2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Using Rule-Based Context Knowledge to Model Table-Top Scenes

Ziyuan Liu Dong Chen Kai M. Wurm Georg von Wichert

***Abstract*— In this paper, we propose a probabilistic method to generate abstract scene graphs for table-top scenes from 6D object pose estimates. We explicitly make use of task-specific context knowledge by encoding this knowledge as descriptive rules in Markov logic networks. Our approach to generate scene graphs is probabilistic: Uncertainty in the object poses is addressed by a probabilistic sensor model that is embedded in a data driven MCMC process. We apply Markov logic inference to reason about hidden objects and to detect false estimates of object poses. The effectiveness of our approach is demonstrated and evaluated in real world experiments.**

I. 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, for example, 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 are 6D object poses that are generated using a feature-based pose estimation approach. This approach estimates object poses either from stereo images or from RGBD point clouds (e.g., generated using the Kinect sensor). A typical result of the approach proposed in this paper is shown in Fig. 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.

Z. Liu and D. Chen are with the Institute of Automatic Control Engineering, Technische Universität München, D-80290, Munich, Germany. ziyuan.liu@tum.de, chendong@mytum.de
K.M. Wurm is with Siemens AG, Corporate Technology, Munich, Germany.
G. von Wichert is with Siemens AG, Corporate Technology, Munich, Germany and Institute for Advanced Study, Techniche Universität München, Munich, Germany georg.wichert@siemens.com

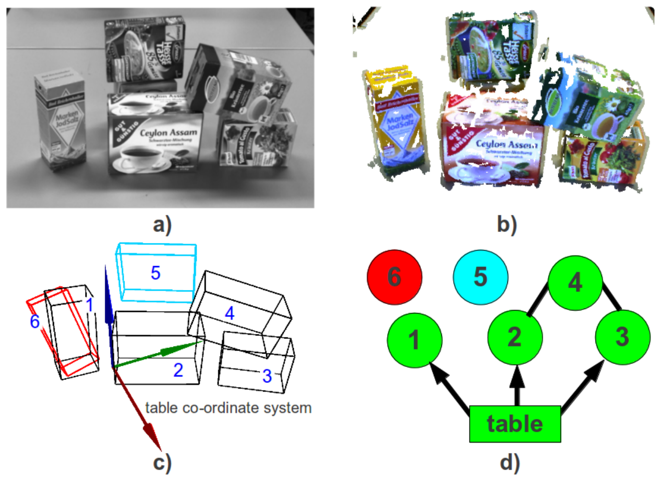


Fig. 1. An example output of our system. a) - b) Sensor input for 6D pose estimation: stereo image (a) or 3D point cloud (b). c) Initial guess of object 6D poses obtained by a feature-based approach. The three axes of the table coordinate system are shown as blue, red and green arrows. d) The scene graph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

The contribution of this paper is a probabilistic approach to generate scene graphs from uncertain 6D pose estimates. We employ *Markov Logic Networks* (MLNs) [15] 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 uncertain scene graph in a data driven Markov chain Monte Carlo (DDMCMC) 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.

II. Related Work

Several previous methods represent knowledge by descriptive logic rules to help robots understand their environment. Typically, ontologies are used to encode knowledge about the composition of the world, e.g., a set of cutlery contains a knife, a fork, a spoon and so on. Using reasoning engines, missing or wrong items can be inferred, and corresponding actions are triggered for the operating robot. Pangercic *et al.* [14] and Tenorth *et al.* [18] use such a system to help robots to perform manipulation tasks. Lim *et al.* [11]

employ such systems to improve perception performance. In addition to ontology-based approaches, Blodow *et al.* [2] use a Markov logic network to address the problem of object identity resolution. Instead of generating scene graphs of the perceived scenes, they focus on inferring the temporal correspondence between observations and entities. By analyzing human verbal input that describes the spatial relations of several objects, Swadzba *et al.* [17] analyze indoor scenes.

A number of approaches analyze scenes using context knowledge but do not encode this knowledge in the form of descriptive logic rules [8], [20], [1]. Grundmann *et al.* [8] employ a physic engine to check the validity of the estimated scene models. Scene models that fail the validity check of the implemented physics engine are given a probability of zero. Another example of using physics engines to check scene validity was presented in [20]. By modeling the relations between objects and their supporting surfaces in the image as a graphical model, Bao *et al.* [1] 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.

III. A Generalizable Knowledge-Supervised MCMC framework

In this paper, we apply our generalizable knowledge-supervised MCMC (KSMCMC) framework [12] 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 $p(M|\mathrm{D})$ 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.

Starting from an initial guess of the model, we apply a data driven MCMC [19] 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:

$$M^* = \arg\max_{M\in\Omega} p(M|\mathrm{D}), \tag{2}$$

where Ω indicates the entire solution space. In section V, we show in detail how to use this framework for the task of generating scene graphs for table-top scenes.

IV. Markov Logic Networks

Since Markov Logic Networks are used to encode context knowledge in our approach, we provide a short introduction to MLNs. For further reading we refer to [15].

Unlike first-order knowledge bases [5], 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 *possible world* [5] less probable, but not impossible (the fewer formulas a possible world violates, the more probable it is). In MLNs, each formula is assigned a weight representing how strong this formula is. According to [15], a MLN is defined as follows:

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 of a possible world x specified by the ground Markov network $M_{L,C}$ is calculated as

$$P(x) = \frac{1}{Z}\exp\left(\sum_i \omega_i n_i(x)\right) \tag{3}$$

where $n_i(x)$ is the number of true groundings of F_i in x. Z is a normalization factor which is the same for all possible worlds with the same number of constants C.

V. Scene Graph Generation

A. 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 common sense 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 [16]. To apply context knowledge to pose estimates, we transform the initial guess of the 6D poses of the objects from the sensor coordinate system into the table coordinate system (see Fig. 1). The rules and the predicates that express context knowledge in our system are shown in Table I.

1) Evidence Predicates: Evidences are abstract terms that are detected from the perceived scene given the object poses and models (see Sec. V-A.6). The predicate *stable(object)* indicates that an object has a stable pose, i.e., it stably lies on a horizontal plane, for example objects #1, #2, #3 and #5 in Fig. 1. Objects #4 and #6, in contrast, have an unstable pose. *table(object)* provides the possibility to model tables as objects, so that we can use it in the reasoning. *contact(object,object)* and *intersect(object,object)* show whether two objects have contact or intersect each other. *higher(object,object)* expresses that the position of the first

index i	weight ω_i	formula F_i	evidence predicate	query predicate
1	∞	!higher(o1,o1)	stable(object)	supported(object)
2	∞	!intersect(o1,o1)	table(object)	supportive(object)
3	∞	!contact(o1,o1)	contact(object,object)	hidden(object)
4	∞	contact(o1,o2) $\rightarrow$ contact(o2,o1)	intersect(object,object)	false(object)
5	∞	intersect(o1,o2) $\rightarrow$ intersect(o2,o1)	hover(object)	
6	∞	higher(o1,o2) $\rightarrow$!higher(o2,o1)	higher(object,object)	
7	∞	table(o1) $\rightarrow$!false(o1)		
8	∞	table(o1) $\rightarrow$!hidden(o1)		
9	∞	table(o1) $\rightarrow$ stable(o1)		
10	∞	stable(o1) $\wedge$ stable(o2) $\wedge$ contact(o1,o2) $\wedge$ higher(o1,o2) $\rightarrow$ supportive(o2) $\wedge$ supported(o1)		
11	log(0.70/0.30)	supported(o1) $\rightarrow$!hidden(o1)		
12	log(0.90/0.10)	!stable(o1) $\rightarrow$!supportive(o1)		
13	log(0.90/0.10)	hover(o1) $\rightarrow$ false(o1) v hidden(o1)		
14	log(0.90/0.10)	intersect(o1,o2) $\rightarrow$ false(o1) v false(o2)		
15	log(0.70/0.30)	supportive(o1) $\rightarrow$!false(o1)		
16	log(0.90/0.10)	stable(o1) $\rightarrow$!false(o1)		

TABLE I
DECLARATION OF RULES AND PREDICATES

attribute is higher than that of the second attribute in the table coordinate system. If an object does not have any contact with other objects including the table, then it is denoted as *hover(object)*.

2) Query Predicates: Using the rules defined above, query predicates are inferred using the evidence. *hidden(object)* expresses that there is an hidden object in the scene under the object that is represented by the attribute (e.g., object #5 in Fig. 1). *false(object)* indicates that the object is a false estimate (e.g., object #6 in Fig. 1). To infer about *hidden(object)* and *false(object)*, two auxiliary query predicates *supportive(object)* and *supported* are defined.

3) Hard Rules: Hard rules are considered to hold in all cases and are assigned a weight of ∞. For instance, rule #1 expresses the fact that an object cannot be higher than itself. Rules #2 and #3 encode that an object does not intersect or have contact with itself. Rules #4, #5, and #6 ensure the commutativity of the corresponding predicates. Rule #7, #8, and #9 express the assumption that in a table-top scene, the table (as an object) is not a false estimate, that it is the lowest object in the scene and that it has a stable pose. Rule #10 describes typical "supportive" and "supported" relations between two objects with a stable pose. These relations do not hold for objects with unstable poses. In Fig. 1, for example, this relation holds between the table and objects #1, #2, and #3 respectively.

4) Soft Rules: Rule #11 encodes that an object that is already known to be supported (rule #10) is not likely to have a hidden object under it. Rule #12 expresses that an object with an unstable pose is unlikely to be supportive. Rule #13 states that a hovering object is either a false estimate or has a hidden support under it. Rule #14 indicates that if two objects intersect, then one of them is probably a false estimate. Rule #15 and #16 mean that a supportive or a stable object is unlikely to be a false estimate.

The choice of the rules is a problem-oriented engineering step, and the rules given in this paper serve as an example of how to encode the properties of typical table top scenes. In order to keep the knowledge base tractable, we need to abstract the scene description and this inevitably introduces modeling errors. This is, why we need to model the context knowledge using uncertain rules.

5) 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 $p1$ [15]. These weights can either be learned [13] or manually designed [9]. 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.

Having defined the predicates and the rules, a MLN initializes a ground Markov network [15] that represents the probability of the modeled scene. In this work, we adapt the ProbCog Toolbox [10] to perform MLN inference. Using Markov Logic inference, we can answer the queries *hidden(object)* and *false(object)* in the form of a probability. By ignoring the normalization factor Z in equation (3), which is the same for all possible worlds (with the same number of constants) and computationally intensive to compute, the unnormalized probability of the perceived scene is calculated. This probability is then used as the prior in equation (1).

6) 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). However, the aforementioned principles generalize over objects with other shapes, as long as evidences can be provided.

To generate evidences, we analyze the oriented bounding boxes of detected objects. 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.

To detect whether two objects have contact or intersect 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, there is contact between object #2 and #4 in Fig. 1;
- There are multiple points of intersection and all points are coplanar and lie on one of the six facets of either OBB (for example, there is contact between object #2 and the table in Fig. 1.

In each of the above three cases, we set *contact(object,object)*=True and *intersect(object,object)*=False. 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 *contact(object,object)*=False and *intersect(object,object)*=True. In all other cases, we set *contact(object,object)*=False and *intersect(object,object)*=False. If an object does not have any contact or intersection with other objects, then we set *hover(object)*=True. If the position of *object1* is higher than the position of *object2* in the table coordinate system, then we set *higher(object1,object2)*=True.

B. 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.* [6]. The basic computational steps are given in Alg. 1. The algorithm is based on SIFT keypoints that are extracted from triangulated stereo images or RGBD measurements, e.g., from the Kinect sensor. In a first step, the SIFT keypoints 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. An object pose hypothesis is then computed from these triples of matched points. Finally, pose hypotheses are clustered and outliers are removed using the RANSAC algorithm [4]. Pose estimation is performed for each new scene but is not repeated during scene graph generation.

Algorithm 1 6D Object Pose Estimation

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

C. Gaussian Sensor Model

To evaluate estimated object poses, we use a Gaussian sensor model similar to the approach proposed by Grundmann *et al.* [7]. For each pose estimate ψ, 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, 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 sensor model is then calculated as

$$\prod_i^n \left(\frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{(x_i - x_i^{\psi})^2}{2\sigma_x^2}} \frac{1}{\sigma_y \sqrt{2\pi}} e^{-\frac{(y_i - y_i^{\psi})^2}{2\sigma_y^2}} \right), \tag{4}$$

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. An illustration is given in Fig. 2.

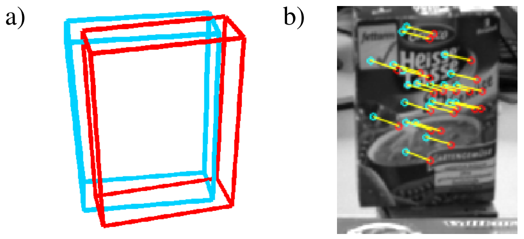


Fig. 2. Evaluation of a pose estimate using the Gaussian sensor model. a) A pose (red) is evaluated against the database object pose (blue). b) Key points in the stereo image that are matched with the object model are shown in cyan (for clarity, only the left camera image is shown). The expected model key points are shown in red. The correspondences between expected and matched key points are shown by yellow lines.

D. Data Driven MCMC

To find the scene graph that best explains the perceived scene, we apply a data driven MCMC (DDMCMC) process [19]. In the t-th iteration with scene graph S_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 (4)). Then the pose estimate with the best weight e_t^* is used to generate a new scene graph S_{t+1}, and this scene graph is accepted by the probability (Metropolis-Hastings algorithm [3])

$$\min \left(1, \frac{Pos(S_{t+1}) \cdot Pro(S_t | S_{t+1})}{Pos(S_t) \cdot Pro(S_{t+1} | S_t)} \right), \tag{5}$$

where $Pos(S_t)$ is the posterior probability of S_t (equation (1)). $Pro(S_{t+1}|S_t)$ is the proposal probability of generating S_{t+1} out of S_t and is calculated as

$$\frac{weight(e_t^*)}{\sum_i^n weight(e_{t,i}) + weight(e_{t,0})}. \tag{6}$$

Similarly, $Pro(S_t|S_{t+1})$ is computed as

$$\frac{weight(e_{t,0})}{\sum_i^n weight(e_{t,i}) + weight(e_{t,0})}. \tag{7}$$

VI. 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 objects or false estimates.

A selection of typical results is shown in Fig. 3. In each row, 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. In addition, all the false estimates and hidden objects are correctly inferred using the defined MLN (Table I). In the used MLN tool [10], 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 in rather complex configurations, similar to those shown in Fig. 3. The 25 scenes contained in all 10 hidden objects and 5 false estimates, all of which were correctly inferred.

In each iteration, the run time of our system is mainly spent on MLN reasoning (including evidence generation) and the DDMCMC 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 DDMCMC process. To get a good scene graph of the perceived scene, our system needs to perform 10 to 15 iterations.

VII. Conclusion

In this paper, we presented 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 common sense rules in Markov logic. Integrating these with an uncertain sensor model, we perform maximum posterior estimation of the scene parameters using a data driven 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.

Acknowledgments

This work is accomplished with the support of the Technische Universität München - Institute for Advanced Study.

The work was partially made possible by funding from the ARTEMIS Joint Undertaking as part of the project R3-COP and from the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS10004E.

References

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

[2] 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.

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

[4] 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.

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

[6] 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.

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

[8] 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.

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

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

[11] 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.

[12] 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.

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

[14] 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.

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

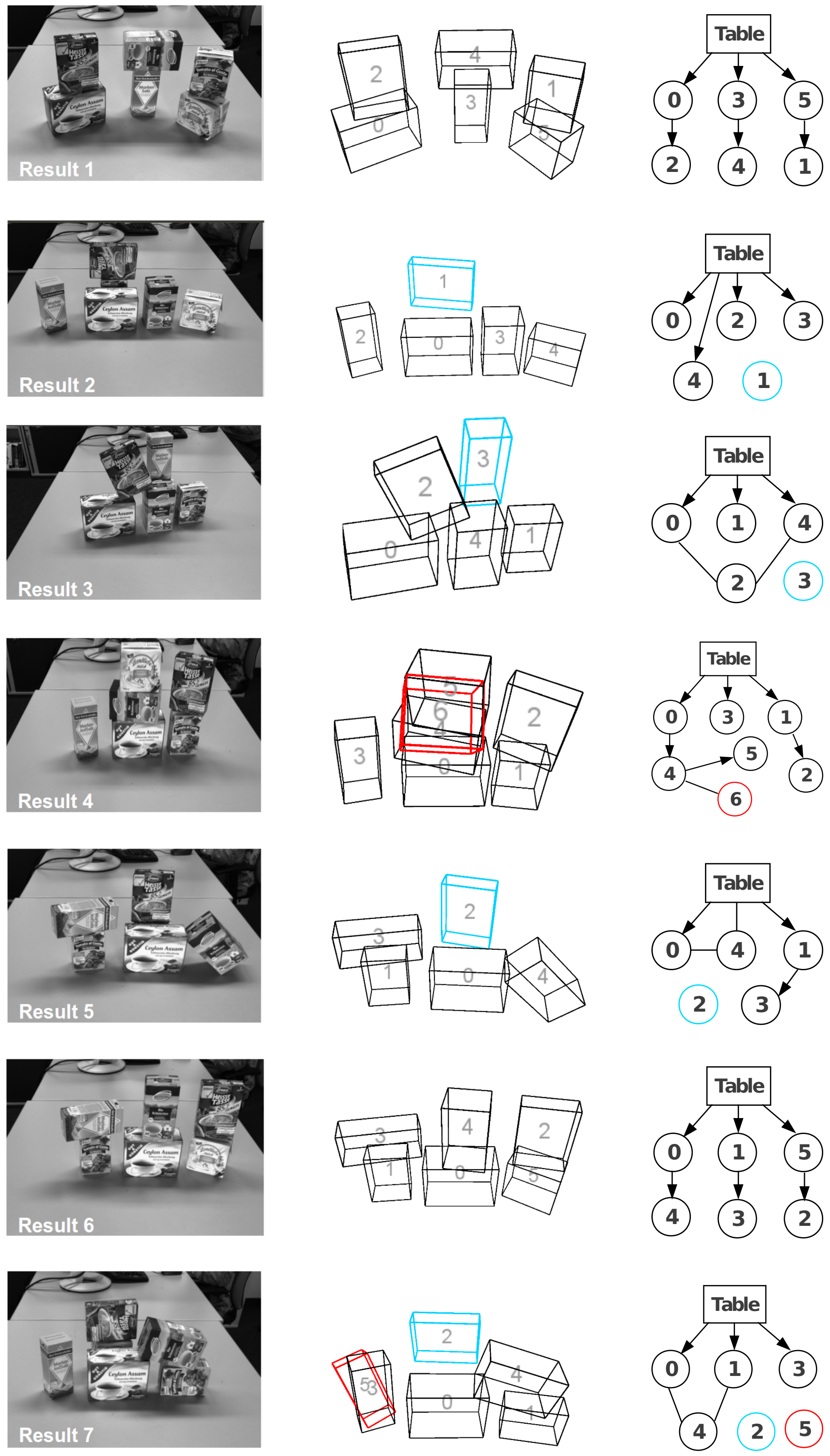
[16] R.B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *IEEE Int. Conf. on Robotics and Automation*, pages 1–4. IEEE, 2011.

[17] 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.

[18] 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.

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

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



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

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

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

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

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

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

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. 3. Experimental results. In each row, the input stereo image (only the left camera image is shown), estimated 6D poses, the resulting scene graph and the query probability are shown. False estimates and objects implying hidden objects are highlighted in red and cyan respectively.