The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Combining Perception and Knowledge Processing for Everyday Manipulation

Dejan Pangercic and Moritz Tenorth and Dominik Jain and Michael Beetz
Intelligent Autonomous Systems Group Technische Universität München
Email: {pangercic, tenorth, jain, beetz}@cs.tum.edu

Abstract— **This paper describes and discusses the K-CoPMan (Knowledge-enabled Cognitive Perception for Manipulation) system, which enables autonomous robots to generate symbolic representations of perceived objects and scenes and to infer answers to complex queries that require the combination of perception and knowledge processing. Using K-CoPMan, the robot can solve inference tasks such as identifying items that are likely to be missing on a breakfast table. To the programmer K-CoPMan, is presented as a logic programming system that can be queried just like a symbolic knowledge base. Internally, K-CoPMan is realized through a data structure framework together with a library of state-of-the-art perception mechanisms for mobile manipulation in human environments. Key features of K-CoPMan are that it can make a robot environment-aware and that it supports goal-directed as well as passive perceptual processing. K-CoPMan is fully integrated into an autonomous mobile manipulation robot and is realized within the open-source robot library ROS.**

I. Introduction

Autonomous robots performing everyday manipulation tasks have to make many decisions that require the combination of perception and knowledge processing. As an illustrative example for knowledge-enabled perception, consider a robot that is to set the table together with a human. In order to implicitly coordinate its course of action with the human, the robot has to fetch missing items. Based on what the robot sees on the table and the time of the day, the robot is to probabilistically infer what meal the table is set for, what is likely to be eaten, and, based on this, which utensils are likely to be required. In this paper, we propose the logic programming system K-CoPMan (Knowledge-enabled Cognitive Perception for Manipulation) that can test and satisfy knowledge preconditions for everyday manipulation. K-CoPMan fulfills three main functions:

1. Providing the robot with abstract symbolic knowledge about perceived scenes. K-CoPMan acquires and stores perceptual data during robot operation, associates data structures with symbolic names that can be used for perceptually grounded knowledge processing. These perceptions extend static perceptual data like environment maps [1]. There are two main perception mechanisms: task-directed and passive perception. *Task-directed perception* provides information necessary for accomplishing manipulation tasks – information about the object to be acted on and the scene context. The *passive perception* is to make the robot environment-aware by also memorizing objects that are not task-relevant at the time of perception.Object information is stored in K-CoPMan at different levels of detail, ranging from raw, sub-symbolic data to symbolic descriptions.

2. Using abstract symbolic knowledge for accomplishing perception tasks. K-CoPMan enables the robot to employ knowledge processing functionalities to simplify perceptual tasks by using (symbolic) models of context, situations, and goal-directed behavior. Using knowledge processing mechanisms and the belief state (memory) of the robot, the robot can for instance point the camera at places where it believes objects to be or exploit the fact that objects inside a cupboard are invisible unless the door is open.

3. Answering new types of queries that require the combination of knowledge processing and perception. For instance, K-CoPMan enables the robot to infer the items that are missing on a table set for a particular meal, the items that have to be put away in order to clean a table, or the items that have to be put into the fridge. Inferring these information requires using a combination of perception and knowledge processing mechanisms.

Technically, K-CoPMan is realized as an interface layer to open-source SWI Prolog. Prolog combines fast inference and computation with declarative, logics-based semantics. Lightweight Prolog inferences can even run in feedback loops up to 10 Hz to make the robot action-aware. Prolog's foreign language interface thereby facilitates the integration of perception routines written in other programming languages like C/C++.

The main contributions of this paper are the following ones: We present K-CoPMan, a logic programming system that integrates state-of-the-art perception and knowledge processing mechanisms for autonomous robot manipulation. The system can infer answers to many queries required for competent everyday manipulation. By automatically acquiring scene models of the relevant regions of interest, such as tables and cupboards [1], the robot becomes aware of its environment. K-CoPMan uses resource-adaptive/on-demand information processing on previously perceived raw sensor data to optimize for operation in parallel to the robot action execution.

The remainder of this paper starts with an overview of the software architecture (Section II). Then, the perception server and the library of perception routines are explained (Section III). Section IV describes the integration of the perceptual mechanisms with the knowledge processing system KnowRob [2]. We conclude with a demonstration scenario, discuss and evaluate an example query.

II. K-CoPMAN System Overview

We apply K-CoPMAN to the autonomous mobile manipulation robot depicted in Figure 2 (right), which is to perform everyday manipulation activities such as setting the table in a kitchen environment. K-CoPMAN controls and uses the sensor system shown in Figure 1. A pair of high-resolution color cameras, a stereo-on-the-chip camera, and a time-of-flight sensor on a pan-tilt sensor head are used for task-directed perception. In addition, a tilting laser scanner mounted on the robot's shoulder continually acquires depth maps of the scene in front of the robot (which are mostly used by the passive perception module).

K-CoPMAN is an extension of KnowRob [2], a system for fast and grounded knowledge processing on autonomous manipulation robots. K-CoPMAN extends KnowRob in two important ways. First, it adds a set of predicates that abstract away from the robot's perceptual mechanisms and transforms the perceptual tasks and their results into a logical representation suitable for knowledge processing and decision-making. Second, K-CoPMAN provides a continual update mechanism for the part of the knowledge base that represents the dynamic world state. This mechanism is to make the robot *environment-aware*, i.e. to always have a rough estimate of the current state of the world. For example, in our application, objects on tables and kitchen counters are declared as a relevant dynamic aspect of the world that should be monitored continually. K-CoPMAN keeps track of the positions of objects on different tables and asserts these percepts as logical facts.

A robot programmer can use KnowRob to define concepts needed for robot control in terms of first-order logical statements. For example, to write plans for joint human-robot table setting tasks, the programmer might want to define the concept of items that are missing on a table in the following way: Missing items on a table where people intend to have a meal *m* are those items that are predicted to be needed for this meal, but cannot be perceived to be already on the table. Having this definition, the programmer can write a plan fragment such as: *keep putting a missing item on the table until no further items are believed to be missing*. In this code fragment, the missing item is a knowledge precondition of the plan step that has to be achieved by computing which items in the environment satisfy the above concept definition.

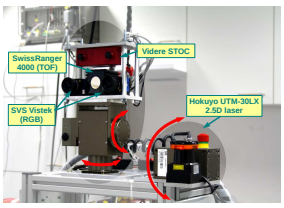


Fig. 1. Setup of the sensor head.

In this setting, K-CoPMAN's task is to test the perception-related parts of the concept definition. Thus, K-CoPMAN translates the conditions to be checked into parametrized perception routines and interprets their results in order to check the conditions. It also controls and supervises the perception processes that are spawned from the information requests, and it stores and manages the results returned by the perception processes.

In order to perform competent perception, it is often helpful to make use of other knowledge stored in KnowRob. In this example, checking the condition requires the robot to identify the right table, which is accomplished using the semantic environment map stored in KnowRob. It allows, for example, to query for objects of type "Table", especially for those that are used for having meals. Similarly, reasoning with perceived information requires the system to explicitly deal with the uncertainty that results from sensors being unreliable, inaccurate, and only providing incomplete information about the world. This functionality is provided by a predicate library that realizes probabilistic first-order reasoning.

A. K-CoPMAN *Components*

Figure 2 shows the embedding of K-CoPMAN into the overall robot control system and the software components of K-CoPMAN within this system. The core of K-CoPMAN is the K-CoPMAN perception server (see Section III). The K-CoPMAN perception server calls the respective perception routines, monitors and manages the perception processes they execute, and stores to and updates the K-CoPMAN data store according to the perception tasks and their results. The second component is the implementation of the K-CoPMAN predicates. The implementation translates information needed to compute the truth value of a predicate into parametrized calls of perception routines and interprets the results returned by these routines in terms of the information requested. The third component is the passive perception component, which continually acquires point cloud data obtained from laser sensor sweeps (Section III provides specifics). As a fourth component, K-CoPMAN uses KnowRob's query interface to communicate with the robot control program. The method *knowrob-query(*q*)* returns a boolean value depending on whether or not *q* is implied by the "virtual" knowledge base. *knowrob-query-var(var,* q*)* returns the bindings of the query variable *var* which renders the logical expression of the query true. The fifth component consists of KnowRob extension libraries for perceptual memory management, first-order probabilistic reasoning and static environment mapping. K-CoPMAN and KnowRob are implemented on our autonomous robot and integrated in ROS (Robot Operating System).

B. *Example Scenario*

Let us now consider our example task of bringing missing items to a breakfast table in more detail. Inferring the missing items is a very complex task and requires the integration of heterogeneous information: Where is the table? What is already on the table? What should be there? Where to find the missing items?

Figure 3 describes the specification of the *missingObjects* predicate. The first three conditions in the predicate require the variable *Table* to be a table in the environment and to have a primary function of having a meal on it. This

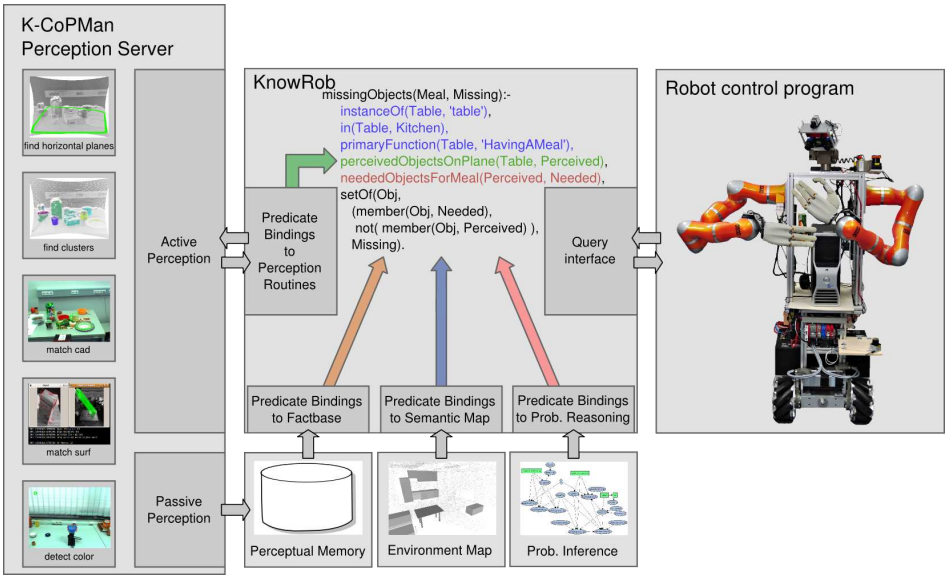


Fig. 2. K-CoPMan's building blocks. **Left)** K-CoPMan perception server with state-of-the-art perception routines and task-directed and passive perception modules. **Middle)** KnowRob with predicates for evoking of perception routines and extension plugins for first-order probabilistic reasoning and knowledge on static objects. **Right)** Robotic manipulator with logical control program.

condition can be met by employing the robot's semantic map of the environment to identify the tables in the environment (visualized in red). The fourth condition tests the set of objects in a given region of interest, which, in our case, is the top of the table, denoted by the variable *Table*. This condition is checked with the perceptual mechanisms of the K-CoPMan perception server which sets up a perception task to detect, categorize and recognize all the objects on the table and binds the result of this perception task to the Prolog variable *Perceived*. The next condition specifies the items that are probably needed on the table. To identify these, we use the first-order probabilistic reasoning component. Schematically, KnowRob converts the predicate *neededObjectsForMeal* into a query $P(on(Obj,Table) \mid Perceived_1, \ldots, Perceived_n)$, which is then computed for all possible objects. Given the result of this probabilistic query, KnowRob binds the set of objects for which the probability value exceeds some threshold θ to the Prolog variable *Needed* (e.g. $\theta = 0.5$ or lower, depending on how conservative we want to be). The last condition then determines the missing items *Missing* as those items that are in the set *Needed* but not in the set *Perceived*.

From the perspective of this paper, the specification of the predicate *perceivedObjectsOnPlane* is most relevant, as it actually uses the capabilities of K-CoPMan.

```
perceivedObjectsOnPlane(Plane, Perceived) :-
    onPlane(Plane),
    setOf(Obj-Hyp,
          ( on(Obj, Plane),
            category(Obj,Cat),
            uniqueId(Id),
            objectInstace(Obj,KnownObj),
            Obj-Hyp = [Id,Obj,Cat,KnownObj]),
          Perceived).
```

The condition collects all object hypotheses generated by the perception routine by producing a unique *Id* for each hypothesis, associating with it the raw sensor data *Obj* that belongs to the hypothesis, categorizing the object hypothesis (*Cat*), and checking whether the hypothesis is a known object instance *KnownObj*, and if so, which one. The perceptual routines needed for the realization of this condition are explained in Figure 4 and the definitions of the predicates in terms of these perception routines can be found in Section IV-B.

III. The K-CoPMan Perception Server

Let us now explain the perception routines supported by K-CoPMan, the passive perception, and the storage and management of perceptual data in more detail.

A. Perception Routines

The K-CoPMan *perception server* provides a set of perception mechanisms for images and point cloud data, including the detection of horizontal planes, point cloud clustering and categorization, CAD model matching, and color-based classification. The mechanisms that are most important for this paper are listed in Figure 4, which shows the name of the routine, how it can be called abstractly, a short functional description, and some sample results. These routines can be used both for task-directed perception and for the passive perception component described below.

B. Passive Perception

The passive perception component is a key mechanism of K-CoPMan, which makes the robot environment-aware. It searches the point clouds of the shoulder laser scanner for regions of interest, such as tables or cupboards. Whenever it finds such a region, it clusters the point cloud data in order to segment objects standing on top of it. A unique identifier is generated for each of these clusters and asserted to the knowledge base (perceptual memory), together with

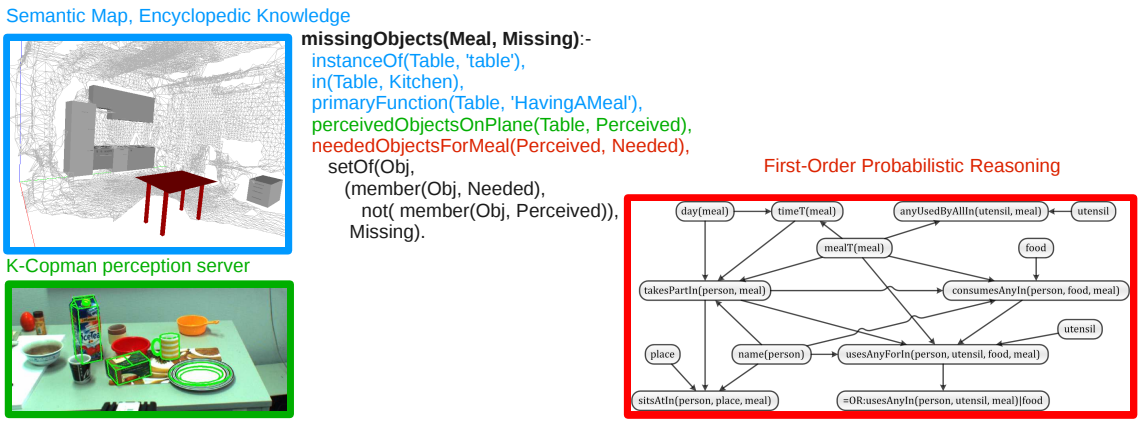


Fig. 3. Query to the K-CoPMAN system for items that are missing on a table with respect to a particular meal. The system first locates the table, perceives the objects on it, queries the probabilistic inference engine for items that are supposed to be on the table and determines those that are missing. BLN graphical model (right) is explained in Section IV-C.

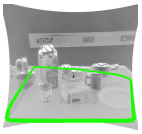
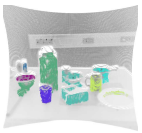
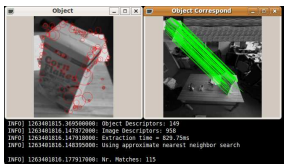
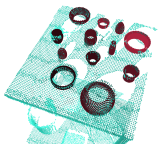
op	abstract routine call	functionality	example results
find-hor-planes	*plane-hyp* ← perceive an object category *plane* orientation *horizontal* size $\geq 0.25m^2$	The routine *find-hor-planes(pointcloud)* estimates surface normals based on local neighborhoods performs region growing on the points with approximately vertical normals. The routine then estimates the best horizontal plane using sample consensus and the minimal bounds thereof (for details see [3]).	
find-clusters	*obj-hyp* ← perceive an object category *pcd-cluster* supported-by *hor-plane*	The routine *find-clusters(pl)* is called with the symbolic name *pl* of a horizontal plane as its parameter and returns a set of names of object hypotheses that are perceived as being supported by *pl* as its result. Each hypothesis name is associated with a subset of point cloud data, which are marked in different colors in the picture on the right.	
match-cad	given *obj-hyp* examine *obj-hyp* object-identity object-pose	*match-cad(obj-hyp, 2D-image)* gets an object hypothesis *obj-hyp* and a 2D color image as its input and performs CAD model matching on the image region that corresponds to *obj-hyp*. The routine returns the object identity of the matching model in the object database and determines the pose of the object. (see [4]).	
match-surf	given *obj-hyp* examine *obj-hyp* object-identity object-pose	*match-surf(pcd-cluster, 2D image)* finds objects in a 2D image using SURF (Speeded Up Robust Feature) features. For each image, we extract the regions of interest representing the objects of interest and compute a vector of SURF features. The next step is to quantize the feature vector into a bag of features using standard K-Means clustering. Then a classification is performed using an SVM classifier with an RBF Laplacian kernel, and the model is used to find the objects in a test phase.	
reconstruct-object	given *obj-hyp* examine *obj-hyp* object-identity surface-of-revolution	The routine *reconstruct-object(pcd-cluster, rotation-axis)* [5] detects surfaces of revolution in point clouds reliably and efficiently. Symmetry assumptions can be hypothesized and verified in order to complete the model from a single view, i.e. to generate data on the occluded parts of the object. These complete models can be used for grasp analysis.	

Fig. 4. Perception routines provided by the CoPMAN Perception Server, their procedure call interface, their functionality and an example result.

information on the region the time at which it was perceived. The identifier can later be used in conjunction with the K-CoPMAN perception server to further examine the cluster, e.g. to categorize/classify the corresponding object.

The example in Figure 5 illustrates the information that is saved for point cloud clusters. Until the object type is determined, K-CoPMAN only knows that it is a *Thing*, the region of interest it was detected in (here: *roi2*), the position of the cluster center, and the corresponding point cloud data.

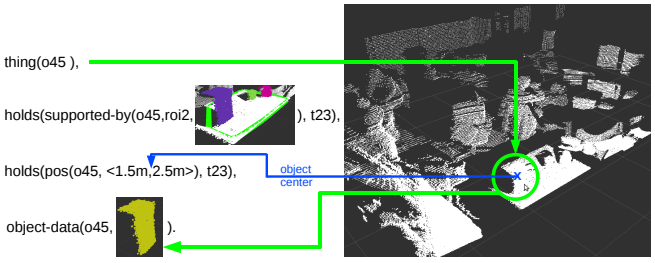


Fig. 5. Information stored in the symbolic knowledge base about a (not yet fully classified) object that was detected on a table.

C. Perceptual Memory

The perceptual memory stores all percepts, making them accessible to future queries. For performance reasons, computations are performed on demand. The passive perception module, for instance, only segments the observed point cloud data and saves the clusters in the memory. Any further processing, such as the classification of the observed objects, is postponed until the information is required for queries involving the respective object identifiers.

IV. Integration with Knowledge Processing

The knowledge processing part in K-CoPMAN is based on KnowRob [2], a relational knowledge processing system especially suited for applications in mobile robotics. KnowRob is specialized in integrating sensor data into the knowledge processing system to perform reasoning on observations from the real world. For this paper, we extended KnowRob with an interface to the K-CoPMAN perception server. This allows for direct reasoning on the perceived objects and their properties, and for applying perception routines to them.

KnowRob is implemented in SWI Prolog and makes use of its support for the extension with custom modules. The robot's knowledge about the world is represented in the Web Ontology Language (OWL) and accessed with Prolog's Semantic Web Library. OWL allows for very structured modeling of the world in terms of classes of objects and instances thereof. For all detected objects, corresponding instances are created; their types are described by classes. KnowRob provides a large amount of encyclopedic knowledge, like descriptions of object classes and their respective properties, as well as common-sense knowledge on, for example, the actions that an object can be used for.

The perception routines described in Section II-B are embedded into Prolog using the foreign language interface (FLI). Prolog predicates are linked to the functions in the perception system and evaluated by calling the corresponding perception routine.

A. Computable Relations

External data can easily be integrated using *computable* relations, which allow to determine whether a relation between object instances holds not only on the static knowledge in the system, but also by querying external data sources. Computables are calculated on demand during the reasoning process. More details on this topic can be found in [2].

In the K-CoPMAN system, computables use attached perception routines to check if a relation holds or not. For instance, the relation *typeOf(Obj, Type)* is evaluated internally by the K-CoPMAN predicate *categorize(Id, Type)*, and the *color* property of an object is determined by a color classification method. In addition to loading data into the system, *computable* relations can also be used to calculate qualitative spatial relations based on the objects' positions, e.g. to determine whether an object is on a table. For these relations, the query is not passed to the perception system, but to a small Prolog program that reads the object positions and dimensions and checks whether the relation holds.

B. K-CoPMAN Predicates

Owing to space limitations, we cannot list all the predicates provided by the K-CoPMAN server. In the following, we list the most relevant ones for the implementation of the *perceivedObjectsOnPlane* predicate:

holds(onPlane(Obj,Plane),ti) is true if *Obj* refers to the raw data of an object hypothesis detected by the perception system when looking at plane *Plane* at time instance *ti*. The predicate is implemented using the perceptual routines *find-hor-planes* and *find-clusters* (see Figure 4).

holds(position(Obj,Pos),ti) is true if *Pos* is the center of mass of the last detection of the object hypothesis *Obj* before *ti*.

holds(spatial-rel(Obj$_1$,Obj$_2$),ti) is true if the object hypotheses Obj_1 and Obj_2 were last detected at the positions Pos_1 and Pos_2, and if these positions satisfy the constraints for *spatial-rel*, e.g. *left-of*. At the moment, we use hard-coded rules to define the spatial relations that depend on the pair of objects at hand but we plan to expand this.

categorize(Obj, Cat) evaluates to true if the point cloud cluster identified by *Obj* can be classified as *Cat* by one of the perception routines in K-CoPMAN. Depending on the perception routine, *Cat* can either be a geometric category, e.g. a cylinder, or an object class like a cup.

C. Probabilistic First-Order Reasoning

In order to cope with non-deterministic domains (e.g. uncertain sensor data), we integrated statistical models, in particular statistical relational models, into our knowledge processing system. By abstracting away from concrete entities and instead representing general principles (of statistical nature) about a domain, statistical relational models represent meta-models for the construction of concrete probability distributions – represented as graphical models – for a given domain of course, i.e. a concrete set of entities that are of interest (see [6]). Specifically, we use Bayesian Logic Networks (BLN) [7], a formalism that combines statistical knowledge (in fragments representing conditional probability distributions) with logical knowledge (sentences in first-order logic). For a given set of entities, a BLN can be instantiated to obtain a ground mixed network [8] or auxiliary Bayesian network that represents a full-joint probability distribution over the relevant propositions about these entities. Given a

model structure and a sufficient amount of relational data – taken directly from our relational knowledge processing system – the parameters of a BLN with given dependency structure can easily be learned, yielding a quantitative representation of statistical dependencies inherent in the data.

To realize our example application, we constructed a model that represents statistical knowledge about table settings. For this model, we used synthetic training data which was generated based on a stochastic process that considered the preferences and habits of six individuals. The model considers the types of meals, the people participating in them (whose preferences the model reflects), the places at which these people sit, the food and drinks they consume as well as the utensils they use to do so. The model's conditional probability fragment structure is shown in Figure 3 (right). Given a partial table setting for one or more persons, the model can be used to infer the probability with which further utensils or food and drinks might be required. Using an appropriately chosen probability threshold, we can thus flexibly perform the task of completing a table setting based on the information we are given.

V. Evaluation

To validate our proposed framework, we performed several experiments on the table scenes depicted in the first row of Figure 6. As we will show, the integrated system can help a robot system make the decisions required for competent operation in the presence of uncertainty. In addition to the example of inferring missing objects, we will present further queries showing the advantages of integrating perception, knowledge processing and probabilistic reasoning.

Scenes 1-3 in Figure 6 show incomplete breakfast settings, whereas scenes 4 and 5 are incomplete lunch settings. The task is to infer which items need to be added to complete the setup. The first row shows the incomplete setup and the lists of objects they involve. In the second row, the table surfaces and clusters identified in the point cloud data are drawn. The clusters were projected onto 2D images and classified with the *match-surf* routine (third row). In the remaining, unoccupied parts of the images, we searched for further objects using combinations of our perception routines.

The set of perceived objects was read into the *KnowRob* system and passed as evidence to the probabilistic reasoning engine which, based on the model described in the previous subsection, infers the table setting that is most likely to be desired. The bottom row visualizes the perceived objects (visualized on the table) and inferred objects (visualized off the table) as instantiated in *KnowRob*. The hue indicates probability: Red corresponds to 1.0, with orange, yellow, green and blue denoting declining probabilities in this order.

As an example query, consider the fourth scene in Figure 6. In terms of the functions and predicates the model considers, the query for potentially missing entities translates to a probabilistic query as follows,

$$
\begin{aligned}
&P(usesAnyIn(P, ?u, M), consumesAnyIn(P, ?f, M) \mid mealT(M) = Lunch \wedge \\
&\quad usesAnyIn(P, Plate, M) \wedge usesAnyIn(P, Knife, M) \wedge \\
&\quad usesAnyIn(P, Fork, M) \wedge usesAnyIn(P, Spoon, M) \wedge \\
&\quad usesAnyIn(P, Napkin, M) \wedge consumesAnyIn(P, Salad, M) \wedge \\
&\quad consumesAnyIn(P, Pizza, M) \wedge consumesAnyIn(P, Juice, M) \wedge \\
&\quad consumesAnyIn(P, Water, M) \wedge takesPartIn(P, M)) \\
&\approx \langle\langle Glass: 1.00, Bowl: 0.85, Cup: 0.51, \ldots\rangle, \\
&\quad \langle Soup: 0.82, Coffee: 0.41, Tea: 0.14, \ldots\rangle\rangle
\end{aligned}
$$

where P is some person participating in the meal M, who is assumed to be using/consuming the objects that were detected, and we ask for the probabilities of corresponding *usesAnyIn* and *consumesAnyIn* atoms. The results above (listed in order of probability) are certainly sensible, given that the presence of a spoon generally implies that something like soup is likely to be consumed, and therefore that a bowl/soup plate is likely to be required. Also, a glass is necessary for the drinks to be consumed.

Further applications of the system, beyond inferring missing objects, are the recognition of an activity/meal based on the objects that were perceived, the detection of misplaced objects (by applying the predicates for computing spatial relations on the perceived objects), and even the identification of potentially superfluous objects (i.e. objects that have a low probability given the other objects).

Since the detected objects are formally represented in the knowledge base, queries can combine object information with background knowledge that describes, for example, their main functionality. For instance, the following query searches for objects that can be used to cut food and that are lying on the table:

```
?- type(Obj, ObjType),
   subClassOf(ObjType, 'KitchenUtensil'),
   onPlane(Obj, T),
   type(Obj, 'Table'),
   primaryFunction(ObjType, 'CuttingFood').
Obj=knife1
```

VI. Related Work

As this paper is situated in the intersection of robotics and artificial intelligence, related work concerns both fields of research. The Shakey robot [9] was one of the first systems to represent perceived world states as first-order knowledge bases and to use this knowledge for decision-making and planning. Unfortunately, subsequent research diverged into two directions investigated by separate communities: (1) object perception and scene understanding and (2) planning and decision making. Perception for autonomous robots largely concentrated on methods and algorithms for object recognition [3], [10], [4], [5], [11], (see also Figure 4). The field of planning and reasoning (with rare exceptions) pursued their research goals under the assumption that representations of the state of the environment are available as a first-order — sometimes probabilistic — knowledge base.

Horswill et al [12] were the first to couple a perception system with Prolog, by grounding the predicates in visual routines of Ullman's visual routines theory. However, their focus was on a real-time implementation of that theory rather than on fostering a competent robot manipulation system.

With regard to the field of knowledge representation, the research in this paper is most closely related to the work on knowledge preconditions [13] that tries to formalize and reason about what agents need to know in order to carry out actions successfully. Powerful scene representations have been investigated for vision-based scene perception.

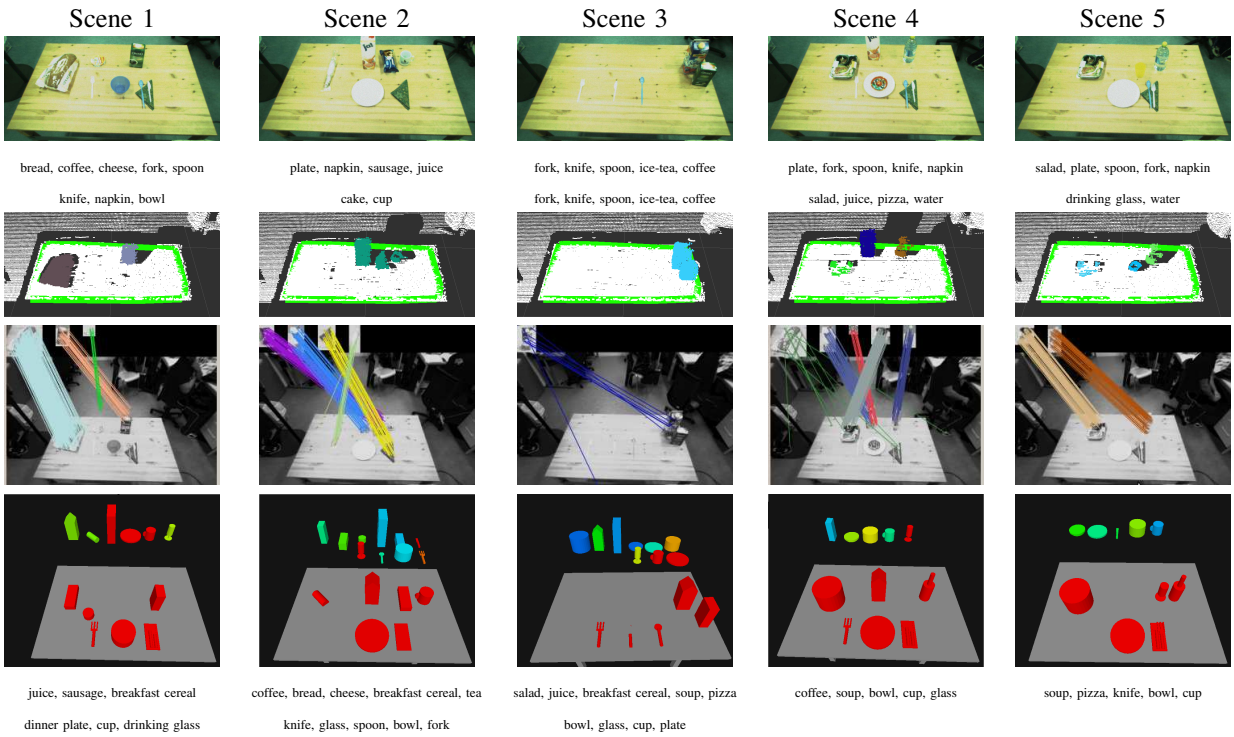


Fig. 6. Evaluation results. **1st row:** Snapshots of test scenes; **2nd row:** object hypotheses; **3rd row:** detection of objects using *match-surf* routine; **4th row:** results of probabilistic inference for *missingObjects query*. Below enlisted objects correspond to the inferred ones (visualized off the table) in left-to-right rear-to-front order.

Neumann et al [14], [15] thus describe a concept of aggregates that are composed of multiple parts and constrained primarily by temporal and spatial relations. The aggregates are represented in an $\mathcal{ALCF}(\mathcal{D})$ Description Logic, and their probabilistic dependencies are captured by Bayesian compositional hierarchies.

Our research is probably best characterized as a specialization of cognitive vision for autonomous robots, which aims at integrating representation, reasoning, learning, and planning mechanisms into vision systems.

VII. Conclusion and Future Work

We presented K-CoPMan, a system that integrates novel perception routines and knowledge processing mechanisms for autonomous robot manipulation. The system abstracts perceptual facts from the real world, utilises symbolic knowledge to boost up perceptual capabilities, and blends in the combination of both in order to answer complex queries such as *what items are missing on the table for a meal*. We verified our approach by showing several queries that demonstrate how the system can contribute to informed decision making.

Including more and more specific perception routines, investigating spatio-temporal reasoning, life-long learning using the passive perception component, and more efficient processing of perceptual data are just some of the items on our future work agenda.

Acknowledgments: This work is supported by excellence cluster CoTeSys and by Willow Garage, Menlo Park, CA.

References

[1] R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz, "Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments," in *IROS*, 2009.
[2] M. Tenorth and M. Beetz, "KnowRob — Knowledge Processing for Autonomous Personal Robots," in *IROS*, 2009.
[3] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz, "Real-time cad model matching for mobile manipulation and grasping," in *Humanoids*, 2009.
[4] M. Ulrich, C. Wiedemann, and C. Steger, "Cad-based recognition of 3d objects in monocular images," in *ICRA*, 2009, pp. 1191–1198.
[5] N. Blodow, R. B. Rusu, Z. C. Marton, and M. Beetz, "Partial View Modeling and Validation in 3D Laser Scans for Grasping," in *Humanoids*, 2009.
[6] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.
[7] D. Jain, S. Waldherr, and M. Beetz, "Bayesian Logic Networks," IAS Group, Fakultät für Informatik, Technische Universität München, Tech. Rep., 2009.
[8] R. Mateescu and R. Dechter, "Mixed deterministic and probabilistic networks," *Annals of Mathematics and Artificial Intelligence*, 2008.
[9] N. J. Nilsson, "Shakey the Robot," AI Center, SRI International, Tech. Rep. 323, 1984.
[10] A. C. Romea, D. Berenson, S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *ICRA*, 2009.
[11] J. Bruce, T. Balch, and M. M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *IROS*, 2000, pp. 2061 – 2066.
[12] I. Horswill, "Integrating vision and natural language without central models," in *In Proc. of the AAAI Fall Symposium on Embodied Language and Action*, 1995.
[13] R. C. Moore, *Reasoning from Incomplete Knowledge in a Procedural Deduction System*, ser. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1975.
[14] B. Neumann and R. Möller, "On Scene Interpretation with Description Logics," in *Cognitive Vision Systems: Samping the Spectrum of Approaches*, ser. LNCS, H. Christensen and H.-H. Nagel, Eds. Springer, 2006, no. 3948, pp. 247–278.
[15] B. Neumann, "Bayesian Compositional Hierarchies - a Probabilistic Structure for Scene Interpretation," in *Logic and Probability for Scene Interpretation*, ser. Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2008.