23], [105] and [89]. An example of this category is illustrated in Fig. 3.6.

In addition to the categories mentioned above, there exist also a few systems which adopt explicitly a compact semantic model to represent the perceived environments. As shown in Fig. 3.7, Shukor et al. [2] proposed a 3D planar model for indoor environments based on knowledge of spatial relationship of room surfaces. Geiger et al. [36] introduced a generative model for explaining urban scenes with semantic types, and realized the entire system using MCMC sampling.

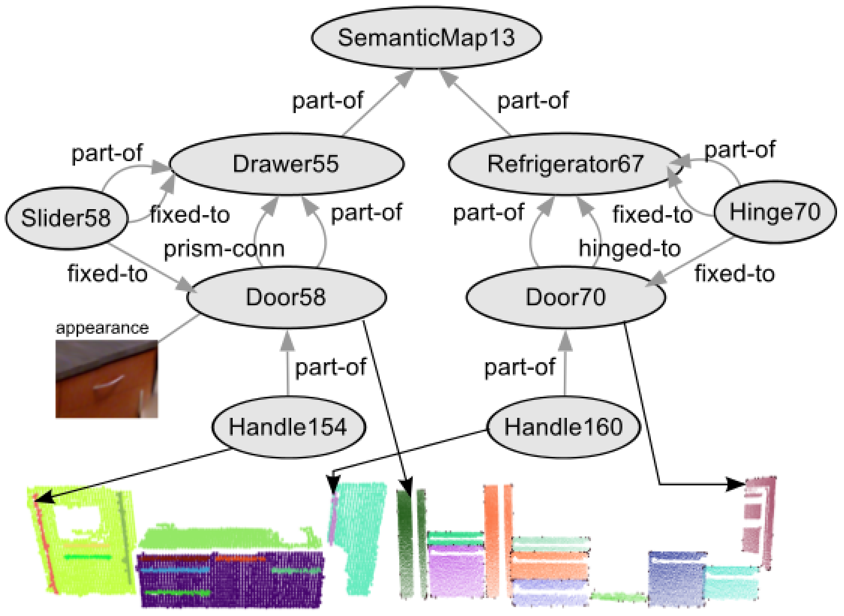


Fig. 3.6: An example of object-based semantic mapping [105].

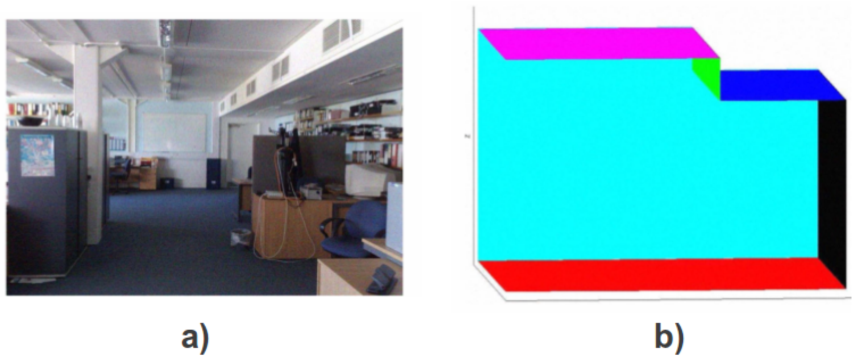


Fig. 3.7: Environment modelling using 3D planar surfaces [2].

3.3 Contributions

Using our knowledge-supervised MCMC sampling approach, we show a new way of automatically generating semantic maps from preprocessed sensor data. This is done by means of a probabilistic generative model and MCMC-based reasoning techniques. We use Bayesian reasoning to build semantic maps, so that they are aligned with the preprocessed sensor observations, that a robot made during an environment exploration and mapping stage. This introduces a bottom-up path into the approach and employs data driven discriminative environment feature detectors to analyze the continuous noisy sensor observations. In principle, our approach demonstrates a new method of finding compact abstract model for input data using predefined abstract terms. Based on these abstract terms, intelligent autonomous systems, such as a robot, should be able to make inference according to specific knowledge base, so that they can better handle the complexity and uncertainty of the real world. Based on Markov logic, we formulate task-specific context knowledge as descriptive logic rules which increase the overall abstraction performance.

Our work differs from previous semantic mapping approaches, that mostly use various classification methods in a bottom-up fashion to label either spatial regions or places based on context or that assign semantic labels directly to portions of the observations. Instead, we construct a parametric, abstract, semantic and top-down representation of the domain under the consideration: a classical indoor environment containing several units of different types, that are connected by doorways.

As output, our system returns a parametric abstract model of the perceived environment that not only accurately represents the environment geometry, but also provides valuable abstract information. The model that we generate, is structured similarly to a scene graph and is perfectly suited for any higher level reasoning and communication purposes.

3.4 An Abstract Parametric Model for Occupancy Grids

Here we aim to construct a probabilistic, abstract and parametric model of the world around the robot, that is essentially based on abstract semantic concepts but at the same time allows to predict the continuous percepts that the robot obtains via its noisy sensors. We call an instance of this model a *semantic world* or a *world*. This model has a form similar to a scene graph, a structure which is widely used in computer graphics. The scene graph (see Fig. 3.8 c) in our case consists of units and doorways connecting the units and can be visualized as a classical floor plan(see Fig. 3.8 b).

The scene graph and thus also the semantically annotated world state is denoted by a vector of hidden parameters W specifying the world state, that generated the occupancy map M we are currently looking at. In the Bayesian framework we can use a maximum posterior approach to infer the most probable state $W^* \in \Omega$ from the space of probable worlds Ω given the map M.

$$W^* = \underset{W \in \Omega}{\operatorname{argmax}}\, p(W|M), \tag{3.1}$$

with

$$p(W|M) \propto p(M|W)p(W). \tag{3.2}$$

Here $p(W|M)$ is the posterior distribution of W given the known map M and $p(W)$ is the prior specifying, which worlds W are possible at all. $p(M|W)$ is the likelihood function describing how probable the observed map M is, given the different probable worlds represented by a parameter vector W. The actual model is represented in the structure of the parameter vector W, while semantically relevant constraints go into the prior $p(W)$.

In our case W contains the scene graph, i.e. the parameters of a floor plan: number of units, their dimensions and connectivity, the location of doorways. Here, units are defined

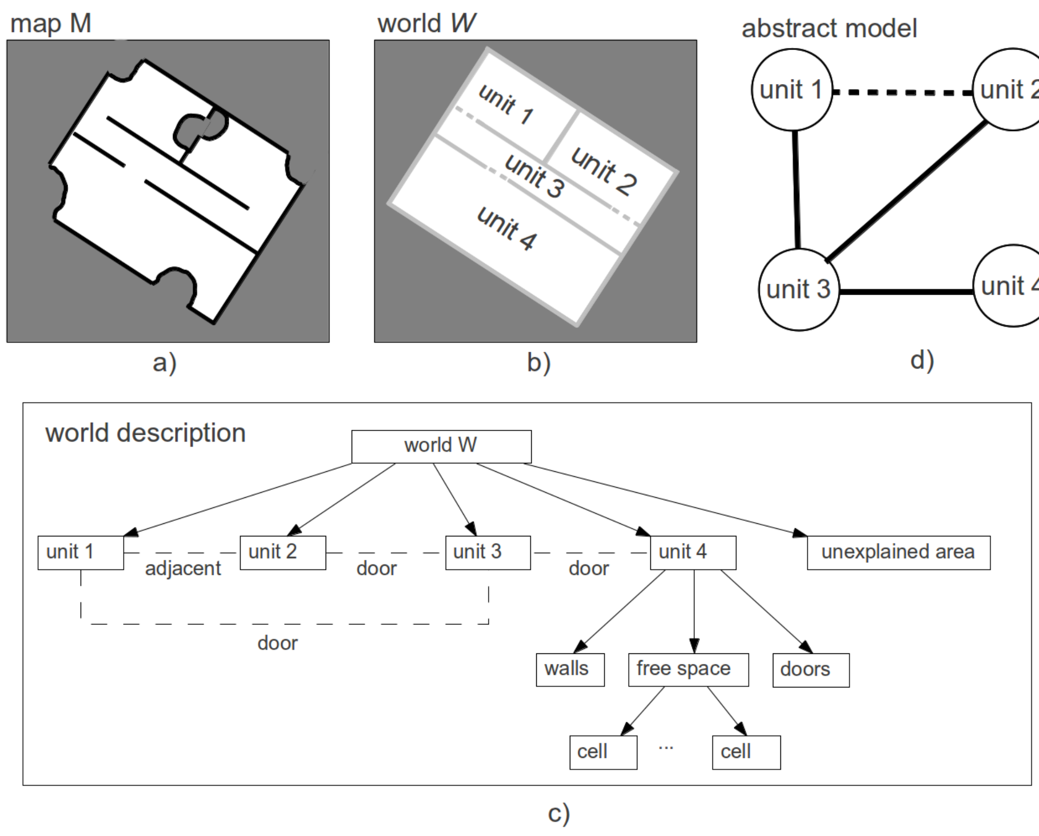


Fig. 3.8: a) A simplified occupancy grid map: Unexplained area is drawn in grey, free space is drawn in white. Occupied area is drawn in black. b) A possible floor plan represented as a scene graph (W): The world (gray=unknown, white=free, solid light-gray=walls, dashed light-gray=doors) is divided into four units and the corresponding unexplained area. The connectivity is given by doors and walls. c) The semantic description of the world in form of the scene graph: Directed links connect nodes. The dashed lines represent connectivity. Like unit 4, each unit has three child nodes: walls, free space, and doors. Note that the lowest level of node in the tree structure is the grid cell that belongs to walls, free space and doors. d) The corresponding abstract model with solid edges indicating connection by a door and dashed edges indicating connection by a wall.

as a rectangular space that is enclosed by four *walls*, and walls are line segments defined by two end points. Connectivity indicates the spatial relationship of different units. Two units could be either adjacent (connection by a wall) or connected by a door.

3.4.1 Using Weak Context Knowledge as Prior

Certain a-priori assumptions about some properties of the structured world are made based on some weak context knowledge as follows:

1) a unit has four walls and possesses a rectangular shape.

2) each cell in the map should only belong to one unit.

These a-priori constraints are enforced by means of the prior $p(W)$ in our generative model (3.2) as simple factors. The prior penalizes worlds that are not fully compliant with the above assumptions:

$$p(W) = \alpha_1 \times \alpha_2, \tag{3.3}$$

where α_1 and α_2 are the corresponding penalization terms for the point 1) and 2) of the prior information respectively, and they are defined as follows:

$$\alpha_1 = \begin{cases} \psi_1^\theta, \text{conflict with point 1)}, \\ 1, \text{otherwise}, \end{cases} \tag{3.4}$$

$$\begin{aligned} \alpha_2 &= \prod_{c(x,y) \in M} \psi_2^{\gamma(c(x,y))}, \\ \gamma(c(x,y)) &= \begin{cases} \sigma(c(x,y)) - 1, \sigma(c(x,y)) > 1, \\ 0, \text{otherwise}, \end{cases} \end{aligned} \tag{3.5}$$

where ψ_1 and ψ_2 are penalization terms with $\psi_1, \psi_2 \in (0,1)$. θ is the number of pairs of adjacent walls whose included angle is not 90 degree ($\pm$tolerance). $c(x,y)$ denotes one grid cell in the map M. $\sigma(c(x,y))$ indicates the number of units, to which $c(x,y)$ belongs. α_2 is a cell-wise penalization of the overlap between different units, i.e. if there is no overlap in one cell $c(x,y)$, then $\sigma(c(x,y))$ is smaller than 1, in which case $\gamma(c(x,y))$ is 0 (no penalization in cell $c(x,y)$). Otherwise, if $\sigma(c(x,y))$ is bigger than 1, which means the cell $c(x,y)$ belongs to more than one unit, then $\gamma(c(x,y))$ is bigger than 0 (penalization in cell $c(x,y)$). Two examples of the functionality of the prior are illustrated in Fig. 3.9 and Fig. 3.10.

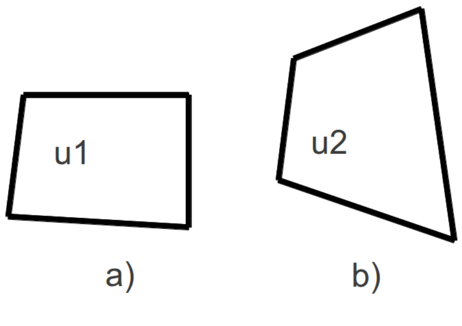


Fig. 3.9: According to the defined weak context knowledge point 1) the unit $u1$ shown in a) is only slightly penalized, because $u1$ is not an exact rectangle. However the unit $u2$ shown in b) is strongly penalized, because $u2$ is no more a rectangle.

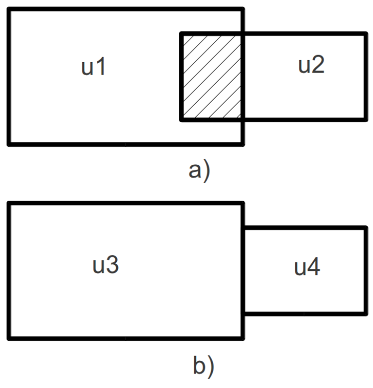


Fig. 3.10: According to the defined weak context knowledge point 2), the world containing $u1$ and $u2$ in a) will be penalized because of the overlap of these two units (shaded area). In contrast, the world containing $u3$ and $u4$ in b) will not be penalized. According to the defined context knowledge the world shown in b) has a higher prior probability.

3.4.2 Definition of Likelihood

For our generative model, we need to specify the likelihood function $p(M|W)$ additionally. Since M is represented by an occupancy grid with statistically independent grid cells $c \in M$, we only need to come up with a model $p(c|W)$ for all cells at their locations (x, y) in the map M:

$$p(M|W) = \prod_{c(x,y) \in M} p(c(x,y)|W). \tag{3.6}$$

For our model $p(c(x,y)|W)$, we first discretize the cell state $M(x,y)$ by classifying the occupancy values into three classes "occupied=2", "unexplained=1" and "free=0" so as to generate the classified map $C_M(x,y)$ according to:

$$C_M(x,y) = \begin{cases} 2, & 0 \leq M(x,y) \leq h_o, \\ 1, & h_o < M(x,y) \leq h_u, \\ 0, & h_u < M(x,y), \end{cases} \tag{3.7}$$

where h_o and h_u are the intensity thresholds for occupied and unexplained grid cells. Based on our model W we can also predict expected cell states $C_W(x,y)$ accordingly:

$$C_W(x,y) = \begin{cases} 2, & (x,y) \in S_w, \\ 1, & (x,y) \in S_u, \\ 0, & (x,y) \in S_f, \end{cases} \tag{3.8}$$

where S_w, S_u and S_f are the set of all the wall cells, unknown cells and free space cells in the world W respectively. $p(c(x,y)|W)$ can then be represented in the form of a table.

$C_W(x,y)$ \ $C_M(x,y)$	0 (occupied)	1 (unexplained)	2 (free)
0 (wall)	0.8	0.1	0.1
1 (unknown)	0.1	0.8	0.1
2 (free)	0.1	0.1	0.8

Tab. 3.1: The lookup table for $p(c(x,y)|W)$.

In principle the likelihood $p(c(x,y)|W)$ plays the role of a sensor model. In our case it captures the quality of the original mapping algorithm producing the grid map (including the sensor models for the sensors used during the SLAM process), and could be learned from labelled training data. In the context of our application, we used some empirically determined values. An example of the values is given in Table 3.1.

3.5 Searching the Solution Space

For solving equation (3.1) we need to efficiently search the large and complexly structured solution space Ω. Here we adopt the approach of [161], who propose a data driven Markov chain Monte Carlo (MCMC) technique for this purpose. The basic idea is to construct a Markov Chain that generates samples W_i from the solution space Ω according to the distribution $p(W|M)$ after some initial burn-in time. One popular approach to construct such a Markov chain is the Metropolis-Hastings (MH) algorithm [13, 21]. In MCMC techniques the Markov chain is constructed by sequentially executing state transitions (in our case from a given world state W to another state W') according to a transition distribution $\Phi(W'|W)$ of the sub-kernels. An example of $\Phi(W'|W)$ is given in Table 3.2. In order for the chain to converge to a given distribution, it has to be reversible and ergodic [13]. The MH algorithm achieves this by generating new samples in three steps:

1. Propose a state transition according to $\Phi(W'|W)$.

2. Generate a new sample W' according to a proposal distribution $Q(W'|W)$.

3. Accept this state transition by the following probability:

$$\lambda(W, W') = \min\left(1, \frac{p(W'|M)Q(W|W')}{p(W|M)Q(W'|W)}\right) \tag{3.9}$$

The resulting Markov chain can be shown to converge to $p(W|M)$. However the selection of the proposal distribution is crucial for the convergence rate. Here, we follow the approach of [161] to propose state transitions for the Markov chain using discriminative methods, which detect relevant environmental features (e.g. walls, doorways) in a bottom-up manner. An overview of our algorithm is described in Algorithm 1.

Algorithm 1 Model generation

```
Require: input map $M$
  while certain accuracy condition not satisfied do
    if input map updated then
      update the classified map $C_M(x, y)$;
      generate unit candidates;
    end if
    denote current world state as $W$;
    select one sub-kernel according to the transition probabilities $\Phi(W'|W)$;
    generate a new world state $W'$ through a state transition using the defined discriminative bottom-up feature detectors;
    calculate the acceptance probability $\lambda(W, W')$ of the generated state according to the MH algorithm (3.9);
    draw a random float number $\theta$, $\theta \in \mathcal{U}[0, 1)$;
    if $\theta < \lambda(W, W')$ then
      accept;
      replace the current world state with $W'$, $W = W'$;
    else
      reject;
      keep current world state $W$;
    end if
  end while
```

3.5.1 MCMC Kernels

In order to design the Markov chain in form of the Metropolis-Hastings algorithm, the kernels that modify the structure of the world are arranged as reversible pairs. Currently, we use five pairs of kernels, and these include:

- Kernel pair 1: ADD or REMOVE one unit.
  - ADD: draw one new unit from certain candidates, then try to add this unit to the world.
  - REMOVE: try to cancel one existing unit from the world.
- Kernel pair 2: SPLIT one unit or MERGE two units.
  - SPLIT: try to decompose one existing unit into two units.
  - MERGE: try to combine two existing units, and generate one new unit out of them.
- Kernel pair 3: SHRINK or DILATE one unit.
  - SHRINK: try to move one wall of one unit along certain orientation, so that the unit becomes smaller.
  - DILATE: similarly to SHRINK, move one wall of one unit, so that the unit becomes bigger.
- Kernel pair 4: ALLOCATE or DELETE one door
  - ALLOCATE: draw one door from the door candidates that are provided by door detector, then try to assign it to two existing units.
  - DELETE: cancel one assigned door.
- Kernel pair 5: INTERCHANGE two units
  - Change the structure of two adjacent units at the same time.

Fig. 3.11 shows an example of the five reversible MCMC kernel pairs, using the same simplified occupancy grid map as in Fig. 3.8. The world W can transit to $W^{'}$, $W^{''}$, $W^{'''}$, $W^{''''}$ and $W^{'''''}$ by applying the sub-kernel REMOVE, MERGE, SHRINK, DELETE and INTERCHANGE, respectively. By contrast, the world $W^{'}$, $W^{''}$, $W^{'''}$, $W^{''''}$ and $W^{'''''}$ can also transit back to W using corresponding reverse sub-kernels.

3.5.2 Discriminative Generation of Chain Transition Proposals

As mentioned above, we use discriminative methods to propose candidates for MCMC kernels, so as to accelerate the convergence of the Markov chain. In the following, we define and explain the discriminative methods used with respect to corresponding MCMC kernels.

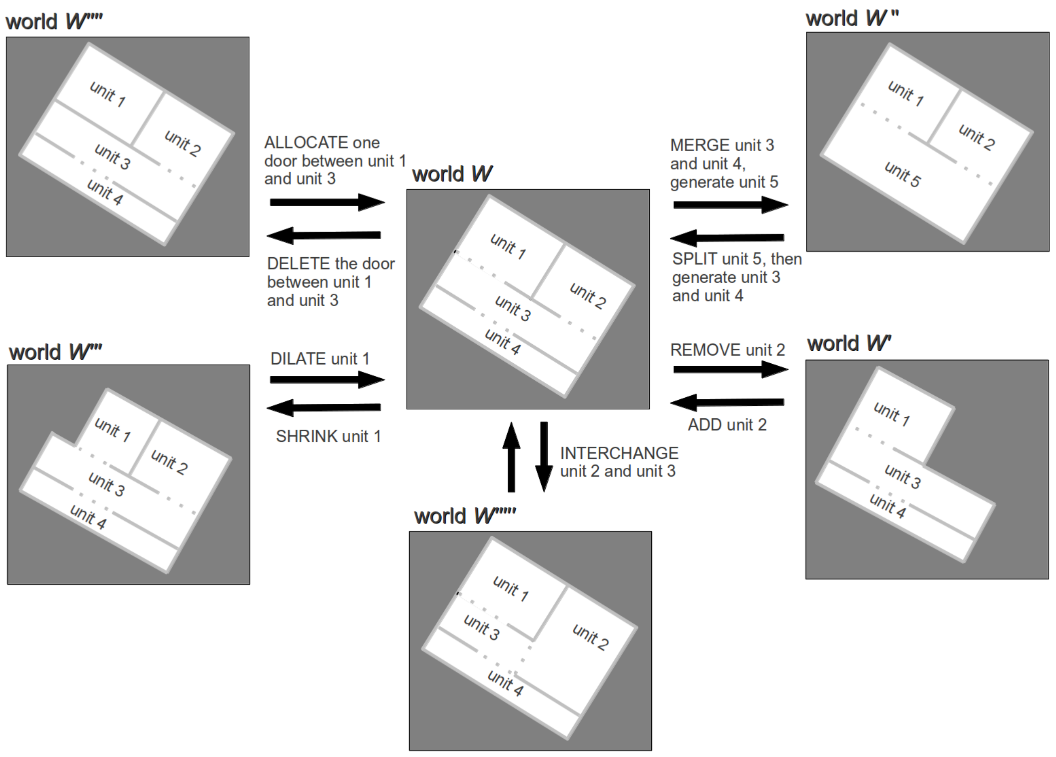

world W''''
unit 1
unit 2
unit 3
unit 4
ALLOCATE one door between unit 1 and unit 3
DELETE the door between unit 1 and unit 3
world W
world W''
MERGE unit 3 and unit 4, generate unit 5
SPLIT unit 5, then generate unit 3 and unit 4
unit 5
world W'''
DILATE unit 1
SHRINK unit 1
REMOVE unit 2
ADD unit 2
world W'
INTERCHANGE unit 2 and unit 3
world W'''''

Fig. 3.11: Reversible MCMC kernel pairs: ADD/REMOVE, SPLIT/MERGE, SHRINK/DILATE, ALLOCATE/DELETE and INTERCHANGE.

ADD

Currently, three unit detectors are adopted to propose unit candidates for ADD: a *wall based unit* (WBU) detector, a *free space unit* (FSU) detector and a *robot position based* (RPB) detector.

WBU detector This unit detector makes use of the well known Hough Transform [57] for edge based wall detection and generates unit candidates using the detected line segments according to the following procedure: We extend the detected line segments so as to divide the map into sub-areas, then these sub-areas are recombined to give new units which obey our prior information. One example of the WBU unit detector is demonstrated in Fig. 3.12. Here, the map is divided into six sub-areas: a1, a2, a3, a4, a5 and a6, and these sub-areas are recombined to generate 18 unit candidates, which are: a1, a2, a3, a4, a5, a6, a1a2, a3a4, a5a6, a1a3, a3a5, a2a4, a4a6, a1a3a5, a2a4a6, a1a2a3a4, a3a4a5a6 and a1a2a3a4a5a6. We sample from the set of all candidate units in a resampling style [74]. Each of the generated unit candidates are weighted according to how well their walls match the observations provided by the occupancy grid map. The weight of a unit ω_r is defined as the lowest wall weight ω_{w_j} among its four walls, where $j, j \in \{r_1, r_2, r_3, r_4\}$, indexes the wall, with $r_i, i \in \{1, 2, 3, 4\}$, indicating the ith wall of unit r:

$$\omega_r = \min_{j \in \{r_1, r_2, r_3, r_4\}} \omega_{w_j}. \tag{3.10}$$

The wall weight $\omega_{w_j}$is calculated as:

$$\omega_{w_j} = \frac{n(w_j)}{l(w_j)}, \tag{3.11}$$

where $l(w_j)$ indicates the length of wall w_j and can be computed from the coordinates of its two end points $(x_{w_{j,1}}, y_{w_{j,1}}), (x_{w_{j,2}}, y_{w_{j,2}})$:

$$l(w_j) = \sqrt{(x_{w_{j,1}} - x_{w_{j,2}})^2 + (y_{w_{j,1}} - y_{w_{j,2}})^2}. \tag{3.12}$$

The term $n(w_j)$ counts the number of wall cells that match with the map:

$$n(w_j) = \sum_{(x,y) \in w_j} t(x, y), \tag{3.13}$$

where

$$t(x, y) = \begin{cases} 1, & C_M(x, y) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.14}$$

Having obtained the weights of all the unit candidates, we implement $Q(W'|W)$ by sampling from the candidates according to their cumulative weights A_r. First, we normalize their weights ω_r:

$$\omega_r^{'} = \frac{\omega_r}{\sum\limits_{r \in B} \omega_r}, \tag{3.15}$$

where B indicates the set of all the unit candidates generated by the WBU detector. Then, we calculate the cumulative weights A_r for unit r:

$$A_r = \sum_{i=1}^{r} \omega_i. \tag{3.16}$$

Finally, we can draw a unit candidate n out of B, by generating a random number $k, k \in [0, 1)$,

$$n = \min\{i | k \leq A_i\}. \tag{3.17}$$

FSU detector Sometimes units will be missed by the edge-based procedure mentioned above. This is often the case for units only partially explored during grid map construction or when walls have been obstructed by furniture and thus not have been perceived by the laser scanner. We therefore use an alternative method for unit detection. This detector works on the basis of connected-components analysis and is referred to as free space unit

(FSU) detector. If there are still regions of the map which are not explained by the world after many MCMC steps (4000 steps in the experiments), then we try to find them using the FSU detector as follows: 1) make a copy of the classified map, which we denote as C'_M; 2) cancel the regions that are already explained by the current semantic world W from C'_M, we denote the rest of C'_M as C''_M; 3) detect unexplained regions in C''_M based on connected-components analysis [19], and generate unit candidates out of them. Fig. 3.13 shows one example of the FSU detector. Here, the world W does not cover the shaded area in the map, then we detect it using the FSU detector and generate unit candidates out of it. Having generated the unit candidates, we use the same sampling technique as done with the WBU detector to propose unit candidates ((3.10) to (3.17)). The number of MCMC steps after which the FSU detector is activated should be big enough so that each of the unexplained regions is relatively small and can be easily used to generate new unit candidates.

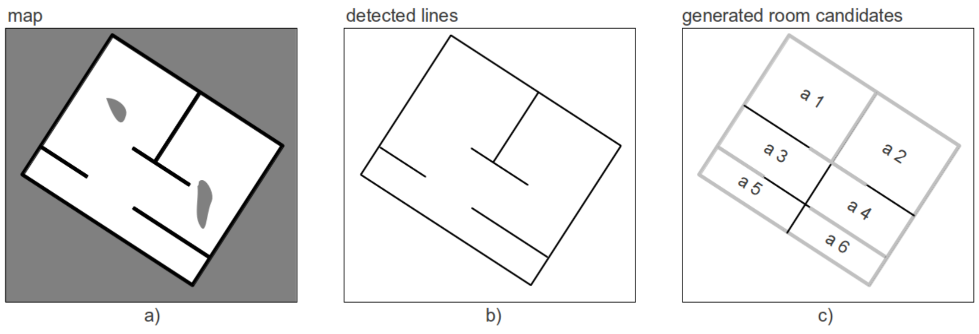


Fig. 3.12: The WBU detector. a) A simplified occupancy grid map. b) The line segments detected by the Hough line detection. c) Divide the map into sub-areas a1, a2, a3, a4, a5 and a6, by extending the detected lines. Generate unit candidates out of these sub-areas. The detected line segments are shown in light-gray.

RPB detector In addition to the detectors mentioned above, we can also use the robot position to generate new units. Following the simple idea that, where the robot currently is must be free space, we generate a small unit with certain minimum size around the robot position. This method is only activated for cases in which robot position is available. Fig. 3.14 depicts three examples of this detector.

SPLIT

For SPLIT we again use the Hough transformation based line detection to propose splitting options. Hough line detection is applied within units that already exist in the world W (member units of W). First, a unit r is randomly chosen according to a uniform

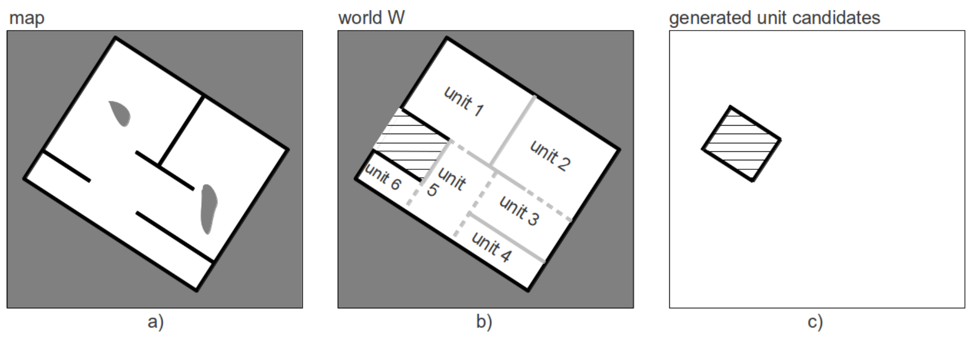


Fig. 3.13: The FSU detector. a) A simplified occupancy grid map. b) The current world W: the shaded area is not explained. c) C''_M: cancel the already explained regions from C'_M and detect unexplained regions based on connected-component analysis, then generate unit candidates out of these detected regions. In this example, a unit candidate is generated from the shaded area.

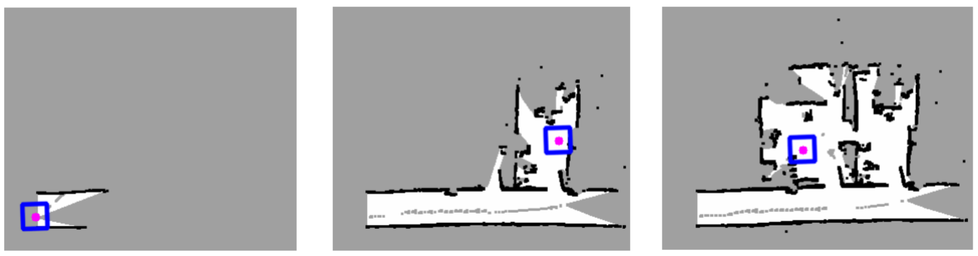


Fig. 3.14: The RPB detector. Three examples of generating a small unit of certain minimum size using robot position as center. Blue rectangles show the generated units, and the violet points show robot position.

distribution, which means every unit contained in the current world has the same chance to be chosen. Then, the Hough line detector is applied within the unit r to detect possible unit splits. Let E_r denote the set of the detected line segments within unit r. Each detected line segment $e, e \in E_r$ is weighted, using its length $l(e)$:

$$\omega_e = l(e), \tag{3.18}$$

where the length $l(e)$ is similarly calculated as done in (3.12). Then we normalize the weights and build the cumulative distribution of E_r. Furthermore, we draw one line segment out of E_r, as done with the WBU detector ((3.15) to (3.17)).

Once a line segment is chosen, it is extended, so that it intersects with the walls of the unit. Currently, only the case is accepted that two opposite walls of the unit are intersected. The case that two neighbour walls are intersected is neglected. With the extended line segment, we propose to split the unit into two units. The MH algorithm then decides whether this action is accepted. A typical example of the SPLIT sub-kernel is shown in Fig. 3.15.

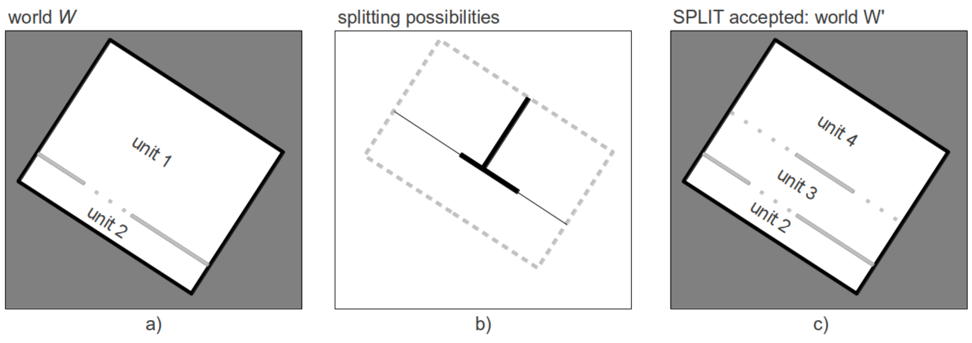


Fig. 3.15: SPLIT sub-kernel. a) The current world W. b) The unit 1 (light-gray dashed rectangle) is chosen for SPLIT. Two line segments are detected (black thick lines) as splitting possibilities. One of them is selected and extended to split the unit (the black thin line). c) After the accepted SPLIT action, the new world $W^{'}$ is created.

MERGE

The sub-kernel MERGE is the inverse of SPLIT. It tries to combine two member units of the current world W, so as to generate a new unit, then the MH algorithm decides whether the proposed new unit is accepted. To do this, the first unit r is drawn from the set of all member units R_W of world W according to a uniform distribution, which means that each member unit has the same possibility to be chosen. Additionally, a second unit s needs to be selected from the rest of the member units, $s \in R_W \setminus r$. For sampling s, we define a new weight $a_r(s)$, which is the reciprocal of the distance $d(c_r, c_s)$ between the center point c_r of unit r and the center point c_s of unit s:

$$d(c_r, c_s) = \sqrt{(c_r.x - c_s.x)^2 + (c_r.y - c_s.y)^2}, \tag{3.19}$$

where $(c_r.x, c_r.y)$ and $(c_s.x, c_s.y)$ are the grid cell index of the two center points. The weight $a_r(s)$ is calculated as follows:

$$a_r(s) = \frac{1}{d(c_r, c_s)}. \tag{3.20}$$

Subsequently, we normalize the weights $a_r(s)$, calculate the cumulative probability and draw the second unit, as done in (3.15) to (3.17). Once the two units are obtained, we try to combine them into one unit. The underlying idea for using $a_r(s)$ in the sampling is that the closer two units are spatially located, the more likely they can be combined. Fig. 3.16 illuminates an example of MERGE.

SHRINK and DILATE

The kernel pair SHRINK/DILATE tries to move a wall w_j of a member unit r in the current world W, so that this unit can better match the map. Here, $j, j \in \{r_1, r_2, r_3, r_4\}$, indexes the wall, with $r_i, i \in \{1, 2, 3, 4\}$, indicating the ith wall of unit r. For selecting the unit r from the set of all member units R_W, we define a new weight b_r:

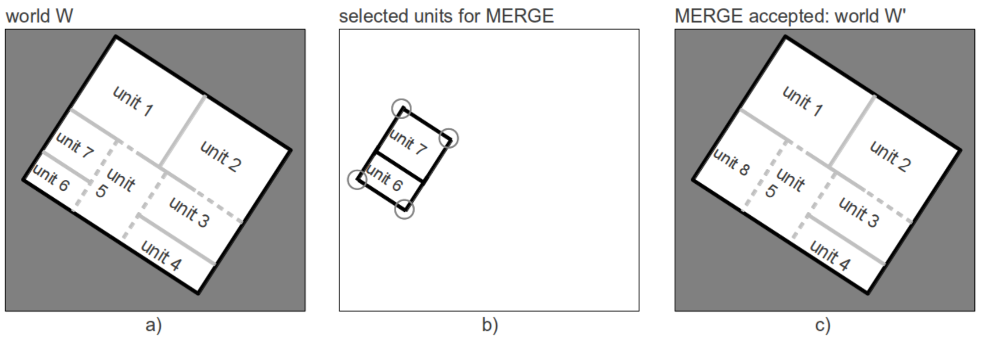


Fig. 3.16: MERGE sub-kernel. a) The current world W. b) The unit 6 is selected as the first unit, the unit 7 is selected as the second unit. Here, unit 7 and unit 5 have better chance to be chosen as the second unit than unit 1, 2, 3 and 4, because unit 5 and 7 are closer to unit 6. The four vertices of the new unit are surrounded by gray circles. c) After the accepted MERGE action, the world $W^{'}$ is generated.

$$b_r = \begin{cases} \frac{1}{\omega_r}, & \frac{1}{\omega_r} \leq h_b \\ h_b, & \text{otherwise}, \end{cases} \tag{3.21}$$

where ω_r is the unit weight defined in (3.10). h_b is a predefined threshold for the weight. Using b_r, a unit is drawn according to (3.15) to (3.17). The reason for this is that the worse a unit matches the map, the more likely it should be changed by SHRINK/DILATE.

Once the unit is selected, one wall w_j needs to be drawn from its four walls. Following the same idea, we define a new weight v_{w_j} for sampling the wall:

$$v_{w_j} = \begin{cases} \frac{1}{\omega_{w_j}}, & \frac{1}{\omega_{w_j}} \leq h_v \\ h_v, & \text{otherwise}, \end{cases} \tag{3.22}$$

where ω_{w_j} is the wall weight defined in (3.11), and h_v is a predefined threshold. Again, the wall is drawn according to v_{w_j}, as done in (3.15) to (3.17). After the wall is selected, we try to shift it parallel to its original orientation using a bias that is drawn from a zero-mean Gaussian distribution. In principle, the algebraic sign of the selected bias decides whether a SHRINK or a DILATE is proposed, e.g. if a positive sign proposes a SHRINK, then a negative sign will propose a DILATE. In general, SHRINK and DILATE sub-kernel have both 50% chance to be proposed. An example of the SHRINK/DILATE kernel pair is shown in Fig. 3.17.

ALLOCATE

A door detector which is based on connected-components analysis [19] proposes door candidates for the sub-kernel ALLOCATE. We draw one door candidate from the set of all candidates according to their weights. Here, the weight ω_g of a door g is similar to the weight of walls ω_{w_j} that is defined in (3.11):

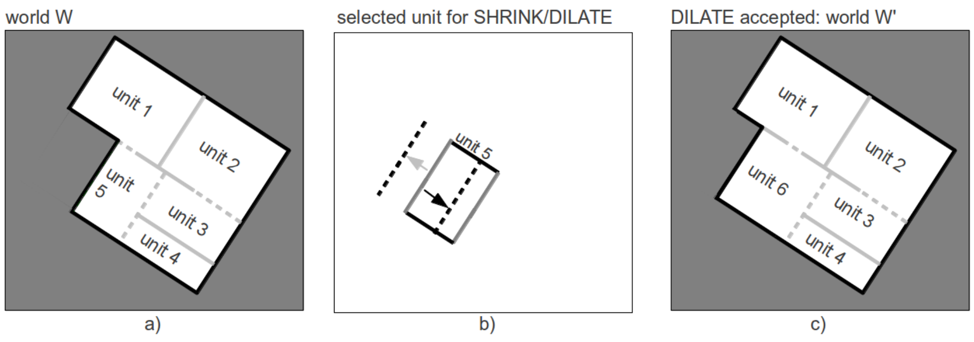


Fig. 3.17: SHRINK/DILATE kernel pair. a) The current world W. b) The unit 5 is selected. Here, the two gray walls are more likely to be chosen, because they match the map worse than the other two black walls. We assume that the left gray wall is selected for SHRINK/DILATE. The light-gray arrow points to one DILATE possibility, whereas the black arrow points to a SHRINK possibility. c) After the accepted DILATE action, the world $W^{'}$ is generated.

$$\omega_g = \frac{n^{'}(g)}{l(g)}, \tag{3.23}$$

where $l(g)$ is calculated the same as in (3.12), and $n^{'}(g)$ is computed as follows:

$$n^{'}(g) = \sum_{(x,y)\in g} t^{'}(x,y), \tag{3.24}$$

where

$$t^{'}(x,y) = \begin{cases} 1, & C_M(x,y) = 2, \\ 0, & \text{otherwise.} \end{cases} \tag{3.25}$$

Using the weight ω_g, one door candidate is drawn from the set of all detected candidates, as done in (3.15) to (3.17). Then, the MH algorithm decides whether this door will be accepted. In the following, we detail how to detect door candidates.

According to the indisputable fact that doors must be located on walls in the real world, we search doors along walls in the structured world in the following steps:

1) Expand each wall in its perpendicular direction, so that a rectangular area is created out of each wall. Note that each wall should be shortened before the expansion, so that the extended area does not overlap each other within one unit.

2) Detect the overlap of these rectangles using connected-components analysis, because the overlap area indicates on which wall the potential door candidates can be found. Localize the wall part that has caused the overlap.

3) Divide the localized wall part averagely into several small segments, so that each segment is equally long. Because each of these segments could be a part of a door,

we weight them according to (3.23) and try to combine the verified segments to build a door (we define a segment as a verified segment, if its weight is bigger than certain weight threshold).

4) Combine the verified segments on each wall, if the distance between them is lower than certain distance threshold. Find the corresponding part of the combined segments on both walls and use them as door candidates.

The above process is demonstrated in Fig. 3.18.

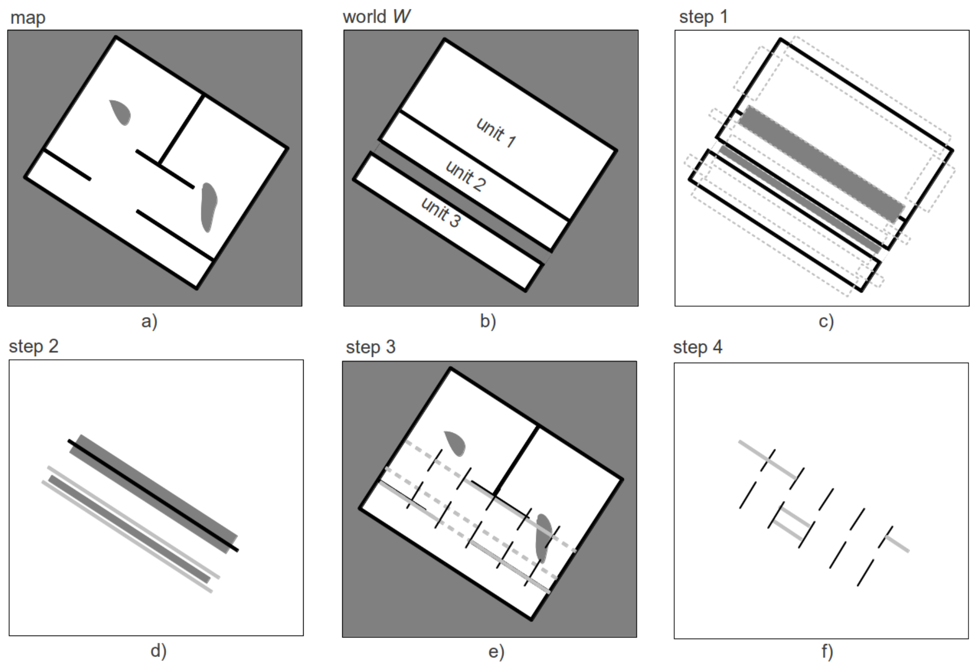


Fig. 3.18: ALLOCATE sub-kernel. a) The occupancy grid map. b) The current world W with no assigned doors. c) Step 1: each wall is shortened and expanded. The expanded areas are marked by light-gray dashed rectangles. The rectangles filled with gray show the overlap areas. d) Step 2: the two pairs of walls that have caused the overlap areas are localized. Note that the black line indicates a pair of overlapping walls, and the other pair of walls is shown in light-gray. e) Step 3: divide the two pairs of walls into six segments, using the black lines. Weight each segment according to how well they match the map. Verified segments are shown in dashed lines, other segments are drawn in solid lines. f) Step 4: combine those verified segments, between which the distance is under certain threshold. Find the corresponding part of the combined segments and use them as door candidates (light-gray lines). Note, all the segments on one wall are detected as verified segment in step 3, which means this whole wall forms a big combined segment, but the final door candidate must correspond to the door candidate on the other wall. That is the reason why only a small door candidate is detected on this wall pair.

After a successful ALLOCATE action, one door is assigned to two units, and each assigned door contains the following information: ID of the two units, ID of the walls on which the door is located, grid cell indices of the door.

INTERCHANGE

The kernel INTERCHANGE tries to restructure two member units of the current world W, so as to generate new solutions. The MH algorithm decides then whether the proposed new solution is accepted. Similar to MERGE, the first unit r is drawn from the set of all member units R_W of world W according to a uniform distribution, which means that each member unit has the same possibility to be chosen. Additionally, a second unit s needs to be selected from the rest of the member units, $s \in R_W \setminus r$. For sampling s, we use the weight $a_r(s)$, which is the reciprocal of the distance $d(c_r, c_s)$ between the center point c_r of unit r and the center point c_s of unit s:

$$d(c_r, c_s) = \sqrt{(c_r.x - c_s.x)^2 + (c_r.y - c_s.y)^2}, \tag{3.26}$$

where $(c_r.x, c_r.y)$ and $(c_s.x, c_s.y)$ are the grid cell index of the two center points. The weight $a_r(s)$ is calculated as follows:

$$a_r(s) = \frac{1}{d(c_r, c_s)}. \tag{3.27}$$

Subsequently, we normalize the weights $a_r(s)$, calculate the cumulative probability and draw the second unit, as done in (3.15) to (3.17). Once the two units are obtained, we try to restructure them. The underlying idea for using $a_r(s)$ in the sampling is that the closer two units are spatially located, the more likely they can be restructured together. Fig. 3.19 illuminates an example of INTERCHANGE. Through INTERCHANGE the world W can transit to W' and vice versa.

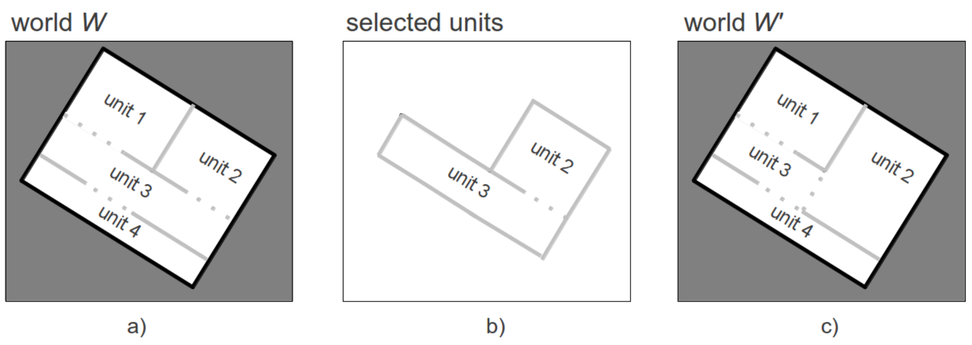


Fig. 3.19: INTERCHANGE. a) The current world W. b) The units 2 and 3 are selected for INTERCHANGE. c) Through INTERCHANGE the world W can transit to W' (a $\rightarrow$ b $\rightarrow$ c) and vice versa (c $\rightarrow$ b $\rightarrow$ a).

REMOVE and DELETE

The sub-kernel REMOVE and DELETE have similar functionality, which is to cancel one existing member unit and one assigned door respectively. There are no special discriminative methods used for these two sub-kernels. They just draw one member from the corresponding set (existing units or assigned doors) and propose to cancel this member, then the MH algorithm decides whether this proposal is accepted. Following the idea that the worse a member matches the map, the more likely it should be cancelled, we use the weight b_r defined in (3.21) for unit sampling. Similarly, we define a new weight z_g for door sampling:

$$z_g = \begin{cases} \frac{1}{\omega_g}, & \frac{1}{\omega_g} \leq h_g \\ h_g, & \text{otherwise,} \end{cases} \tag{3.28}$$

where ω_g is the door weight defined in (3.23), and h_g is a predefined threshold.

3.5.3 Proposal Probability

The proposal probability $Q(W^{'}|W)$ describes how probable it is that the world W transits to the world $W^{'}$, and by contrast, $Q(W|W^{'})$ is the probability for transiting back to the world W, given the world $W^{'}$. Intuitively, $Q(W^{'}|W)$ is the product of the normalized weight of the selected elements (unit candidate, splitting line, wall etc.) in the corresponding MC sub-kernel defined in the previous section. For instance, in the ADD or REMOVE sub-kernel, $Q(W^{'}|W)$ is equal to the corresponding normalized weight of the selected unit candidate or that of the selected member unit. $Q(W^{'}|W)$ in ALLOCATE and DELETE can be calculated similarly to that in ADD and REMOVE respectively. In SPLIT, $Q(W^{'}|W)$ is the product of the corresponding normalized weight of the selected member unit and that of the selected splitting line. In SHRINK/DILATE, $Q(W^{'}|W)$ is product of three terms: the corresponding normalized weight of the member unit, that of the selected wall and that of the generated Gaussian bias. Similarly, $Q(W^{'}|W)$ of MERGE and INTERCHANGE is calculated as the product of the corresponding normalized weight of the first unit and that of the second unit.

Compared with $Q(W^{'}|W)$, the calculation of $Q(W|W^{'})$ is less intuitive, because the back transition is virtual and must be defined. For ADD, $Q(W|W^{'})$ should perform the same function as the sub-kernel REMOVE, namely, the world W' transits back to the world W by cancelling the unit that is added in the transition from W to W', thus $Q(W|W^{'})$ of ADD should be the normalized weight of the added unit in the sub-kernel REMOVE. $Q(W|W^{'})$ of REMOVE can also be calculated as the normalized weight of the unit, that is canceled in the transition from W to W', in the sub-kernel ADD. Analogously, $Q(W|W^{'})$ of SHRINK, DILATE, MERGE, SPLIT, ALLOCATE and DELETE can be calculated in a style similar to $Q(W^{'}|W)$ in their corresponding reverse sub-kernels. In addition, the SHRINK/DILATE pair just tries to move one wall of the selected unit using a relatively small bias, thus the resulting world W' is similar to W. For computational simplicity, we assume that $Q(W^{'}|W)$ and $Q(W|W^{'})$ are equal in the SHRINK/DILATE pair and in the INTERCHANGE pair.

iteration β / sub-kernel	$\beta \leq 1000$	$1000 < \beta \leq 4000$	$\beta > 4000$
ADD	0.8	0.05	0.05
REMOVE	0.2	0.05	0.05
SPLIT	0	0.15	0.15
MERGE	0	0.15	0.15
SHRINK	0	0.15	0.15
DILATE	0	0.15	0.15
INTERCHANGE	0	0.3	0.2
ALLOCATE	0	0	0.05
DELETE	0	0	0.05

Tab. 3.2: Transition probabilities $\Phi(W'|W)$ of MC sub-kernels.

3.6 Intermediate Results Using Only Weak Context Knowledge as Prior

We apply our algorithm to several occupancy grid maps. The selection probabilities of the MC sub-kernels are listed in Table 3.2. Here, the selection probabilities depend partially on the iteration index β. At the beginning ($\beta \leq 1000$), the world W does not contain much information about the map, so we mainly apply ADD to propose new units into the world. For $\beta > 1000$, the selection probability of ADD is set to be very low (0.05), because most part of the map is already explained during the initial exploration ($\beta \leq 1000$). In addition, we activate SPLIT, MERGE, SHRINK, DILATE and INTERCHANGE to change the form of the member units of the world. For $\beta > 4000$, we activate ALLOCATE and DELETE, so that the connectivity information is explored and attached to the world. Moreover, the FSU detector is also activated for $\beta > 4000$ to detect left-over free space regions. Table 3.2 effectively implements a heuristic scheduling policy.

Fig. 3.20 depicts the process of Markov chain convergence by showing the evolution of the log posterior $\log(P(W|M))$ along the development of the Markov chain on the left side. In an initial burn-in process the chain quickly approaches its target distribution $P(W|M)$. This is indicated by the rapid increase of the posterior in the beginning. We can also see the discontinuities at the iteration 1000 and 4000 which show the effect of the scheduling policy. The jump at iteration 1000 is a consequence of the activation of new transition kernels that greatly improve the system capability to structurally adapt the world state W

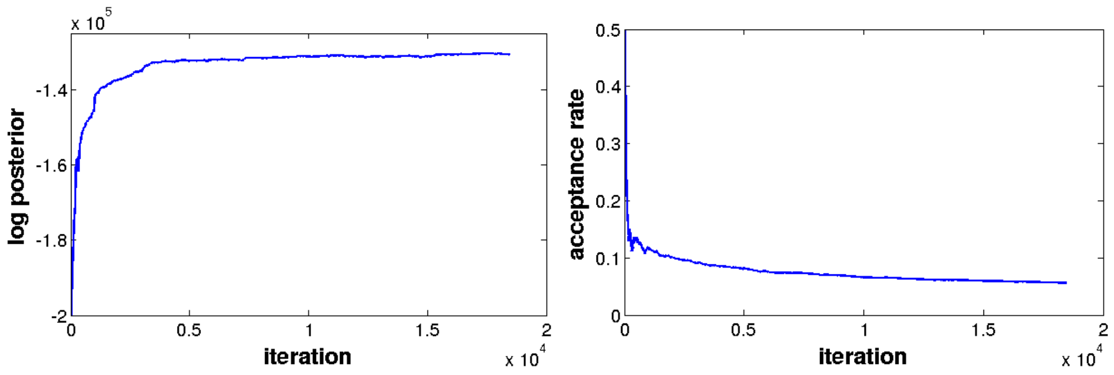


Fig. 3.20: The typical development of the posterior probability $P(W|M)$ (left) for the input map shown in Fig. 3.21 and the acceptance rate of the proposed state transitions (right) along the Markov chain development in terms of iterations.

to the observations in the map. After the initial phase, the chain samples from $P(W|M)$ and produces samples that are slight variations of the world W and do not significantly improve the situation any more.

This convergence process can also be seen by looking at the development of the acceptance rate of the Metropolis-Hastings algorithm (see Fig. 3.20, right). In the early phases, the acceptance rate is comparatively high, which means that most of the transitions proposed by $Q(W'|W)$ are accepted, since they correspond to a significantly improved explanation of the map M by the model W'. Towards the end, the acceptance rate stabilizes on a low level.

Fig. 3.21 shows a typical result of the overall process. Here, part a) shows an original input occupancy map M [58]. Part b) shows the classified map $C_M(x, y)$ that is defined in (3.7), with black, gray and white indicating occupied, unexplained and free cells respectively. Part c) visualizes the world state W representing our structured semantic model. Here black, gray, white and cyan show the wall, unknown, free and door way cells respectively. In part d), walls (blue) and doors (cyan) of the world W are directly plotted onto the original input map M, so as to give a more intuitive comparison.

Fig. 3.22 depicts an example of the underlying MCMC sampling. The MCMC sampling starts in Fig. 3.22-a. By applying state transitions realized by the defined kernels shown in Fig. 3.11, the sampled semantic world is gradually changed with respect to the input map. Three subsequent intermediate results are illustrated by Fig. 3.22-b, c and d. As shown in these intermediate results, the semantic world is adapted to better match the input map.

A result using a different input map [58] is illustrated in Fig. 3.23. Compared with the map used in Fig. 3.21, this map is relatively simple. Fig. 3.24 shows the progression of chain convergence in terms of log-posterior and acceptance rates. Again, we can clearly identify the burn-in phase and the effects of our scheduling policy at iterations 1000 and 4000.

The computational cost strongly depends on the size of the occupancy grid map, and

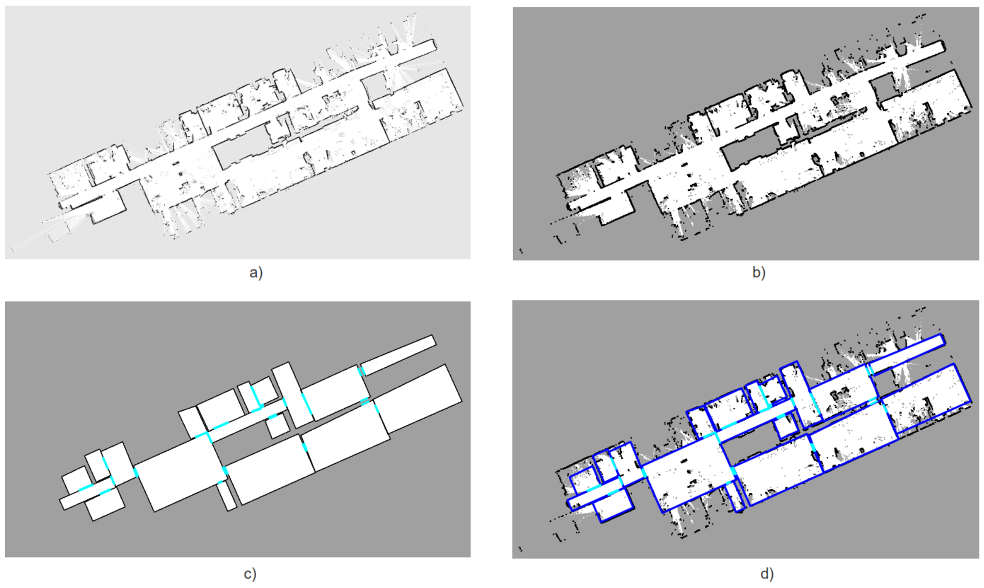


Fig. 3.21: Analysis of the "ubremen-cartesium" dataset [58]. a) The occupancy grid map M. b) The classified map $C_M(x, y)$ with three intensity values (black=wall, grey=unexplained, white=free). c) The analyzed world W (black=wall, gray=unknown, white=free, cyan=door). d) Walls (blue) and doors (cyan) of the world W drawn into the original map M.

the major part of the computation is spent in the evaluation of the generative model. The computation speed of analyzing the map in Fig. 3.21 (size:1237×672) is around 30 iterations per second (ips), and in general it takes about 10000 to 20000 iterations, until the Markov chain reaches a good state, so the computation time on the current PC is around 5 to 10 minutes. By contrast, the map used in Fig. 3.23 is much smaller (size:556×322), and the same computer reaches around 140 ips, which leads to an overall computation time of 1 to 2 minutes. Currently, we use a single-threaded implementation, where at each iteration only one sub-kernel is tested for the sampling. One of the most important features of the Markov chain is that the current state is only dependent on the previous one, therefore, it is theoretically possible to do multiple tests using different sub-kernels at each iteration, then only the successful test results are saved for the sampling. Using today's powerful off-the-shelf multi-core CPUs, this idea should lead to a much less computation time.

3.7 The Big Challenge: Topological Defects

So far we effectively produced samples from $P(W|M)$, which represents the distribution of probable worlds W given the observation M, using MCMC sampling. However, it is not enough to use weak context knowledge only to define the prior distribution, which simply describes the very basic features of the environments (rectangularity and overlapping). In this way, the resulting prior distribution is not peaked enough to provide topological constraints to the sampling process. Although the obtained worlds provide a good cell-wise

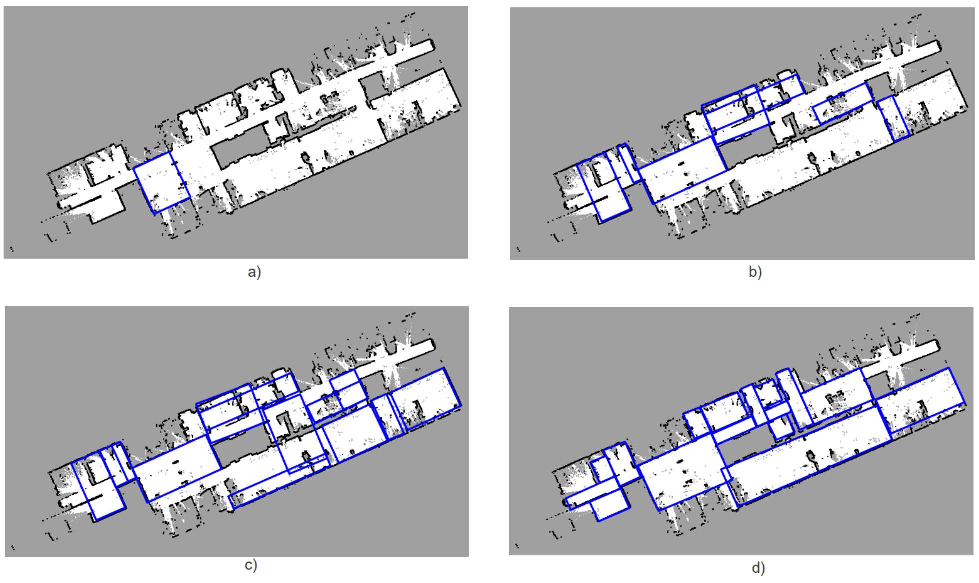


Fig. 3.22: The process of MCMC sampling starts from a random initial guess (figure a). By applying the kernels shown in Fig. 3.11, the semantic world model is adapted to the input map, three subsequent intermediate results are demonstrated in figure b, c and d.

matchness to input maps, they contain topological defects which look wrong for a human observer, i.e., as human observers we do not consider these solutions as optimal.

Here we take the sampled worlds of the input map shown in Fig. 3.21 as an example. Two alternative explanations of this input map are depicted in Fig. 3.25 and Fig. 3.26. In Fig. 3.27, these two samples are compared with the result shown in Fig. 3.21. The topological defects are marked by red arrows. Using a more complex understanding of typical architecture, it is not difficult to identify these topological defects and to sketch a almost perfect explanation for this input map (see Fig. 3.28). In order to generate such explanations, we need to extend the abstract parametric model with semantically meaningful terms, based on which more advanced context knowledge can be applied to describe the environments more precisely, i.e., rule-based context knowledge.

3.8 Semantic Extension of the Parametric Model

In this section we show in detail how to instantiate and adopt our knowledge-supervised MCMC for the task “semantic robot indoor mapping”. For this purpose we need to extend our current parametric model with semantically meaningful, abstract terms. The extended model provides a semantic level explanation (such as “type” and “relation”) and geometrical estimation of the perceived environment. Explanation of our model is given in Fig. 3.29.

Our parametric model explains indoor environments in terms of basic indoor space

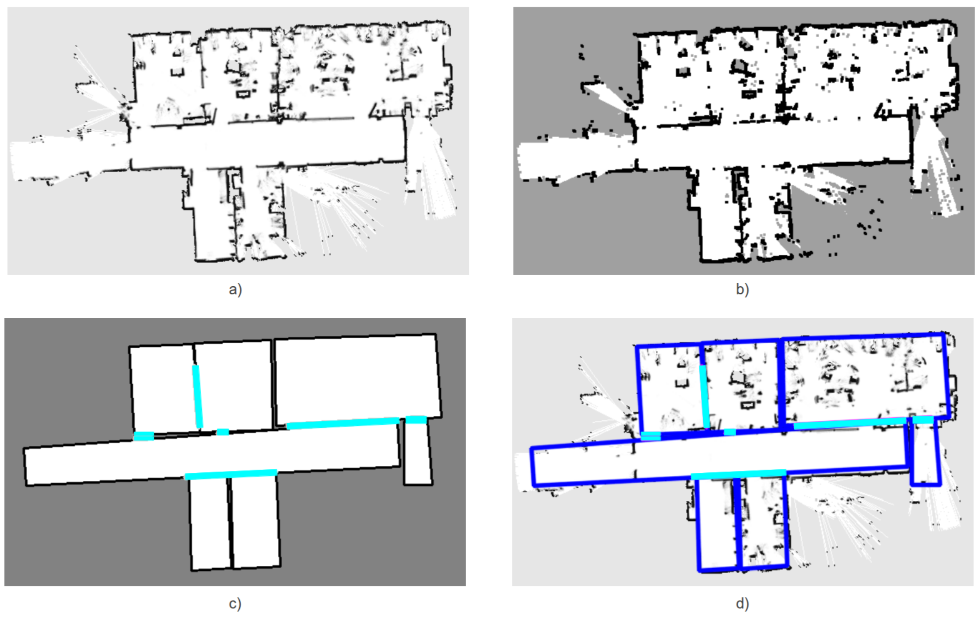


Fig. 3.23: Analysis of the "albert-b-laser" dataset [58]. a) The occupancy grid map. b) The classified map (black=wall, grey=unexplained, white=free). c) The analyzed world (black=wall, gray=unknown, white=free, cyan=door). d) Walls (blue) and doors (cyan) of the world drawn into the original map.

types, such as "room", "corridor", "hall" and so on, and we denote it as W:

$$W := \{U, T, R\}, \tag{3.29}$$

where $U = \{u_i | i = 1, \dots, n\}$ represents the set of all n units. Each unit u_i has a rectangle shape, and its geometry (size, position and orientation) is represented by its four vertices. The four edges of a unit are its walls. Doors are small line segments of free cells that are located in walls and connect to another unit. Unknown cells of the input map that are located within a unit are considered as object cells. All cells within a unit that do not belong to object cells are considered as free space of the unit. $T = \{t_i | i = 1, \dots, n\}$ is the set of type of each individual unit, with $t_i \in \{\text{room,corridor,hall,other}\}$. Here "other" indicates unit types that are not "room", "corridor" or "hall". $R = \{r_{p,q} | p = 1, \dots, n; q = 1, \dots, n\}$ is a $n \times n$ matrix, whose element $r_{p,q}$ describes the relation between the unit u_p and the unit u_q, with $r_{p,q} = r_{q,p} \in \{\neg\text{adjacent,adjacent}\}$. If two units share a wall, we define their relation as "adjacent", otherwise "$\neg$adjacent". By default, we define a unit u_p is not adjacent to itself, i.e. $r_{p,p} = \neg\text{adjacent}$. In the following, we call each instance of the parametric abstract model a "semantic world" or "world".

For evaluating how well a semantic world W matches with the input grid map M, we still use the posterior probability defined in (3.2):

$$p(W|M) \propto p(M|W) \cdot p(W).$$

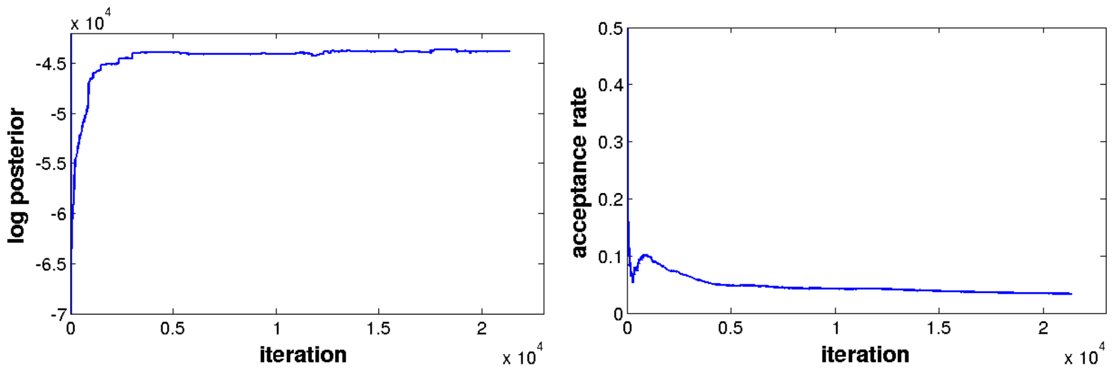


Fig. 3.24: The typical development of the posterior probability $P(W|M)$ (left) for the input map shown in Fig. 3.1 and the acceptance rate of the proposed state transitions (right) along the Markov chain development in terms of iterations.

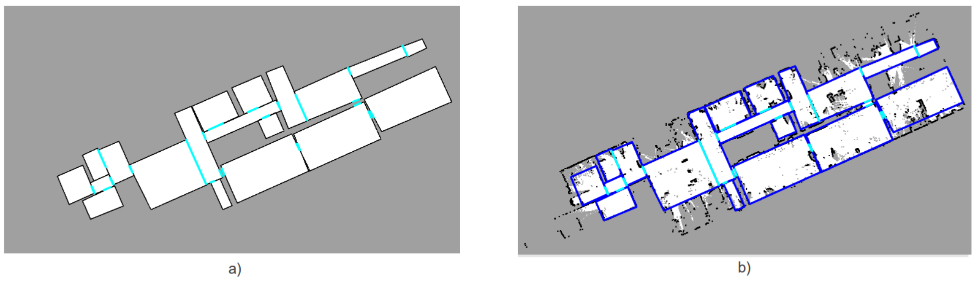


Fig. 3.25: Alternative explanation 1 of the "ubremen-cartesium" dataset [58].

Here, the term $p(M|W)$ is usually called *likelihood* and indicates how probable the input is for different worlds. The term $p(W)$ is called *prior* and describes the belief on which worlds are possible at all. In the following, we formulate task-specific context knowledge as descriptive rules in Markov Logic Networks (MLNs) [117] and show how to define the likelihood and the prior using the inference results of MLNs in a systematic way. For details on MLNs, we refer to [117].

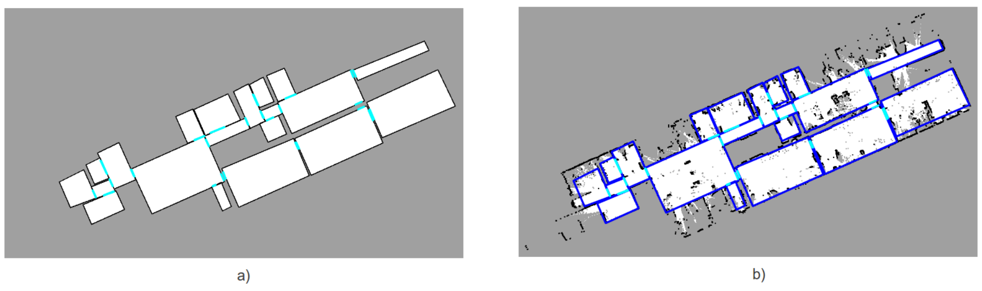


Fig. 3.26: Alternative explanation 2 of the "ubremen-cartesium" dataset [58].

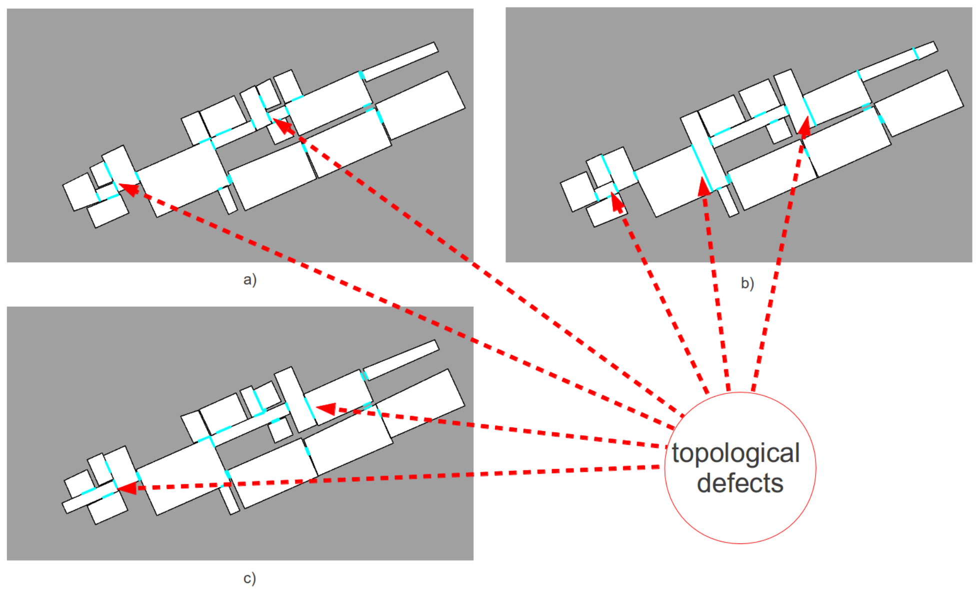


Fig. 3.27: Topological defects of the solutions for the "ubremen-cartesium" dataset [58].

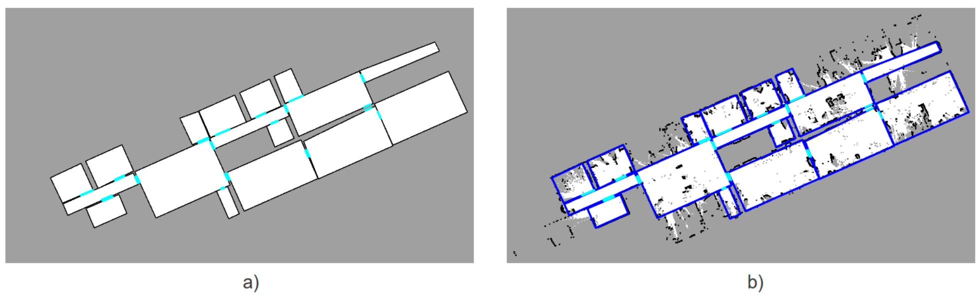


Fig. 3.28: A almost perfect explanation for the "ubremen-cartesium" dataset [58].

3.8.1 Inference Using Rule-Based Context Knowledge

In general, context knowledge describes our prior belief for a certain domain, such as that the ground becomes wet after it has rained. Rather than exact quantitative information, context knowledge provides advisory qualitative information for our judgements. Such information is very valuable in handling problems of high dimensionality where computation suffers due to the huge state space. In the domain of robot indoor mapping, we formulate following context knowledge:

- There are four types of space units: room, corridor, hall and other.
- Two units are either adjacent (neighbours) or not adjacent.
- The type of a unit is dependent on its geometry and size.
- In contrast to rooms, corridors have multiple doors.
- Connecting walls of two adjacent rooms have the same length.

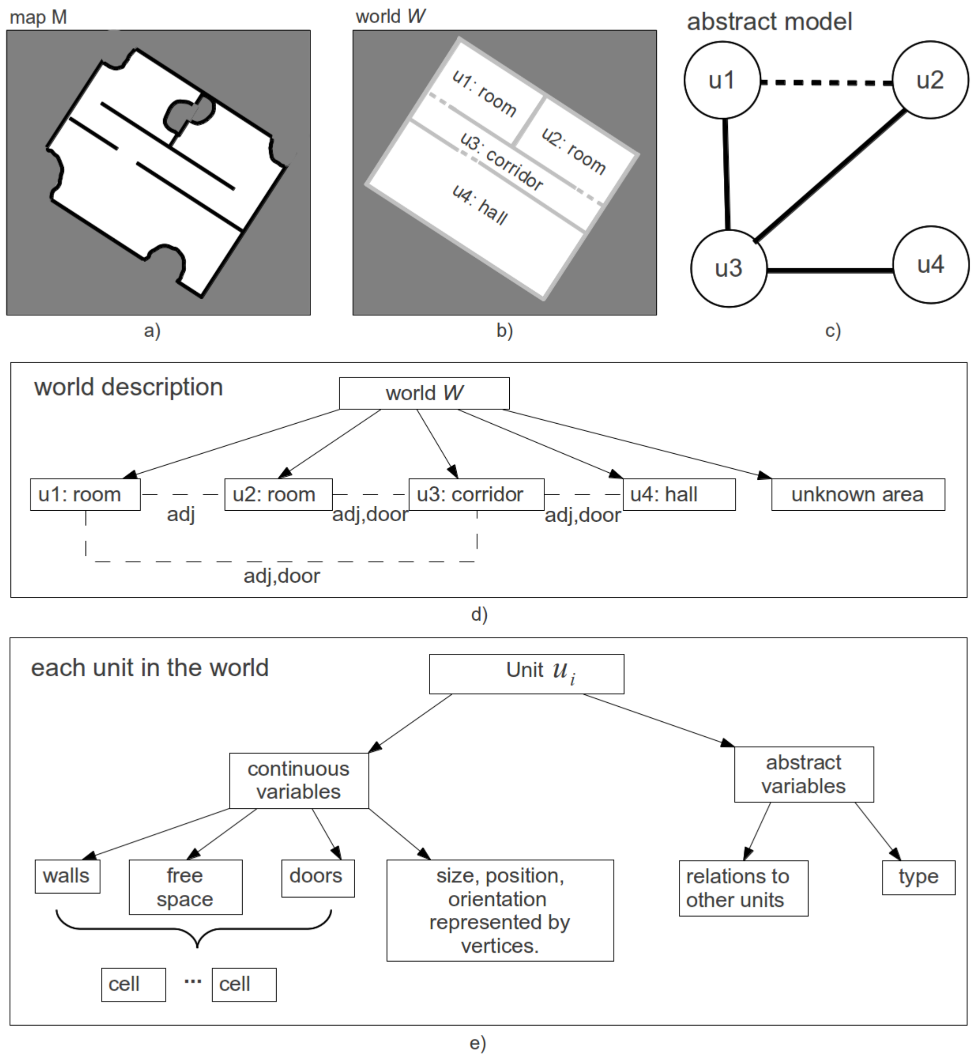


Fig. 3.29: The extended abstract parametric model. a) Input occupancy grid map (gray=unknown, white=free, black=occupied). b) The corresponding parametric model (gray=unknown, white=free, solid light-gray=walls, dashed light-gray=doors), while assuming the fundamental units have rectangle shape (represented by their four vertices). The units are explained as room, corridor and hall. c) The abstract representation. d) The semantic world described as a scene graph. If two units share a wall, then they are adjacent (e.g. u_1 and u_3). In addition, connectivity between two units through a door is also detected. e) Each unit in the world contains continuous and abstract variables. The four edges of each unit are its walls. Doors are small line segments comprised of free cells that are located on walls and connect to another unit. All the cells within a unit are considered to belong to free space of the unit. Size, position and orientation of each unit are implicitly represented by its four vertices.

With the help of MLNs, we formulate our context knowledge as descriptive rules in Table 4.3. Based on these rules, query defined in Table 4.2 can be made given evidence shown in Table 4.1. Using these rules, we try to formulate the features of certain indoor environments with rectangular space units. The choice of the rules is a problem-oriented engineering step, and the rules shown here serve as a good example.

predicate	explanation
$RoLi(u_p)$	Unit u_p has a room-like geometry.
$CoLi(u_p)$	Unit u_p has a corridor-like geometry.
$HaLi(u_p)$	Unit u_p has a hall-like geometry.
$MulDoor(u_p)$	Unit u_p has multiple doors.
$Adj(u_p, u_q)$	Unit u_p and u_q are adjacent.

Tab. 3.3: Definition of evidence predicates

predicate	explanation
$Room(u_p)$	Unit u_p has the type of room.
$Corr(u_p)$	Unit u_p has the type of corridor.
$Hall(u_p)$	Unit u_p has the type of hall.
$Other(u_p)$	Unit u_p has the type of other.
$SaLe(u_p, u_q)$	Unit u_p and u_q have each a wall with the same length.

Tab. 3.4: Definition of query predicates

Before we can make inference in MLNs, the evidence defined in Table 4.1 need be provided as input for MLNs, which includes geometry evidence, relation evidence and evidence on doors. To provide the first, we use a classifier that categorizes the geometry of a unit into "room-like", "corridor-like" or "hall-like" according to its size and length/width ratio. The general idea of this classifier is shown in Table 3.6.

Relation evidence is detected based on image processing techniques: we first dilate all four walls of each unit, and then relation $r_{p,q}$ between the unit u_p and u_q is decided according to connected-components analysis [19]. An example of relation detection is depicted in Fig. 3.30, where R is given by

$$R = \begin{bmatrix} \neg\text{adj} & \text{adj} & \neg\text{adj} \\ \text{adj} & \neg\text{adj} & \text{adj} \\ \neg\text{adj} & \text{adj} & \neg\text{adj} \end{bmatrix}. \tag{3.31}$$

basic features:
$Adj(u_p, u_q) \rightarrow Adj(u_q, u_p)$
$SaLe(u_p, u_q) \rightarrow SaLe(u_q, u_p)$
reasoning on type:
$HaLi(u_p) \rightarrow Hall(u_p)$
$HaLi(u_p) \rightarrow \neg Room(u_p)$
$HaLi(u_p) \rightarrow \neg Corr(u_p)$
$HaLi(u_p) \rightarrow \neg Other(u_p)$
$RoLi(u_p) \rightarrow \neg Hall(u_p)$
$CoLi(u_p) \rightarrow \neg Hall(u_p)$
$RoLi(u_p) \wedge \neg MulDoor(u_p) \rightarrow Room(u_p)$
$RoLi(u_p) \wedge \neg MulDoor(u_p) \rightarrow \neg Corr(u_p)$
$RoLi(u_p) \wedge MulDoor(u_p) \rightarrow Other(u_p)$
$CoLi(u_p) \wedge \neg MulDoor(u_p) \rightarrow Other(u_p)$
$CoLi(u_p) \wedge MulDoor(u_p) \rightarrow Corr(u_p)$
$CoLi(u_p) \wedge MulDoor(u_p) \rightarrow \neg Room(u_p)$
reasoning on $SaLe$:
$\neg Adj(u_q, u_p) \rightarrow \neg SaLe(u_p, u_q)$
$Room(u_p) \wedge Room(u_q) \wedge Adj(u_p, u_q) \rightarrow SaLe(u_p, u_q)$
$Room(u_p) \wedge Hall(u_q) \wedge Adj(u_p, u_q) \rightarrow \neg SaLe(u_p, u_q)$
$Room(u_p) \wedge Corr(u_q) \wedge Adj(u_p, u_q) \rightarrow \neg SaLe(u_p, u_q)$
$Hall(u_p) \wedge Corr(u_q) \wedge Adj(u_p, u_q) \rightarrow \neg SaLe(u_p, u_q)$
$Other(u_p) \wedge Hall(u_q) \wedge Adj(u_p, u_q) \rightarrow \neg SaLe(u_p, u_q)$
$Other(u_p) \wedge Corr(u_q) \wedge Adj(u_p, u_q) \rightarrow \neg SaLe(u_p, u_q)$
$Other(u_p) \wedge Room(u_q) \wedge Adj(u_p, u_q) \rightarrow \neg SaLe(u_p, u_q)$

Tab. 3.5: Context knowledge defined as descriptive rules

Similar to relation detection, doors are detected as small open line segments which are located on the connecting walls of two neighbour space units. Details on door detection can be found in section 3.5.2.

Given necessary evidence, we can make inference in MLNs and use the inference results to calculate the prior and likelihood. In this work, we have used the ProbCog Toolbox [67] to perform MLN inference. Currently, we use hard evidences for knowledge processing, however, our system is also able to process soft evidences, as long as the evidences are provided in the soft form.

size \ ratio	small	big
small	room-like	corridor-like
big	hall-like	hall-like

Tab. 3.6: The classifier providing geometry evidence.

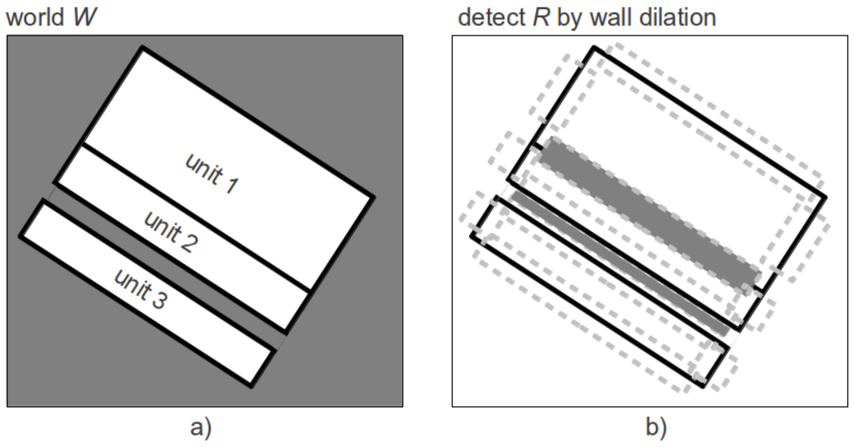


Fig. 3.30: An example of relation detection. **a)** A semantic world W containing three units (black=wall, white=free, gray=unknown). **b)** All four walls of each unit are dilated, with dashed rectangles in light-gray representing the dilated walls. The overlap of the dilated walls is shown in dark-gray which indicates the relation of "adjacent". The overlap is detected using connected-components analysis [19]. In this example, unit 1 and unit 3 are not adjacent; unit 2 and unit 3 are adjacent; unit 1 and unit 2 are adjacent.

3.8.2 Inference-Based Prior and Likelihood Redefinition

Prior

According to the model definition in equation (3.29), the prior $p(W)$ is given by

$$\begin{aligned} p(W) &= p(U,T,R)\cdot\alpha_1\cdot\alpha_2 \\ &= p(U|T,R)\cdot p(T,R)\cdot\alpha_1\cdot\alpha_2. \end{aligned} \tag{3.32}$$

Here, α_1 and α_2 are the factors describing the very basic environmental features (rectangularity and overlap) defined in equation (3.3). $p(U|T,R)$ can be seen as a factor expressing the dependency of the geometry parameters of the underlying units (see Fig. 3.29-e) on the abstract terms in the MLNs. In our case, the geometry (size, position and orientation) of a unit is described by its four vertices. Furthermore, we define

$$p(U|T,R) := \eta \prod_{p,q\in[1,2,\ldots,n]} b(u_p,u_q), \tag{3.33}$$

with

$$b(u_p, u_q) = \begin{cases} e^{-\frac{d^2}{2\sigma^2}}, p(SaLe(u_p, u_q)|\text{evidence}) > \text{threshold} \\ \qquad \text{and} \;\; p \neq q, \\ 1, \;\; \text{otherwise}, \end{cases} \tag{3.34}$$

where n is the total number of units, and d represents the length difference of the connecting walls of two adjacent units. $e^{-\frac{d^2}{2\sigma^2}}$ indicates a Gaussian function with mean at zero. $p(SaLe(u_p, u_q)|\text{evidence})$ is one of the inferences that we can make in MLNs. η is the normalization factor which ensures that $p(U|T, R)$ integrates to one. At the current stage, we assume that $p(T, R)$ follows a uniform distribution. However, it is possible to learn this distribution given proper training data.

So far, the prior $p(W)$ is defined based on the inference results of MLNs, which enforces that the semantic worlds that comply with the context knowledge have high prior probability. Note that the worlds that contradict the context knowledge are not given a zero prior probability, instead, they become less probable.

Likelihood

Let $c(x, y)$ be the grid cell with the coordinate (x, y) in the input map M, then we define the likelihood $p(M|W)$ as follows:

$$p(M|W) = \prod_{c(x,y) \in M} p(c(x, y)|W). \tag{3.35}$$

Here, the term $p(c(x, y)|W)$ is a sensor model and evaluates the match between the world W and input map M. Essentially, $p(c(x, y)|W)$ captures the quality of the original mapping algorithm producing the grid map which is used as input in our system. For calculating $p(c(x, y)|W)$, we discretize the cell state $M(x, y)$ of the input map into three classes “occupied”, “unknown” and “free”, by thresholding its occupancy values. Our semantic world W contains also three types of cell states, which are

- “wall”: cells on the four edges of a unit.
- “free”: cells that are located within a unit.
- “unknown”: cells that are located outside all units.

In this way, our sensor model $p(c(x, y)|W)$ is realized as a “3×3” look-up table. Given the extended model defined in equation (3.29), the sensor model can be further factorized:

$$p(c(x, y)|W) = p(c(x, y)|U, T, R). \tag{3.36}$$

Here U contains the geometrical parameters of the underlying units. T indicates the semantic type of the units. U and T strongly influence the values of the sensor model.

Since R only describes the relation between two individual units, it does not influence the sensor model. Thus equation (3.36) becomes

$$p(c(x,y)|W) = p(c(x,y)|U,T). \tag{3.37}$$

In this way, the sensor model turns into a semantic sensor model that is not only dependent on the geometrical parameters but also on the semantic type of the underlying units which are inferred in MLNs.

In the real world, it is more likely that rooms and halls contain objects than corridors do, because the functionality of corridors is connecting other units, rather than placing objects. Thus, we propose to make the values of our semantic sensor model dependent on the type (decided based on the inference results in MLNs) of the underlying unit. The fundamental idea is to set the values of the semantic sensor model in such a way that it does not strongly penalize the mismatch between the input and the semantic world within the units of non-corridor types. Thus the existence of false positives (potential object) is made more probable in the units of non-corridor types. An example of the effect of our semantic sensor model is depicted in Fig. 3.31.

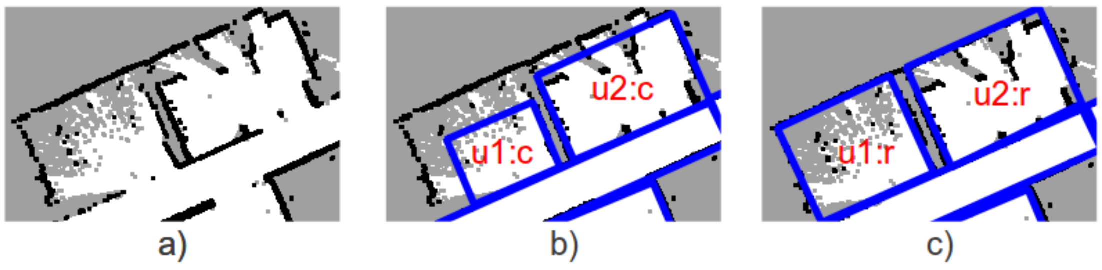


Fig. 3.31: Effect of our semantic sensor model. **a)** The input map. **b)** The highest likelihood solution for unit 1 and unit 2 if they are of the type "corridor". **c)** The highest likelihood solution for unit 1 and unit 2 if they are of the type "room". Note that the type of a unit is decided based on the inference results in MLNs (Table 4.3).

Having redefined the prior and likelihood using inference results of MLNs, a knowledge-supervised MCMC sampling is formed. This process follows the same procedure as described in section 3.5.

3.9 Experiments and Discussions

So far we have extended the parametric model with semantically meaningful, abstract terms. Based on these terms we defined rule-based context knowledge to describe the features of indoor environments. Our system, as a whole, is realized as a knowledge-supervised MCMC sampling. The performance of our system is evaluated using various data sets, which include publicly available benchmark data obtained from the Robotics Data Set Repository (Radish) [58], data acquired using our own mobile robot and simulated data of open source simulators.

3.9.1 Evaluation Using Publicly Available Benchmark Data

Without loss of generality, we evaluate our system using publicly available benchmark data. These data are real world data and were acquired with real robots by various researchers. In the following we show the performance of our system on two data sets obtained from [58].

Fig. 3.32 shows the performance of our system on a big data set. As input, an occupancy grid map M ("ubremen-cartesium" dataset [58]) of an entire floor of a building is used. This map is a big matrix ("1237×672") containing occupancy values. We classify these values as $\{occupied, unknown, free\}$ so as to generate the classified input map C_M (Fig. 3.32-b). Starting from a random initial guess, the semantic world W is adapted to better match the input map M by stochastically applying the defined kernels. After certain burn-in time, we get the most likely semantic world W^* comprised of 17 units (each of which is represented by a rectangle) as shown in Fig. 3.32-c. The topology of the perceived environment is generated by connecting the centres of doors with the centres of space units. Not only does W^* accurately represent the geometry of the input map, W^* is also a parametric abstract model (Fig. 3.32-e) of the input map that provides valuable abstract information, such as type, adjacency and connectivity by doors. In Fig. 3.32-d, the semantic world is plotted onto the input map to provide a direct comparison. The type and the ID of each unit are shown in red at its centre. The unknown areas in each unit, which are generally caused by non-transparent objects like furniture, are highlighted by the color green. In addition, unexplored areas are also captured by our model (marked by magenta "N"). These areas are too small to be recognized as space units but are evidence for physically existing space.

Fig. 3.33 shows the posterior distribution $p(W|M)$ built by 1000 samples after the underlying Markov chain has converged, i.e. W^* obtained. In Fig. 3.33-b, we can see that except some small variations, these 1000 semantic worlds are almost the same, which indicates, that the whole Markov chain stays stable and that the convergence is well retained.

Compared with using weak context knowledge only (section 3.4.1), our current system employs rule-based context knowledge in a systematic way (in the form of MLNs), so that the input map is explained according to the underlying model structure. In this way, the big challenge presented in section 3.7 is solved. A performance comparison between using weak context knowledge only and using rule-based context knowledge is depicted in Fig. 3.34. Three high likelihood samples obtained by using weak context knowledge only are shown in Fig. 3.34-a,b,c. They essentially represent the local maxima of the likelihood shown in Fig. 2.2. Although all these three results provide good match to the input map (in terms of high likelihood), they have topological defects (highlighted by magenta circles), which contradict our knowledge (low prior). In this case, all pairs of connecting walls of

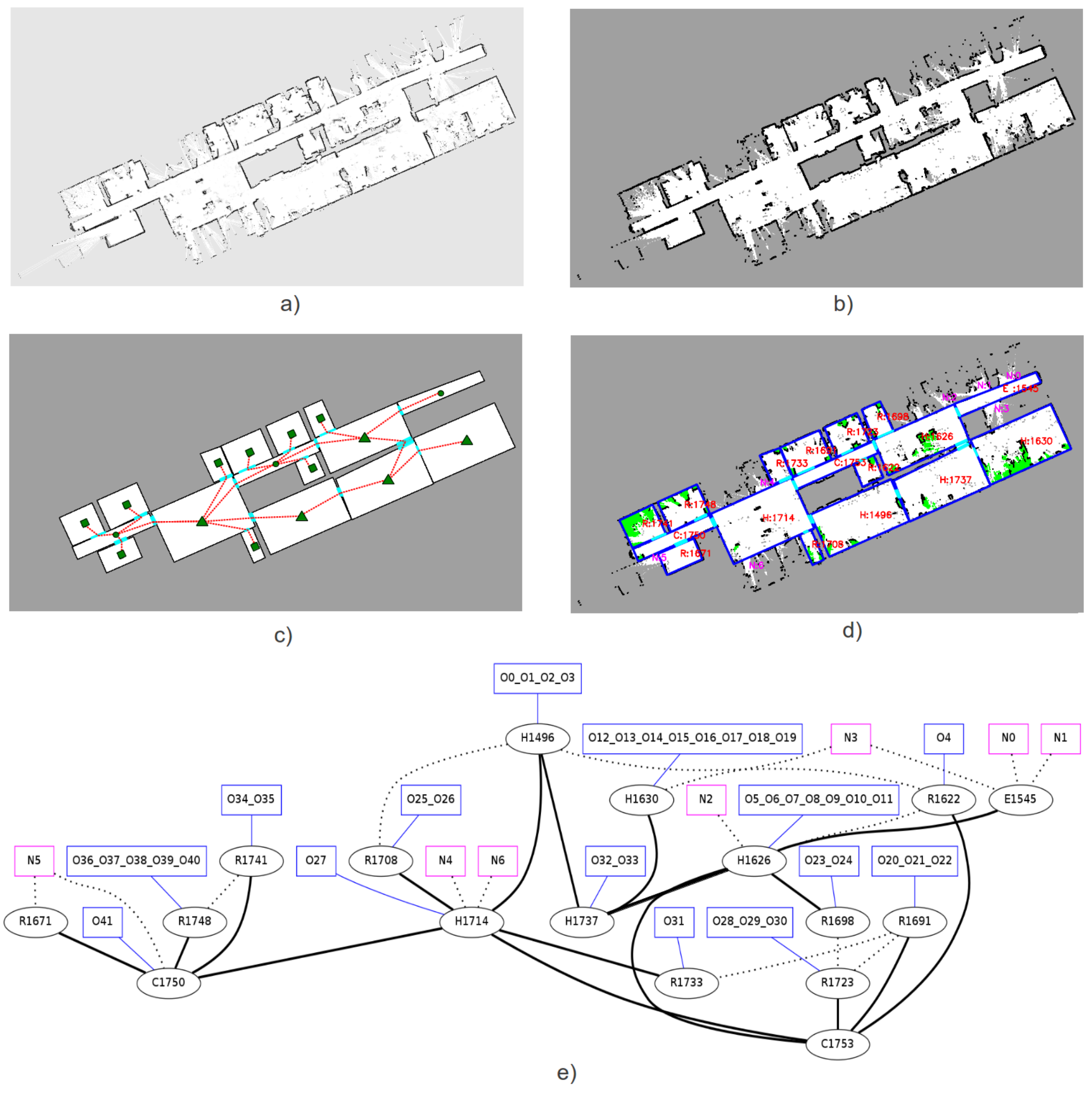


Fig. 3.32: The performance of our system. a) The original occupancy grid map M obtained from [58]. b) The corresponding classified map C_M (black=occupied, gray=unknown, white=free). c) Our semantic world W^* (black=wall, gray=unknown, white=free). Topology is shown by red lines, with cyan representing detected doors. Small triangles, circles and rectangles show the geometric center of hall, corridor and room. d) The semantic world W^* plotted onto the classified map C_M (black=occupied, blue=wall, gray=unknown, white=free, cyan=door, green=unknown cells in units). The type and ID of each unit are shown at its center (R=room, H=hall, C=corridor, E=other). Unexplored area is detected using connected-components analysis [19] and is marked by magenta "N". e) The corresponding abstract model of W^* (ellipse node=space unit with ID, blue rectangle node=unknown cells within units, magenta rectangle node=unexplored area with ID, black solid edge=connected by door, black dashed edge=adjacent without door).

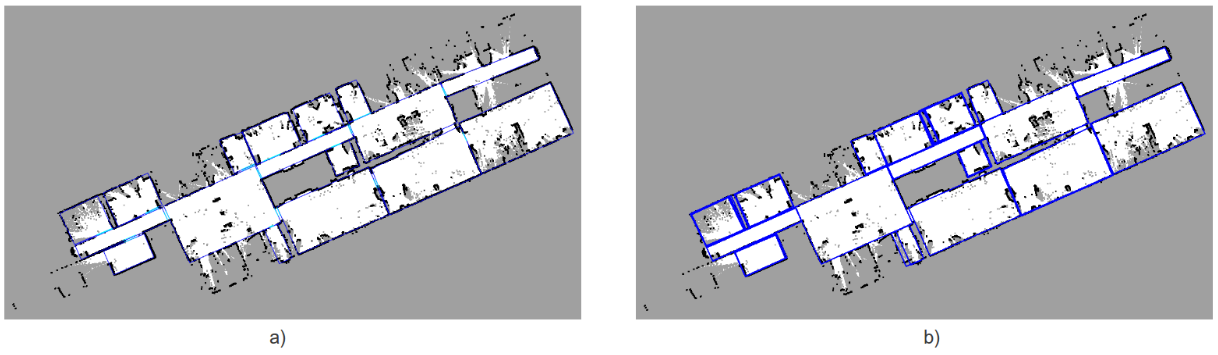


Fig. 3.33: a) After the underlying Markov chain has converged, the world W^* is obtained. Here we purposefully plot the world using very thin blue lines so that the corresponding distribution built by multiple worlds can be better seen. b) The posterior distribution $p(W|M)$ illustrated by 1000 semantic worlds obtained after getting W^* (figure a). Except small variations, these 1000 semantic worlds are almost the same, which indicates, that the Markov chain stays stable and that the convergence is well retained.

adjacent rooms that should have the same length are drawn in orange in Fig. 3.34-a,b,c. The length difference of each pair of these connecting walls results in a penalization in prior probability (equation (3.34)). By applying our rule-based context knowledge, local maxima of the likelihood with topological defects are suppressed so that they have a low posterior probability. In this way a semantic world that has high likelihood and high prior, i.e. high posterior (Fig. 3.34-d), is obtained by our knowledge-supervised MCMC sampling. In addition, various poor local matches (highlighted by magenta rectangles in Fig. 3.34-a,b,c) are corrected by the semantic sensor model used in our system.

Even using the same semantic world (Fig. 3.33-a) as the start state of the Markov chain, the posterior distribution obtained by using weak context knowledge only and that obtained by using rule-based context knowledge are different. This effect is illustrated in Fig. 3.35 by plotting 1000 accepted worlds together. Here we purposefully plot each sample (world) using very thin lines. We can see that the semantic worlds obtained using rule-based context knowledge show much smaller variations than the ones obtained using weak context knowledge only. It is obvious that the underlying Markov chain obtained using rule-based context knowledge is more stable and converges better (smaller variance).

Fig. 3.36 shows the performance of our system using another data set obtained from [58]. As can be seen, the obtained semantic world accurately represents the geometry of the environments captured in the input map and explains the perceived environments with the correct topology. The corresponding posterior distribution after the convergence is depicted in Fig. 3.37. Again, we can obviously tell that our system constructs a stable Markov chain that produces a fine semantic world (abstract model) for the input map (data).

By setting a proper height to walls and doors, we can generate 3D semantic worlds

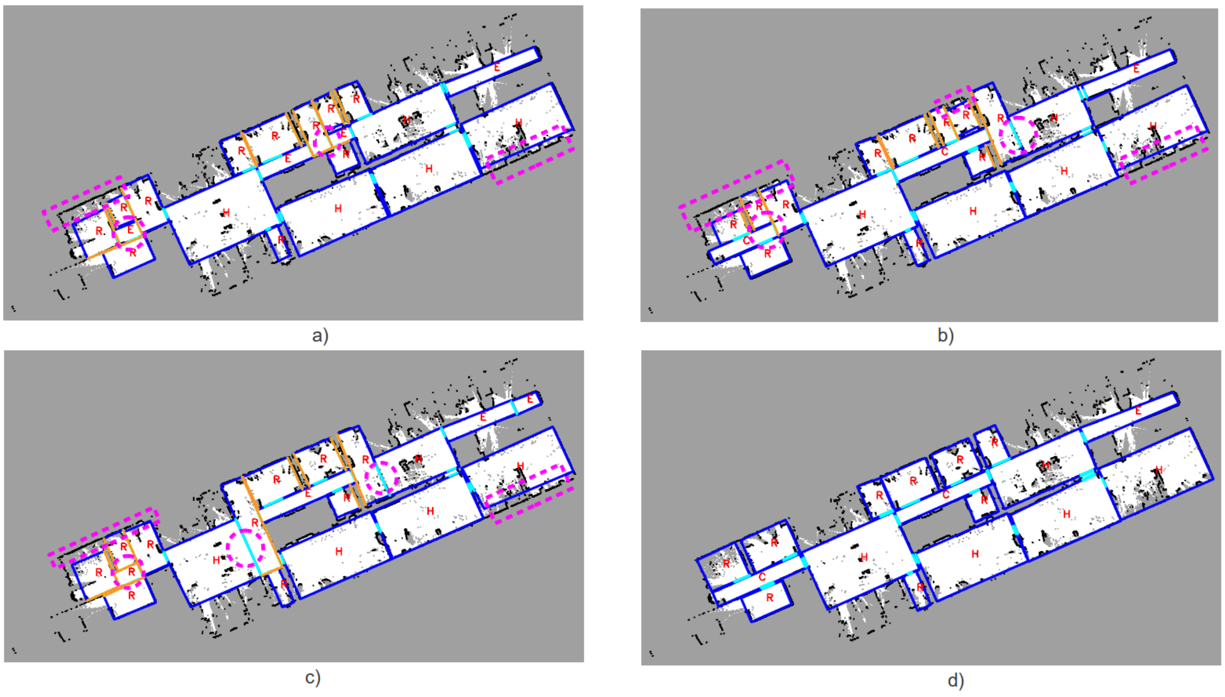
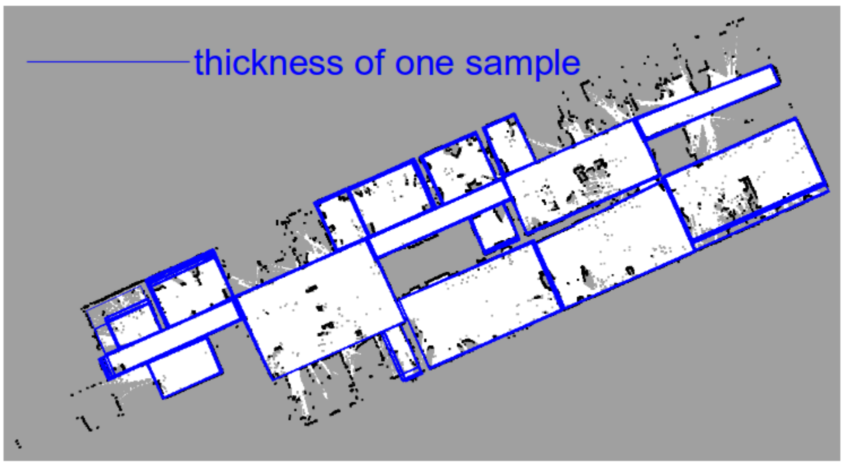


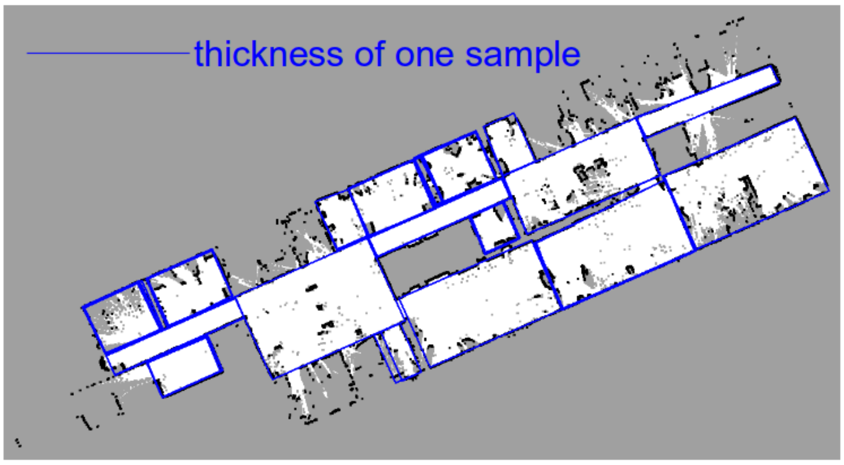
Fig. 3.34: Comparison of the performance between using weak context knowledge only (figure a, b and c) and using rule-based context knowledge in the form of MLNs (figure d). Topological defects of the intermediate results are highlighted by magenta dashed circles. Poor local matches that are improved by semantic sensor model are highlighted by magenta dashed rectangles. Each pair of connecting walls that should have the same length are drawn in orange (figure a, b and c). Type of each unit is shown at its center.

based on our parametric abstract model. These 3D worlds can be well used for simulation purposes. Two examples are depicted in Fig. 3.38.

The computational cost is strongly dependent on the size of the input map and consists of two parts, which are MCMC operations and knowledge processing in MLNs. With a single-threaded implementation on an Intel i7 CPU, the speed of MCMC operations is around 30 iterations per second for the map shown in Fig. 3.32. The speed of knowledge processing in MLNs depends on one hand on the tool (the software implementation of MLNs) that one uses. On the other hand, it depends on the number of optimization iterations set in the tool. In our case, we could get a satisfactory result in 5-8 seconds using the tool in [67] per processing. To analyze a grid map, we first start our system using weak context knowledge only. After enough context is available (e.g. coverage of the input map greater than 80%), rule-based context knowledge processing is enabled to help better explain the input map. In this way, we could obtain a good result within a reasonable processing time, which is, 20 to 25 minutes for the map shown in Fig. 3.32.



a)



b)

Fig. 3.35: Starting from the world state shown in figure 3.33-a, we plot 1000 accepted samples obtained by using weak context knowledge only (figure a) and using rule-based context knowledge (figure b) onto the classified map. It is obvious that the underlying Markov chain converges better using rule-based context knowledge.

3.9.2 Evaluation Using Data Acquired by Our Own Robot

In addition to publicly available benchmark data, we test our system on our own mobile robot (see Fig. 3.39) as well, which is equipped with three laser scanners, a Kinect camera and a stereo camera system. In our experiments, we mainly used the two laser scanners that are situated at the front and the back side of our robot to sense the robot's operating environments. While our robot travels in the environment, the obtained laser scans are fed into the Gmapping algorithm [41] to generate an occupancy grid map of the perceived environment. Subsequently, the resulting grid map is used as input in our system to produce the corresponding semantic world. An example of online experiment is depicted in Fig. 3.40. Here, the violet point shows the robot position. The input map is being updated while our robot explores the environment. The obtained semantic world is directly plotted

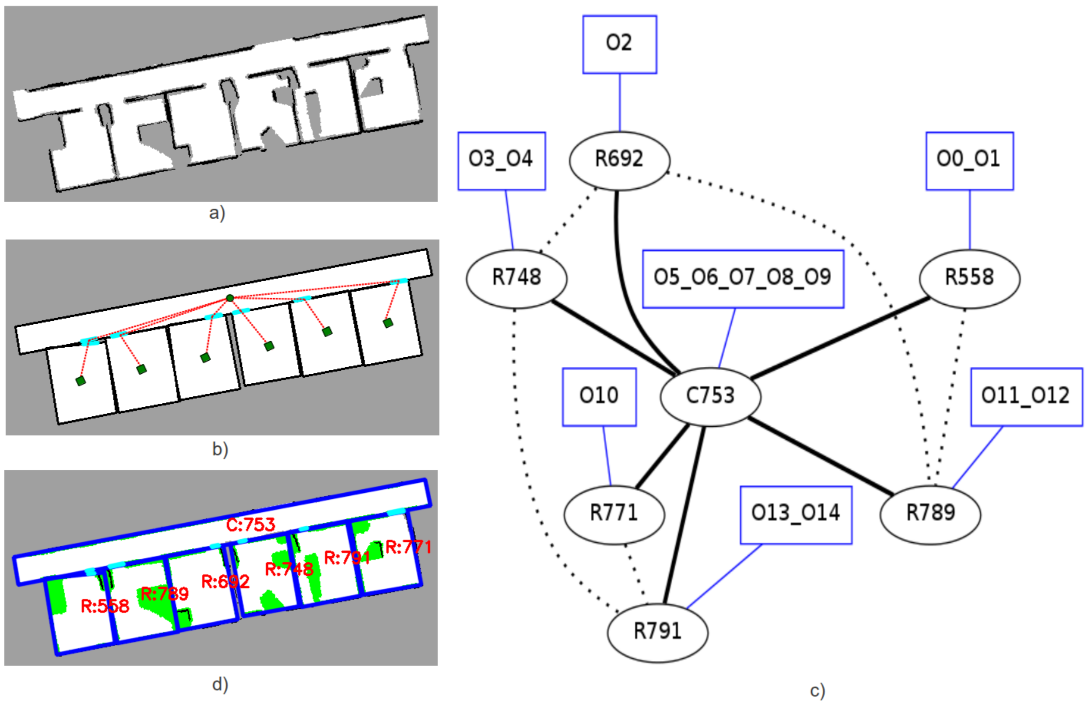


Fig. 3.36: The performance of our system for another data set from [58]. a) The classified map. b) The corresponding semantic world. c) The corresponding abstract representation of the obtained semantic world. d) A direct comparison between the map and the resulting semantic world.

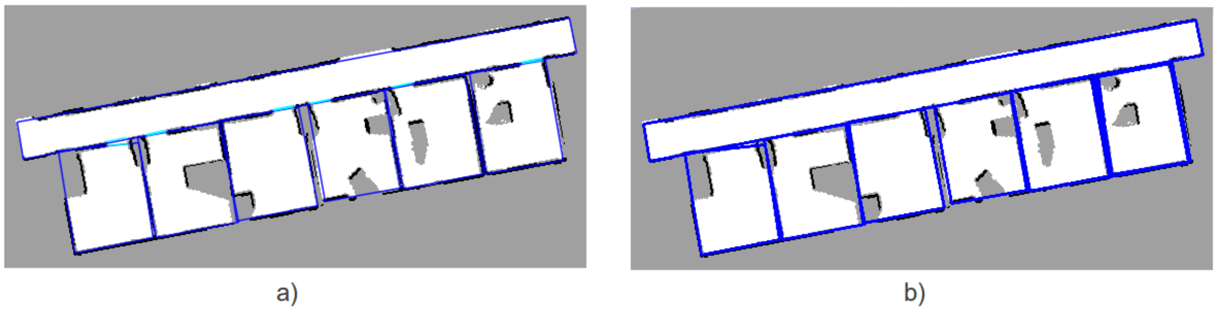


Fig. 3.37: Posterior distribution (figure b) illustrated by plotting 1000 accepted samples together after getting the result in figure a.

onto the grid map. Walls and doors of our semantic world are shown in blue and cyan respectively.

Combining our parametric abstract model with object pose estimation, we obtain a coherent 3D semantic map of indoor environments. The general principle is illustrated in Fig. 3.41. 3D objects that are captured by RGB-D images of the Kinect camera are recognized and localized by the algorithm in [151]. Given such a map, a robot not only knows the geometry and topology of the environment, it also has a prior belief on the pose of objects and their belongingness. Furthermore, this map provides robots semantic

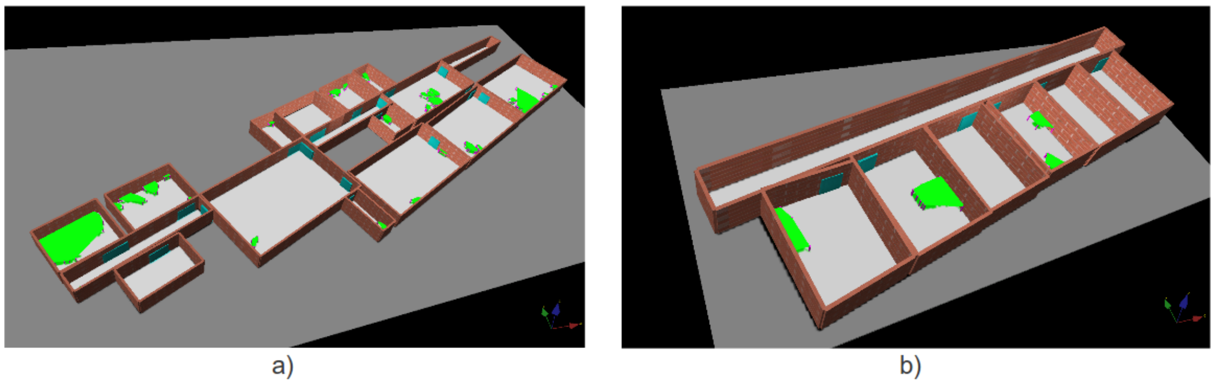


Fig. 3.38: Corresponding 3D semantic worlds of the results shown in Fig. 3.32-c and Fig. 3.36-b. These 3D worlds can be well used for simulation purposes.

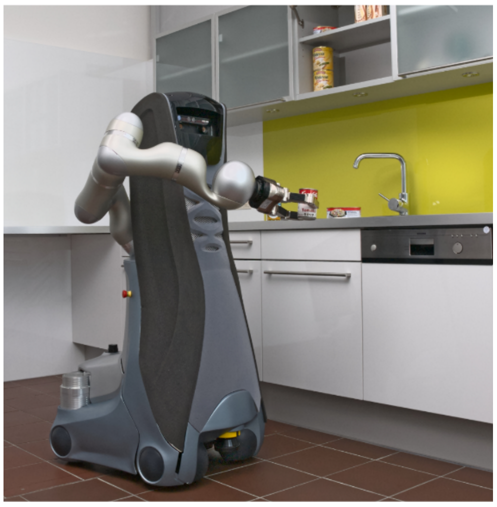


Fig. 3.39: Our mobile robot equipped with three laser scanners, a Kinect camera and a stereo camera system.

understanding of the underlying environment, which can benefit high level robotic applications. For instance, if a robot needs to plan a path from position A that is located in space unit $R1$ to position B in unit $R4$, instead of a brute-force path planning $A \rightarrow B$, which happens directly on the coordinate level, the robot can plan a path based on the topology given by the coherent semantic map. As shown in Fig. 3.42, the actual path planning can be divided into a few parallel steps on the semantic level: $A \rightarrow door(R1, C3)$, $door(R1, C3) \rightarrow door(C3, R4)$ and $door(C3, R4) \rightarrow B$. Since the positions $door(R1, C3)$ and $door(C3, R4)$ are also given by the semantic map, the above path planning steps can be easily done using standard methods like the A* algorithm [51].

Fig. 3.43 shows the result of our system for an indoor office environment, which contains five furnished office rooms and a big corridor. In the grid map of this environment (Fig. 3.43-a), we can see that the five office rooms are quite cluttered (because of the existence of furniture and things). In spite of the clutter, our system still provides a good semantic

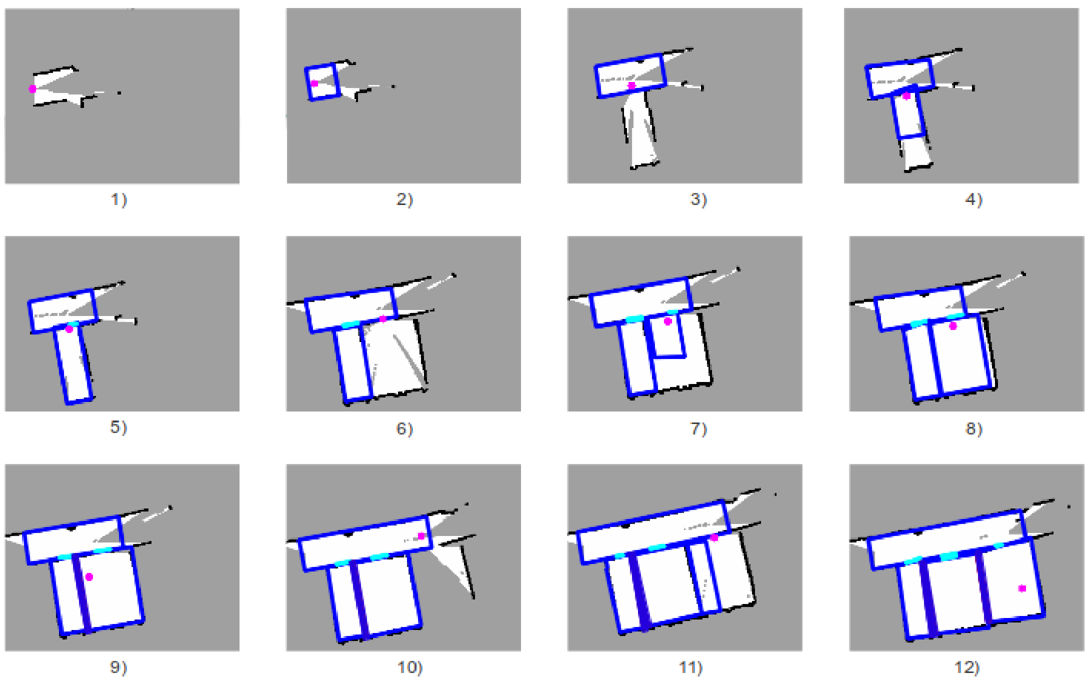


Fig. 3.40: An example of online experiment. Here, the violet point shows the robot position. The input map is being updated while our robot explores the environment. The obtained semantic world is directly plotted onto the grid map. Walls and doors of our semantic world are shown in blue and cyan respectively.

world that correctly explains the environment with six space units (five rooms and one corridor) and the correct topology, as shown in Fig. 3.43-b and c. By setting a proper height to walls (blue) and doors (green), we show the 3D parametric model along with the detected 3D objects in Fig. 3.43-d. Here the table planes, on which the 3D objects are detected, are shown in cyan. The current robot pose (represented by a cluster of coordinate systems), the current laser scan (shown by red points) and the current color point cloud are highlighted by the dashed yellow rectangle. In the direct comparison between the grid map and the resulting semantic world, as shown in Fig. 3.43-d, it is obvious that the resulting semantic world accurately approximates the geometry of the grid map which essentially captures the environment geometry. Fig. 3.43-e depicts the details of object recognition and localization. In space units $R1, R2$ and $R5$, several 3D objects and tables are recognized and localized regarding their 6D pose (x,y,z,roll,pitch,yaw).

By plotting 1000 samples together we show the resulting posterior distribution in Fig. 3.44. Here each sample is drawn in very thin lines as depicted by Fig. 3.44-a. Again, we can obviously see that our system constructs a stable Markov chain that well converges to the goal distribution.

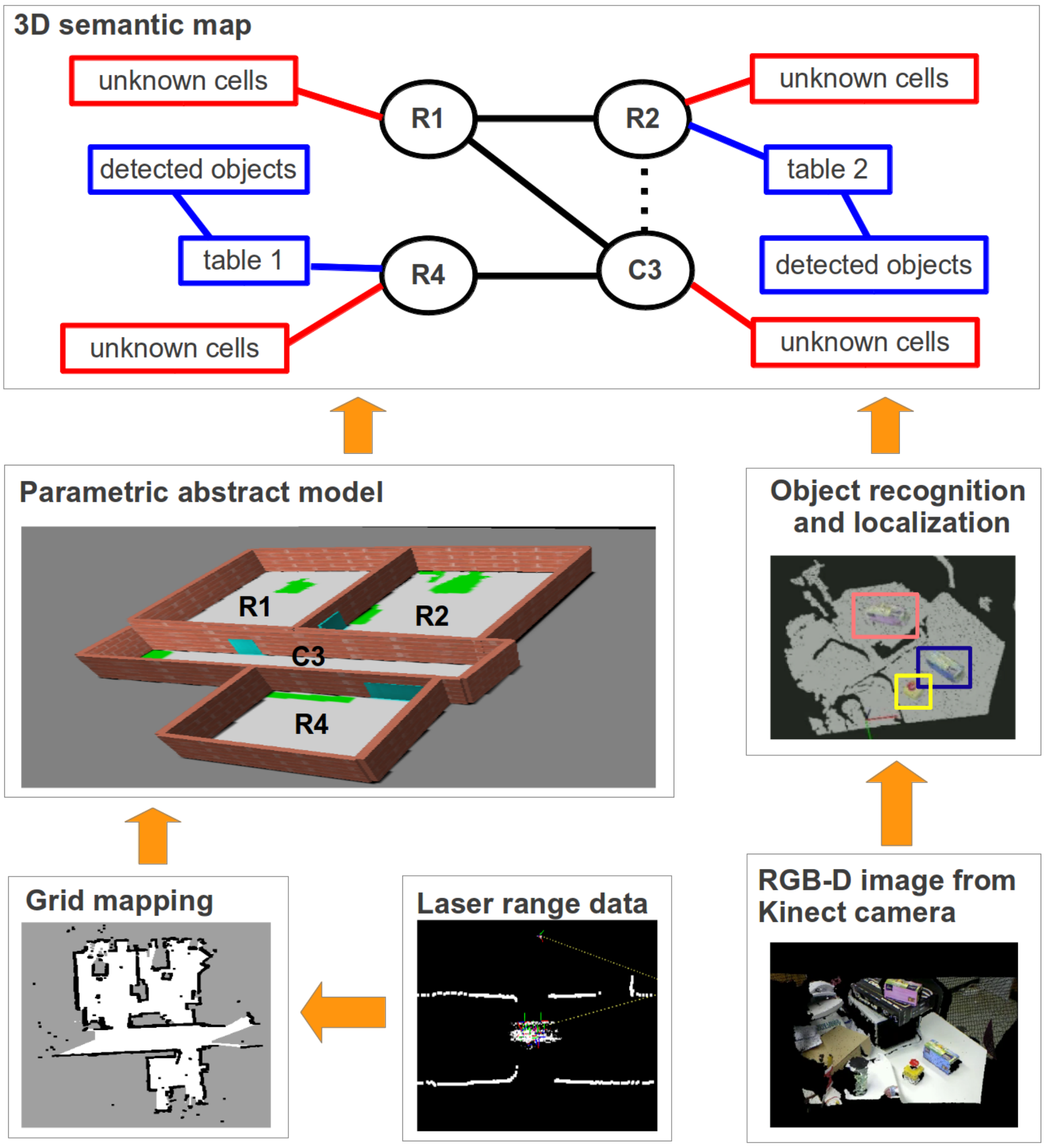


Fig. 3.41: A coherent 3D semantic map obtained by combining our parametric model with object pose estimation.

3.9.3 Evaluation Using Simulated Data Obtained from Open Source Simulators

In addition to the above experiments with real world data, we evaluate our system in simulation as well. In the real world, we do not encounter so many indoor environments of different structures, in which we can test our system. Thus it is quite helpful to evaluate our system in simulation where a big number of different environments can be manually created. Without loss of generality, we use open source simulators and 3D environment models for this purpose, which are publicly available in the internet. Here we have used the ROS [60] integration of the Gazebo simulator [59] to simulate a PR2 robot [35] and

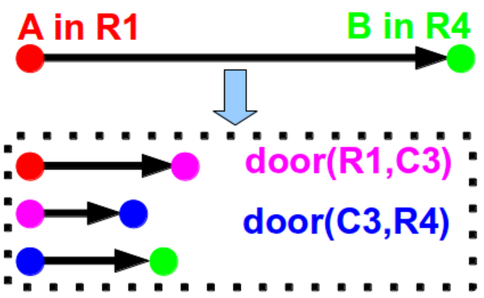


Fig. 3.42: Path planning on the semantic level. Instead of a brute-force planning from A to B, the robot can plan a path based on the topology given by the semantic map in a few parallel steps.

its operating environments. An example of this robot and a simulated 3D environment is depicted by Fig. 3.45.

Fig. 3.46 to Fig. 3.50 show five simulation results. In these figures, sub-figures a) show snapshots of the 3D environments simulated by the Gazebo simulator. Sub-figures b) depict the corresponding grid maps generated by the Gmapping algorithm, after the simulated robot has perceived the environments. Sub-figures c) illustrate the resulting semantic worlds with their topology, where the geometric centers of halls, corridors and rooms are shown by small triangles, circles and rectangles respectively. A direct comparison is shown in sub-figures d) by plotting the semantic worlds onto the corresponding grid maps. Finally, the corresponding abstract representation of the semantic worlds are illustrated in sub-figures e).

We purposefully chose these five environments to test our system, because they represent several common environment types which are often found in the reality. The environment shown in Fig. 3.46 represents the type, in which a big hall is surrounded by a lot of satellite rooms. Fig. 3.47 depicts a complex indoor environment consisting of many space units. In this environment, rooms are located in a row and are connected by corridors, which separate halls from rooms. Another environment of this kind is illustrated in Fig. 3.48. Fig. 3.49 shows a classical office environment, where ten rooms are situated in two rows and connected by a long corridor. Another environment comprised of three halls and a corridor, which is like an exhibition centre, is depicted in Fig. 3.50. As we can see, our system performs very well in all the five environments, i.e. the resulting semantic world well explains the corresponding environment with a correct number of space units and an appropriate topology. Moreover, as the quantitative evaluation in the following sub-section will show, the resulting semantic worlds accurately represent the geometry of the perceived environments as well.

3.9.4 Quantitative Evaluation

In order to quantitatively evaluate our approach, we compute $K(W, M)$, the *cell prediction rate* capturing the predictive power of the semantic world model W with respect to an input map M:

$$K(W, M) = \frac{\sum\limits_{c(x,y) \in M} l(c(x, y))}{t_M},$$

with

$$l(c(x, y)) = \begin{cases} 1, & C_M(x, y) = C_W(x, y), \\ 0, & \text{otherwise}, \end{cases} \tag{3.39}$$

where t_M is the number of all grid cells in the map M. $c(x, y)$ indicates one grid cell located at the position (x, y). $C_M(x, y)$ and $C_W(x, y)$ are previously defined in equation (3.7) and (3.8). $K(W, M)$ of the maps shown in this chapter is given in Table 3.7. In this table, we can see that the $K(W, M)$ for the three real world data sets (Fig. 3.32, Fig. 3.36 and Fig. 3.43) is above 90%. The mismatch is mainly due to the clutter caused by furniture and things. For the five simulation data sets (Fig. 3.46, Fig. 3.47, Fig. 3.48, Fig. 3.49 and Fig. 3.50), the $K(W, M)$ is above 94%, where the mismatch lies mainly in some not-fully-explored areas. Such areas are evidences for partially explored space units in corresponding environments which are too small to be recognized. To sum up, our system accurately represents the geometry of the perceived environments in all experiments.

Tab. 3.7: Cell prediction rate $K(W, M)$.

	Fig. 3.32	**Fig.** 3.36	**Fig.** 3.43	**Fig.** 3.46
Percentage	93.6%	91.0%	90.1%	95.4%
	Fig. 3.47	**Fig.** 3.48	**Fig.** 3.49	**Fig.** 3.50
Percentage	96.0%	95.2%	96.3%	94.4%

3.10 Conclusions

In this chapter, we used our knowledge-supervised MCMC sampling approach to solve the problem of semantic robot mapping. In principle, our approach demonstrates a new method of finding compact abstract model for input data using predefined abstract terms. Based on these abstract terms, intelligent autonomous systems, such as a robot, should be able to make inference according to specific knowledge base, so that they can better handle the complexity and uncertainty of the real world. Based on Markov logic, we formulate task-specific context knowledge as descriptive logic rules which increase the overall abstraction performance.

Using our knowledge-supervised MCMC sampling approach, we showed a new way of automatically generating semantic maps from preprocessed sensor data. This is done by means of a probabilistic generative model and MCMC-based reasoning techniques.

We use Bayesian reasoning to build semantic maps, so that they are aligned with the preprocessed sensor observations, that a robot made during an environment exploration and mapping stage. This introduces a bottom-up path into the approach and employs data driven discriminative environment feature detectors to analyze the continuous noisy sensor observations.

Our work differs from previous semantic mapping approaches, that mostly use various classification methods in a bottom-up fashion to label either spatial regions or places based on context or that assign semantic labels directly to portions of the observations. Instead, we construct a parametric, abstract, semantic and top-down representation of the domain under the consideration: a classical indoor environment containing several units of different types, that are connected by doorways.

As output, our system returns a parametric abstract model of the perceived environment that not only accurately represents the environment geometry, but also provides valuable abstract information. The model that we generate, is structured similarly to a scene graph and is perfectly suited for any higher level reasoning and communication purposes. By constructing the prior distribution of the semantic maps using inference results, high likelihood results with topological defects (contradiction with context knowledge) are suppressed. By means of MCMC sampling, such results can be ruled out, as shown in Fig. 3.34. By modelling context knowledge in MLNs, we can assign semantic information, e.g. the type, to the data. This allows us to use a semantically informed sensor model to better explain the observations. Experiments using real world data and simulated data showed promising results and thus confirmed the effectiveness of our approach.

Currently, we assume units having a rectangular shape, which however does not imply that our approach is restricted to this shape only. The general idea behind this is to demonstrate the exploitation of abstract (uncertain) rules on how the environment might be structured. Adhering to these rules helps the robot to interpret the noisy sensor data more correctly. In fact, units of other shapes can be introduced to the approach in that we update the prior, add discriminative methods for proposing units of other shapes and implement functionalities for carrying out new geometrical operations (e.g. for SHRINK/DILATE and SPLIT/MERGE). However, the general approach will be the same.

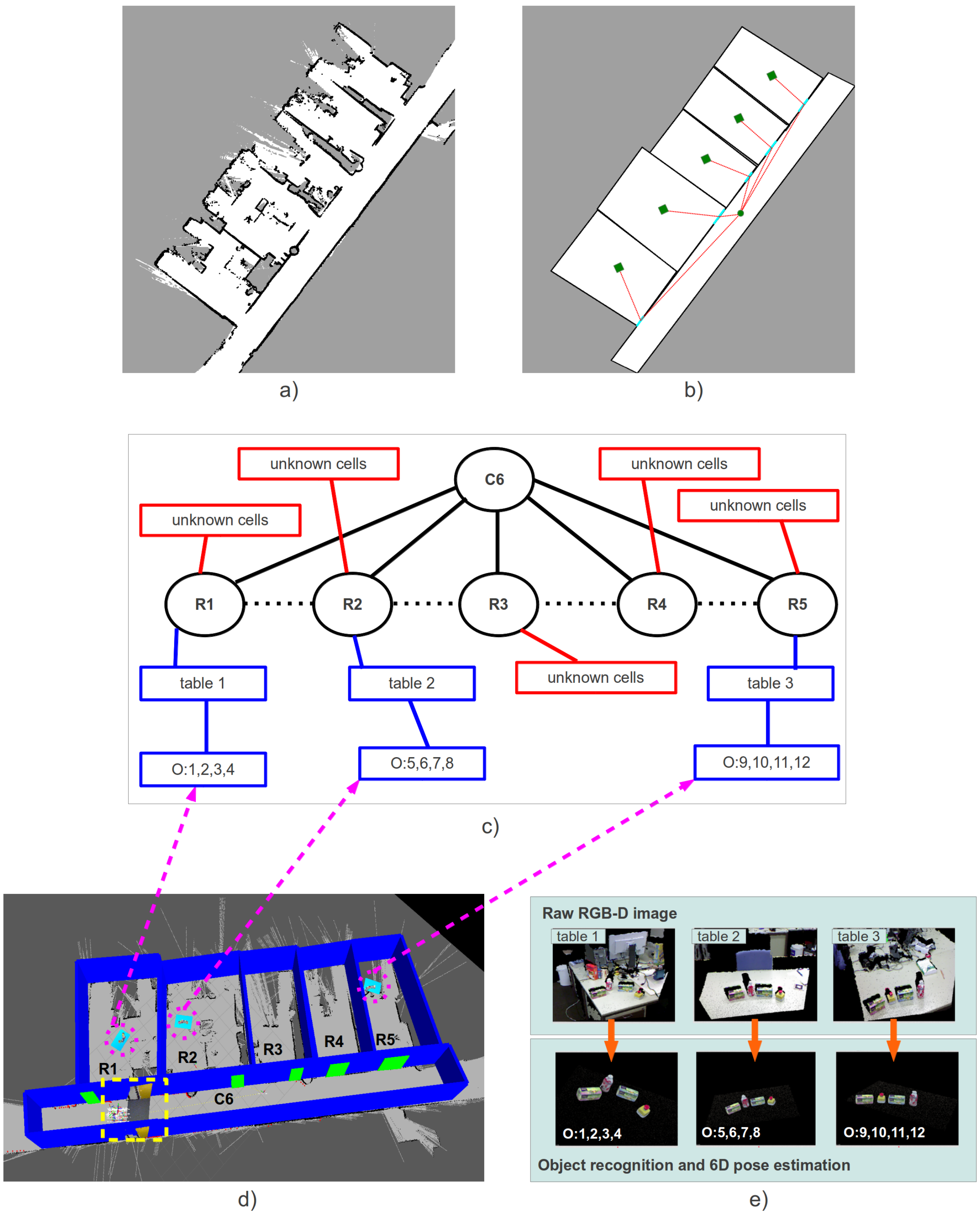


Fig. 3.43: A 3D semantic map. a) The resulting grid map of a cluttered office environment. b) The semantic world produced by our system. c) The abstract representation of the semantic world. d) By setting a proper height to walls and doors, we plot the 3D parametric model directly onto the corresponding grid map (blue=walls, green=doors, cyan=detected tables with 3D objects). The current robot pose (represented by a cluster of coordinate systems), the current laser scan (shown by red points) and the current color point cloud are highlighted by the dashed yellow rectangle. e) Details on 3D object localization.

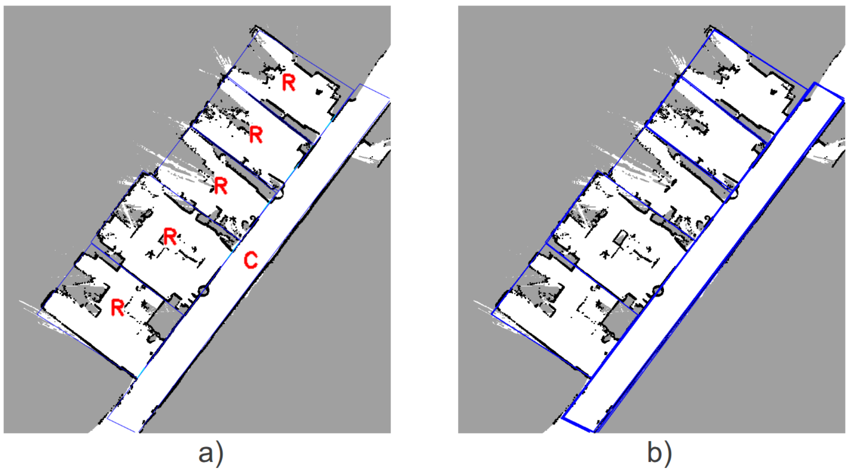


Fig. 3.44: a) The same semantic world as shown in Fig. 3.43-b plotted in very thin lines. b) Posterior distribution built by 1000 samples after the underlying Markov chain has reached the state in figure a.

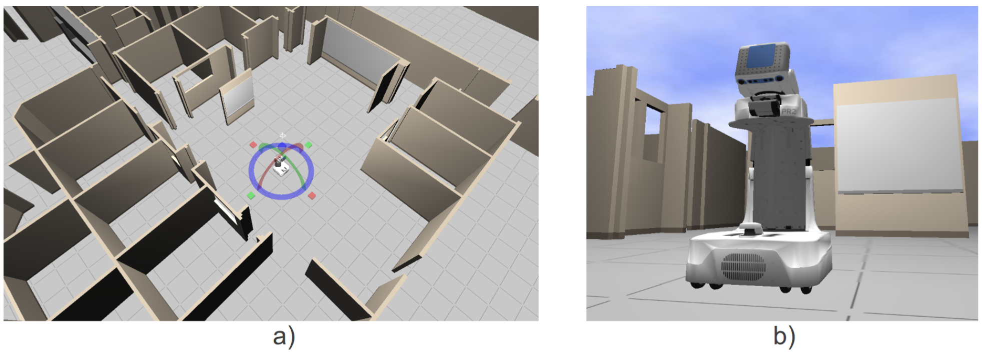


Fig. 3.45: An example of the simulated PR2 robot and its operating environment. a) A simulated 3D environment. b) A simulated PR2 robot.

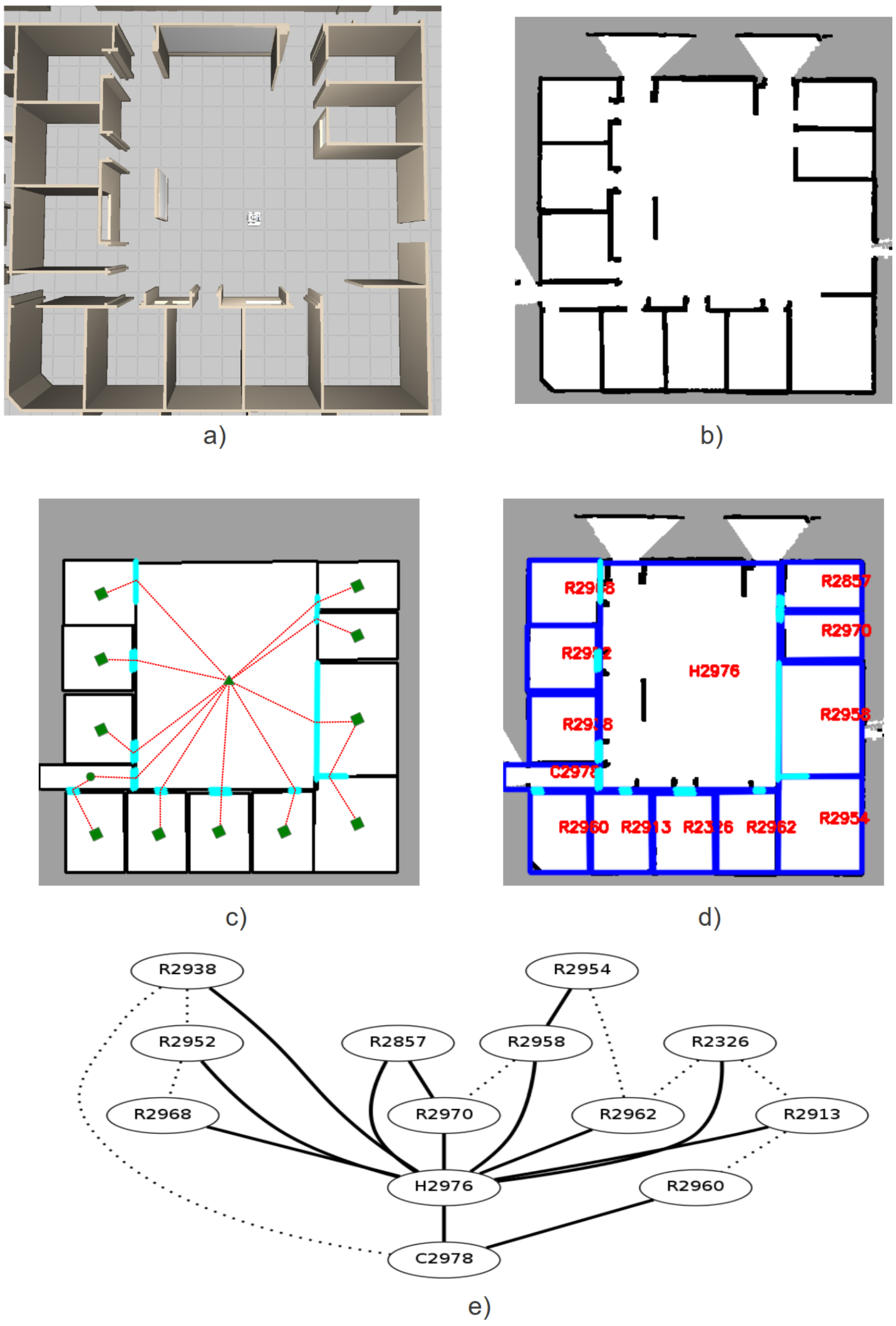


Fig. 3.46: Simulation result no. 1. a) A snapshot of the simulated environment in the Gazebo simulator. b) The corresponding grid map generated by the Gmapping algorithm. c) The resulting semantic world with its topology obtained by our system. Small triangles, circles and rectangles show the geometric center of halls, corridors and rooms. d) A direct comparison between the grid map and the semantic world. e) The abstract representation of the semantic world.

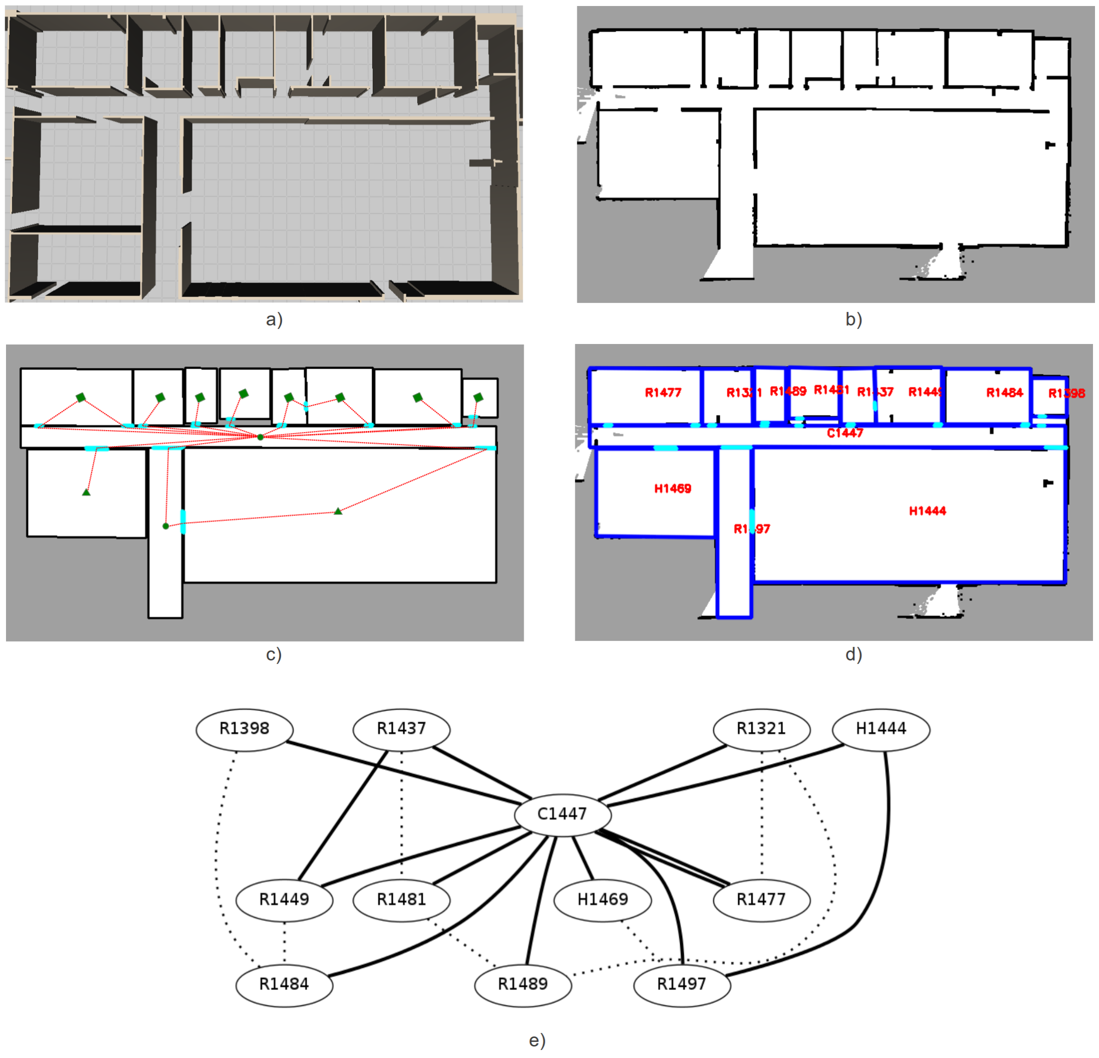


Fig. 3.47: Simulation result no. 2. a) A snapshot of the simulated environment in the Gazebo simulator. b) The corresponding grid map generated by the Gmapping algorithm. c) The resulting semantic world with its topology obtained by our system. Small triangles, circles and rectangles show the geometric center of halls, corridors and rooms. d) A direct comparison between the grid map and the semantic world. e) The abstract representation of the semantic world.

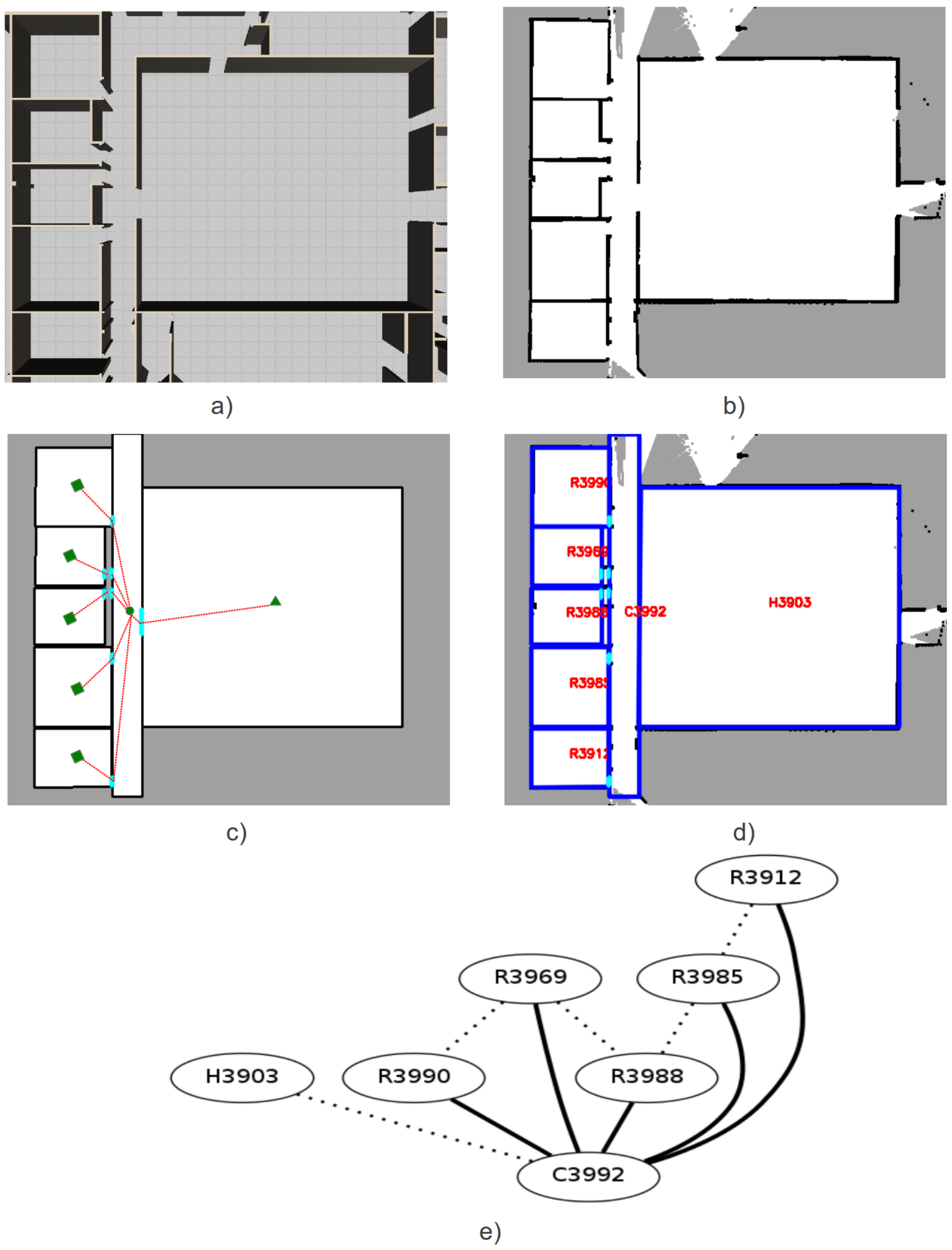


Fig. 3.48: Simulation result no. 3. a) A snapshot of the simulated environment in the Gazebo simulator. b) The corresponding grid map generated by the Gmapping algorithm. c) The resulting semantic world with its topology obtained by our system. Small triangles, circles and rectangles show the geometric center of halls, corridors and rooms. d) A direct comparison between the grid map and the semantic world. e) The abstract representation of the semantic world.

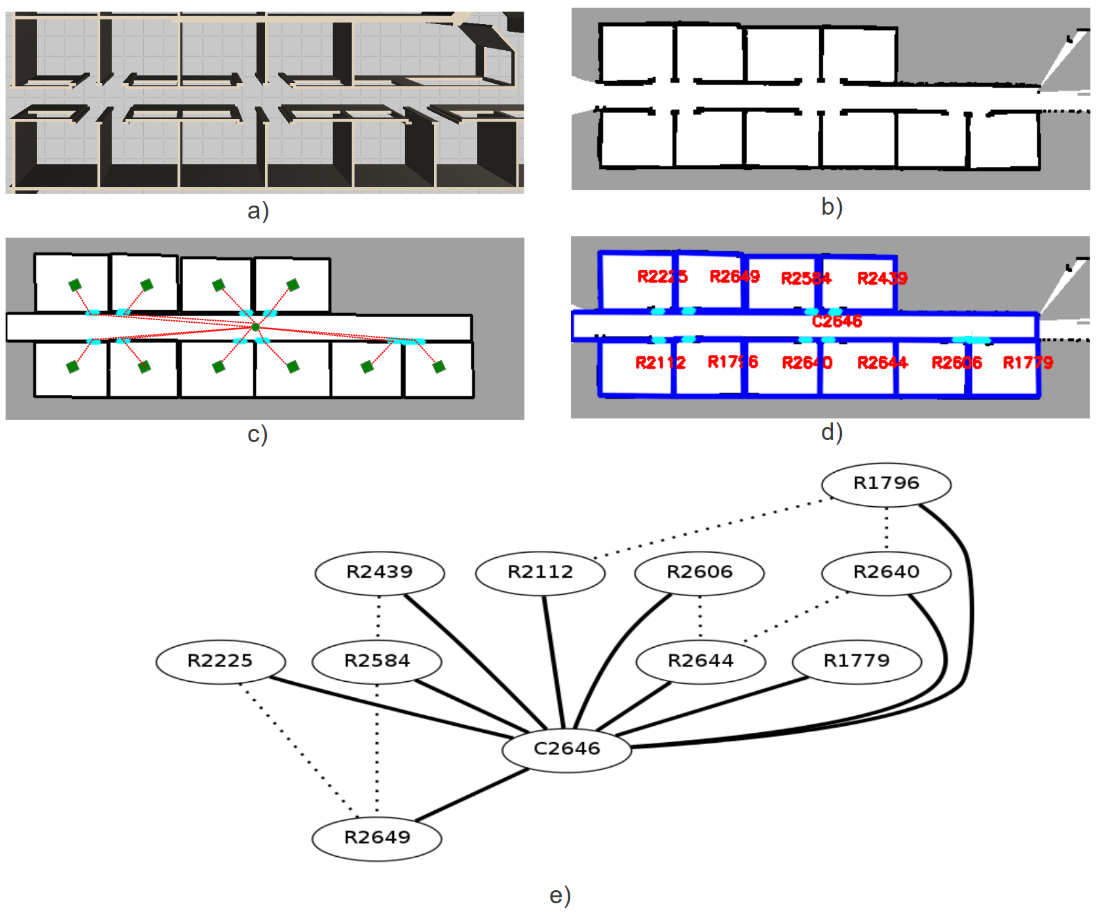


Fig. 3.49: Simulation result no. 4. a) A snapshot of the simulated environment in the Gazebo simulator. b) The corresponding grid map generated by the Gmapping algorithm. c) The resulting semantic world with its topology obtained by our system. Small triangles, circles and rectangles show the geometric center of halls, corridors and rooms. d) A direct comparison between the grid map and the semantic world. e) The abstract representation of the semantic world.

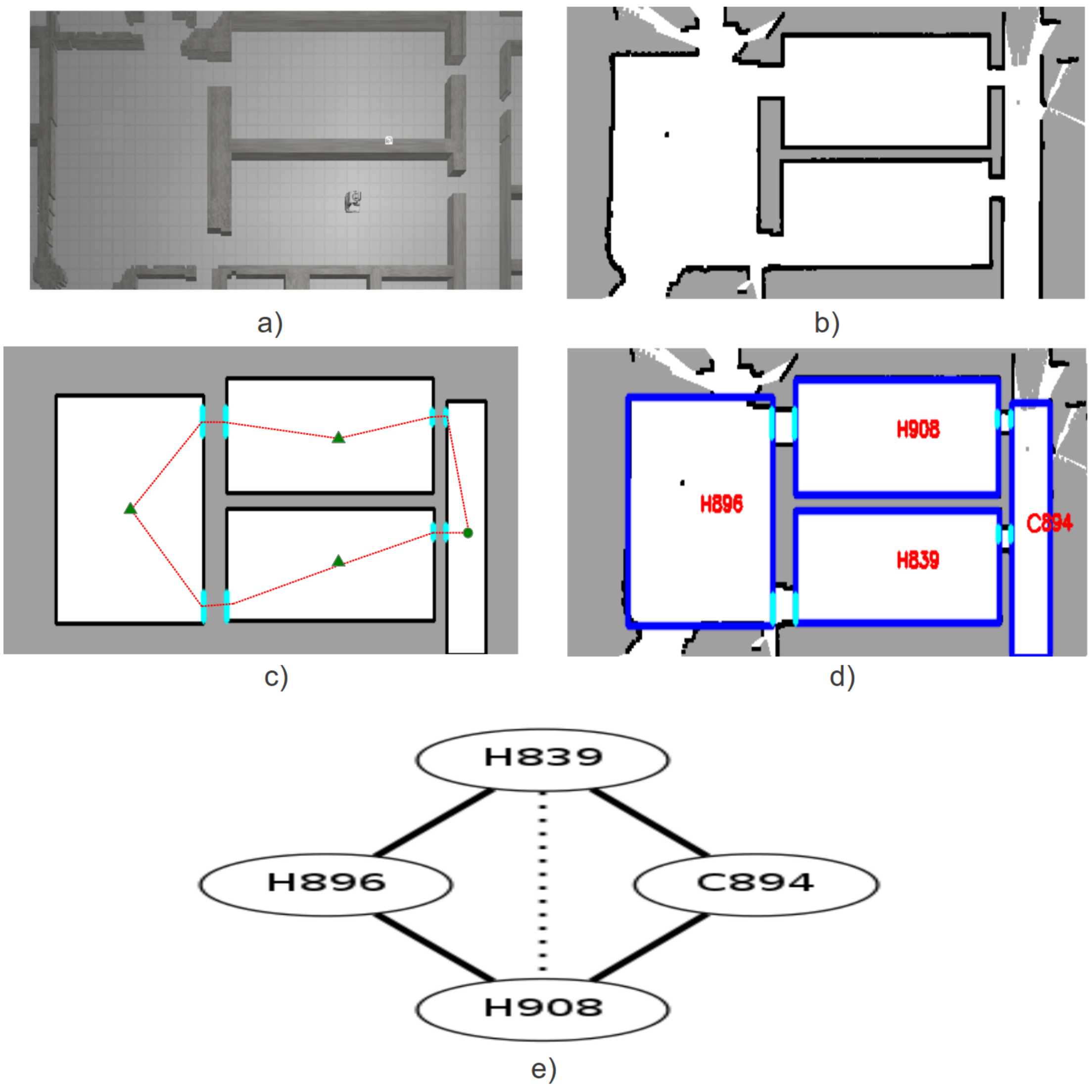


Fig. 3.50: Simulation result no. 5. a) A snapshot of the simulated environment in the Gazebo simulator. b) The corresponding grid map generated by the Gmapping algorithm. c) The resulting semantic world with its topology obtained by our system. Small triangles, circles and rectangles show the geometric center of halls, corridors and rooms. d) A direct comparison between the grid map and the semantic world. e) The abstract representation of the semantic world.

4 Use Case: Table-Top Scene Analysis

So far we have shown how to use our knowledge-supervised MCMC sampling technique to solve the problem of semantic indoor mapping. Our technique is not restricted to this use case. It also generalizes over other applications. In the following, we demonstrate the effectiveness of our knowledge-supervised MCMC sampling technique in another typical robotic domain, which is table-top scene analysis.

4.1 Introduction

For autonomous robots to successfully perform manipulation tasks, such as cleaning up and moving things, they need a structural understanding of their environment. It is not sufficient to provide geometry scene knowledge alone, i.e., the locations of the objects relevant to the manipulation task. The robots planning components require additional information about the composition and inter-object relations within the scene. Imagine, a robot that is asked to fetch one of the objects shown in Fig. 4.1. It is important for the robot to understand for example that

- to move object #3, object #4 should be moved first, otherwise object #4 will fall while moving object #3.
- object #6 is a false estimate thus can not be moved.
- there is something hidden under object #5.

In this chapter, we propose a probabilistic method to generate abstract scene graphs for table-top scenes that can answer such questions. The input to our algorithm is 6D object poses that are generated using a feature-based pose estimation approach [88, 6, 22]. Object poses can be estimated either from stereo images or from RGBD point clouds. A typical result of the procedure is shown in Fig. 4.1.

Our scene graph for table-top scenes describes the composition of the perceived scene and the relations between the objects, such as “support” and “contact”. To efficiently generate such scene graphs, we explicitly formulate and use context knowledge, which we encode in logic rules that typically hold for table-top scenes, e.g., “objects do not hover over the table”, “objects do not intersect with each other” and so on.

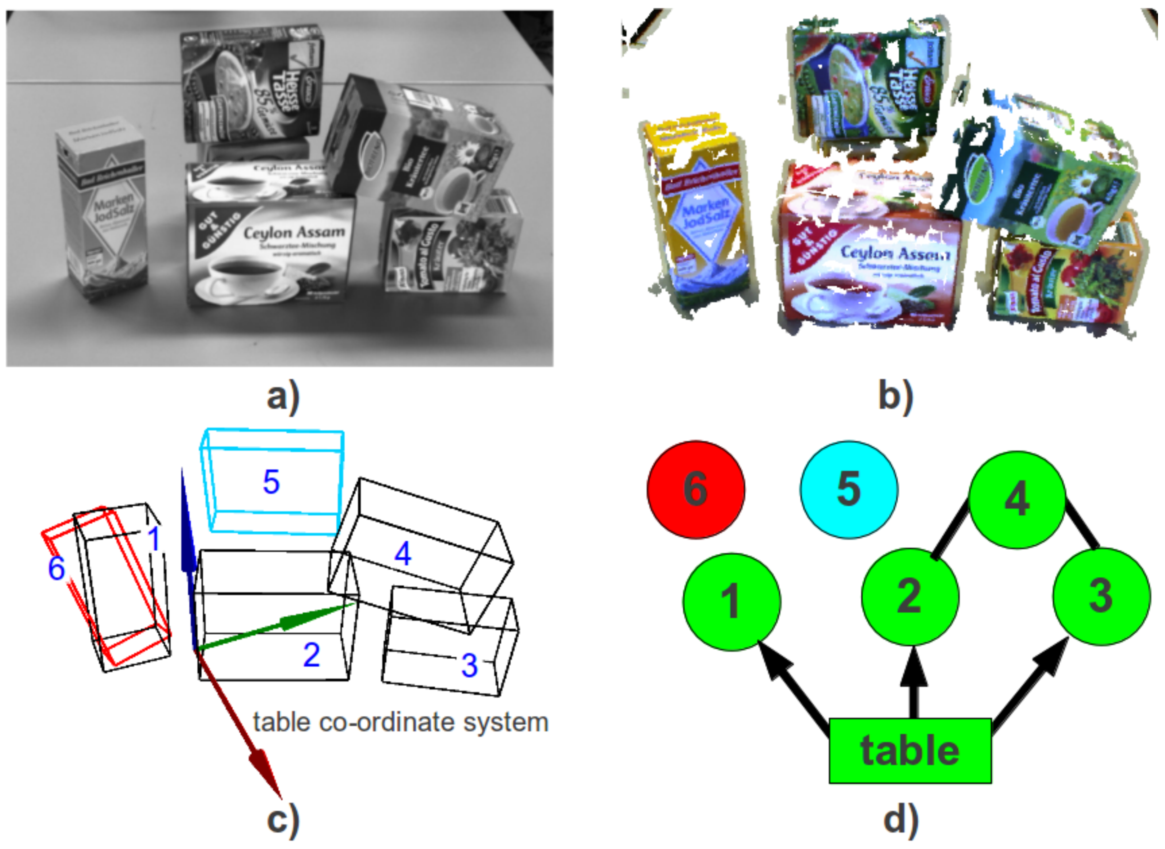


Fig. 4.1: An example output of our system. a) - b) Sensor input for 6D pose estimation: stereo image (a) or 3D point cloud (b). c) Initial guess of object 6D poses obtained by a feature-based approach. The three axes of the table coordinate system are shown as blue, red and green arrows. d) The scene graph generated by our system. Arrows indicate the relation "stable support". Undirected lines show the relation "unstable contact". Object #5 is considered to have a "hidden object" under it. object #6 is considered to be a "false estimate".

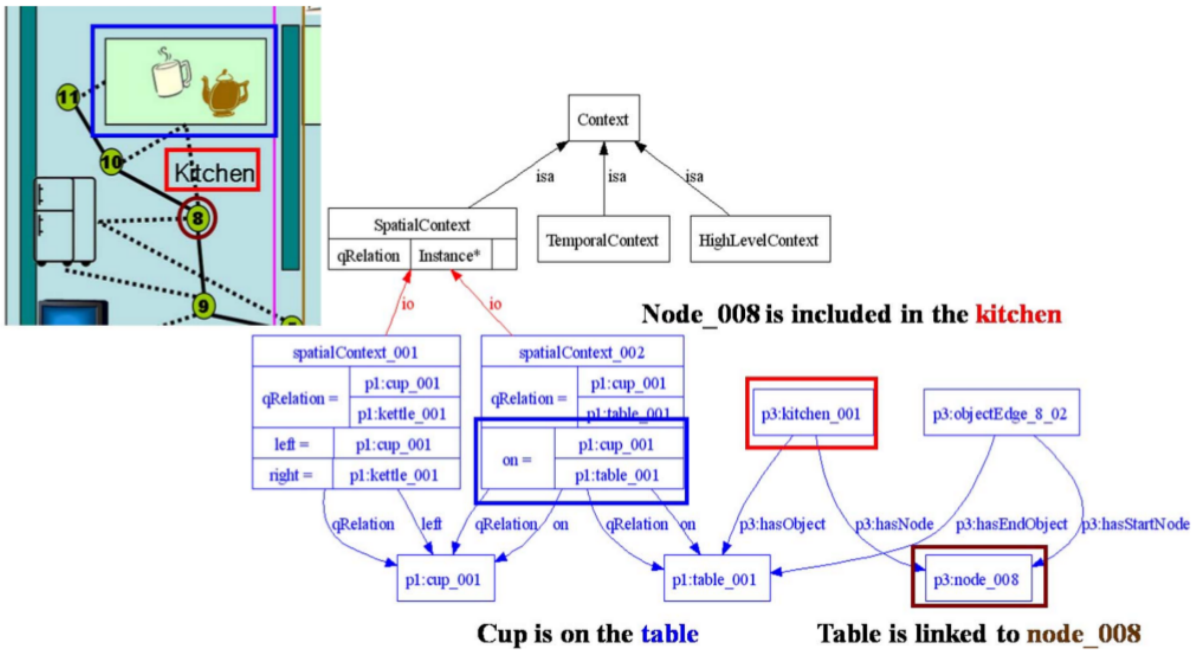


Fig. 4.2: An example of the approach proposed in [83]. Ontology schema and the instance defining where to go to find the cup.

4.2 Related Work

Scene analysis is a big definition and contains several aspects, such as object identification [132, 25], localization [33, 79, 52], discovery [73, 72, 152] and so on. Literally it means to analyze a scene which is comprised of different composites. Depending on the context, each individual scene could be very different. For a table-top scene, the scene may contain several objects which are commonly found on tables, such as, books and computers. For a traffic scene, the scene is totally different and normally consists of cars, humans and other relevant objects. The goal of different applications of scene analysis is not the same either. Some approaches concentrate on identifying and localizing the involved objects, while some other approaches try to discover objects in a cluttered environment. In the following, we provide a short review on existing approaches, and we focus on the ones that use context knowledge to analyze the perceived scene.

4.2.1 Using Knowledge as Logic Rules

Several previous methods represent context knowledge as descriptive logic rules to help robots understand the perceived scenes. Using description logic [7, 42, 17], ontologies [56, 145] are used to encode context about the composition of scenes, such as, a set of cutlery contains a knife, a fork, a spoon and so on. Using reasoning engines, such as Racer [49], Pellet [124] and FaCT++ [142], missing or wrong items in the scene can be inferred so as to give robots higher-level understanding of the perceived scene. In addition, robot actions are sometimes also encoded as ontologies. Different steps for performing a certain task, such as setting up a table, are defined as the composites. Based on inference results of the defined ontologies, corresponding actions are triggered for the operating robot.

Lim et al. [83] presented an ontology-based knowledge framework, in which they model robot knowledge as a semantic network. This framework is comprised of two parts: knowledge description and knowledge association. Knowledge description combines knowledge regarding perceptual features, part objects, metric maps, and primitive behaviours with knowledge about perceptual concepts, objects, semantic maps, tasks, and contexts. Knowledge association adopts both unidirectional and bidirectional rules to perform logical inference. This framework enabled their robot to complete a "find a cup" task in spite of hidden or partial data. An example of this approach is depicted in Fig. 4.2.

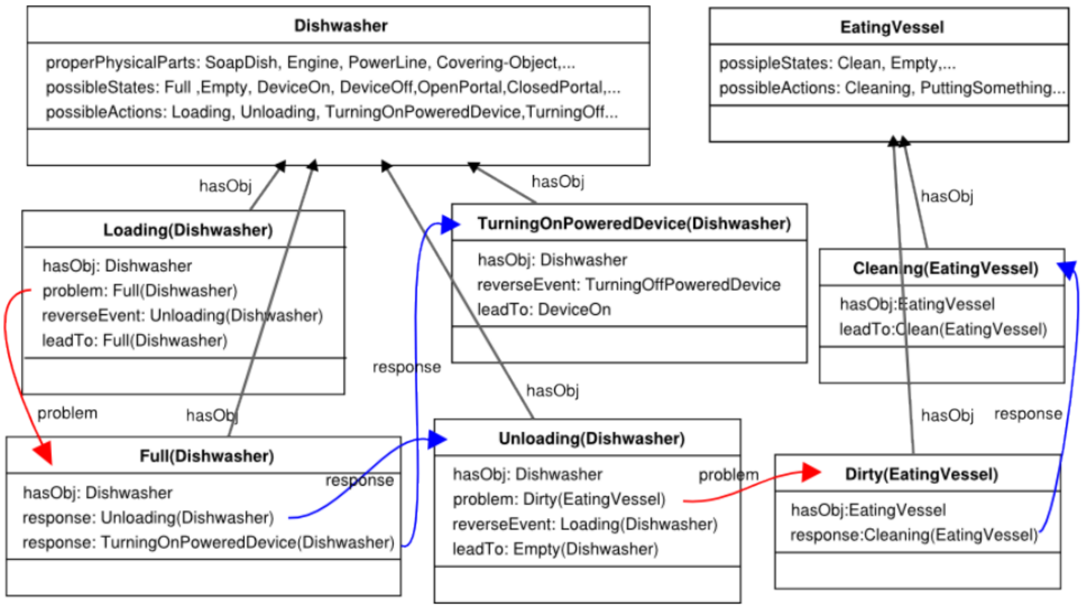
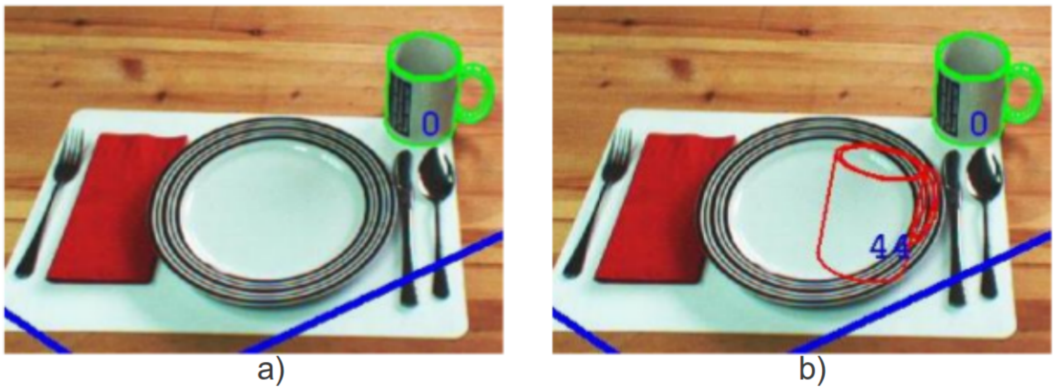

Dishwasher
properPhysicalParts: SoapDish, Engine, PowerLine, Covering-Object,...
possibleStates: Full ,Empty, DeviceOn, DeviceOff,OpenPortal,ClosedPortal,...
possibleActions: Loading, Unloading, TurningOnPoweredDevice,TurningOff...
EatingVessel
possipleStates: Clean, Empty,...
possibleActions: Cleaning, PuttingSomething...
hasObj
hasObj
hasObj
Loading(Dishwasher)
hasObj: Dishwasher
problem: Full(Dishwasher)
reverseEvent: Unloading(Dishwasher)
leadTo: Full(Dishwasher)
TurningOnPoweredDevice(Dishwasher)
hasObj: Dishwasher
reverseEvent: TurningOffPoweredDevice
leadTo: DeviceOn
Cleaning(EatingVessel)
hasObj:EatingVessel
leadTo:Clean(EatingVessel)
response
hasObj
problem
hasObj
hasObj
response
Full(Dishwasher)
hasObj: Dishwasher
response: Unloading(Dishwasher)
response: TurningOnPoweredDevice(Dishwasher)
response
Unloading(Dishwasher)
hasObj: Dishwasher
problem: Dirty(EatingVessel)
reverseEvent: Loading(Dishwasher)
leadTo: Empty(Dishwasher)
problem
Dirty(EatingVessel)
hasObj:EatingVessel
response:Cleaning(EatingVessel)

Fig. 4.3: An example of the approach proposed in [135]: common-sense knowledge about a dishwasher. The diagram shows some relations, actions, states, and objects that are closely related with the concept Dishwasher.

Tenorth et al. [135] proposed a system for building environment models for robots by combining different types of knowledge. Spatial information about objects in the environment is combined with encyclopedic knowledge to inform robots about the types and properties of objects. In addition, common-sense knowledge is used to describe the functionality of the involved objects. Furthermore, by learning statistical relational models, another type of knowledge is derived from observations of human activities. By providing robots deeper knowledge about objects, such as their types and their functionalities, this system helps robots accomplish complex tasks like cleaning dishes. An example describing a dishwasher presented in this work is shown in Fig. 4.3.



```
Place the PLACEMAT in **front** of the chair.
Place the NAPKIN just **left** of the center of the placemat.
Place the PLATE(ceramic, paper or plastic, Ceramic preferre
 in the **center** so that it just covers the right side of
 the napkin.
Place the FORK on the **side** of the napkin.
Place the KNIFE to the **right** so that the blade faces the
plate.
Place the SPOON right **next** to the knife.
Place the CUP to the top right **corner** of the placemat.
```

c)

Fig. 4.4: An example of the approach proposed in [106]. a) Correct result for the green mug. b) Hallucinated red Mug, correctly discarded by the proposed system. c) Instructions from http://www.wikihow.com/Set-a-Table (set of objects in capital, spatial relations in bold).

Pangercic et al. [106] proposed a top-down guided 3D model-based vision algorithm for assistive household environments. They use "how-to" instructions which are parsed and extracted from the wikihow.com webpage [61] (one of the largest resources of natural language task descriptions in the world) to shape the top-down guidance. The robots knowledge base is represented in Description Logics (DL) using the Web Ontology Language (OWL) [91]. Based on the knowledge base, inferences are obtained using SWI-Prolog queries [154]. Using this system, the task "how to set a table" is accomplished, in which a robot makes a table ready for a meal according to the instructions obtained from the wikihow webpage. An example of this system is demonstrated in Fig. 4.4.

In addition to ontology-based approaches, some other methods use first-order logic [127, 30] or variants of first-order logic, such as Markov logic networks [117] or Bayesian logic networks [68], to formulate context knowledge to solve scene analysis. Blodow et al. [15] use a Markov logic network to address the problem of object identity resolution. Instead

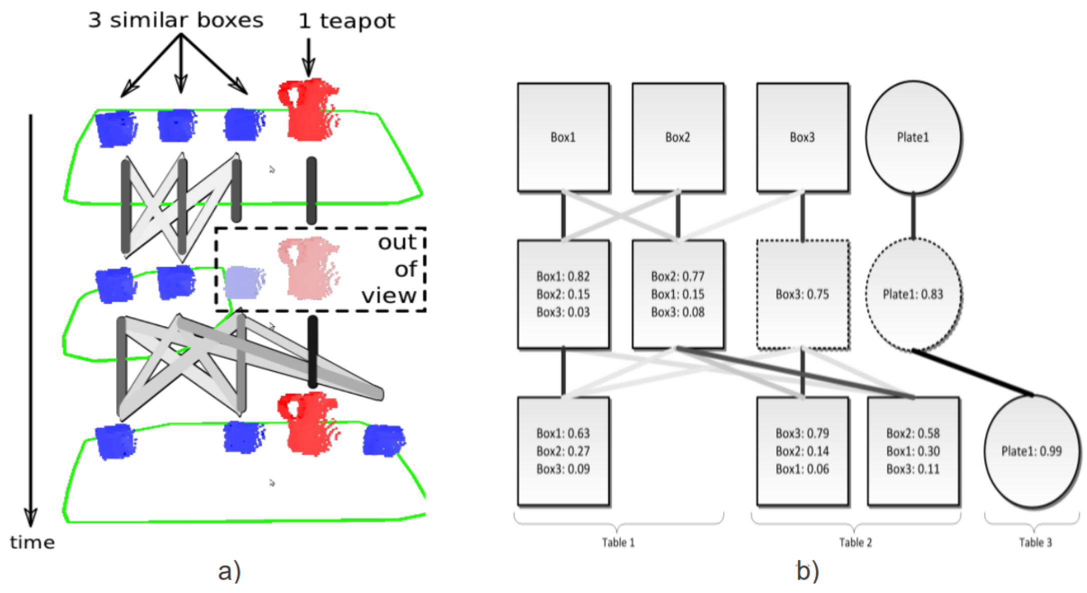


Fig. 4.5: An example of the approach proposed in [15]. a) Illustration the output of the proposed system (boxes and teapot clusters detected on a table) over three time steps. b) An example of inference results. Every square/circle represents an observation that was made (cluster). Dotted squares/circles are in fact not observed but are shown in order to illustrate beliefs on unobserved areas.

of generating scene graphs of the perceived scenes, they focus on inferring the temporal correspondence between observations and entities. By keeping track of where objects of interest are located, they aim to provide robots environment awareness, so that robots are able to infer which observations refer to which entities in the real world. An example of this approach is shown in Fig. 4.5.

The approaches introduced here use knowledge in the form of logic rules for scene analysis, however, none of them used such knowledge to provide a semantic abstract scene graph of the perceived scenes. Instead, they apply knowledge for some other purposes, such as object detection and reasoning about the scenes.

4.2.2 Using Knowledge through HRI

In addition to the approaches, which encode context knowledge as logical knowledge bases, there exist also methods that exploit context knowledge through human robot interaction (HRI) [39]. Motivated from psycholinguistic studies, Swadzba et al. [133] proposed a computational model for arranging objects into a set of dependency trees via spatial relations extracted from human verbal input. Assuming that objects are arranged in a hierarchical manner, they predict intermediate structures which support other object structures, such as, “soft toys lie on the table”. The objects at the leaves of the trees are assumed to be known and used to compute potential planar patches for their parent nodes. The computed patches are adapted to real planar surfaces, so that wrong object assignments are

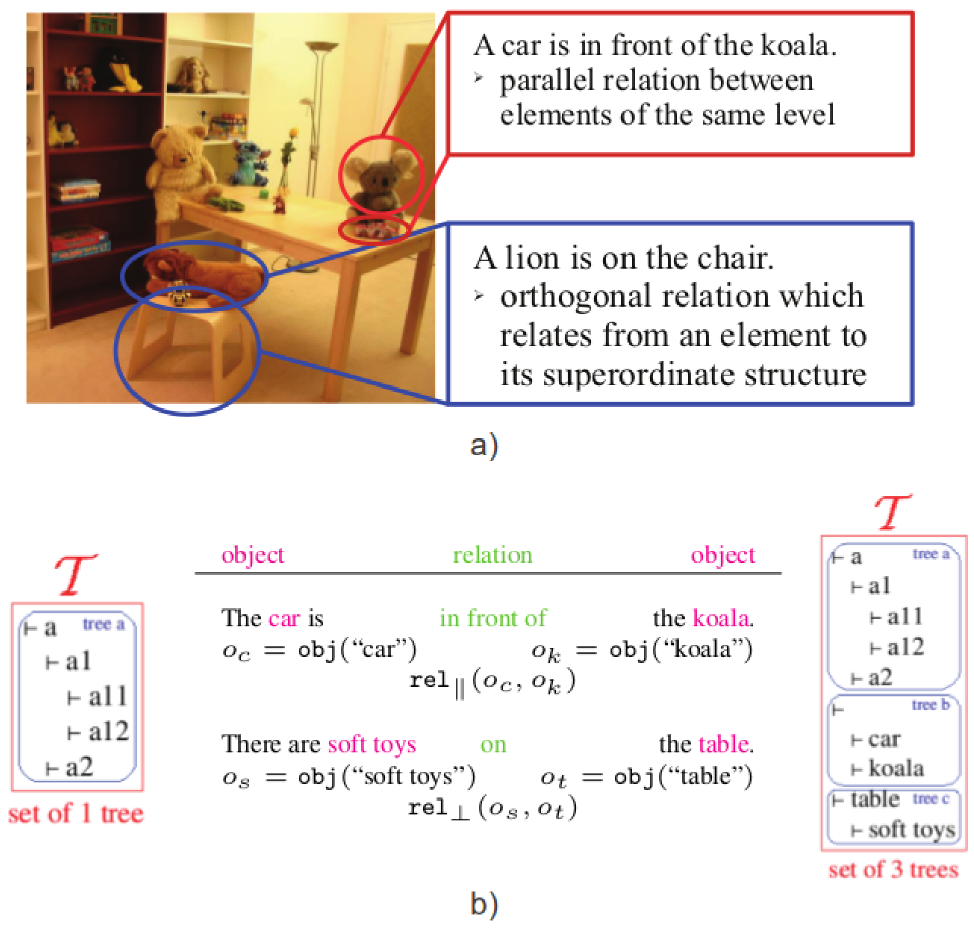


Fig. 4.6: An example of the approach proposed in [133]. a) Two typical relations (parallel and orthogonal) between scene elements described by human verbal input. b) Examples of the defined tree structure.

corrected. In addition, new object relations which were not given in the verbal descriptions could also be introduced. In this way, they could generate a model of the scene through context encoded in the verbal input of human observers. An example of this approach is depicted in Fig. 4.6. Other examples of this kind of approaches can be found in [160], [132] and [90]. Such approaches focus mainly on using methods of HRI, such as human verbal input, to describe a scene or detect objects.

4.2.3 Using Knowledge in Other Form

In addition to the approaches introduced above, a number of approaches analyze scenes using context knowledge in other form. Grundmann et al. [46] proposed a method to increase the estimation accuracy of independent sub-state estimation using statistical dependencies in the prior. The dependencies in the prior are modelled by physical relations. They use a physics engine to test the validity of the sampled physical relations. Scene models that fail the validity check of the implemented physics engine are given a probability of zero. In this way, a better approximation of the joint posterior is achieved. Another example of

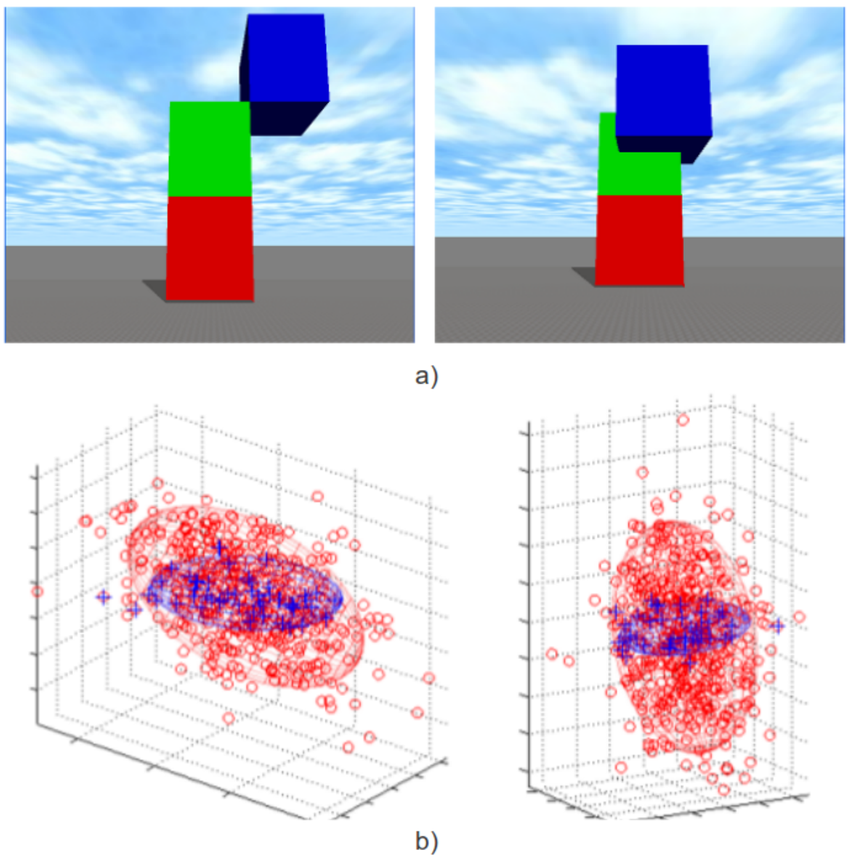


Fig. 4.7: Using a physics engine to check the validity of scene models. a) An example from the approach presented in [148]. b) Improved estimation (shown by blue) of the joint posterior from the approach presented in [46].

using physics engines to check scene validity was presented in [148]. Fig. 4.7 illustrates an example of both approaches.

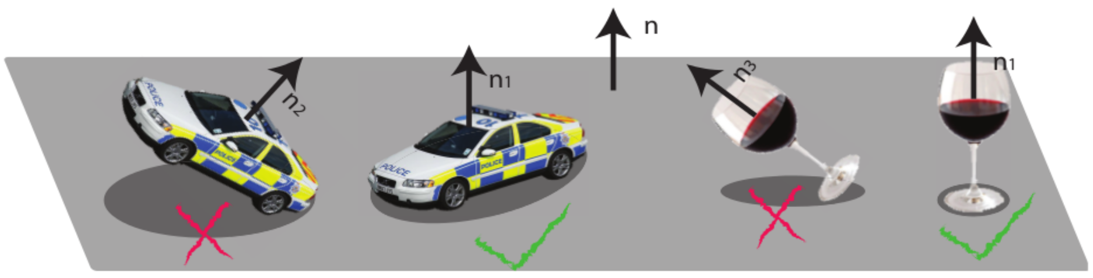


Fig. 4.8: An example from the approach presented in [8]. If the normal of a plane is n, objects lying on this plane tend to share the same normal direction $n_1//n$. Objects whose normal is not parallel to n (e.g. n_2 and n_3) are unlikely to sit on that supporting plane.

By modelling the relations between objects and their supporting surfaces in the image as a graphical model, Bao et al. [8] formulate the problem of objects detection as an optimization problem, in which parameters such as the object locations or the focal length of the camera are optimized. They follow the intuition that objects' location and pose in the 3D space are not arbitrarily distributed but rather constrained by the fact that objects must lie on one or multiple supporting surfaces. Such supporting surfaces are modelled by means of hidden parameters. The solution to the problem is finding the set of parameters that maximizes the joint probability. An example of this approach is demonstrated in Fig. 4.8.

Approaches introduced here have in common that they do not use knowledge in the form of logical knowledge bases, which are a more sophisticated and theoretically sounder way of applying knowledge. Instead, they encode knowledge as simple factors or constraints and use these directly in the continuous domain. In addition, none of these approaches provided a semantic abstract scene graph of the perceived scenes.

4.3 Contributions

Other than the approaches introduced above, we propose a probabilistic approach to generate abstract scene graphs from uncertain 6D pose estimates. We focus on generating a semantic understanding of the perceived scenes that well explains the composition of the scene and the inter-object relations. The proposed system is realized by our knowledge-supervised MCMC sampling technique. We employ *Markov Logic Networks* (MLNs) [117] to encode the underlying context knowledge as descriptive logic rules. In addition, we use a probabilistic sensor model to encode the fact that measurements are subject to significant uncertainty. We integrate the measurements with the uncertain scene graph in a data driven MCMC process. Our system is fully probabilistic and links the high-level abstract scene description to uncertain low level measurements. Moreover, false estimates of the object poses and hidden objects of the perceived scenes can be systematically detected using the Markov logic inference techniques.

4.4 A Generalizable Knowledge-Supervised MCMC Sampling Framework

In this chapter, we apply our generalizable knowledge-supervised MCMC (KSMCMC) sampling framework to interpret table-top scenes. The fundamental idea of our framework is to define an abstract model M to explain data D with the help of rule-based context knowledge (defined in MLNs). According to the Bayes' theorem, a main criterion for evaluating how well the abstract model M matches the input data D is the *posterior* probability of the model conditioned on the data $p(M|D)$ which can be calculated as follows:

$$p(M|D) \propto p(D|M) \cdot p(M). \tag{4.1}$$

Here, the term $p(D|M)$ is usually called the *likelihood* and indicates how probable the observed data are for different settings of the model. The term $p(M)$ is the *prior* describing what kind of models are possible at all. We propose to realize the prior by making use of context knowledge in the form of descriptive rules, so that the prior distribution is shaped in such a way that impossible models are ruled out. Calculations of the prior and likelihood are explained in section 4.7 and section 4.8 respectively.

Starting from an initial guess of the model, we apply a data driven MCMC [143] process to improve the quality of the abstract model. Our goal is then to find the model M^* that best explains the data and meanwhile complies with the prior, which leads to the maximum of the posterior probability:

$$M^* = \underset{M \in \Omega}{\operatorname{argmax}}\ p(M|D), \tag{4.2}$$

where Ω indicates the entire solution space. Details about this data driven MCMC process are provided in section 4.9.

4.5 Rule-Based Context Knowledge

Objects on a table-top are not arranged arbitrarily but they follow certain physical constraints. In our system, we formulate such constraints as context knowledge using descriptive rules. This knowledge helps to model table-top scenes efficiently by ruling out impossible scenes. We express physical constraints in a table coordinate system. This table coordinate system can be efficiently detected from the sensor input, e.g., using the Point Cloud Library [119]. To apply context knowledge for scene modelling, we transform the initial guess of the 6D poses of the objects from the sensor coordinate system into the table coordinate system (see Fig. 4.9).

4.5.1 Evidence Predicates

Evidences are abstract terms that are detected from the perceived scene given the object poses and models. To formulate the knowledge as descriptive rules, we first define several evidence predicates that describe the features of table-top scenes. These predicates are shown in Table 4.1 and have the following meanings:

- *stable(object)*: this predicate indicates that an object has a stable pose, i.e., it stably lies on a horizontal plane. For instance, objects #1, #2, #3 and #5 in Fig. 4.9-b have a stable pose. Objects #4 and #6, in contrast, have an unstable pose.
- *table(object)*: this predicate provides the possibility to model tables as objects, so that we can use it in the reasoning.
- *contact(object,object)*: this predicate indicates whether two objects have contact with each other. In the scene shown in Fig. 4.9-b, for instance, there exists contact between object #2 and #4, and between object #3 and #4. By contrast, contact does not exist between object #4 and #5, or between object #2 and #5.

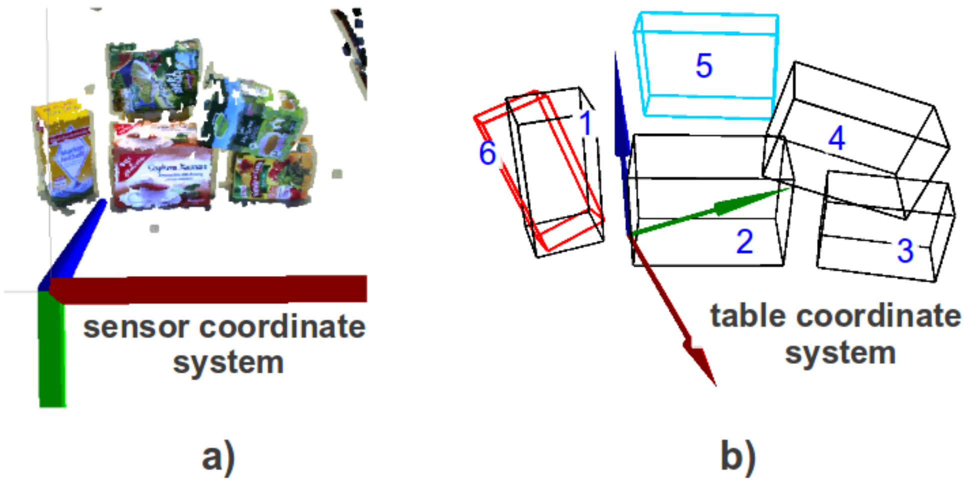


Fig. 4.9: a) The object poses are initially calculated in the sensor coordinate system. b) To Apply context knowledge in a table-top scene, the objects poses are transformed into the table coordinate system.

- *intersect(object,object)*: this predicate indicates whether two objects intersect with each other. In the scene shown in Fig. 4.9-b, intersection only exists between object #1 and #6. The predicates *intersect(object,object)* and *contact(object,object)* are mutually exclusive.
- *higher(object,object)*: this predicate expresses that the position of the first attribute is higher than that of the second attribute in the table coordinate system. In the scene shown in Fig. 4.9-b, for instance, this predicate is true for (#5,#2), (#4,#2) and (#4,#3).
- *hover(object)*: this predicate means that an object does not have any contact with other objects including the table. In the scene shown in Fig. 4.9-b, this predicate is true only for object #5.

evidence predicates
stable(object)
table(object)
contact(object,object)
intersect(object,object)
hover(object)
higher(object,object)

Tab. 4.1: Declaration of evidence predicates

4.5.2 Query Predicates

Having defined the evidence predicates, we formulate context knowledge as descriptive rules using Markov logic in Table 4.3. Using these rules, query predicates are inferred given the evidence. In principle, the query predicates represent the questions that Markov logic can answer given the defined knowledge base. These query predicates are listed in Table 4.2 and have the following interpretations:

- *hidden(object)*: this predicate expresses that there is an hidden object in the scene under the object that is represented by the attribute. In the scene shown in Fig. 4.9-b, this predicate is true for object #5.
- *false(object)*: this predicate indicates that the object represented by the attribute is a false estimate. In the scene shown in Fig. 4.9-b, this predicate is true for object #6.
- *supportive(object)*: this predicate indicates that the object represented by the attribute physically supports other objects.
- *supported(object)*: this predicate indicates that the object represented by the attribute is physically supported by other objects. *supportive(object)* and *supported(object)* are two auxiliary query predicates which are used to infer about *hidden(object)* and *false(object)*. The meaning of these two predicates will be explained together with the defined rules in the following section.

query predicates
supported(object)
supportive(object)
hidden(object)
false(object)

Tab. 4.2: Declaration of query predicates

4.5.3 Context Knowledge Defined as Logic Rules

According to the certainty of knowledge, knowledge can be defined as soft rules or hard rules in Markov logic. Knowledge with great certainty that holds in all cases are defined as hard rules. Hard rules are assigned a weight of ∞ in Markov logic. By contrast, soft rules are used to encode uncertain knowledge and are given a probabilistic weight in Markov logic representing the uncertainty of the corresponding knowledge. Here, we define several hard and soft rules (see Table 4.3) to model table-top scenes.

index i	weight ω_i	formula F_i
r_1	∞	!higher(o1,o1)
r_2	∞	!intersect(o1,o1)
r_3	∞	!contact(o1,o1)
r_4	∞	contact(o1,o2) $\rightarrow$ contact(o2,o1)
r_5	∞	intersect(o1,o2) $\rightarrow$ intersect(o2,o1)
r_6	∞	higher(o1,o2) $\rightarrow$!higher(o2,o1)
r_7	∞	table(o1) $\rightarrow$!false(o1)
r_8	∞	table(o1) $\rightarrow$!hidden(o1)
r_9	∞	table(o1) $\rightarrow$ stable(o1)
r_{10}	∞	stable(o1) $\wedge$ stable(o2) $\wedge$ contact(o1,o2) $\wedge$ higher(o1,o2) $\rightarrow$ supportive(o2) $\wedge$ supported(o1)
r_{11}	log(0.70/0.30)	supported(o1) $\rightarrow$!hidden(o1)
r_{12}	log(0.90/0.10)	!stable(o1) $\rightarrow$!supportive(o1)
r_{13}	log(0.90/0.10)	hover(o1) $\rightarrow$ false(o1) v hidden(o1)
r_{14}	log(0.90/0.10)	intersect(o1,o2) $\rightarrow$ false(o1) v false(o2)
r_{15}	log(0.70/0.30)	supportive(o1) $\rightarrow$!false(o1)
r_{16}	log(0.90/0.10)	stable(o1) $\rightarrow$!false(o1)

Tab. 4.3: Declaration of rules

Hard Rules In all, ten hard rules are defined, and they are explained as follows:

r_1: This rule expresses the fact that an object cannot be higher than itself.

r_2: This rule indicates that an object does not intersect with itself.

r_3: This rule encodes the fact that an object does not have contact with itself.

r_4: This rule means that the predicate *contact(object,object)* is commutative, i.e., given that object o_1 has contact with o_2, the statement that object o_2 has contact with o_1 is true.

r_5: This rule means that the predicate *intersect(object,object)* is commutative, i.e., given that object o_1 intersects with o_2, the statement that object o_2 intersects with o_1 is true.

r_6: This rule means that the predicate *higher(object,object)* is not commutative, i.e., given that object o_1 is higher than o_2, the statement that object o_2 is higher than o_1 is wrong.

r_7: This rule expresses that in a table-top scene, the table (as an object) is not a false estimate.

r_8: This rule expresses that in a table-top scene, the table (as an object) is the lowest object in the scene and has no hidden object under it.

r_9: This rule expresses that in a table-top scene, the table (as an object) has a stable pose.

r_{10}: This rule describes "supportive" and "supported" relations between two objects with a stable pose. These relations do not apply for objects with unstable poses. In the scene shown in Fig. 4.9-b, for example, this relation holds between the table and objects #1, #2, and #3 respectively.

Soft Rules In addition to the hard rules, six soft rules are defined, and they are explained as follows:

r_{11}: This rule encodes the assumption that an object that is already known to be supported (through rule #10) is not likely to have a hidden object under it.

r_{12}: This rule expresses the assumption that an object with an unstable pose is unlikely to be supportive.

r_{13}: This rule states the assumption that a hovering object is either a false estimate or has a hidden support under it.

r_{14}: This rule states the assumption that if two objects intersect, then one of them is probably a false estimate.

r_{15}: This rule indicates the assumption that a supportive object is unlikely to be a false estimate.

r_{16}: This rule indicates the assumption that an object with a stable pose is unlikely to be a false estimate.

The choice of the rules is a problem-oriented engineering step, and the rules given here serve as an example of how to encode the properties of typical table-top scenes.

4.5.4 Weights in Log-odd Form

Rules #11 to #16 are soft and are therefore given a weight in the log-odd form describing our belief on how often the corresponding uncertain knowledge holds. A weight in the log-odd form $log(p1/p2)$ with $p1, p2 \in (0, 1)$ and $p1 + p2 = 1$, means that the corresponding rule holds with the probability of $p1$ [117]. These weights can either be learned [87, 64, 65] or manually designed [66]. In our work, we use two belief levels $log(0.90/0.10)$ (very sure) and $log(0.70/0.30)$ (relatively sure) to encode the uncertainty of knowledge. Using Markov Logic inference, we can answer the queries *hidden(object)* and *false(object)* in the form of a probability.

4.5.5 Evidence Generation

To do inference in MLNs, necessary evidences must be given as input. In this work we focus on objects with a regular shape, in particular, objects that can be well represented by an oriented bounding box (OBB) [11]. However, the aforementioned principles generalize over objects with other shapes, as long as evidences are provided accordingly. In the following we elaborate on how to generate evidences by analyzing the oriented bounding boxes of detected objects:

- *stable*(*object*): if any edge of an object OBB is parallel to the vertical axis of the table coordinate system, we define this object to have a stable pose, i.e., *stable*(*object*)=True. Examples are shown in Fig. 4.10. Here object #0, #1, #3 and #4 have a stable pose. In contrast, object #2 has a unstable pose.

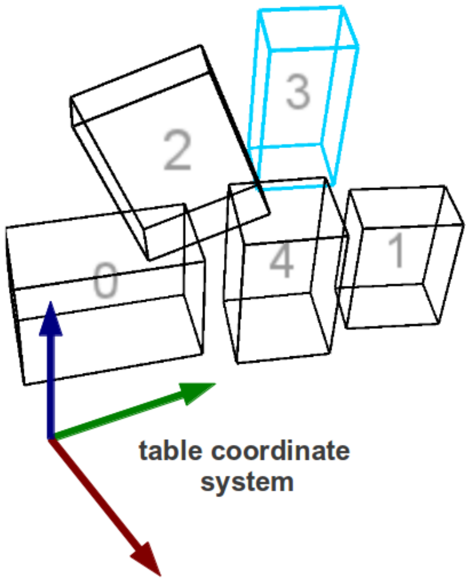


Fig. 4.10: An example scene.

- *contact(object,object)*: to detect whether two objects have contact with each other, we search for points of intersection between the OBB of these two objects. If two OBBs contact but do not intersect each other, there are three possible cases:
  - There is only one point of intersection, and it coincides with one of the six vertices of either OBB.
  - There are multiple points of intersection and all points are co-linear and lie on one of the twelve edges of either OBB (for example, the contact between object #2 and #4 in Fig. 4.10).

  - There are multiple points of intersection and all points are coplanar and lie on one of the six facets of either OBB (for example, the contact between object #0 and the table in Fig. 4.10).

  In each of the above three cases, we set *contact(object,object)*=True and *intersect(object,object)*=False.

- *intersect(object,object)*: *contact(object,object)* and *intersect(object,object)* are mutually exclusive, i.e., they can not be true at the same time. If there exist points of intersection between two OBBs, and none of the above cases applies, or if an OBB completely contains the other OBB, then we set *intersect(object,object)*=True. In all other cases, we set *contact(object,object)*=False and *intersect(object,object)*=False. An example of the case that two objects intersect with each other is depicted by Fig. 4.11. Here *intersect(object,object)* is true for object #3 and #5. Points of intersection are shown by gray spheres.

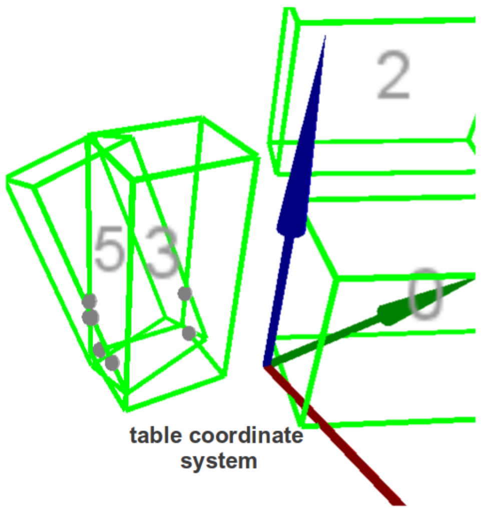


Fig. 4.11: An example of the case that two objects intersect with each other. Points of intersection are shown by gray spheres.

- *hover(object)*: if an object does not have any contact or intersection with other objects including the table, then we set *hover(object)*=True. An example for this case is the object #3 in Fig. 4.10.

- *higher(object,object)*: if the position of *object1* is higher than the position of *object2* in the table coordinate system, then we set *higher(object1,object2)*=True.

4.6 Estimation of Object Poses

To determine 6D object poses, we apply a pose estimation approach that is similar to the approach presented by Grundmann et al. [44]. The basic computational steps are given in Alg. 2. The algorithm is based on Scale-invariant feature transform (SIFT) keypoints [88] that are extracted from triangulated stereo images or RGBD measurements, e.g., from the Kinect sensor.

Algorithm 2 6D Object Pose Estimation

```
Require:
    $z$, input measurement
    $K$, object database
Ensure:
    $H$, set of pose hypotheses
 1: extract SIFT keypoints from $z$
 2: match keypoints to database $K$
 3: for all object models $k \in K$ do
 4:    for $i$ iterations do
 5:       randomly choose three keypoints matched to $k$
 6:       compute object pose hypothesis from matches
 7:    end for
 8:    cluster pose hypotheses for object $k$
 9:    add clustered hypotheses to $H$
10: end for
```

In a first step, the SIFT keypoints of the observed objects are matched to a database K of object models. In our work, we use the object database of the Deutsche Servicerobotik Initiative (DESIRE) project [1]. The object models are generated by an accurate 3D modelling device which is equipped with a turn table, a movable stereo camera pair and a digitizer. The turn table is used to place the object which should be scanned. The stereo camera pair acquires stereo images of the target object. Since this camera pair is movable, together with the turn table, stereo images of the target object can be obtained for many view angles. Mounted at a fixed position, the digitizer provides 3D data using structured light. The modelling device is well calibrated and can therefore generate an accurate 3D model of the target object. More details about the hardware setup of this modelling device can be found in [158]. An example of the modelling process is illustrated in Fig. 4.12. As shown in Fig. 4.12-a, the stereo camera pair first acquires stereo images of the target object from all possible view angels. Then 2D SIFT keypoints are extracted from these stereo images. Through keypoints matching and triangulation, a 3D point cloud (4.12-c) is generated out of the matched 2D SIFT keypoints. Based on this point cloud and some further optimization steps, the final object model is generated in the form of a textured

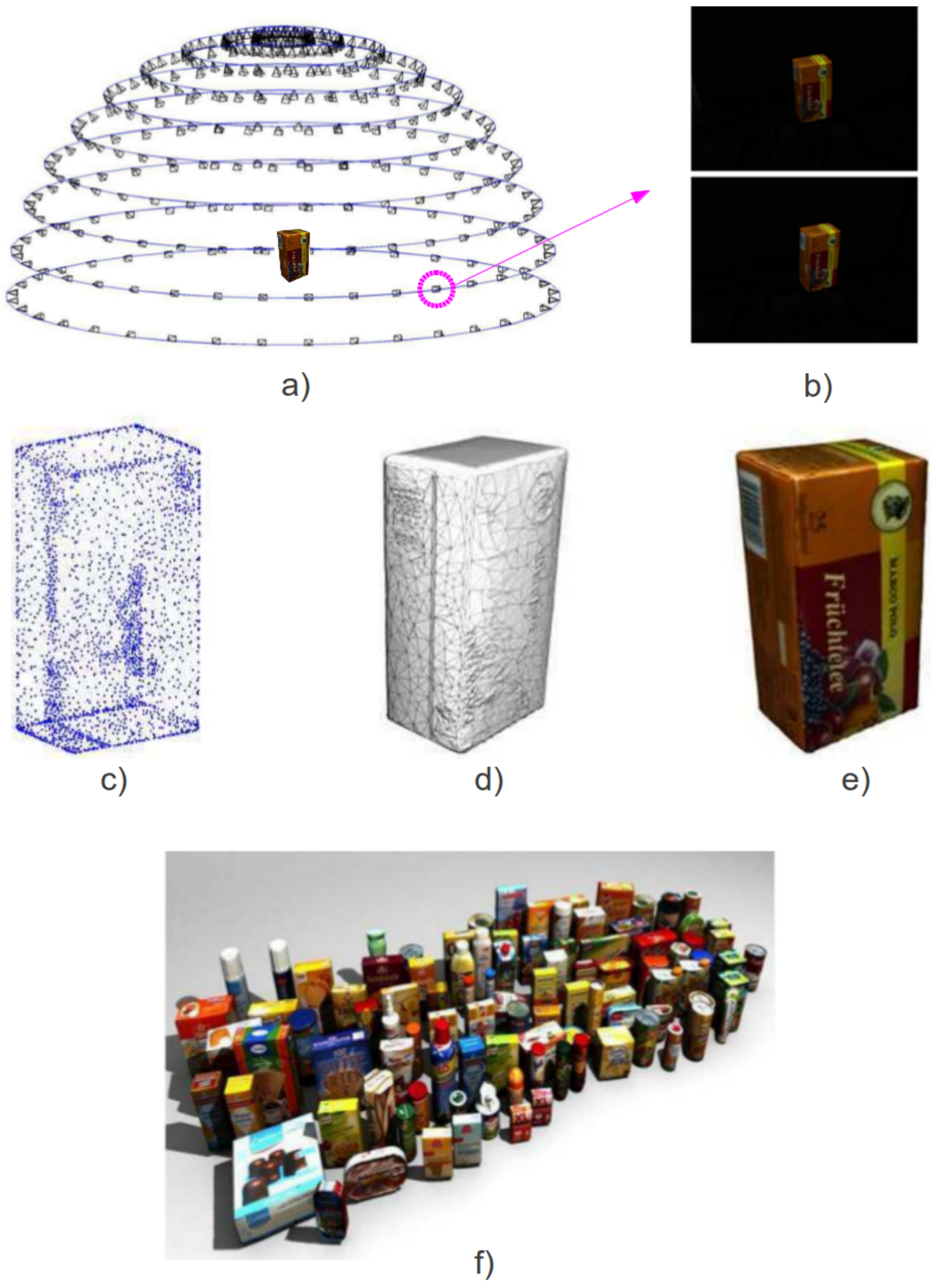


Fig. 4.12: An example of the modelling process. All sub-figures are obtained from [47]. a) Camera poses made possible by the turn table and the camera movement. b) A stereo image pair obtained from the highlighted (magenta) camera pose. c) High resolution point cloud obtained through triangulation of matched feature points in stereo images. d) Generated triangle mesh. e) Textured triangle mesh. f) An overview of the object database.

mesh of 3D SIFT keypoints (4.12-e). In 4.12-f, an overview of the DESIRE object database which contains 100 house-hold items is demonstrated.

For each object model $k \in K$, a maximum of i hypotheses is generated. To generate hypotheses, three keypoints are chosen randomly from the set of keypoints that has been matched to model k. Here, the keypoints extracted from the stereo images must undergo a certain matching scheme to check their validity. The matching scheme is depicted in Fig. 4.13. First of all, the extracted keypoints are checked by stereo matching, i.e., to check whether a keypoint found in the left image can also be found in the right image, or vice versa. The keypoints that have survived the stereo matching are matched with the object database separately. If a keypoint in the left image and its corresponding keypoint in the right image (that has been matched through stereo matching) refer to the same point in the object database, then this keypoint is a valid keypoint and can be used for pose estimation.

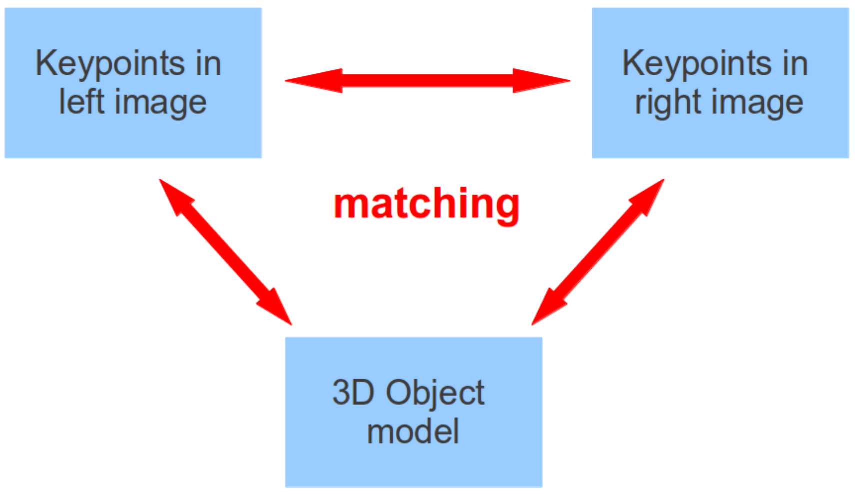


Fig. 4.13: Matching scheme for the SIFT keypoints extracted from images.

An object pose hypothesis is then computed from triples of these matched points. Finally, pose hypotheses are clustered, and outliers are removed using the RANSAC algorithm [28]. An example of pose estimation is depicted in Fig. 4.14. As shown in Fig. 4.14-a, the SIFT keypoints that are detected in the stereo image are firstly checked by stereo matching. Matched keypoints pairs are linked by yellow lines. These stereo-matched keypoints are further compared with the object database. In Fig. 4.14-b and c, the database-matched keypoints are shown by cyan in the left and the right image. Using these matched keypoints, pose hypotheses are generated which are shown in Fig. 4.14-d. Pose estimation is performed for each new scene but is not repeated during scene graph generation.

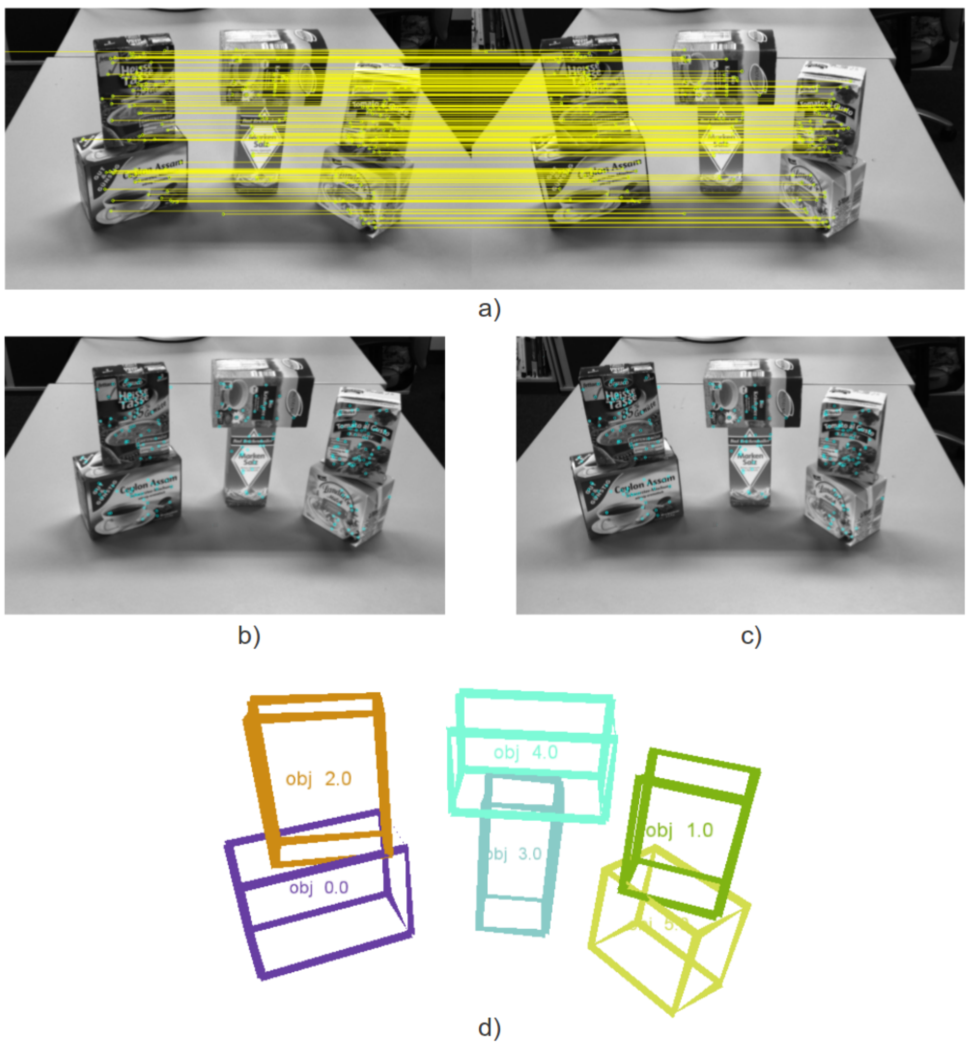


Fig. 4.14: An example of pose estimation. a) Stereo matching of detected SIFT keypoints. b) Database-matched keypoints of the left image. c) Database-matched keypoints of the right image. d) Generated pose hypotheses using the matched keypoints.

4.7 Calculation of Prior Probability

Having defined the predicates and the rules, a knowledge base is formulated in the form of a Markov logic network (MLN). A MLN initializes a ground Markov network [117], if it is provided with a finite set of constants. In our application, the detected objects and the table form the set of constants. The probability of a possible world x (a hypothesis of scene graph) is given by the probability distribution that is represented by this ground Markov network. As shown in the equation (2.4), this probability is calculated as follows:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i \omega_i n_i(x)\right),$$

where $n_i(x)$ is the number of true groundings of formula F_i in x, and ω_i is the weight of F_i. Z is a normalization factor. As can be seen in the above equation, the probability of a possible world is equal to the exponentiated sum of weights of formulas that are satisfied in this possible world divided by the normalization factor Z.

By ignoring the normalization factor Z, which is the same for all possible worlds, the unnormalized probability is used as the prior probability in equation (4.1):

$$p(M) = \exp\left(\sum_i \omega_i n_i(x)\right). \tag{4.3}$$

In this work, we adapt the ProbCog Toolbox [67] to perform MLN inference and to calculate this unnormalized probability.

4.8 Calculation of Likelihood

To evaluate estimated object poses, we use a Gaussian sensor model as likelihood, which is similar to the approach proposed by Grundmann et al. [45]. For a pose estimate ψ, which corresponds to a scene graph M, we first determine the set of keypoints that have been matched in the object database. Let $(x_i, y_i),\ i = 1, 2, \cdots, n$, be the set of 2D image coordinates of the key points in the stereo image that are matched to the object database. Using the pin hole camera model [116], we project the model keypoints (x_i, y_i) into the image and denote the resulting set of coordinates as (x_i^{ψ}, y_i^{ψ}). The likelihood $p(D|M)$ in equation (4.1) is then calculated as

$$p(D|M) = \prod_i^n \left(\frac{1}{\sigma_x\sqrt{2\pi}}\, e^{-\frac{(x_i - x_i^{\psi})^2}{2\sigma_x^2}}\, \frac{1}{\sigma_y\sqrt{2\pi}}\, e^{-\frac{(y_i - y_i^{\psi})^2}{2\sigma_y^2}}\right), \tag{4.4}$$

where σ_x and σ_y are the standard deviation in x- and y-direction of the image coordinates. In our experiments, we use a standard deviation of 1 pixel for σ_x and σ_y. An illustration is given in Fig. 4.15. Here a pose hypothesis (shown in red) is evaluated against the database object pose (shown in blue). Key points in the stereo image that are matched with the object model are shown in cyan. For clarity, only the left camera image is shown. The projected model key points are shown in red. The correspondences between projected and matched key points are shown by yellow lines. The sensor model is calculated based on such correspondences.

4.9 Data-Driven MCMC

To find the scene graph that best explains the perceived scene, we apply a data driven MCMC process [143]. In the t-th iteration with scene graph M_t, we generate n new pose estimates $e_{t,i}, i = 1, 2, \cdots, n$, by adding Gaussian noises to the current pose estimate $e_{t,0}$

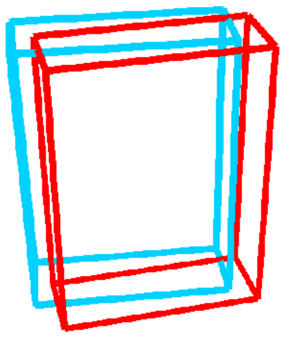
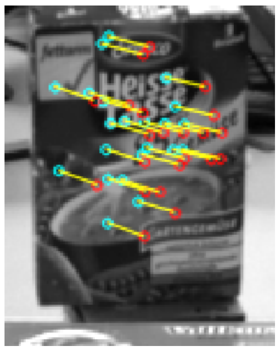
a)

b)


Fig. 4.15: Evaluation of a pose estimate using the Gaussian sensor model. a) A pose (red) is evaluated against the database object pose (blue). b) Key points in the stereo image that are matched with the object model are shown in cyan (for clarity, only the left camera image is shown). The projected model key points are shown in red. The correspondences between projected and matched key points are shown by yellow lines.

and weight them using the sensor model (equation (4.4)).

An example of generating new pose estimates is given in Fig. 4.16. The pose estimate with the best weight e_t^* is used to generate a new scene graph M_{t+1}. This scene graph is accepted by the probability $\lambda(M_t, M_{t+1})$, using the Metropolis-Hastings algorithm [20]:

$$\lambda(M_t, M_{t+1}) = \min\left(1, \frac{P(M_{t+1}|D)\cdot Q(M_t|M_{t+1})}{P(M_t|D)\cdot Q(M_{t+1}|M_t)}\right), \tag{4.5}$$

where $P(M_t|D)$ is the posterior probability of M_t (equation (4.1)). $Q(M_{t+1}|M_t)$ is the proposal probability of generating M_{t+1} out of M_t and is calculated as

$$Q(M_{t+1}|M_t) = \frac{weight(e_t^*)}{\sum_i^n weight(e_{t,i}) + weight(e_{t,0})}. \tag{4.6}$$

Similarly, $Q(M_t|M_{t+1})$ is computed as

$$Q(M_t|M_{t+1}) = \frac{weight(e_{t,0})}{\sum_i^n weight(e_{t,i}) + weight(e_{t,0})}. \tag{4.7}$$

Here, $weight(e_{t,i})$ is the sensor model that is calculated using $(e_{t,i})$ as pose estimate (equation (4.4)).

4.10 Experiments

We conducted numerous real world experiments to evaluate our approach. In each experiment, a number of household objects was placed on a table and a sensor measurement was taken. We then applied our approach to generate a scene graph and to infer hidden

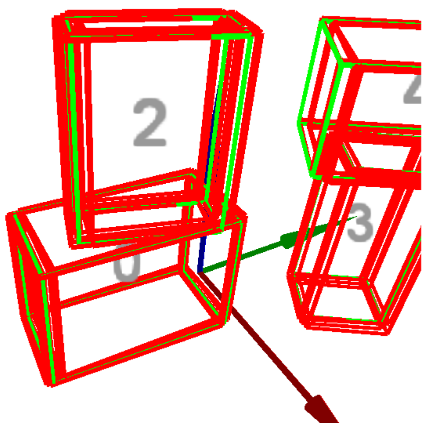


Fig. 4.16: Generating new pose estimates (red) by adding Gaussian noises to the current pose estimate (green).

objects or false estimates.

A selection of typical results is shown in Fig. 4.17 to 4.23. In each figure, the left camera image of the stereo image, the estimated poses, the resulting scene graph and the corresponding query probabilities are shown. False estimates and objects implying the existence of hidden objects are highlighted in red and cyan respectively. It can be seen, that all the perceived scenes are correctly represented by our scene graphs. Arrows indicate that an object stably supports another object. Undirected lines mean that two objects have an unstable contact.

4.10.1 Inference

In our experiments, the defined knowledge base (Table 4.3) is used to reason about false estimates and hidden objects in the perceived scenes. In all experiments, the false estimates and hidden objects are correctly inferred. In the used MLN tool [67], the query probabilities are calculated based on certain sampling methods, and their values v are normalized ($v \in [0, 1]$). We interpret these values as follows:

- If the value is around 0.5, i.e., $0.4 < v < 0.6$, the uncertainty of the corresponding query is the biggest, and we do not make decisions, e.g., *false(2)* and *hidden(0)* in result #3.

- If the value is greater than a given threshold, i.e., $v > 0.6$, the corresponding query is considered to be true, e.g., *false(5)* and *hidden(2)* in result #7.

- If the value is lower than a given threshold, i.e., $v < 0.4$, the corresponding query is considered to be false, e.g., *false(0)* and *hidden(4)* in result #1.

We manually labelled 25 complex table-top scenes. Each of the scenes contained several household objects of various types and had rather complex configurations, similar to those shown in Fig. 4.17 to 4.23. The 25 scenes contained in all 10 hidden objects and 5 false estimates, all of which were correctly inferred.

To check the robustness of our system, all the experiments were carried out 20 times. The generated scene graphs stay the same. In addition, false estimates and hidden objects in the scenes are also correctly inferred by the defined MLN in all repeated experiments.

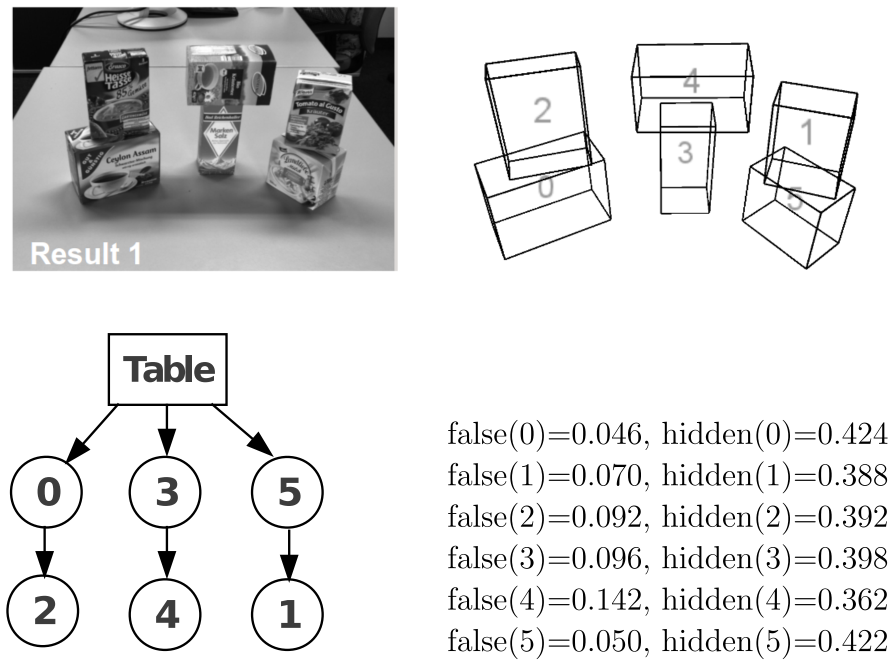
4.10.2 Runtime

In experiments, we have also tested the run time performance of the proposed system. In each iteration, the run time of our system is mainly spent on MLN reasoning (including evidence generation) and the MCMC process. With a single-threaded implementation on an Intel i7 CPU, the average processing time of each iteration for the experiments shown in this chapter is 2.18 seconds. 68.8% of this processing time is spent on MLN reasoning, and the other 31.2% is spent on the MCMC process. To get a good scene graph of the perceived scene, our system needs to perform 10 to 15 iterations.

4.11 Conclusions

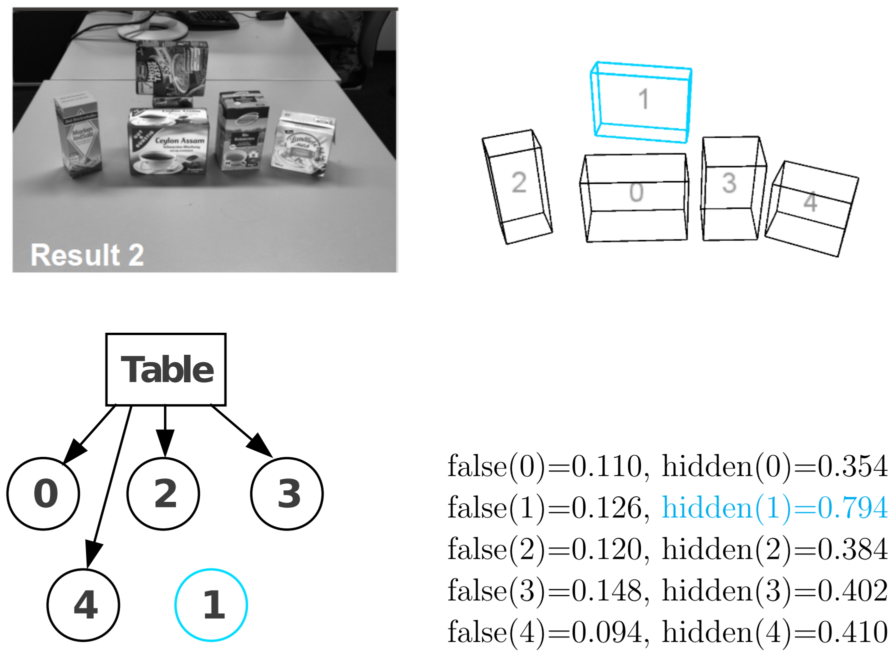
In this chapter, we used our knowledge-supervised MCMC sampling technique to model table-top scenes. Our system, as a whole, demonstrates a probabilistic approach to generate abstract scene graphs for table-top scenes using object pose estimation as input. Our approach explicitly makes use of task-specific context knowledge by defining this knowledge as descriptive logic rules in Markov logic. Integrating these with a probabilistic sensor model, we perform maximum posterior estimation of the scene parameters using our knowledge-supervised MCMC process.

We evaluated our approach using real world scenes. Experimental results confirm that our approach generates correct scene graphs which represent the perceived table-top scenes well. By reasoning in the defined MLN, false estimates of the object poses and hidden objects of the perceived scenes were correctly inferred.



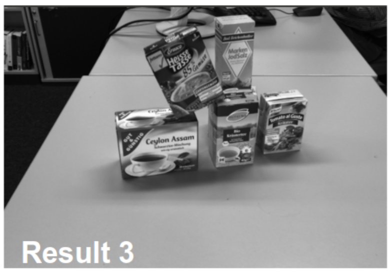
false(0)=0.046, hidden(0)=0.424
false(1)=0.070, hidden(1)=0.388
false(2)=0.092, hidden(2)=0.392
false(3)=0.096, hidden(3)=0.398
false(4)=0.142, hidden(4)=0.362
false(5)=0.050, hidden(5)=0.422

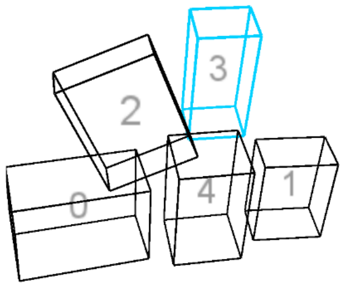
Fig. 4.17: Experimental result 1. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.

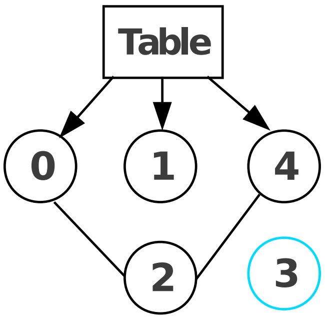


false(0)=0.110, hidden(0)=0.354
false(1)=0.126, hidden(1)=0.794
false(2)=0.120, hidden(2)=0.384
false(3)=0.148, hidden(3)=0.402
false(4)=0.094, hidden(4)=0.410

Fig. 4.18: Experimental result 2. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.

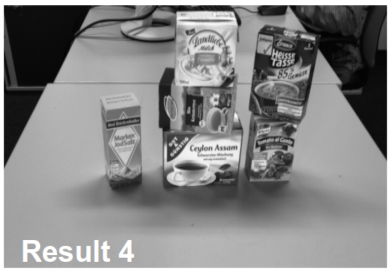


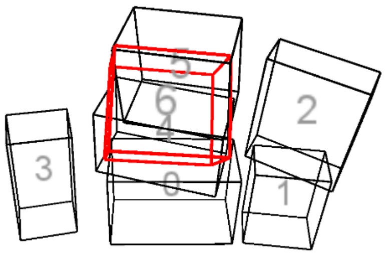


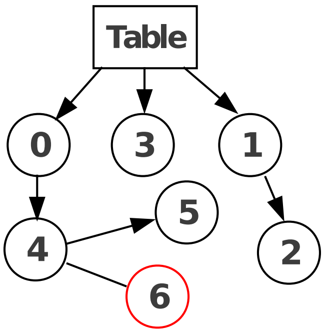


false(0)=0.074, hidden(0)=0.418
false(1)=0.080, hidden(1)=0.376
false(2)=0.518, hidden(2)=0.510
false(3)=0.106, hidden(3)=0.852
false(4)=0.090, hidden(4)=0.350

Fig. 4.19: Experimental result 3. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.

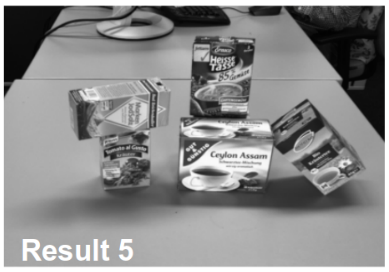


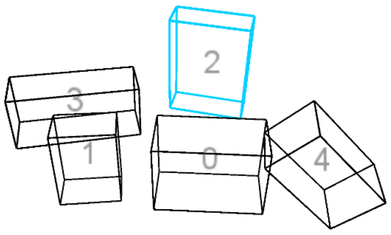


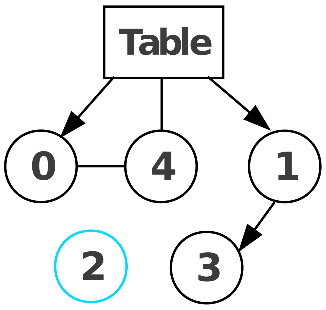


false(0)=0.074, hidden(0)=0.406
false(1)=0.036, hidden(1)=0.424
false(2)=0.092, hidden(2)=0.384
false(3)=0.102, hidden(3)=0.386
false(4)=0.094, hidden(4)=0.400
false(5)=0.130, hidden(5)=0.348
false(6)=0.912, hidden(6)=0.466

Fig. 4.20: Experimental result 4. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.

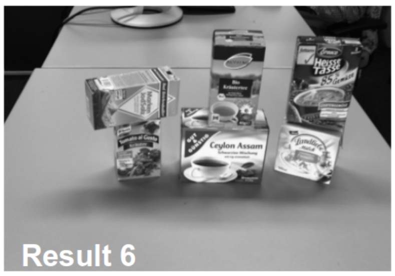


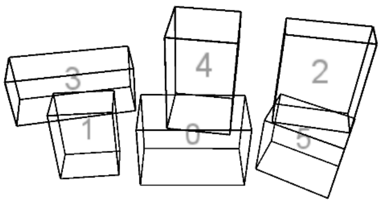


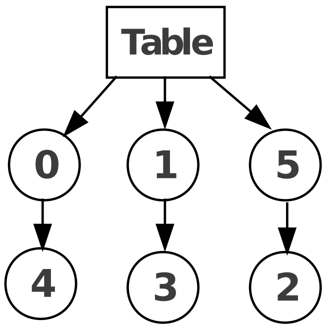


false(0)=0.082, hidden(0)=0.428
false(1)=0.044, hidden(1)=0.432
false(2)=0.150, hidden(2)=0.816
false(3)=0.110, hidden(3)=0.382
false(4)=0.496, hidden(4)=0.478

Fig. 4.21: Experimental result 5. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.

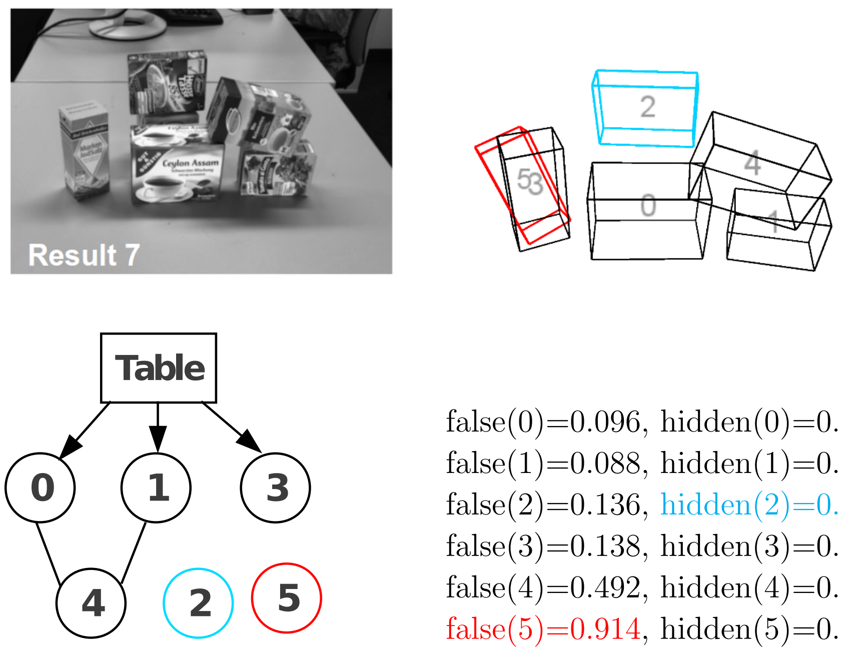






false(0)=0.070, hidden(0)=0.416
false(1)=0.036, hidden(1)=0.454
false(2)=0.158, hidden(2)=0.384
false(3)=0.110, hidden(3)=0.400
false(4)=0.152, hidden(4)=0.374
false(5)=0.074, hidden(5)=0.358

Fig. 4.22: Experimental result 6. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.



false(0)=0.096, hidden(0)=0.404
false(1)=0.088, hidden(1)=0.406
false(2)=0.136, hidden(2)=0.808
false(3)=0.138, hidden(3)=0.386
false(4)=0.492, hidden(4)=0.506
false(5)=0.914, hidden(5)=0.460

Fig. 4.23: Experimental result 7. The input stereo image (upper left), estimated 6D poses (upper right), the resulting scene graph (lower left) and the query probability (lower right) are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.

5 Conclusions and Future Directions

5.1 Concluding Remarks

Data processing handles data directly in the continuous domain, and it refers to the collection and manipulation of items of data to produce meaningful information. By contrast, knowledge processing is one of the core research fields of Artificial Intelligence, and it mainly deals with knowledge representation and reasoning in the symbolic domain. Although data processing and knowledge processing have many successful applications in their own domain, the combination of the both in a systematic and theoretically sound way is rarely seen. Such a combination is good at handling the complexity and uncertainty of the real world, and therefore it can help autonomous systems to tackle highly complex tasks.

In this dissertation, a knowledge-supervised MCMC (KSMCMC) sampling technique is developed, which provides autonomous systems the ability to abstract and to infer based on given knowledge and data. Within the framework of KSMCMC, knowledge processing and data processing can fully deploy their own strength and meanwhile work together as a consistent unity. We propose to realize the KSMCMC sampling technique by combining Markov logic and data driven MCMC sampling, because the former is a powerful tool for modelling uncertain knowledge, and the latter provides an efficient way of drawing samples from unknown complex distributions.

Based on Markov logic, task-specific context knowledge can be formulated as descriptive logic rules. These rules define the system behaviour on higher levels, regardless of the exact setting of the environment and the exact sensor readings. This abstractly defined behaviour is applicable to a wider range of situations and thus increases the overall robustness of the system. As a whole, KSMCMC is a new method of fitting abstract semantic models to input data by combining high-level knowledge processing with low-level data processing in a probabilistic and systematic way.

The effectiveness of the proposed KSMCMC sampling technique was demonstrated in two typical tasks in the robotic domain: semantic mapping and scene analysis. In chapter 3, we proposed a new system for semantic indoor mapping based on KSMCMC. This system takes preprocessed sensor data (in the form of occupancy grids) as input and generates a parametric, abstract, semantic and top-down representation of the perceived environments under the consideration: a classical indoor environment containing several units of different types connected by doorways. The generated parametric abstract models not only accurately represent the geometry of the perceived environments, they also provide valuable abstract information which could benefit higher level reasoning and communication purposes. Other than previous semantic mapping approaches that mainly focused on labelling the environments using semantic tags, the proposed system produces parametric models in a probabilistic generative manner. Task-specific context knowledge is defined as descriptive logic rules in Markov logic networks and is used to guide the sampling process to the desired environment representations, i.e. those comply with the defined knowledge, and at the same time, match the data well.

In chapter 4, the KSMCMC sampling technique was used to build a system for modelling table-top scenes. The proposed system employs a probabilistic approach to generate abstract scene graphs for table-top scenes using 6D object pose estimation as input. This system explicitly makes use of context knowledge that describes how such table-top scenes could be constructed. This knowledge is defined as descriptive logic rules in Markov logic and is used to calculate the probability of scene graphs. Combining the probability of scene graphs with a probabilistic sensor model, maximum posterior estimation of the scene parameters is performed using KSMCMC. The proposed system was evaluated using real world scenes. Experimental results confirmed that this system generates correct scene graphs which well represent the perceived table-top scenes. These scene graphs explain the composition of the observed scenes correctly and provide valuable semantic information on the inter-object relations which is very useful for robot manipulation tasks. By reasoning in the defined Markov logic network, false estimates of the object poses and hidden objects of the perceived scenes are correctly inferred.

5.2 Future Directions

Combining knowledge processing and data processing into a systematic and consistent unity is an interdisciplinary task. The work presented in this dissertation demonstrates a solid contribution in this direction. The ideas discussed in this dissertation also motivate several interesting future research directions:

- *Efficiency of knowledge processing*: the run time efficiency of knowledge processing depends mainly on the size of the represented knowledge base and the effectiveness of the reasoner (software implementation). In general, the more complex and powerful a knowledge base is, the less efficient the processing will be. This effect restricts the use of sophisticated knowledge bases. To fully deploy the power of knowledge processing, more work should be done to solve the efficiency issue.

- *Other knowledge representation formalisms*: in this dissertation, we used Markov logic for knowledge processing. There exist some other formalisms for knowledge representation, which could also be useful, such as description logic and semantic network. A future research direction would be to try out and evaluate these formalisms or their combinations.

- *Application to other tasks*: in this dissertation, the proposed KSMCMC sampling technique was used to handle two typical challenges in robotics, which are semantic mapping and scene analysis. However, its usefulness is not restricted to these two tasks only. This technique can be well used for tasks that are concerned with semantic and abstract interpretation of data sets. Thus, another line of research work would be to apply this approach to other tasks.

Regarding the two use cases of the proposed KSMCMC sampling technique, there exist also some future research directions, as listed in the following:

- *Semantic mapping*: at the current stage, semantic models are extracted from 2D map data of indoor environments, an extension to 3D scenarios and outdoor environments would be an interesting future direction. Another research direction would be the integration of this type of semantic knowledge into the perception procedures at the run time of the robot.

- *Scene analysis*: currently, objects with a regular shape that can be well represented by an oriented bounding box are used for scene analysis. This box shape is mainly used to simplify the discriminative evidence generation. A possible future direction could be the extension to objects with irregular shapes.

List of Figures

List of Tables

Bibliography

[1] Deutsche servicerobotik initiative, http://www.service-robotik-initiative.de/, 2009.

[2] SA Abdul Shukor, KW Young, and EJ Rushforth. 3d modeling of indoor surfaces with occlusion and clutter. In *IEEE International Conference on Mechatronics*, pages 282–287. IEEE, 2011.

[3] S.Y. An, L.K. Lee, and S.Y. Oh. Fast incremental 3d plane extraction from a collection of 2d line segments for 3d mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[4] Adrien Angeli, Stéphane Doncieux, J-A Meyer, and David Filliat. Incremental vision-based topological slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1031–1036. IEEE, 2008.

[5] Adrien Angeli, Stéphane Doncieux, J-A Meyer, and David Filliat. Visual topological slam and global localization. In *International Conference on Robotics and Automation*, pages 4300–4305. IEEE, 2009.

[6] Pedram Azad, Tamim Asfour, and Ruediger Dillmann. Stereo-based 6d object localization for grasping with humanoid robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 919–924. IEEE, 2007.

[7] Franz Baader. *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.

[8] S. Y. Bao, M. Sun, and S. Savarese. Toward coherent object detection and scene layout understanding. *Image and Vision Computing*, 2012.

[9] Moshe Bar. Visual objects in context. *Nature Reviews Neuroscience*, 5(8):617–629, 2004.

[10] J. Barwise. An introduction to first-order logic. *Studies in Logic and the Foundations of Mathematics*, 90:5–46, 1977.

[11] Michael Bender and Manfred Brill. *Computergrafik*, volume 2. Hanser, 2003.

[12] Irving Biederman et al. Perceiving real-world scenes. *Science*, 177(43):77–80, 1972.

[13] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2007.

[14] J-L Blanco, J-A Fernández-Madrigal, and Javier Gonzalez. A new approach for large-scale localization and mapping: Hybrid metric-topological slam. In *IEEE International Conference on Robotics and Automation*, pages 2061–2067. IEEE, 2007.

[15] N. Blodow, D. Jain, Z. Marton, and M. Beetz. Perception and probabilistic anchoring for dynamic world state logging. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

[16] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools, 2000*, 2000.

[17] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *KR*, pages 2–13, 1998.

[18] Kelvin Chan and Jay Liebowitz. The synergy of social network analysis and knowledge mapping: a case study. *International journal of management and decision making*, 7(1):19–35, 2006.

[19] Fu Chang, Chun jen Chen, and Chi jen Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93:206–220, 2004.

[20] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[21] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. 49(4):327–335, 1995.

[22] Alvaro Collet, Dmitry Berenson, Siddhartha S Srinivasa, and Dave Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pages 48–55. IEEE, 2009.

[23] F.R. Corrêa and J. Okamoto. Semantic mapping with image segmentation using conditional random fields. In *International Conference on Advanced Robotics*, pages 1–6. IEEE, 2009.

[24] MJ Cubison, H Coe, and M Gysel. A modified hygroscopic tandem dma and a data retrieval method based on optimal estimation. *Journal of aerosol science*, 36(7):846–865, 2005.

[25] Robert Dale and Ehud Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[26] Austin Eliazar and Ronald Parr. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *International Joint Conferences on Artificial Intelligence*, volume 3, pages 1135–1142, 2003.

[27] F Ferreira, I Amorim, R Rocha, and J Dias. T-slam: Registering topological and geometric maps for robot localization in large environments. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 392–398. IEEE, 2008.

[28] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 1981.

[29] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 100(1):67–92, 1973.

[30] Melvin Fitting. *First-order logic and automated theorem proving.* Springer, 1996.

[31] Carl French. *Data Processing and Information Technology.* Cengage Learning Business Press, 1996.

[32] E. Fukuda, M. Kimura, K. Miura, H. Fuji, and M. Tazawa. A new knowledge-based expert system for inspection of ulsi process flow. In *IEEE/SEMI International Semiconductor Manufacturing Science Symposium*, pages 85 –88, may 1991.

[33] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *IEEE 12th International Conference on Computer Vision*, pages 670–677. IEEE, 2009.

[34] Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722, 2010.

[35] Willow Garage. Pr2, 2011.

[36] A. Geiger, M. Lauer, and R. Urtasun. A generative model for 3d urban scene understanding from movable platforms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1945–1952. IEEE, 2011.

[37] M.R. Genesereth and N.J. Nilsson. *Logical foundations of artificial intelligence*, volume 9. Morgan Kaufmann Los Altos, CA, 1987.

[38] N. Goerke and S. Braun. Building semantic annotated maps by mobile robots. In *Proceedings of the Conference Towards Autonomous Robotic Systems*, 2009.

[39] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[40] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[41] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[42] Benjamin N Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web*, pages 48–57. ACM, 2003.

[43] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[44] T. Grundmann, R. Eidenberger, M. Schneider, M. Fiegert, and G. v Wichert. Robust high precision 6d pose determination in complex environments for robotic manipulation. In *Proc. Workshop Best Practice in 3D Perception and Modeling for Mobile Manipulation at ICRA*, 2010.

[45] T. Grundmann, W. Feiten, and G. v. Wichert. A gaussian measurement model for local interest point based 6 dof pose estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2085–2090, 2011.

[46] T. Grundmann, M. Fiegert, and W. Burgard. Probabilistic rule set joint state update as approximation to the full joint state estimation applied to multi object scene analysis. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[47] Thilo Grundmann. Scene analysis for service robots. *Ph.D. Dissertation*, 2012.

[48] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? In *Handbook on ontologies*, pages 1–17. Springer, 2009.

[49] Volker Haarslev and Ralf Möller. Racer: A core inference engine for the semantic web. In *EON*, volume 87, 2003.

[50] Dirk Hahnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 206–211. IEEE, 2003.

[51] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[52] Hedi Harzallah, Frédéric Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *IEEE 12th International Conference on Computer Vision*, pages 237–244. IEEE, 2009.

[53] Xuming He, Richard S Zemel, and Miguel A Carreira-Perpinán. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–695. IEEE, 2004.

[54] Martin Hepp, Dumitru Roman, et al. An ontology framework for semantic business process management. In *Wirtschaftsinformatik (1)*, pages 423–440, 2007.

[55] Edward H Herskovits and Gregory F Cooper. Kutato: An entropy-driven system for construction of probabilistic expert systems from databases. *arXiv preprint arXiv:1304.1088*, 2013.

[56] Thomas Hofweber. Logic and ontology. 2008.

[57] P. Hough. Method and means for recognizing complex patterns. In *U.S. Patent 3069654*, 1962.

[58] A. Howard and N. Roy. The robotics data set repository (radish), http://radish.sourceforge.net/, 2003.

[59] http://www.gazebosim.org/wiki. Gazebo, 2013.

[60] http://www.ros.org/wiki/. Robot operating system, 2013.

[61] http://www.wikihow.com/Main Page. wikihow, 2013.

[62] Dong Huang, Yi Yang, and J. Calmet. A knowledge-based security policy framework for business process management. In *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, pages 154–160, Dec. 2006.

[63] D Hummel and G Redeker. A new vortex flow experiment for computer code validation. Technical report, DTIC Document, 2003.

[64] Tuyen N Huynh and Raymond J Mooney. Discriminative structure and parameter learning for markov logic networks. In *Proceedings of the 25th international conference on Machine learning*, pages 416–423. ACM, 2008.

[65] Tuyen N Huynh and Raymond J Mooney. Max-margin weight learning for markov logic networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 564–579. Springer, 2009.

[66] D. Jain. Knowledge engineering with markov logic networks: A review. *Evolving Knowledge in Theory and Applications*, page 16.

[67] D. Jain. Probcog toolbox, http://ias.cs.tum.edu/software/probcog, 2011.

[68] Dominik Jain, Stefan Waldherr, and Michael Beetz. Bayesian logic networks. *IAS Group, Fakultät für Informatik, Technische Universität München, Tech. Rep*, 2009.

[69] Stanisław Jaśkowski. Propositional calculus for contradictory deductive systems. *Studia Logica*, 24(1):143–157, 1969.

[70] I. Jebari, S. Bazeille, E. Battesti, H. Tekaya, M. Klein, A. Tapus, D. Filliat, C. Meyer, R. Benosman, E. Cizeron, et al. Multi-sensor semantic mapping and exploration of indoor environments. In *IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 151–156. IEEE, 2011.

[71] G. Jeong and H.S. Yang. Context-aware activity recognition by markov logic networks of trained weights. In *16th International Conference on Virtual Systems and Multimedia (VSMM)*, pages 5–12. IEEE, 2010.

[72] Hongwen Kang, Martial Hebert, and Takeo Kanade. Discovering object instances from scenes of daily living. In *IEEE International Conference on Computer Vision*, pages 762–769. IEEE, 2011.

[73] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *IEEE International Conference on Robotics and Automation*, pages 2088–2095. IEEE, 2013.

[74] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Computational and Graphical Statistics*, 5(1):1–25, 1996.

[75] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

[76] Kazuhiro Kosuge and Yasuhisa Hirata. Human-robot interaction. In *IEEE International Conference on Robotics and Biomimetics*, pages 8–11. IEEE, 2004.

[77] A.K. Krishnan and K.M. Krishna. A visual exploration algorithm using semantic cues that constructs image based hybrid maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1316–1321. IEEE, 2010.

[78] Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

[79] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2129–2142, 2009.

[80] George G Lendaris. Conceptual graph knowledge systems as problem context for neural networks. In *IEEE International Conference on Neural Networks*, pages 133–140. IEEE, 1988.

[81] V. Leung and S. Herbin. Flexible tracklet association for complex scenarios using a markov logic network. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1870–1875. IEEE, 2011.

[82] Zhao Li. Spelling correction system based on ocr. *Ordnance Industry Automation*, 9:029, 2010.

[83] G. H. Lim, I. H. Suh, and H. Suh. Ontology-based unified robot knowledge for service robots in indoor environments. *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Human*, 41(3):492–509, 2011.

[84] B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1471, 2005.

[85] Pamela Lipson, Eric Grimson, and Pawan Sinha. Configuration based scene classification and image indexing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1007–1013. IEEE, 1997.

[86] Z. Liu and G. von Wichert. Extracting semantic indoor maps from occupancy grids. *Robotics and Autonomous Systems*, 2013.

[87] D. Lowd and P. Domingos. Efficient weight learning for markov logic networks. *Knowledge Discovery in Databases*, pages 200–211, 2007.

[88] D. G. Lowe. Object recognition from local scale-invariant features. In *The proceedings of the seventh IEEE international conference on Computer vision*, volume 2, pages 1150–1157, 1999.

[89] J. Mason and B. Marthi. An object-based semantic world model for long-term change detection and semantic querying. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[90] Nikolaos Mavridis and Deb Roy. Grounded situation models for robots: Where words and percepts meet. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4690–4697. IEEE, 2006.

[91] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(2004-03):10, 2004.

[92] Stephen J Mellor and Marc J Balcer. *Executable UML: a foundation for model-driven architecture*. Addison-Wesley Professional, 2002.

[93] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[94] David Milne, Ian H. Witten, and David M. Nichols. A knowledge-based search engine powered by wikipedia. In *Proceedings of the sixteenth ACM conference on Information and Knowledge management*, pages 445–454, November 2007.

[95] Domain-Driven Data Mining and A Framework. Domain-driven, actionable knowledge discovery. 2007.

[96] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Innovative Applications of Artificial Intelligence Conference*, pages 593–598, 2002.

[97] Kevin Murphy, Antonio Torralba, and William Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. *Advances in neural information processing systems*, 16, 2003.

[98] Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993.

[99] Allen Newell and Herbert A Simon. *GPS, a program that simulates human thought.* Defense Technical Information Center, 1961.

[100] Allen Newell and Fred M Tonge. An introduction to information processing language v. *Communications of the ACM*, 3(4):205–211, 1960.

[101] D. Nienhuser, T. Gumpp, and JM Zollner. Relevance estimation of traffic elements using markov logic networks. In *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1659–1664. IEEE, 2011.

[102] C. Nieto-Granda, J.G. Rogers, A.J.B. Trevor, and H.I. Christensen. Semantic map partitioning in indoor environments using regional analysis. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1451–1456. IEEE, 2010.

[103] A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[104] tephen E Palmer. The effects of contextual scenes on the identification of objects. *Memory & Cognition*, 3(5):519–526, 1975.

[105] D. Pangercic, B. Pitzer, M. Tenorth, and M. Beetz. Semantic object maps for robotic housework-representation, acquisition and use. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012.

[106] D. Pangercic, M. Tenorth, D. Jain, and M. Beetz. Combining perception and knowledge processing for everyday manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[107] Devi Parikh, C Lawrence Zitnick, and Tsuhan Chen. From appearance to context-based recognition: Dense labeling in small images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[108] J.T. Park and J.B. Song. Hybrid semantic mapping using door information. In *8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 128–130. IEEE, 2011.

[109] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[110] M. Persson, T. Duckett, C. Valgren, and A. Lilienthal. Probabilistic semantic mapping with a virtual sensor for building/nature detection. In *International Symposium on Computational Intelligence in Robotics and Automation*, pages 236–242. IEEE, 2007.

[111] M. Pfingsthorn, A. Birk, and N. Vaskevicius. Semantic annotation of ground and vegetation types in 3d maps for autonomous underwater vehicle operation. In *OCEANS 2011*, pages 1–8. IEEE, 2011.

[112] T. Poklemba, I. Sivy, and Z. Havlice. Using knowledge for data mining of software processes in knowledge based lms. In *9th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 171 –174, oct. 2011.

[113] A. Pronobis and P. Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *IEEE International Conference on Robotics and Automation*, pages 3515–3522. IEEE, 2012.

[114] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[115] A. Ranganathan and F. Dellaert. Semantic modeling of places using objects. In *Robotics: Science and Systems*, 2007.

[116] Lord Rayleigh. X. on pin-hole photography. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 31(189):87–99, 1891.

[117] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

[118] Bryan Russell, Antonio Torralba, Ce Liu, Rob Fergus, and William T Freeman. Object recognition by scene alignment. In *Advances in Neural Information Processing Systems*, pages 1241–1248, 2007.

[119] R.B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE, 2011.

[120] R.B. Rusu, Z.C. Marton, N. Blodow, A. Holzbach, and M. Beetz. Model-based and learned semantic object labeling in 3d point cloud maps of kitchen environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3601–3608. IEEE, 2009.

[121] V. Sakenas, O. Kosuchinas, M. Pfingsthorn, and A. Birk. Extraction of semantic floor plans from 3d point cloud maps. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 1–6. IEEE, 2007.

[122] I. Shim, Y. Choe, and M.J. Chung. 3d mapping in urban environment using geometric featured voxel. In *International Conference on Ubiquitous Robots and Ambient Intelligence*, pages 804–805. IEEE, 2011.

[123] Graeme Simsion and Graham Witt. *Data modeling essentials*. Morgan Kaufmann, 2004.

[124] Evren Sirin and Bijan Parsia. Pellet system description. In *Proc. of the Int. Workshop on Description Logics, DL*, volume 6, 2006.

[125] K. Sjoo. Semantic map segmentation using function-based energy maximization. In *IEEE International Conference on Robotics and Automation*, pages 4066–4073. IEEE, 2012.

[126] Raymond M Smullyan. *First-order logic*, volume 6. Springer, 1968.

[127] Raymond M Smullyan. *First-order logic*. Courier Dover Publications, 1995.

[128] John F Sowa. Semantic networks. *Encyclopedia of Cognitive Science*, 2006.

[129] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 33–40. ACM, 2006.

[130] Thomas M Strat and Martin A Fischler. Context-based vision: recognizing objects using information from both 2 d and 3 d imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1050–1065, 1991.

[131] G. Sukhatme, M. Batalin, V. Chen, and W. Kaiser. Towards spatial and semantic mapping in aquatic environments. International Conference of Robotics and Automation, 2008.

[132] Yuyin Sun, Liefeng Bo, and Dieter Fox. Attribute based object identification. In *IEEE International Conference on Robotics and Automation*, 2013.

[133] A. Swadzba, S. Wachsmuth, C. Vorwerg, and G. Rickheit. A computational model for the alignment of hierarchical scene representations in human-robot interaction. In *IJCAI*, pages 1857–1863, 2009.

[134] Tong Tao, Stephen Tully, George Kantor, and Howie Choset. Incremental construction of the saturated-gvg for multi-hypothesis topological slam. In *IEEE International Conference on Robotics and Automation*, pages 3072–3077. IEEE, 2011.

[135] M. Tenorth, L. Kunze, D. Jain, and M. Beetz. Knowrob-map-knowledge-linked semantic object maps. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2010.

[136] Moritz M. Tenorth. Knowledge processing for autonomous robots. *Ph.D. Dissertation*, 2011.

[137] E.A. Topp and H.I. Christensen. Topological modelling for human augmented mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2257–2263. IEEE, 2006.

[138] Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):169–191, 2003.

[139] Antonio Torralba, Kevin P Murphy, and William T Freeman. Contextual models for object detection using boosted random fields. In *Advances in neural information processing systems*, pages 1401–1408, 2004.

[140] Antonio Torralba, Kevin P Murphy, William T Freeman, and Mark A Rubin. Context-based vision system for place and object recognition. In *IEEE International Conference on Computer Vision*, pages 273–280. IEEE, 2003.

[141] Leon Edgar Truesdell. *The development of punch card tabulation in the Bureau of the Census, 1890-1940: with outlines of actual tabulation programs*. USGPO, 1965.

[142] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *Automated reasoning*, pages 292–297. Springer, 2006.

[143] Z. Tu, X. Chen, A.L. Yuille, and S.C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.

[144] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):657–673, 2002.

[145] Paolo Valore. *Topics on General and Formal Ontology*. Polimetrica, 2006.

[146] Maarten H Van Emden and Robert A Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM)*, 23(4):733–742, 1976.

[147] Jakob Verbeek and William Triggs. Scene segmentation with crfs learned from partially labeled images. 2007.

[148] N. Wagle and N. Correll. Multiple object 3d-mapping using a physics simulator. Technical report, University of Colorado at Boulder, 2010.

[149] Bing Wang, Jiting Yang, and Hongpei Liu. Understanding the mechanism of social network in the knowledge transfer process. In *Technology Management for Global Economic Growth (PICMET)*, pages 1 –6, July 2010.

[150] J. Wang and P. Domingos. Hybrid markov logic networks. In *Proceedings of the 23rd national conference on Artificial intelligence*, volume 2, pages 1106–1111, 2008.

[151] Wei Wang, Lili Chen, Dongming Chen, Shile Li, and Kolja Kühnlenz. Fast object recognition and 6d pose estimation using viewpoint oriented color-shape histogram. In *IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, to appear.

[152] Markus Weber, Max Welling, and Pietro Perona. Towards automatic discovery of object categories. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 101–108. IEEE, 2000.

[153] Felix Werner, Frederic Maire, Joaquin Sitte, Howie Choset, Stephen Tully, and George Kantor. Topological slam using neighbourhood information of places. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4937–4942. IEEE, 2009.

[154] Jan Wielemaker, S Ss, and I Ii. Swi-prolog 2.7-reference manual. 1996.

[155] Patrick H Winston. Learning structural descriptions from examples. Technical report, DTIC Document, 1970.

[156] D.F. Wolf and G.S. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, 2008.

[157] Lior Wolf and Stanley Bileschi. A critical view of context. *International Journal of Computer Vision*, 69(2):251–261, 2006.

[158] Zhixing Xue, Alexander Kasper, J Marius Zoellner, and Ruediger Dillmann. An automatic grasp planning system for service robots. In *International Conference on Advanced Robotics*, pages 1–6. IEEE, 2009.

[159] Hongfeng Yin. Method and system of knowledge based search engine using text mining, August 14 2007. US Patent 7,257,530.

[160] Xiaodong Yu, Cornelia Fermuller, Ching Lik Teo, Yezhou Yang, and Yiannis Aloimonos. Active scene recognition with vision and language. In *IEEE International Conference on Computer Vision (ICCV)*, pages 810–817. IEEE, 2011.

[161] S.C. Zhu, R. Zhang, and Z. Tu. Integrating top-down/bottom-up for object recognition by data driven markov chain monte carlo. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2000.

[162] X. Zhu, H. Zhao, Y. Liu, Y. Zhao, and H. Zha. Segmentation and classification of range image from an intelligent vehicle in urban environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1457–1462. IEEE, 2010.

List Of Own Publications And Collaborations

Journal Articles:

[1] Z. Liu and G. von Wichert. Extracting semantic indoor maps from occupancy grids. *Robotics and Autonomous Systems*, 2013.

[2] Z. Liu and G. v. Wichert. A generalizable knowledge framework for semantic indoor mapping based on markov logic networks and data driven mcmc. *Future Generation Computer Systems*, 2013.

Conference Proceedings:

[3] Z. Liu, D. Chen, K. M. Wurm, and G. von Wichert. Using rule-based context knowledge to model table-top scenes. In *IEEE International Conference on Robotics and Automation*. IEEE, 2014.

[4] Z. Liu, W. Wang, D. Chen, and G v. Wichert. A coherent semantic mapping system based on parametric environment abstraction and 3d object localization. In *European Conference on Mobile Robots (ECMR)*, pages 234–239, 2013.

[5] D. Chen, Z. Liu, and G v. Wichert. Grasping on the move: A generic arm-base coordinated grasping pipeline for mobile manipulation. In *European Conference on Mobile Robots (ECMR)*, pages 234–239, 2013.

[6] Z. Liu and G. von Wichert. Applying rule-based context knowledge to build abstract semantic maps of indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5141 –5147, Nov. 2013.

[7] Z. Liu, D. Chen, and G. von Wichert. Online semantic exploration of indoor maps. In *IEEE International Conference on Robotics and Automation*, pages 4361–4366. IEEE, 2012.

[8] Ziyuan Liu, Dong Chen, and Georg von Wichert. 2d semantic mapping on occupancy grids. In *Proceedings of ROBOTIK 2012 - 7th German Conference on Robotics*, pages 1–6. VDE, 2012.

[9] Z. Liu, D. Lee, and Sepp W. Particle filter based monocular human tracking with a 3d cardbox model and a novel deterministic resampling strategy. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3626 – 3631, Sep. 2011.

[10] Christian Seitz, Christoph Legat, and Ziyuan Liu. Flexible manufacturing control with autonomous product memories. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2010.