Elsevier Editorial System(tm) for Robotics and Computer Integrated Manufacturing
Manuscript Draft

Manuscript Number: RCIM-D-14-00090R1

Title: Table-Top Scene Analysis Using Knowledge-Supervised MCMC

Article Type: SI:Knowledge Driven Robotics

Keywords: Knowledge Representation and Reasoning; Robotics; Scene Analysis; Abstract Models; Semantic Modelling

Corresponding Author: Mr. ziyuan liu,

Corresponding Author's Institution:

First Author: ziyuan liu

Order of Authors: ziyuan liu; Dong Chen; Kai M Wurm; Georg von Wichert

Abstract: In this paper, 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 explicitly make use of task-specific context knowledge by encoding this knowledge as descriptive rules in Markov logic networks. We use a probabilistic sensor model to encode the fact that measurements are subject to significant uncertainty. We integrate the measurements with the abstract 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 defined Markov logic knowledge base. The effectiveness of our approach is
demonstrated and evaluated in real world experiments.

Cover letter for submission of manuscript

Dear Editor,

I am enclosing herewith a manuscript entitled “Table-Top Scene Analysis Using Knowledge-Supervised MCMC” for publication in Journal Robotics and Computer-Integrated Manufacturing.

The corresponding author of this manuscript is Ziyuan Liu, with the following contact information:
Email: ziyuan.liu@tum.de
Phone: +49 176 3499 4703.

Sincerely,

Ziyuan Liu

Institute of Automatic Control Engineering
Technische Universität München
Theresienstr. 90, 80333, Munich, Germany

*Highlights (for review)

A probabilistic approach for generating abstract scene graphs from uncertain 6D pose estimates is proposed for understanding table-top scenes.
The proposed system is realized by the knowledge-supervised MCMC sampling technique.
Task-specific context knowledge is defined as descriptive rules in Markov logic networks.
The proposed system is fully probabilistic and links the high-level abstract scene description to uncertain low level measurements.
False estimates and hidden objects of the perceived scenes are systematically inferred using the defined knowledge base.

***Detailed Response to Reviewer#1**

We heartfully thank the anonymous reviewers for their valuable comments that were greatly helpful for improving the quality of this paper!

According to the request of a reviewer, this paper is slightly restructured, and it has now 7 sections (instead of 8). Section 3 “Contributions” of the initial submission is arranged as a subsection of section 2 “Related work” in the current version.

Answers to reviewers' questions are given below in color blue.

Reviewer #1:

The abstract and highlights are all appropriate for the paper, but the term MCMC in the title is never defined and should be explained, especially the role of Monte-Carlo methods, which don't appear to be used.

An introduction about MCMC methods is added in the first paragraph of section 3.

In the description of Markov networks in Section 5, the nodes in the graph are defined but the arcs are not and there is no explanation of what the cliques represent. This makes it hard to understand their role in the algorithm.

In the theory of Markov networks, the arcs (edges) in a Markov network indicate dependency between nodes. If there is a dependency between two nodes, then these two nodes are connected by an arc. Through the connection by arcs, cliques of nodes are generated, and they are used to calculate (with the help of potential functions) the probability distribution that the underlying Markov network represents. In the theory of Markov logic networks (MLN), each ground MLN instantiates a Markov network, with ground atoms being the nodes. The dependencies (arcs) between nodes are decided by the logic formula (rules) that are defined in the MLN. In this way, the nodes (ground atoms), whose predicate symbols are used in the same logic formula, are joined together by a clique. To keep the paper in a compact form, some details of the underlying theories are not given in this paper. Instead, references are provided which give more insights into the theories.

The set of rules used to interpret the scenes looks reasonable but it would have been helpful if the authors had explained how and why the particular set was chosen and the weights were selected. Also, it would be interesting to know how sensitive the method is to the rules and weights.

The choice of rules is indeed a problem-oriented engineering procedure. It mainly depends on what kind of features one wants to describe about the scene and what inferences one wants to make. In our case, we want to describe the scene in terms of inter-object relations such as support and contact. The inferences that we want to make are: whether a detected object is a false estimate and whether there is a hidden object. The definition of rules begins by writing down the rules in the form of natural language. This is an 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.

We use the log-odd form log(p1/p2) to formulate weights, because this is an intuitive way of expressing how sure we are about the rules. With p1, p2 \in (0,1) and p1+p2=1, log(p1/p2) means that a rule holds with the probability of p1. Then, in this way, log(0.5/0.5) simply indicates a rule holds with the probability of 50%, which actually means we do not know anything about the uncertainty of this rule. In our work, we use two belief levels: log(0.9/0.1) and log(0.7/0.3). For the rules that we are very sure about, log(0.9/0.1) is used as weight. For the rules that we are only relatively sure about, log(0.7/0.3) is used. The key idea here is to use several reasonable belief levels to group the uncertainty of the defined rules into different categories. For our application we consider two levels are enough, and it is possible to use more belief levels. Inferences in Markov logic networks are performed based on sampling methods, and we need to avoid using weights like log(0.5/0.5) or log(0.56/0.44), because such weights are non-informative and almost mean that we do not know anything about the rule, in which case the corresponding rule should not have been defined in the first place.

As long as the above aspects are taken into account, the method is not so sensitive to the weights, because the possible world that complies with the defined knowledge base will always have the highest probability among all other possible worlds.

In Section 6.1.3, please describe the predicates in the same order as in Table 3.

Accordingly corrected.

In Figure 6, the points in images b and c are not visible. Use larger points and greater contrast.

Accordingly corrected.

In Section 6.6, the authors should explain what the benefit is of adding Gaussian noise to the position estimates from the sensor data. Why not simply use the values provided by the sensors?

The pose estimate that is used as input in our system is obtained from a SIFT-based keypoint matching method. Because of the limited accuracy of sensor data (triangulated stereo images) and imperfect object-database, the input pose estimate is therefore imperfect. By adding Gaussian noise to position estimates, we try to optimize the pose estimate. Section 5.6 is accordingly extended.

Section 7 on evaluation needs to be expanded. It is not very informative to present only cases where the system works flawlessly. The authors should present cases where the system fails and explain under what conditions this can be expected to occur.

The discriminative generation of evidences which is described in section 5.1.5 is important for our system to deliver correct inferences. If something goes wrong with evidence generation, e.g., a “supportive” and “supported” relation is missed, the inference results would be less accurate. Since evidence generation is done based on discriminative methods, such errors could happen (but very rarely), when it comes to some near-to-threshold cases, e.g. when we define that if the closest distance between two objects is less than 0.5 cm, then we consider that they have a contact, then for cases, in which the closest distance between two objects is 0.6 cm or 0.7 cm, no contact will be detected. A discussion on this point is added in section 5.1.5.

The authors do not present any conclusions or suggestions for improvements of their work. Section 8 should be expanded or, at the very least, renamed "Summary."

This section is now extended with future directions.

We heartfully thank the anonymous reviewers for their valuable comments that were greatly helpful for improving the quality of this paper!

According to the request of a reviewer, this paper is slightly restructured, and it has now 7 sections (instead of 8). The section 3 “Contributions” of the initial submission is arranged as a subsection of section 2 “Related work” in the current version.

Answers to reviewers' questions are given below in color blue.

Reviewer #2: Originality

- The authors claim that their probabilistic approach to generate graph scenes is different than other existing approaches presented in section 2.
- The use of Markov logic networks seems to provide very effective and accurate results, although the experiments always use the same number of objects arranged in a similar manner.

In our system, a SIFT-based keypoints matching method is used to estimate the 6D poses of the perceived objects. The estimated poses are then used as input to generate abstract scene graphs for the perceived scenes. Pose estimation is done based on extensive matching of SIFT keypoints between the sensor data and the database of learned objects. Since the database of learned objects is very big in size, the matching procedure is very computationally demanding. On our current computer (Intel i7 with 6GB RAM) the maximum number of objects which can be localized at the same time is 6. This is the reason why we used this number of objects in the experiments. In summary, the limitation does not lie in our KSMCMC framework, instead, the limitation is the computational complexity of pose estimation that serves as input to our system.

- It would be interesting if the authors could make a comparison between their work and some other related work, even if the techniques used are different. This could help analyze how this work contributes in this context.

Scene analysis is a very general term which contains a lot of different aspects, such as object localisation, object discovery and so on. In our work we focus on generating abstract scene graphs for the perceived scenes by taking the estimated object 6D poses as input. The approaches described in Section 2 “related work” already cover most of the existing systems which are related with scene analysis, although most of them use different techqiues to address different aspects of scene analysis. On the other hand, we need to focus on this area, i.e. scene analysis, so that we do not get off the topic.

- There are some previous works (references 45 and 51) that are mentioned in the text as having a similar approach as the one used in this paper. Can we consider the use of Markov logic networks as the "new element" in the proposed approach? Please, provide more details on how these papers relate each other?

In general, references 45 and 51 are not competing approaches to our approach, because they focus on estimating object 6D poses. Our approach uses these estimated poses as input to do something else, which is to provide abstract scene graphs for the perceived scenes. In our system, we use a method that is similar to reference 45 to estimate object 6D poses. Reference 51 has provided us an example on how we can calculate the data likelihood. The reason why references 45 and 51 are cited is that we consider these two approaches a great inspiration for us.

Scientific quality

- Section 2 covers several representative works already developed, which contributes to a general review of existing approaches.
- It is not clear why object #6 will be detected? Figures 1 and 2 do not clearly shown such object. Once detected, it is not clear why it will be considered as a false estimate? This question relates to rule r13. The same occurs with object #3 in Figure 3.

The pose estimate that is used as input in our system is obtained from a SIFT-based keypoint matching method. Because of the limited accuracy of sensor data (triangulated stereo images) and imperfect object-database, the input pose estimate is therefore also imperfect. Object #6 is detected, because the yellow salt box is recognized as two objects, once as object #1 and once as object #6. This is also one of the reasons why we would like to use a knowledge-supervised approach to distinguish between the correct and the wrong estimate in a top-down manner. Of course, such wrong estimates happen rarely, but the point here is, how we could find it out in a systematic way, once it happens. Once object #6 is detected, an intersection between object #6 and object #1 occurs, and this makes both objects a candidate for false estimate according to rule r14.

Rule r13 states that a hovering object is either a false estimate or has a hidden support under it. This makes the probability of object #3 being a false estimate or having a hidden support very high. Since object #3 also has a stable pose, which means that it is unlikely a false estimate (rule r16), objects #3 is finally considered to have a hidden support.

- In Section 3, it is mentioned that measurements are subject to significant uncertainty. I would like to see some discussion regarding this point. What is the level or percentage of such uncertainty? How it influences in the results, as the results presented seem to be very accurate.

From our point of view, the significant uncertainty related with measurements comes from two major sources, which are the limited accuracy of sensor data (triangulated stereo images) and the imperfect object-database. In our application, we use 3D point clouds that are generated from triangulated stereo images to estimate 6D object poses. Although it is in general difficult to quantify the level of uncertainty, it is obvious to see that the accuracy of the generated 3D point clouds is not as high as that we want to have, especially when it comes to small objects like those we used in the experiments. An example is that the surface points of a box are not quite aligned, instead, they float below or above the corresponding surface. The other major source of uncertainty is the imperfect object-database, which is generated by scanning the objects using a 3D scanner. Given these uncertainties, the pose estimate is also imperfect. Thus, we use a knowledge-supervised approach to examine the pose estimate in a top-down manner. In addition, the pose estimate is locally optimized by the underlying MCMC process.

- In subsection 6.1, it is mentioned that some constraints must be followed in order to arrange the objects. The experimental scenarios use objects that intersect or are supported by other objects. How real are these scenarios? If we have a table with only hover objects, what will be the resulting graph? All objects will be considered hidden or false estimates? I would like to see an evaluation of more complex scenarios, with a variable number of objects, and also more complex relations (for example, an object inside another object, a pile of boxes or papers, etc).

There is a misunderstanding here. By saying “Objects on a table-top are not arranged arbitrarily but they follow certain physical constraints ”, we mean that there are certain physical rules like gravity and forces, which apply to the objects on a table-top and “arrange” them in a certain way. An example of these rules is defined in the Markov logic network to model table-top scenes. To illustrate the general idea, a simplification is made by using objects with a box shape. This simplification eases the generation of evidences which serve as input in the knowledge reasoning process. The scenarios used in our experiments are all real world scenarios.

If we have a table with hover objects only (hypothetically), the probability for them being a false estimate or having hidden support would be high according to rule r13. Since in this hypothetical case, we do not have enough information about “support” and “intersect”, the reasoning reuslts would be as follows: for those objects, which have a stable pose, the probability of being a false estimate is lowered according to rule r16, and they would be primarily considered to have hidden support; for those objects, which have an unstable pose, the probability for being a false estimate and the probability for having hidden support are both high, and more information is needed so as to make a nonambiguous reference. Regarding more complex scenarios, as we mentioned in one of the previous answers, the database of learned objects is very big in size, thus the matching procedure is very computationally demanding. On our current computer, the maximum number of objects which can be localized at the same time is 6. This is the reason why we used this number of objects in the experiments, although our KSMCMC framework is capable of processing scenes with more objects.

- Figure 2 does not clearly shown the contact between objects #2 and #3, as stated in page 6.

There is indeed no contact between #2 and #3. This was a mistake, and the contact was meant for #2 and #4. This is corrected now.

- If we change the values used in the belief levels (log-odd form), what will be the impact in the system's performance and accuracy? The authors mention learning-based and manually designed techniques, but do not provide further details on such approaches. On the other hand, the title mention knowledge-supervised. I would like to see some discussion here, in order to provide the reader with some basic knowledge on what approach is more suitable for this kind of application.

We use the log-odd form log(p1/p2) to formulate weights, because this is an intuitive way of expressing how sure we are about the rules. With p1, p2 \in (0,1) and p1+p2=1, log(p1/p2) means that a rule holds with the probability of p1. Then, in this way, log(0.5/0.5) simply indicates that a rule holds with the probability of 50%, which actually means we do not know anything about the uncertainty of this rule. In our work, we use two belief levels: log(0.9/0.1) and log(0.7/0.3). For the rules that we are very sure about, log(0.9/0.1) is used as weight. For the rules that we are only relatively sure about, log(0.7/0.3) is used. The key idea here is to use several reasonable belief levels to group the uncertainty of the defined rules into different categories. For our application we consider two levels are enough, and it is possible to use more belief levels. Inferences in Markov logic networks are performed based on sampling methods, and we need to avoid using weights like log(0.5/0.5) or log(0.56/0.44), because such weights are non-informative and almost mean that we do not know anything about the rule, in which case the corresponding rule should not have been defined in the first place. As long as these aspects are taken into account, the method is not so sensitive to the weights, because the possible world that complies with the defined knowledge base will always have the highest probability among all other possible worlds.

To keep this paper in a compact form (which already has 17 pages), we did not provide details on the learning-based and manually-designed techqiues. If readers are interested in this topic, details can be found in the corresponding references which are cited in our paper.

An extended discussion about our knowledge-supervised MCMC method is added in the second paragraph of section 3.

- How the maximum value is set during the hypotheses generation? How we can choose a "good" value?

The maximum number of hypotheses depends on how many outliers are generated and on how many object instances can be expected in a scene. A higher value increases the robustness of the algorithm but also comes at the cost of an increased runtime. In our experiments, we set this value to 50. This is an emperical value that led to good object detection at a reasonable runtime.

- Regarding performance, the authors mention an average time of 2, 18 seconds for each experiment. This is the mean time for 10 or 15 interactions? Although the scenarios are very similar, the average time is the same in all experiments? It is possible to parallelize your system: generating a graph scene while determining the (partial) objects relationships? This could be useful or even possible?

The average time of 2.18 seconds is meant for each iteration. For 10 iterations, the total time would be 21.8 seconds. This average time is the average of all the experiments shown in this paper. Generating a scene graph and determining relationships can not be parallelized.

Relevance to the field of this journal

- I think this paper could be quite revelant and interesting to this journal, since the suggestions made here are met.

Presentation

- The sentence "(In addition to) the approches introduced above" appears many times in sections 2 and 3.

Accordingly corrected.

- Section 3 consists of a single paragraph with the same text used in the Abstract. I suggest removing this section or to incorporate it in Section 4.

We understand the reviewer's concern. This paper is quite long, a separate section of contributions can help readers easily catch the key ideas of the paper. This part does not fit into the theory of KSMCMC, so we suggest incorporating it into related work.

- In subsection 5.2, put a period in the sentence "... concept of Markov logic networks. For more details on..."

Accordingly corrected.

- There are some incomplete references: 12, 26, 34, 35, 46, and 47.

Accordingly corrected.

- The final version will continue to be colorful, as colors are used in many figures and mentioned in their related paragraphs?

Although the figures are drawn in colors, they are also readable in black/white print because of the contrast of the used colors. In addition, the objects are also numbered, and this makes the figures easy to understand.

Table-Top Scene Analysis Using Knowledge-Supervised MCMC

Ziyuan Liu[1], Dong Chen

Siemens AG, Corporate Technology, Munich, Germany
Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany

Kai M. Wurm

Siemens AG, Corporate Technology, Munich, Germany

Georg von Wichert

Siemens AG, Corporate Technology, Munich, Germany
Institute for Advanced Study, Techniche Universität München, Lichtenbergstrasse 2a, D-85748 Garching, Germany

Abstract

In this paper, 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 explicitly make use of task-specific context knowledge by encoding this knowledge as descriptive rules in Markov logic networks. We use a probabilistic sensor model to encode the fact that measurements are subject to significant uncertainty. We integrate the measurements with the abstract 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 defined Markov logic knowledge base. The effectiveness of our approach is demonstrated and evaluated in real world experiments.

Keywords: Knowledge Representation and Reasoning, Robotics, Scene Analysis, Abstract Models, Semantic Modelling

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. 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 paper, 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. Object poses can be estimated either from stereo images or from RGBD point clouds. A typical result of the procedure is shown in Fig. 1.

Email addresses: ziyuan.liu@tum.de (Ziyuan Liu), chendong@mytum.de (Dong Chen), georg.wichert@siemens.com (Georg von Wichert)

[1] Corresponding author. Postal address: Karlstr. 45, room 5001, 80333, Munich, Germany. Telephone: +49-89-289-26900. Fax: +49-89-289-26913.

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”, or “objects do not intersect with each other”.

The remainder of this paper is structured as follows: in section 2, we review related work in the field of context-based scene analysis. In section 2 we list our contributions. In section 3, we explain the fundamental idea of our generalizable knowledge framework. In section 4, we briefly introduce the theory of Markov logic networks. In section 5, we elaborate on how to use our knowledge framework to solve the problem of table-top scene analysis. In section 6, we evaluate the performance of our system using real world data. In the end, we conclude in section 7.

2. Related Work

Scene analysis involves several different aspects, such as object identification [1, 2], object localization [3, 4, 5], and object discovery [6, 7, 8]. Depending on the context, each individual scene can be very different. For a table-top scene, a scene may contain several objects that are commonly found on tables, such as, books and computers. In traffic analysis, a scene may be something completely different and consist of cars, traffic lights 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 objects involved, while 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 information to analyze the perceived scene.

Several previous methods represent context knowledge as descriptive logic rules to help robots understand the perceived scenes. Using description logic [9, 10, 11], ontologies [12, 13] are used to encode knowledge about the composition of scenes, for example, that a set of cutlery consists of a knife, a fork, and a spoon. Using reasoning engines, such as Racer [14], Pellet [15] and FaCT++ [16], 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 of 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. [17] 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 behaviors 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.

Tenorth et. al. [18] 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 of 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.

Pangercic et. al. [19] proposed a top-down guided 3D model-based vision algorithm for assisting household environments. They use “how-to” instructions which are parsed and extracted from the wikihow.com webpage [20] to shape the top-down guidance. The robots knowledge base is represented in Description Logics (DL) using the Web Ontology Language (OWL) [21]. Based on the knowledge base, inferences are obtained using SWI-Prolog queries [22]. Using this system, the task “how to set a table” is accomplished, in which a robot prepares a table for a meal according to the instructions obtained from the wikihow webpage.

Some other methods use first-order logic [23, 24] or variants of first-order logic, such as Markov logic networks [25] or Bayesian logic networks [26], to formulate context knowledge to solve scene analysis. Blodow et. al. [27] use a Markov logic network to identify objects. Instead 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.

Additionally, there exist also methods that exploit

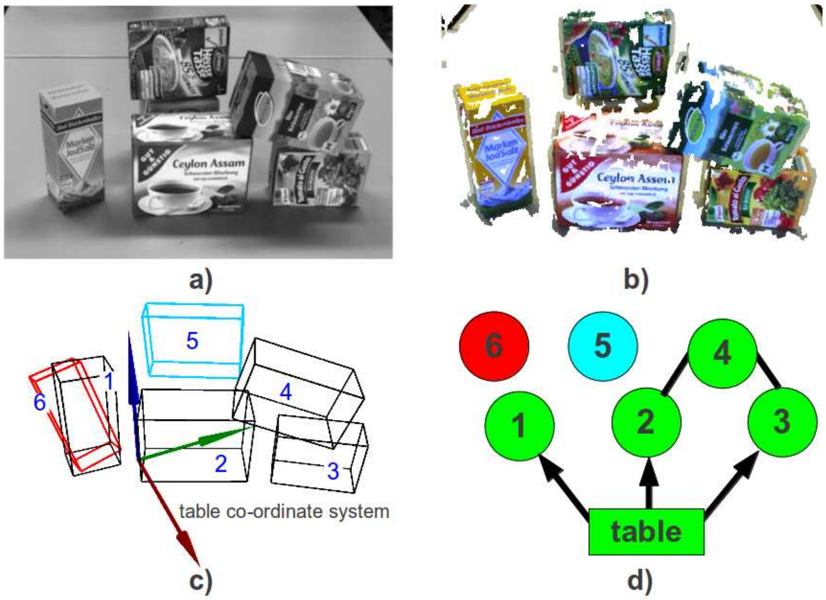


Figure 1: An example output of our system. a) - b) Sensor input for 6D pose estimation: stereo image (a) or 3D point cloud (b). c) Initial guess of object 6D poses obtained by a feature-based approach. The three axes of the table coordinate system are shown as blue, red and green arrows. d) The scene 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".

context information through human robot interaction (HRI) [28]. Motivated from psycholinguistic studies, Swadzba et. al. [29] 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 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. Other examples of this kind of approaches can be found in [30], [1], and [31].

A number of approaches analyze scenes using context information in other form. Grundmann et. al. [32] 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 modeled 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 using physics engines to check scene validity was presented in [33].

By modeling the relations between objects and their supporting surfaces in the image as a graphical model, Bao et. al. [34] 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 modeled by means of hidden parameters. The solution to the problem is finding the set of parameters that maximizes the joint probability. However, this approach aimed at detecting objects using context information and did not provide an abstract understanding of the perceived scenes.

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 [35]. We employ *Markov Logic Networks* (MLNs) [25] to encode the underlying context knowledge, since MLNs are able to model uncertain knowledge by combining first-order logic with probabilistic graphical models. We use a probabilistic sensor model to encode the fact that measurements are subject to significant uncertainty. We integrate the measurements with the abstract 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.

3. A Generalizable Knowledge-Supervised MCMC Sampling Framework

Markov chain Monte Carlo (MCMC) [36] methods are a class of algorithms for sampling from probability distributions. In MCMC methods, a Markov chain, whose equilibrium distribution is identical with the desired distribution, 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.

In this paper, we apply our generalizable knowledge-supervised MCMC (KSMCMC) sampling framework [35], which is a modern extension of MCMC methods, to interpret table-top scenes. KSMCMC is a combination of Markov logic networks [25] and data-driven MCMC sampling [37]. Based on Markov logic networks, 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 combining high-level knowledge processing with low-level data processing in a probabilistic and systematic way.

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 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 $p(M|\mathrm{D})$ which can be calculated as follows:

$$p(M|\mathrm{D}) \propto p(\mathrm{D}|M) \cdot p(M). \tag{1}$$

Here, the term $p(\mathrm{D}|M)$ 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 5.4 and section 5.5 respectively.

Starting from an initial guess of the model, we apply a data driven MCMC 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:

$$M^* = \underset{M \in \Omega}{\operatorname{argmax}}\ p(M|\mathrm{D}), \tag{2}$$

where Ω indicates the entire solution space. Details on this process are provided in section 5.6.

4. Markov Logic Networks

Before explaining the theory of Markov Logic Networks (MLNs), we briefly introduce the two fundamental ingredients of MLNs, which are Markov Networks and First-Order Logic.

4.1. Markov Networks

According to [38], 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

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}), \tag{3}$$

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

$$Z = \sum_{x \in \mathbb{X}} \prod_k \phi_k(x_{\{k\}}). \tag{4}$$

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

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j \omega_j f_j(X)\right), \tag{5}$$

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, please refer to [38].

4.2. 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, please refer to [39].

- *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 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 to represent 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*.

4.3. 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. The definition of a MLN is [25]:

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 $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ 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

$$\begin{aligned} P(X = x) &= \frac{1}{Z} \exp\left(\sum_i \omega_i n_i(x)\right) \\ &= \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)}, \end{aligned} \tag{6}$$

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 $\phi_i(x_{\{i\}}) = e^{\omega_i}$. For more details on MLN, please refer to [25].

5. Scene Graph Generation

5.1. 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 [40]. To apply context knowledge to scene analysis, we transform the initial guess of the 6D poses of the objects from the sensor coordinate system into the table coordinate system (see Fig. 2).

5.1.1. 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 properties of table-top scenes. The predicates are given in Table 1 and encode the following properties:

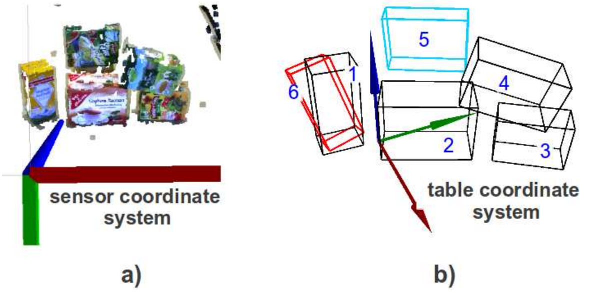


Figure 2: a) The object poses are initially calculated in the sensor coordinate system. b) To apply context knowledge in a table-top scene, the objects poses are transformed into the table coordinate system.

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

Table 1: Declaration of evidence predicates

- *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. 2-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. 2-b, for instance, there is contact between object #2 and #3, and between object #3 and #4. By contrast, there is no contact between object #4 and #5, or between object #2 and #5.

- *intersect(object,object)*: this predicate indicates whether two objects intersect each other. In the scene shown in Fig. 2-b, intersection only occurs 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. 2-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. 2-b, this predicate is only true for object #5.

5.1.2. Context Knowledge Defined as Logic Rules

Having defined the evidence predicates, we formulate context knowledge as descriptive rules using Markov logic. Knowledge can be defined as soft rules or hard rules in Markov logic to express uncertainty. Knowledge 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 to model table-top scenes (see Table 2).

Hard Rules.

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. 2-b, for example, this relation holds between the table and objects #1, #2, and #3 respectively.

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) → contact(o2,o1)
r_5	∞	intersect(o1,o2) → intersect(o2,o1)
r_6	∞	higher(o1,o2) → !higher(o2,o1)
r_7	∞	table(o1) → !false(o1)
r_8	∞	table(o1) → !hidden(o1)
r_9	∞	table(o1) → stable(o1)
r_{10}	∞	stable(o1) ∧ stable(o2) ∧ contact(o1,o2) ∧ higher(o1,o2) → supportive(o2) ∧ supported(o1)
r_{11}	log(0.70/0.30)	supported(o1) → !hidden(o1)
r_{12}	log(0.90/0.10)	!stable(o1) → !supportive(o1)
r_{13}	log(0.90/0.10)	hover(o1) → false(o1) v hidden(o1)
r_{14}	log(0.90/0.10)	intersect(o1,o2) → false(o1) v false(o2)
r_{15}	log(0.70/0.30)	supportive(o1) → !false(o1)
r_{16}	log(0.90/0.10)	stable(o1) → !false(o1)

Table 2: Declaration of rules

Soft Rules.

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.

5.1.3. Query Predicates

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 3 and have the following interpretations:

- *supportive(object)*: this predicate indicates that the object represented by the attribute physically supports other objects.

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

Table 3: Declaration of query predicates

- *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)*.

- *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. 2-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. 2-b, this predicate is true for object #6.

5.1.4. 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 $p1, p2 \in$

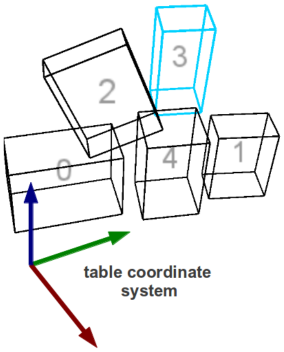


Figure 3: An example scene consisting of five objects represented by their corresponding oriented bounding box.

(0, 1) and $p1+p2 = 1$, means that the corresponding rule holds with the probability of $p1$ [25]. These weights can either be learned [41, 42, 43] or manually designed [44]. 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.

5.1.5. 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) [45]. 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. 3. Here object #0, #1, #3 and #4 have a stable pose. In contrast, object #2 has a unstable pose.

- *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. 3).
  - 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. 3).

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

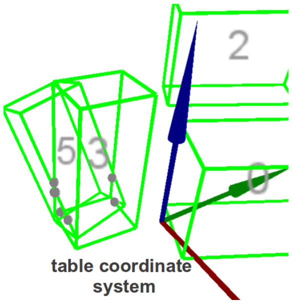


Figure 4: An example of two objects intersecting each other. Points of intersection are shown by gray spheres.

- *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. Here *intersect(object,object)* is true for object #3 and #5. 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. 3.

- *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.

The performance of discriminative generation of evidences is important for our system to deliver correct inferences. If something goes wrong with evidence generation, e.g., a “supportive” or “supported” relation is missed, the inference results would be less accurate. Since evidence generation is done based on discriminative methods, errors could happen (but very rarely), when it comes to some near-to-threshold cases. For instance, if we define that two objects are considered to have a contact if the closest distance between these two objects is less than 0.5 cm. Then for the cases, in which the closest distance between two objects is 0.6 cm or 0.7 cm, no contact will be detected, although the detection of contact would be more favourable in this case.

5.2. 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. [46]. The basic computational steps are given in Alg. 1. The algorithm is based on Scale-invariant feature transform (SIFT) keypoints [47] that are extracted from triangulated stereo images or RGBD measurements, e.g., from the Kinect sensor.

Algorithm 1 6D Object Pose Estimation

```
Require:
    $z$, input measurement
    $D$, object database
Ensure:
    $H$, set of pose hypotheses
extract SIFT keypoints from $z$
match keypoints to database $D$
for all object models $d \in D$ do
    for $i$ iterations do
        randomly choose three keypoints matched to $d$
        compute object pose hypothesis from matches
    end for
    cluster pose hypotheses for object $d$
    add clustered hypotheses to $H$
end for
```

In a first step, the SIFT keypoints of the observed objects are matched to a database D of object models. For each object model $d \in D$, 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 d. 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. 5. 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.

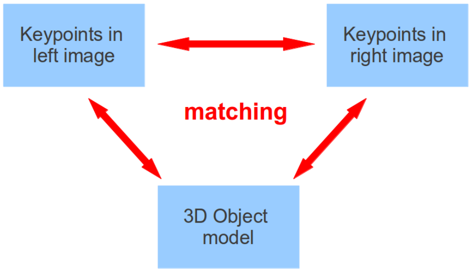


Figure 5: Matching of SIFT keypoints extracted from images.

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 [48].

An example of pose estimation process is depicted in Fig. 6. First, the keypoints extracted from the stereo images are matched (see Fig. 6-a, matched keypoint pairs are visualized by yellow lines). Then, matched keypoints are compared to the object database. In Fig. 6-b and c, keypoints found in the object database are shown in yellow. For clarity, only the key points of an object are shown. Using these matched keypoints, pose hypotheses are generated which are shown in Fig. 6-d. Pose estimation is performed for each new scene but is not repeated during scene graph generation.

5.3. Object Database

We use the object database of the Deutsche Servicerobotik Initiative (DESIRE) project [50]. The object models are generated using an accurate 3D modelling device which is equipped with a turn table, a movable stereo camera pair and a digitizer. An example of the modelling process is illustrated in Fig. 7. As shown in Fig. 7-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 (7-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 mesh of 3D SIFT keypoints (7-e). In 7-f, an overview of the DESIRE object database which contains 100 house-hold items is demonstrated.

5.4. Calculation of Prior Probability

Having defined the predicates and the rules, a knowledge base is formulated in the form of a Markov logic

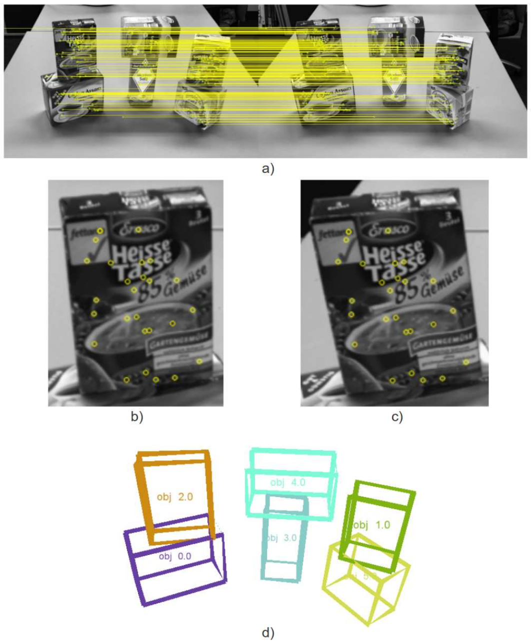


Figure 6: An example of pose estimation. a) Stereo matching of detected SIFT keypoints. b) Database-matched keypoints of an object in the left image. c) Database-matched keypoints of the same object in the right image. d) Generated pose hypotheses.

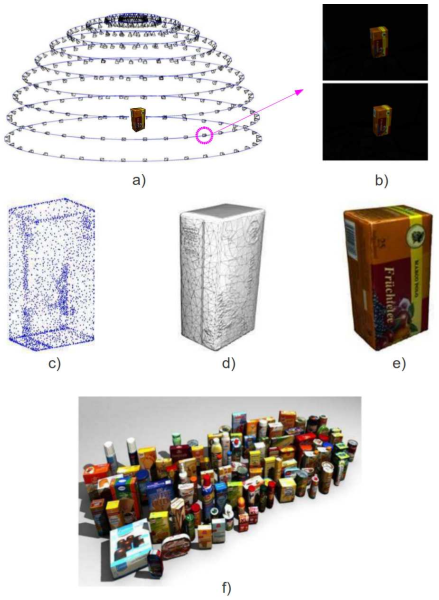


Figure 7: An example of the modelling process. All sub-figures are obtained from [49]. a) Camera poses made possible by the turn table and the camera movement. b) A stereo image pair obtained from the highlighted (magenta) camera pose. c) High resolution point cloud obtained through triangulation of matched feature points in stereo images. d) Generated triangle mesh. e) Textured triangle mesh. f) An overview of the object database.

network (MLN). A MLN initializes a ground Markov network [25], 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 M) is given by the probability distribution that is represented by this ground Markov network. As shown in the equation (6), 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 (1):

$$p(M) = \exp\left(\sum_i \omega_i n_i(x) \right). \tag{7}$$

In this work, we adapt the ProbCog Toolbox [51] to perform MLN inference and to calculate this unnormalized probability.

5.5. *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. [52]. 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 [53], 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 $p(\mathrm{D}|M)$ in equation (1) is then calculated as

$$p(\mathrm{D}|M) = \prod_i^n \left(\frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{(x_i - x_i^\psi)^2}{2\sigma_x^2}} \frac{1}{\sigma_y \sqrt{2\pi}} e^{-\frac{(y_i - y_i^\psi)^2}{2\sigma_y^2}} \right), \tag{8}$$

where σ_x and σ_y are the standard deviation in x- and y-direction of the image coordinates. We use this sensor model to determine the likelihood in equation (1). In our experiments, we use a standard deviation of 1 pixel for σ_x and σ_y.

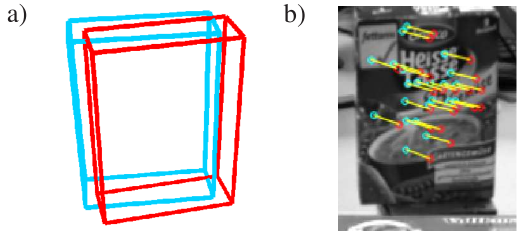


Figure 8: Evaluation of a pose estimate using the Gaussian sensor model. a) A pose (red) is evaluated against the database object pose (blue). b) Key points in the stereo image that are matched with the object model are shown in cyan (for clarity, only the left camera image is shown). The projected model key points are shown in red. The correspondences between projected and matched key points are shown by yellow lines.

An illustration of this method is given in Fig. 8. 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 to 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.

5.6. *Data Driven MCMC*

Because sensor data could be noisy and the used object database is imperfect, the input pose estimate of the observed objects is also imperfect. To find the scene graph that best explains the perceived scene, we apply a data driven MCMC process [37]. 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 $e_{t,0}$ and weight them using the sensor model (equation (8)). In this way, the input pose estimate is optimized.

An example of generating new pose estimates is given in Fig. 9. 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 p_a, using the Metropolis-Hastings algorithm [54]:

$$p_a = \min\left(1, \frac{P(M_{t+1}|D) \cdot Q(M_t|M_{t+1})}{P(M_t|D) \cdot Q(M_{t+1}|M_t)}\right), \tag{9}$$

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

$$Q(M_{t+1}|M_t) = \frac{weight(e_t^*)}{\sum_i^n weight(e_{t,i}) + weight(e_{t,0})}. \tag{10}$$

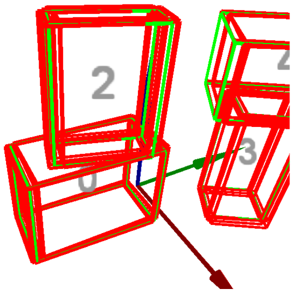


Figure 9: Generating new pose estimates (red) by adding Gaussian noises to the current pose estimate (green).

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

$$Q(M_t|M_{t+1}) = \frac{weight(e_{t,0})}{\sum_i^n weight(e_{t,i}) + weight(e_{t,0})}. \tag{11}$$

Here $weight(e_{t,i})$ is the likelihood calculated using $e_{t,i}$ as pose estimate (equation (8)).

6. Evaluation

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 objects or false estimates.

A selection of typical results is shown in Fig. 10 to 16. 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.

6.1. *Inference*

In our experiments, the defined knowledge base (Table 2) 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 [51], 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 labeled 25 complex table-top scenes. Each of the scenes contained several household objects of various types and had rather complex configurations. 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.

6.2. 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 paper 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.

7. Conclusion

In this paper, 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.

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.

Acknowledgements

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

References

[1] Y. Sun, L. Bo, D. Fox, Attribute based object identification, in: IEEE International Conference on Robotics and Automation, 2013.

[2] R. Dale, E. Reiter, Computational interpretations of the gricean maxims in the generation of referring expressions, Cognitive Science 19 (2) (1995) 233–263.

[3] B. Fulkerson, A. Vedaldi, S. Soatto, Class segmentation and object localization with superpixel neighborhoods, in: IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 670–677.

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

[5] H. Harzallah, F. Jurie, C. Schmid, Combining efficient object localization and image classification, in: IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 237–244.

[6] A. Karpathy, S. Miller, L. Fei-Fei, Object discovery in 3d scenes via shape analysis, in: IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 2088–2095.

[7] H. Kang, M. Hebert, T. Kanade, Discovering object instances from scenes of daily living, in: IEEE International Conference on Computer Vision, IEEE, 2011, pp. 762–769.

[8] M. Weber, M. Welling, P. Perona, Towards automatic discovery of object categories, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, IEEE, 2000, pp. 101–108.

[9] F. Baader, The description logic handbook: theory, implementation, and applications, Cambridge university press, 2003.

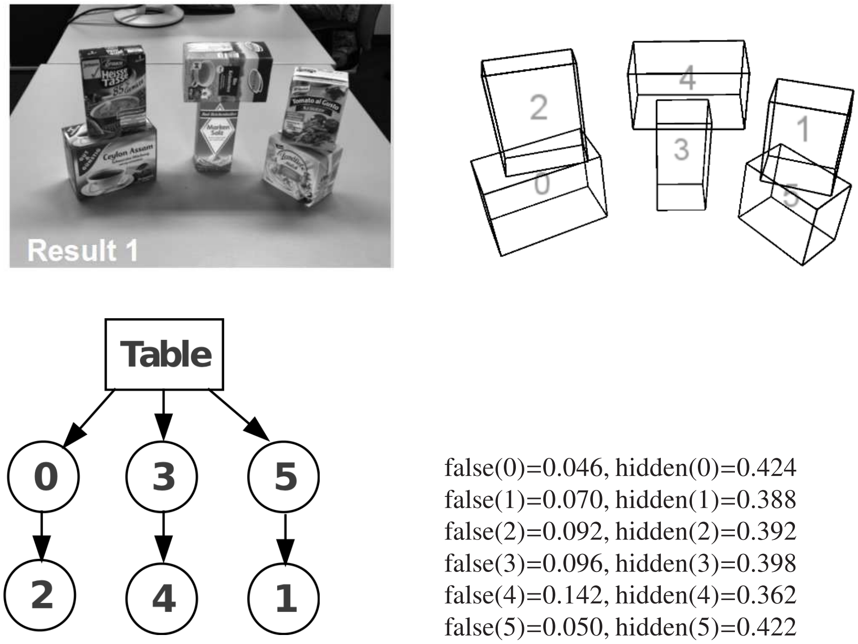
[10] B. N. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: Combining logic programs with description logic, in: Proceedings of the 12th international conference on World Wide Web, ACM, 2003, pp. 48–57.

[11] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Description logic framework for information integration, in: KR, 1998, pp. 2–13.

[12] T. Hofweber, Logic and ontology, 2008.

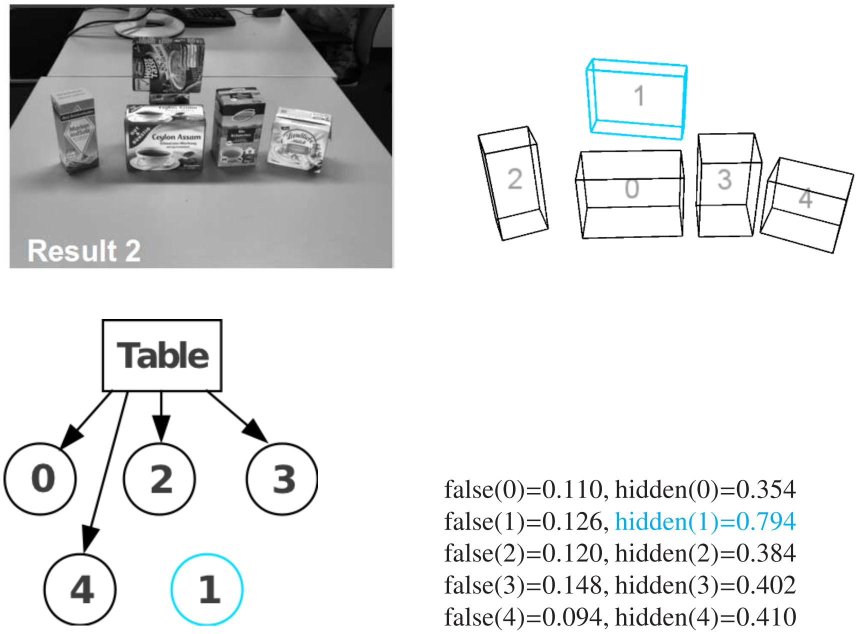
[13] P. Valore, Topics on General and Formal Ontology, Polimetrica, 2006.

[14] V. Haarslev, R. Möller, Racer: A core inference engine for the semantic web., in: EON, Vol. 87, 2003.



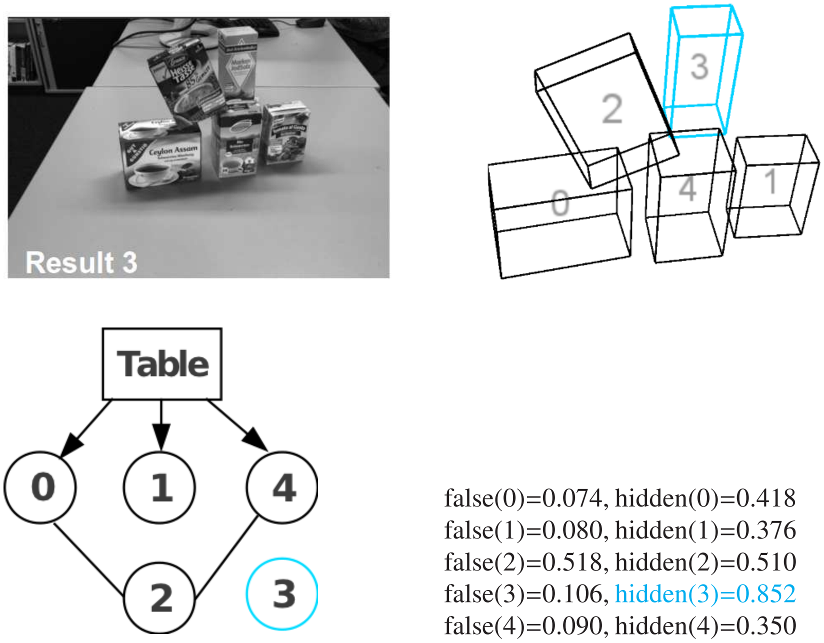
false(0)=0.046, hidden(0)=0.424
false(1)=0.070, hidden(1)=0.388
false(2)=0.092, hidden(2)=0.392
false(3)=0.096, hidden(3)=0.398
false(4)=0.142, hidden(4)=0.362
false(5)=0.050, hidden(5)=0.422

Figure 10: 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.



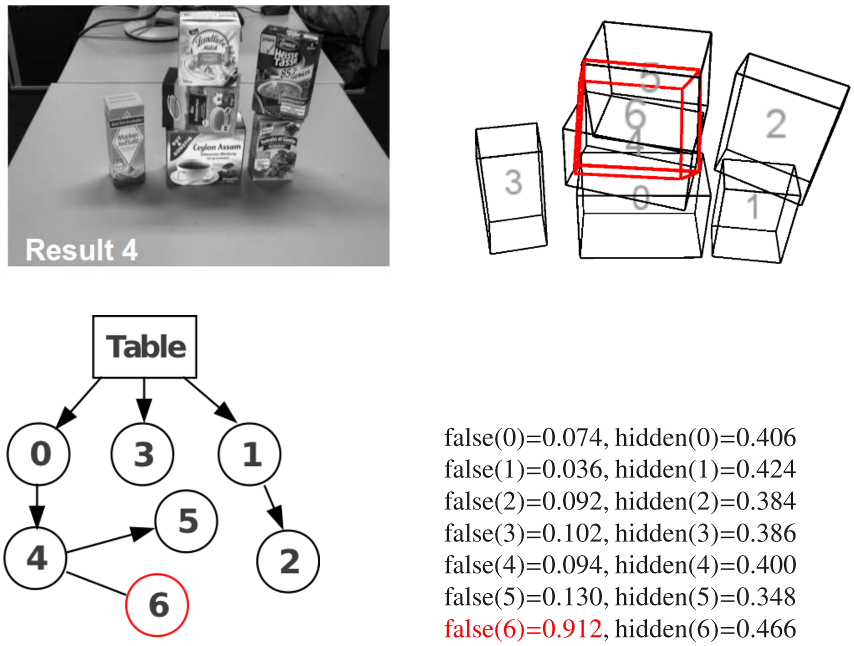
false(0)=0.110, hidden(0)=0.354
false(1)=0.126, hidden(1)=0.794
false(2)=0.120, hidden(2)=0.384
false(3)=0.148, hidden(3)=0.402
false(4)=0.094, hidden(4)=0.410

Figure 11: 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.



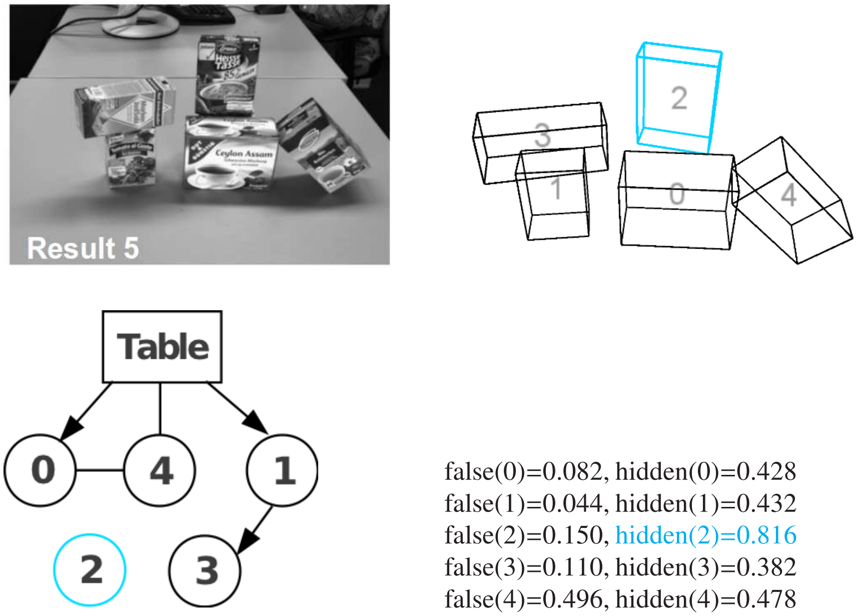
false(0)=0.074, hidden(0)=0.418
false(1)=0.080, hidden(1)=0.376
false(2)=0.518, hidden(2)=0.510
false(3)=0.106, hidden(3)=0.852
false(4)=0.090, hidden(4)=0.350

Figure 12: 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.



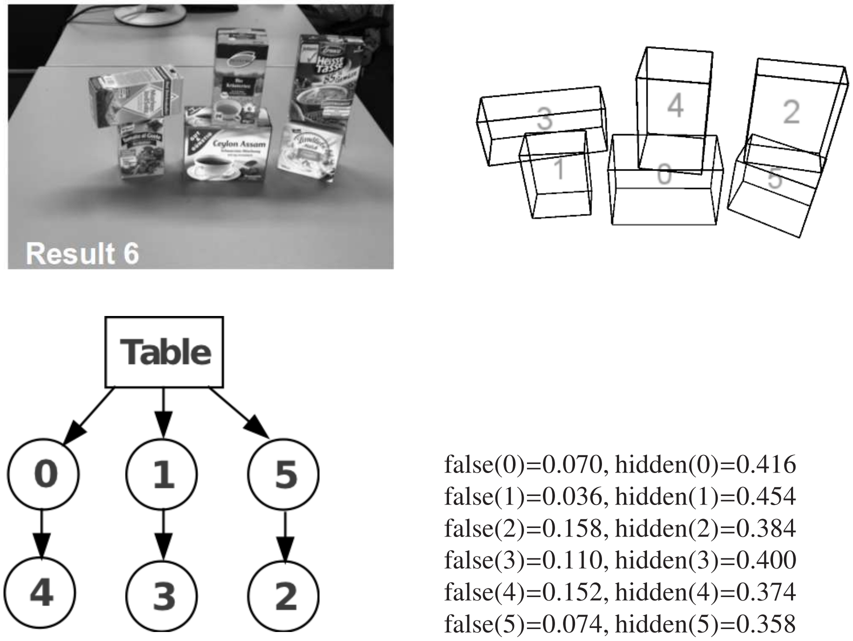
false(0)=0.074, hidden(0)=0.406
false(1)=0.036, hidden(1)=0.424
false(2)=0.092, hidden(2)=0.384
false(3)=0.102, hidden(3)=0.386
false(4)=0.094, hidden(4)=0.400
false(5)=0.130, hidden(5)=0.348
false(6)=0.912, hidden(6)=0.466

Figure 13: 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.



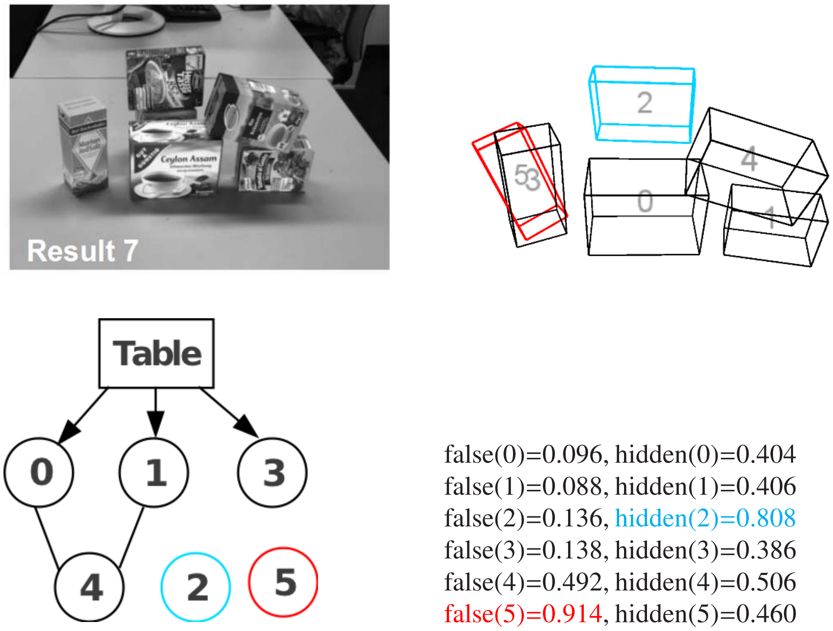
false(0)=0.082, hidden(0)=0.428
false(1)=0.044, hidden(1)=0.432
false(2)=0.150, hidden(2)=0.816
false(3)=0.110, hidden(3)=0.382
false(4)=0.496, hidden(4)=0.478

Figure 14: 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

Figure 15: 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

Figure 16: 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.

[15] E. Sirin, B. Parsia, Pellet system description, in: Proc. of the Int. Workshop on Description Logics, DL, Vol. 6, 2006.

[16] D. Tsarkov, I. Horrocks, Fact++ description logic reasoner: System description, in: Automated reasoning, Springer, 2006, pp. 292–297.

[17] G. H. Lim, I. H. Suh, 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) (2011) 492–509.

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

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

[20] http://www.wikihow.com/Main Page, wikihow (2013).
URL `http://www.wikihow.com/Main-Page`

[21] D. L. McGuinness, F. Van Harmelen, et al., Owl web ontology language overview, W3C recommendation 10 (2004-03) (2004) 10.

[22] J. Wielemaker, S. Ss, I. Ii, Swi-prolog 2.7-reference manual.

[23] R. M. Smullyan, First-order logic, Courier Dover Publications, 1995.

[24] M. Fitting, First-order logic and automated theorem proving, Springer, 1996.

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

[26] D. Jain, S. Waldherr, M. Beetz, Bayesian logic networks, IAS Group, Fakultät für Informatik, Technische Universität München, Tech. Rep, 2009.

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

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

[29] A. Swadzba, S. Wachsmuth, C. Vorwerg, G. Rickheit, A computational model for the alignment of hierarchical scene representations in human-robot interaction., in: IJCAI, 2009, pp. 1857–1863.

[30] X. Yu, C. Fermuller, C. L. Teo, Y. Yang, Y. Aloimonos, Active scene recognition with vision and language, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 810–817.

[31] N. Mavridis, D. Roy, Grounded situation models for robots: Where words and percepts meet, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 4690–4697.

[32] T. Grundmann, M. Fiegert, 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.

[33] N. Wagle, N. Correll, Multiple object 3d-mapping using a physics simulator, Tech. rep., University of Colorado at Boulder (2010).

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

[35] Z. Liu, 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.

[36] R. M. Neal, Probabilistic inference using markov chain monte carlo methods, 1993.

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

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

[39] M. Genesereth, N. Nilsson, Logical foundations of artificial intelligence, Vol. 9, Morgan Kaufmann Los Altos, CA, 1987.

[40] R. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 1–4.

[41] D. Lowd, P. Domingos, Efficient weight learning for markov logic networks, Knowledge Discovery in Databases (2007) 200–211.

[42] T. N. Huynh, R. J. Mooney, Discriminative structure and parameter learning for markov logic networks, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 416–423.

[43] T. N. Huynh, R. J. Mooney, Max-margin weight learning for markov logic networks, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2009, pp. 564–579.

[44] D. Jain, Knowledge engineering with markov logic networks: A review, Evolving Knowledge in Theory and Applications, 2011.

[45] M. Bender, M. Brill, Computergrafik, Vol. 2, Hanser, 2003.

[46] T. Grundmann, R. Eidenberger, M. Schneider, M. Fiegert, 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.

[47] D. G. Lowe, Object recognition from local scale-invariant features, in: The proceedings of the seventh IEEE international conference on Computer vision, Vol. 2, 1999, pp. 1150–1157.

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

[49] T. Grundmann, Scene analysis for service robots, Ph.D. Dissertation, 2012.

[50] Deutsche servicerobotik initiative, http://www.service-robotik-initiative.de/ (2009).
URL `http://www.service-robotik-initiative.de/`

[51] D. Jain, Probcog toolbox, http://ias.cs.tum.edu/software/probcog (2011).
URL `http://ias.cs.tum.edu/software/probcog`

[52] T. Grundmann, W. Feiten, 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), 2011, pp. 2085–2090.

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

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

Ziyuan Liu received his B.E. degree in Mechatronics from the TongJi University, Shanghai, China, in 2008. He received his M.S. degree in 2010 from the Institute of Automatic Control Engineering at Technische Universität München, Munich, Germany, where he is a Ph.D. candidate currently. His research interests are knowledge-based inference, semantic perception and sampling methods.

Dong Chen received his Diploma (MSc) from the department of Electrical and Information Technology at Technische Universität München, Munich, Germany. Currently he is a Ph.D. candidate at the Institute of Automatic Control Engineering at Technische Universität München. His research interests include mobile robot manipulation and control.

Kai M. Wurm joined Siemens in 2012 and is a member of Siemens Corporate Technology. He is an engineer at the research and technology center and part of the robotics and autonomous systems group. From 2007 to 2012 he worked as a research scientist at the University of Freiburg (Germany). In 2012, he received his PhD (Dr. rer. nat.) in computer science. His research interests lie in the fields of robot navigation, high-level control, terrain classification, SLAM, and 3D perception.

Georg von Wichert received his Diploma (MSc) in Electrical and Control Engineering from Darmstadt University of Technology in 1992. From 1992 to 1998 he was a research and teaching assistant at the Institute of Control Engineering at Darmstadt University of Technology. In Darmstadt he also received the Ph.D. degree in Electrical Engineering in 1998. Since 1998 his is with Siemens Corporate Research and Technologies, where he currently holds the position of the program manager for Cognitive Autonomous System. At the same time he is a fellow of the Institute for Advanced Study at Technische Universität München.