Technical Report 75

ROBOSHERLOCK: Unstructured Information Processing for Robot Perception

Michael Beetz[1],
Ferenc Balint-Benczedi[1],
Nico Blodow[2],
Daniel Nyga[1],
Thiemo Wiedemeyer[1],
Zoltan-Csaba Marton[3]

[1]TZI-Universität Bremen
[2]TU-München
[3]Deutsches Zentrum für Luft und Raumfahrt

TZI-Bericht Nr. 75
2014

TZI-Berichte

Herausgeber:
Technologie-Zentrum Informatik und Informationstechnik
Universität Bremen
Am Fallturm 1
28359 Bremen
Telefon: +49 421 218 94090
Fax: +49 421 218 94095
E-Mail: hq@tzi.de
http://www.tzi.de

ISSN 1613-3773

ROBOSHERLOCK: Unstructured Information Processing for Robot Perception

Michael Beetz[1], Ferenc Bálint-Benczédi[1], Nico Blodow[2], Daniel Nyga[1],
Thiemo Wiedemeyer[1], Zoltán-Csaba Márton[3]
{beetz, balintbe, nyga,wiedemeyer}@cs.uni-bremen.de, blodow@cs.tum.de, zoltan.marton@dlr.de

***Abstract*— We present ROBOSHERLOCK, an open source software framework for implementing perception systems for robots performing human-scale everyday manipulation tasks. In ROBOSHERLOCK, perception and interpretation of realistic scenes is formulated as an unstructured information management (UIM) problem. The application of the UIM principle supports the implementation of perception systems that can answer task-relevant queries about objects in a scene, boost object recognition performance by combining the strengths of multiple perception algorithms, support knowledge-enabled reasoning about objects and enable automatic and knowledge-driven generation of processing pipelines. We demonstrate the potential of the proposed framework by three feasibility studies of systems for real-world scene perception that have been built on top of ROBOSHERLOCK.**

I. INTRODUCTION

Robots for mobile manipulation of objects of daily use need to be equipped with powerful perception capabilities. A robot acting in a human household environment, for example, needs to effectively and robustly perceive objects like cups, plates, cooking pots, glasses, silverware pieces, packagings of groceries, tools and utensils, electrical devices and many more. All of these objects exhibit very different perceptual characteristics, such as texture, monochromy, shininess, dullness, shape, translucence, size or text – to name only a few – or combinations of all of those. In addition, robots are equipped with different kinds of sensors like RGB-D cameras, monocular cameras or laser scanners and existing perception routines are unequally well-suited to be applicable to different kinds of sensory input. Recent research in robot perception, however, has focused on investigating subsets of those perception problems, and specialized perception algorithms have been developed and tested in rather homogeneous and controlled environments in order to satisfy the presumptions of the respective algorithm. Despite work in individual subfields of robot perception has made tremendous progress in recent years by developing perception algorithms that handle particular subsets of the above-mentioned problems with high accuracy, the 'omni-potential' perception routine still remains undeveloped and proficiently perceiving wide varieties of heterogeneous objects in real-world robotics applications remains challenging.

There is a huge gap between the perceptual capabilities that robots performing human-scale manipulation tasks require and the functionality today's perception algorithms provide robots with. These gaps exist in multiple dimensions:
1) ***There exists no single comprehensive perception algorithm*** that shows promise in handling all of the different perceptual characteristics of objects and the diversity of tasks that a robot in human environments has to perform. As an example, consider a robot that is to fill a drinking mug with orange juice from a bottle. The bottle typically is translucent, but it also has a textured label with a brand logo and text stating that it contains orange juice, whereas the drinking mug can be recognized mainly by its shape. As another example, consider kitchen utensils, pieces of cutlery or electrical devices, which consist of shiny metal parts and parts made of plastic which are dull. Although there exist sophisticated algorithms for dealing with particular subsets of these perceptual characteristics, proficiently and robustly handling *all* of them still goes beyond the capabilities of today's perception methods.
2) ***Robot perception must go beyond object categorization***. For perceiving objects of daily use with the purpose of competently manipulating them, it is insufficient for a robot to merely assign a class label to a particular region of interest in the sensor data, but a wide breadth of purposeful perceptual functionality is required. The robot has to detect, localize, categorize, and reconstruct the objects it is seeing and to decompose them into their functional parts. Consider a robot that is to make pancakes, for instance. Pouring the batter into the pan requires the robot to identify and localize the opening of the bottle containing the batter, detect whether or not it is closed by a cap, unscrew the cap if necessary, and localize the center of the pan, where the batter needs to be poured. For flipping the pancake, the robot has to identify the handle of the spatula as well as its blade for grasping it correctly and moving it under the center of the pancake. Thus, robot perception systems need to be able to answer queries about particular properties of objects that must be known for successfully performing a task. This makes perception for robots highly *task-dependent* and *knowledge-intensive*.

In this work, we present ROBOSHERLOCK, an opensource framework for building perception systems for robotic agents that are to perform human-scale manipulation tasks. ROBOSHERLOCK enables programmers to combine perception, representation, and reasoning methods in order to scale the perception capabilities of robots towards the needs implied by general manipulation tasks. ROBOSHERLOCK supports the implementation of perception systems that

[1] Institute for Artificial Intelligence and the TZI Center for Computing Technologies, University of Bremen, Germany [2] Intelligent Autonomous Systems Group, Technische Univerität München [3] Institute of Robotics and Mechatronics German Aerospace Center (DLR), Germany

- ... ***can be equipped with ensembles of expert perception algorithms*** with complementary, similar or overlapping functionality instead of relying on one particular perception algorithm. ROBOSHERLOCK provides control mechanisms and data structures to direct the algorithms to synergetically cooperate, to communicate relevant information, fuse their results and hence to combine the strengths of individual methods.
- ... ***can be tasked***. The robot control program can request the perception system to detect objects that satisfy a given description, ask it to examine aspects of the detected objects such as their 3D form, pose, state, etc. In this view, perception can be viewed as a question answering system that answers the queries of the robot's control system based on perceived scenes.
- ... ***can enhance perception with knowledge and reasoning.*** In ROBOSHERLOCK the robot can reason about the objects to be detected and examined and the respective task and environment context to make the perceptual processes faster and more robust. Knowledge processing also helps the system to interpret the results returned by the perception algorithms and thereby increase the set of perceptual tasks that can be accomplished.

ROBOSHERLOCK has been designed with two major implementational aspects in mind: (1) it does not replace any existing perception system or algorithm but rather enables easy integration of previous work, in a unifying framework that allows these systems to synergistically work together and (2) new methods can be easily wrapped into ROBOSHERLOCK processing modules to extend and improve existing functionality and performance.

In order to provide these services and scale towards realistic sets of objects, environments, and perceptual tasks, ROBOSHERLOCK considers perception as *content analytics in unstructured data*. Content analytics (CA) denotes the discipline of applying methods from the field of statistical data analysis to large amounts of data in order to extract semantically meaningful knowledge from those. The data are considered *unstructured* since they lack explicit semantics and structure. The paradigm of *unstructured information management* (UIM) offers an implementational framework for realizing high-performance CA systems. The perhaps most prominent example of a UIM system is *Watson* [1], a question answering system that has won the US quiz show *jeopardy!*, competing against the champions of the show and demonstrating an unprecedented breadth of knowledge that has been acquired from automatic analysis of documents, and the crucial ability to correctly judge it's own competence in answering questions. In UIM, pieces of unstructured data, such as web pages, text documents or images are processed by a collection of specialized information extraction algorithms (*annotators*), and each algorithm contributes pieces of knowledge with respect to its expertise. Thereby, outputs of different algorithms are allowed to be complementary, overlapping or even contradictory. Hence subsequently, the collected annotations are rated and consolidated to come to a consistent final decision.

The main contribution of this paper is the application of the concepts of UIM analytics to the domain of robot perception. We explain and show how these principles can be used to design and realize robot perception systems that are taskable and can scale towards broader ranges of perception tasks and improve robustness and performance by exploiting ensembles of possibly knowledge-enabled perception experts. We present the capabilities of the proposed framework in a kitchen scenario with objects of daily use, and demonstrate its applicability and transferability in a chemical experiment setting.

The remainder of the paper is organized as follows. Section II presents our vision on how perception tasks should be formulated, followed by a description of our conceptual framework in Section III. Section IV describes the components and processing modules currently implemented in ROBOSHERLOCK, which have been used to demonstrate its capabilities in Section V. Related and future work are discussed in Sections VI and VII.

II. ROBOSHERLOCK OVERVIEW

In robot perception, raw sensor data (of arbitrary kinds) can be regarded as unstructured documents. ROBOSHERLOCK creates object hypotheses for pieces of sensor data that it believes to represent objects or object groups. Subsequent perception algorithms (specialized routines called *experts*) analyze these hypotheses and annotate their results as semantic meta data to the hypotheses with respect to their expertise. Further algorithms then test and rank possible answers to the given perceptual task based on the combination of sensor and meta data. To this end, the framework supports the application of different algorithms to different objects with different properties and combine the answers. It does so by enabling perceptual capabilities to reason about which method to apply to which perceptual subtask, by providing means for the communication among expert methods, and by providing infrastructure that supports reasoning about how results of different experts should be combined. A key aspect of the system is the seamless integration of knowledge processing into the perception processes. In the following, we will explain its key concepts in more detail.

A robot uses ROBOSHERLOCK to perceive the information that it needs in order to accomplish its manipulation tasks. The robot can formulate the perception tasks and issue them to ROBOSHERLOCK in the following two ways:

1) **detect *obj-descr*** asks ROBOSHERLOCK to detect objects in the sensor data that satisfy the description *obj-descr* and return the detected matching *object hypotheses*. In this command format the robot can describe a red spoon that it is looking for as *(an object (category spoon) (color red))*. ROBOSHERLOCK returns a failure if it cannot detect a matching object.
2) **examine *obj-hyp attributes*** asks the perception system to examine a given hypothesis *obj-hyp* in order to extract additional information as requested by *attributes* and add the information to *obj-hyp*. The examination

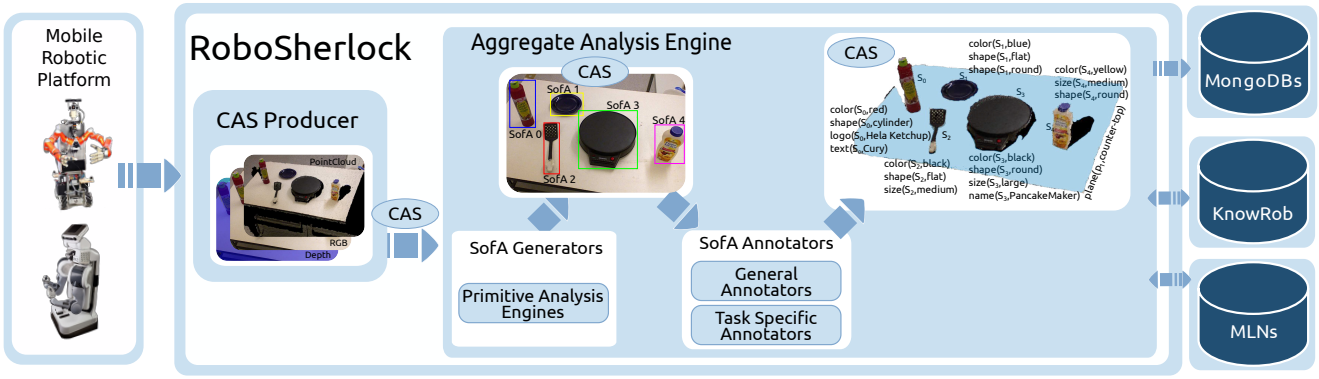

Fig. 1: Example of a pipeline execution in RoboSherlock.

of object hypotheses enables the robot to perceive information such as the exact pose of the object, its 3D model, or the parts of object which are needed for competent object manipulation but not yet included in the object hypotheses.

Let us consider three example object detection tasks that ROBOSHERLOCK can perform. The first one is to look for something that can hold 1 liter of water:

```
(detect (an object
          (category container)
          (capacity (≥ 1 liter))))
```

This perception task is challenging because 'container' is an abstract category that subsumes a broad variety of categories that vary widely in their visual appearance. In addition, the capacity of an object is not directly observable but must be inferred from the dimensions of the object. Performing this perception task requires ROBOSHERLOCK to combine perception with knowledge and reasoning.

The second perception task is to find a Kellog's conflakes box on the table which can be stated as:

```
(detect (an object
          (form cuboid)
          (logo "Kellogg's") (text "cornflakes")
          (location (a location (on (an object (category table)))))
```

Here ROBOSHERLOCK has to employ different perception methods, some for determining the shape of objects and others for recognizing logos and reading the text. Alternatively, if the robot already knows the Kellogg's cornflakes box beforehand it can learn the visual appearance of the object and recognize it using learned visual features. While the application of multiple methods can increase the robustness of recognition methods it also introduces complications such as handling the cases in which the results are inconsistent.

The third perception task is to look for the lid of the tube that contains acid.

```
(detect (an object-part
          (category lid)
          (part-of (an object (category tube)
                    (contains (some stuff (category acid)))))))
```

To accomplish this perception task ROBOSHERLOCK has to detect a specific part of a previously detected object.

ROBOSHERLOCK is designed to accomplish such perception tasks in complex scenes that include objects with different perceptual characteristics. To do so, ROBOSHERLOCK perceives objects taking the scene context into account and employs different perception methods and reasons about which methods to apply to which objects. ROBOSHERLOCK can use background knowledge to simplify perception tasks, for example when the object it is looking for has a salient distinctive characteristic, or in order to interpret the perception results by inferring whether or not the object could hold a liter of stuff. In order to improve robustness ROBOSHERLOCK can also reason about how to combine the results of different perception methods.

A. *High-level View on* ROBOSHERLOCK

Figure 1 shows the high-level organization of ROBOSHERLOCK. Performing perception tasks starts with the robot capturing images using different cameras (high-resolution RGB, RGB-D, stereo cameras, or thermo camera) and other sensors. The images together with data structures that combine the images with meta data constitute the *Common Analysis Structure (CAS)*, the main data structure that ROBOSHERLOCK and its components are working on.

After the initial CAS is created, ROBOSHERLOCK runs object hypothesis generators on the images in order to detect data pieces in the images that might correspond to objects and object groups in the environment. In Figure 1 the object hypothesis generators have detected raw point clusters that are indicated by the colored bounding boxes. For each hypothesis a designated data structure called SOFA (Subject of Analysis) is created and assigned a systemwide unique name. SOFAs collect and organize all pieces of information that different ROBOSHERLOCK components infer about the respective object hypothesis.

After the SOFAs of the CAS are created, other ROBOSHERLOCK modules, called SOFA annotators, run on the detected SOFAs, interpret the respective hypotheses, apply vision algorithms to the corresponding image pieces, interpret results, etc. The results of these interpretation processes are then asserted as logical annotations to the SOFAs. For example, a SOFA annotator might run a color classifier on

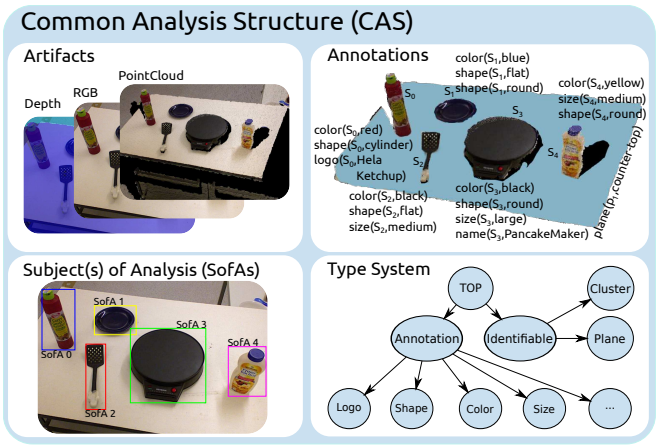

Fig. 2: CAS of a kitchen table scene.

the image piece corresponding to the SOFA s and store the recognized color c as an assertion *color(s,c)* in the annotations of s.

The SOFA annotators of ROBOSHERLOCK fall into two categories. First, general-purpose ones that run on every detected object hypothesis. Second, perception task-specific SOFA annotators that perform computations and image interpretation methods that contribute to the accomplishment of the given perception task.

Running general-purpose SOFA annotators, including color classifiers, object size estimators, shape estimators, logo detectors, text readers, etc. results in a small set of facts for each SOFA. In the CAS (step 2) in Figure 1 the annotations of SOFA s_0 (the ketchup bottle) include the following assertions: *color(s_0,red)*, *shape(s_0,cylinder)*, *text(s_0,curry)*, and *logo(s_0,Hela Ketchup)*. ROBOSHERLOCK can make use of these logical assertions for selecting appropriate perception methods, for equipping interpretation methods with background knowledge, and for interpreting the results of perception algorithms in context.

Task-specific SOFA annotators infer specific information pieces about SOFAs. Typically, the robot does not care about the volume of the objects it sees unless it is looking for objects with volume constraints. In our examples above we had the perception task to look for a container that can hold 1 liter. This task triggers task-specific SOFA annotators that are capable of checking whether observed objects are containers and that can estimate the capacity of objects based on their estimated sizes. Testing these properties is done through the application of knowledge processing and reasoning techniques which will be detailed in Section III-C.

III. CONCEPTUAL FRAMEWORK

The key concepts that ROBOSHERLOCK builds upon are

- the *common analysis structure (CAS)* with the SOFAs to be analyzed, which is the common data structure and knowledge base of the ROBOSHERLOCK system;
- the *analysis engines (AEs)* (SOFA generators and the SOFA annotators) share and operate on the CAS by generating, interpreting, and refining SOFAs; and
- the *knowledge representation and processing mechanisms* that the analysis engines can use as resources for reasoning about the objects and scenes they interpret.

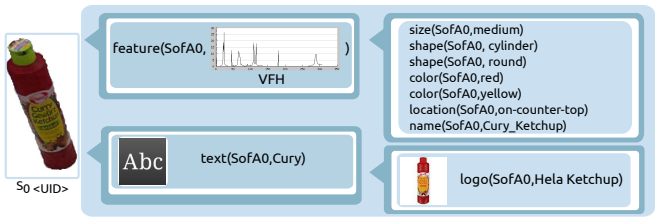

Fig. 3: SOFA of a ketchup bottle with annotations

The components work together according to the data analytics information processing paradigm, which we have discussed in our introduction. In this section we will describe and explain these concepts in greater detail.

A. Common Analysis Structure (CAS)

The CAS is the data structure that is used by the components of the ROBOSHERLOCK-based perception systems in order to communicate with each others. A CAS contains:

- ***images*** the sensor data that are interpreted (e.g. RGB or depth image);
- **SOFA*s*** (subjects of analysis), which name data pieces in the image that the perception system assumes to have structure and can be assigned semantic meaning;
- ***Annotations*** (meta data), which represent information about SOFAs that interpretation algorithms have inferred; and
- a common ***type system*** that ensures that the different components of the perception system share a common vocabulary

Figure 2 depicts a CAS of a scene on a table as it is produced by one of our ROBOSHERLOCK-based perception systems. The SOFAs of the CAS are the individual point clusters and parts thereof. The CAS also contains annotations for each SOFA, which are represented as symbolic facts and information about the underlying type system.

SOFAs are data pieces of the artifact for which the perception system has generated the hypothesis that these data pieces have some deeper structure and can be assigned some semantic meaning. For example, for point clusters on a supporting surface perception systems might want to generate hypotheses that these clusters correspond to objects or object groups in the environment.

Figure 3 shows a SOFA that depicts a bottle, the SOFA includes component SOFAs, namely the lid and the body of the bottle.

In ROBOSHERLOCK the annotations for SOFAs are a set of logical assertions of the form $\langle$*attr-name*$\rangle$($\langle$*sofa-id*$\rangle$, $\langle$*attr-val*$\rangle$), where $\langle$sofa-id$\rangle$ is the identifier of a SOFA, $\langle$attr-name$\rangle$ is an attribute of this SOFA and $\langle$attr-val$\rangle$ the value for this attribute. For example, the annotation *shape(S_0,cylinder)* asserts that SOFA S_0 has the shape *cylinder*. Figure 3 shows the annotations for

the different object hypotheses (SOFAs) $S_{\langle UID \rangle}$ which constitute the results of different perception AEs. *UID*s are globally unique identifiers to ensure inconsistencies in the knowledge base containing all SOFAs. Annotations can be semanctical (size, shape, color etc.) or numerical (eg, the VFH [2] shape descriptor). Numerical annotations often serve as the basis for image interpretation (e.g. classification), which in turn might result in semantic labels.

B. Analysis Engines and Annotators

Analysis Engines (AEs) are the core processing elements of ROBOSHERLOCK. AEs share and operate on the cas by generating, interpreting and refineing SOFAs. AE can be considered as being primitive or aggregate analysis engine. Primitive AEs can be categorized based on their capabilities into two kinds: The first kind, called *hypotheses generators* analyzes artifacts and generates SOFAs, and the second one, called *annotators* annotates the SOFAs. *Hypotheses generators* are to find regions in the raw data that might be of interest for the task currently being executed.

Annotators further investigate SOFAs by generating new and revising old annotations. To this end, they take the part of the artifact that corresponds to the SOFA, transformations and interpretations of it as their input. To perform their computations in more informed manners they have also access to the annotations of this and all other SOFAs.

Because annotators might employ heuristic interpretation methods or are more or less reliable and accurate, the set of annotations is allowed to be inconsistent or contradictory. Inconsistencies and erroneous annotations can be handled by subsequent reasoning and hypothesis testing and ranking. ROBOSHERLOCK maintains book keeping information to facilitate the informed resolution between conflicting annotations in later processing stages. This book keeping information includes justifications of annotations in terms of the name of the AE that generated it and confidence values that some AEs return. A particularly powerful method for resolving inconsistencies in ROBOSHERLOCK is the application of first-order probabilistic reasoning described in Section III-E.

Aggregate analysis engines solve more complex tasks. They are a combination of primitive analysis engines thus creating perception pipelines. Aggregate AEs can be run sequentially or in parallel in order to annotate the data, at the end of each execution cycle all information from the CAS being stored in a data base. A more detailed explanation about the AEs used in ROBOSHERLOCK to solve perception problems will be given in Chapter IV.

Figure 1 shows an example perception pipeline where the resulting CAS is passed to an aggregate AE that consists of (1) primitive AEs that generate object and object group hypotheses and represent them as SOFAs and (2) annotators that interpret and analyze these SOFAs in order to add informative and semantic meta data to the SOFAs.

C. Knowledge Processing

ROBOSHERLOCK uses annotations in form of logical atoms in order to be able to ask structured queries and perform logical reasoning about annotations added by the analysis engines. The main reasoning engine used in ROBOSHERLOCK is based on SWI-Prolog, a general purpose logic programming language providing bindings for the most common procedural languages like C++, Java. AEs can be equipped with Prolog programs, which consists of facts and rules. Relevant annotations are automatically asserted as facts into the Prolog knowledge base.

Semantic annotations make use of the terminology of concepts and predicates of the KNOWROB knowledge base [3], a comprehensive knowledge processing system for autonomous robots. The use of KNOWROB concepts and predicates ensures that ROBOSHERLOCK can make use of a large body of knowledge about objects and relations between them. This knowledge is valuable for informing perception routines and specialize perception tasks in context-specific ways as well as for interpreting the results of expert routines.

Knowledge and reasoning are key valuable resources for scaling the perceptual capabilities of robots. They enable the robot to specialize the perception tasks it has to solve in order to resolve ambiguities in the perception data, make perception methods more robust and efficient by exploiting the information that is already available, and reason about the capabilities of the AEs.

In ROBOSHERLOCK, we can, for example, state that an object is a container if it is annotated with a category that is a specialization of the concept 'container' as:

category(Obj,'container') :-
category(Obj,Cat), subclass(Cat,'container').

We can also equip the AE with a rule that allows it to infer the volume of an object:

volume(Obj,Vol) :-
category(Obj,'container'), geom-primitive(Obj,'cylinder'),
radius(Obj,Radius), height(Obj,Height),
*Vol = pi * Radius * Radius * Height.*

Using the Prolog rules above an AE can detect containers in a given scene that can hold at least 1 liter by issuing a query of the form

object(Obj), category(Obj,container), volume(Obj,Vol), Vol $\geq$ *1.*

The execution of a Prolog program is triggered by the query of the AE. Logically, the Prolog engine tries to find a resolution refutation of the negated query. If the negated query can be refuted, it follows that the query, with the appropriate variable bindings in place, is a logical consequence of the program. In that case, all generated variable bindings are reported to AE as an answer.

D. Reasoning about and composition of perception pipelines

Another important application of knowledge processing in ROBOSHERLOCK is the context-directed application of AEs. For example, we might formulate a set of rules that state which kinds of objects a particular method applies to with competence. We can do this by specifying rules such as

applicable('Moped',SOFA) :-
degreeOfTexturedness(SOFA,Degree), Degree $\geq$ *0.6,*
distance(SOFA,robot-camera,Dist), Dist $\leq$ *1.5 m,*

which specifies that the *'Moped'*-AE is applicable if the object has a certain minimum texturedness level and the distance from which the object is perceived is not too high.

Although at the moment rules are only defined for a limited set of AEs, the system can already deduce whether or not particular annotators are needed for the perception task to be successfully executed. This is the case for the 'goggles' annotator AE. Because of its slow response rate ($\sim$1s per query), we only execute the goggles annotator if the perception task explicitly asks for results that at the moment we could only receive from this particular module. Once every AE has its own rule defined, the system will be able to automatically pipelines based on the task description.

E. Probabilistic Knowledge Representation and Reasoning

One of the key features of ROBOSHERLOCK is its ability to benefit from the combination of multiple specialized perception algorithms and systems, which are integrated in form of AEs wrapped around these methods. Since AEs are being applied to SOFAs independently of each others, their outputs may be complementary, overlapping or even contradictory. It is therefore important to have means available to combine annotations of different experts in a meaningful way in order to come up with a final consistent ensemble result. ROBOSHERLOCK has an integrated engine for learning of and reasoning about probabilistic first-order knowledge bases, which we use for consolidation of inconsistent annotations. In [4], we have shown how first-order probabilistic models, such as Markov logic networks (MLNs), can be used to combine the strengths of different perception algorithms and the object recognition performance of perception systems can be significantly boosted.

IV. IMPLEMENTATION

Percepetion algortihms in ROBOSHERLOCK are implemented as primitive analysis engines, where each AE can assert facts into the knowledge base and reason about its own capabilities. In Figure 4 we present some of the commonly used AEs that are implemented, and their respective logical assertions. The majority of these AEs wrap around perception algorithms from PCL [5] or OpenCV [6] that are commonly used in robotics, most of which can be assigned to one of the two categories of AEs that we described earlier.

A. Hypothesis Generation

Object hypothesis generators are tasked with generating SOFAs. In most cases these are specialized segmentation algorithms in order to find possible locations of items, and deal with objects that exhibit different perceptual characteristics such as ordinary objects of daily use, flat objects such as pieces of paper or small shiny objects such as knives and forks.

These methods are complemented with mechanisms to combine their results in a consistent manner including the following ones:

- attention mechanisms that detect points of interest in pixel coordinates in order to create regions of interest (point and extents) in the camera's frame.

AE Category	AE Name	Logical assertions
SOFA-Generators	plane-detector	*plane(sofa)*
	flat-object-detctor	*cluster(sofa)*
	pcd-cluster-detector	*cluster(sofa)*
	MOPED[7]	*cluster(sofa)*
High-frequency AEs	color	*color(sofa,col)*
	size	*size(sofa,s)*
	location	*location(sofa,s)*
	primitive-shape[8]	*shape(sofa,s)*
Low-frequency and event-driven AEs	goggles	*text(sofa,txt)* *logo(sofa,lg)*
Object recognition	LINE-MOD[9]	*name(sofa,id)*
Object categorization	MOPED	*category(sofa,cat)*
	MLN	*category(sofa,cat)*

Fig. 4: Some of ROBOSHERLOCK's Analysis Engines.

- image segmentation algorithms (e.g. color-based) can generate masks or region maps, referencing the respective part of the image.
- point cloud segmentation relying on supporting planar structures can generate index vectors.

Note that most of these types can be converted between each other relatively trivially, e.g. projecting a point cluster from a point cloud into a camera image, or transforming an image region to a grasping pose in robot-local coordinates. In a well-integrated platform such as the PR2, this can even be done at a later point in time from another vantage point to some extent. This allows us to retrieve the camera image region of interest corresponding to a 3D point cluster, enabling the combination of image analysis annotations and point cloud processing. The AEs that are most commonly used in ROBOSHERLOCK are RANSAC based plane segmentation to find the supporting plane that objects can lie on, Euclidean clustering in 3D space to find 3D clusters and a color based segmentation in order to find flat objects. Although these AEs implement very simple examples of segmentation we will show how they complement each other and come up with better object hypotheses. Having better segmentation algorithms (e.g. ones that deal with cluttered scenes) would furthermore improve the capabilities of the system.

B. Object Annotators

Object annotators are the subclass of AEs that interpret SOFAs and generate annotations. In general, annotators wrap existing perception algorithms and result in numerical or symbolic values that are linked to the SOFA.

As we described in Section II-A annotators can be either general-purpose ones or task-specific. The AEs shown in Figure 4 are of the former kind. This is because task-specific annotators are domain specific, and do not wrap around any existing algorithm. An example of a task-specific annotator would be one that finds the center of a pan, since this is where the robot would like to pour the batter, or deduces which container can hold a certain amount of liquid.

General-purpose annotators can further categorized based on the type of perception algorithms they implement. For instance, the location annotator uses a 3D semantic map of our environment [10] to annotate an object cluster with

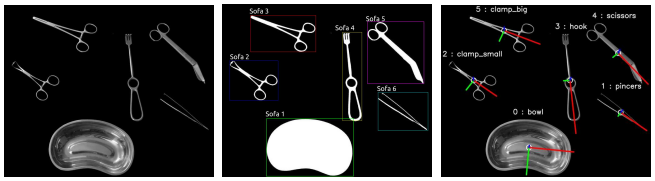

Fig. 5: Surgical utensils on a table: GreyScale *(left)*, SOFAs *(center)* and annotations *(right)*

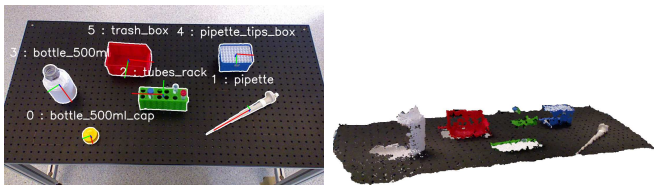

Fig. 6: Pipetting scene as seen by the robot. *Left: RGB, Right: PointCloud*

a semantically meaningful object location (e.g. "on top of counter_top"). The PCL Feature extractor can process any point cluster (with estimated normals) and compute any open source 3D feature implemented in PCL (VFH, CVFH, FPFH, Spin images, RIFT, SHOT etc). Color, size or primitive shape annotators compute symbolical values for the SOFAs they are processing. All of the aforementioned AEs, we consider as high-frequency AEs, purely because of their execution times (order of milliseconds).

On the other there are annotators like Google Goggles, wrapping around a web service and smart phone app allowing the analysis of an image. Google goggles generates a highly structured list of matches including product descriptions, bar codes, logo/brand recognition, OCR text recognition or a list of similar images, but being a web service, response times for analyzing all SOFAs is in the order of seconds. That is why we consider them low-frequency annotators.

Some annotators wrap around existing perception frameworks, namely BLORT[], Moped and Linemod. These are object recognition or categorization engines, and are worth executing, when the task involves an object that at least one of them was trained on.

V. EXPERIMENTS

Since the contributions are neither individual algorithms nor a monolithic system, but a framework, and since it covers a considerably wider scope than previous work, it is hard to quantitatively assess the quality of the proposed approach. It is also hard to compare it to existing perception systems used in robotics, since it builds on top of these systems, and offers developers the possibility to wrap their framework in ROBOSHERLOCK.

We therefore showcase the capabilities of ROBOSHERLOCK on three tasks of a robotic agent performing different experiments: (1) pick and place of surgical utensils (2) a robot performing pipetting in a chemical laboratory (3) reasoning about objects and their properties.

Execution of each task is shown in the video accompanying our paper.

A. Task 1: Pick and Place Surgical Utensils

The first task involves a robotic agent picking up surgical tools, and putting them in a bowl. This task is solved by a single aggregate analysis engine that generates the SOFAs and annotates them by assigning a class label and a 3D pose.

The input image, the generated SofAs, and their respective annotations are shown in Figure 5. Generating the SOFAs was done using a simple color based segmentation, class labels were assigned by using a simple nn-classifier trained on Hu-moments[11] for the objects that were used in the experiment. Because of the flat nature of the objects the 6 DOF poses of the objects were calculated using the camera parameters.

B. Task 2: Chemical experiment

The robot task was to pick up the pipette, mount a tip on it, get some solution from the bottle and release it into one of the tubes found in the rack. Finally the robot should release the contaminated pipette tip into the trash box. The challenges for perception here are (1) not all objects can be perceived using RGB-D sensors (see Fig. 6, right, hence ROBOSHERLOCK uses a combination of color segmentation and point cloud clustering) and (2) some of the perception tasks needed are based on common sense knowledge: the opening of a container is the top part of an object.

There is no one unique solution to perceive all of the objects, and their parts, but having several expert algorithms combined with knowledge processing significantly increases the success rate of finding the relevant items on the table. Detecting the labels of the objects and estimating their 3D pose, as in Task 1, is not sufficient for the successful execution of pipetting. We precisely need to identify where the pipette tip needs to enter the bottle, or find the tubes in the rack.

To successfully accomplish this task we made use of ROBOSHERLOCKs reasoning capabilities, formulating rules like:

fitCircle(Obj,Radius) :-
category(Obj,'container'), object-part(Obj,Opening),
geom-primitive(Obj,'circular'),
radius(Opening,Radius),

which deduces the radius of the circle that needs to be fit in order to find the opening on top a container, or the holes in the rack where tubes can be found.

C. Task 3: Reasoning about objects and their properties

One of the key aspects of ROBOSHERLOCK is its ability to handle complementary, overlapping or even contradictory annotations. In this last task, the system has to successfully identify objects of daily use in the case where all of our annotations from Fig. 4 were run on the table top scene.

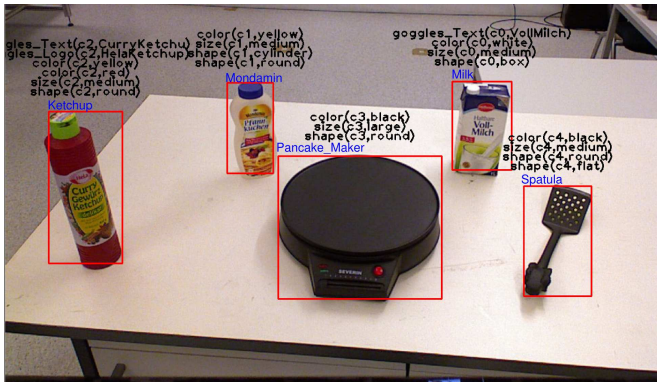

Fig. 7: SOFA s with annotations and the MLN prediction

We used a Markov logic network to combine annotations while taking into account the co-occurrences of objects in the environment as reported in [4]. The annotations of the single perception algorithms as well as the prediction are presented in 7.

VI. Related Work

There are a lot of works oriented at creating perception libraries which are a collection of task specific algorithms, e.g. ECTO [1], PCL, OpenCV and the STAIR Vision Library [12].The perception tasks that such a robot has to accomplish go substantially beyond what is supported by current perception libraries and frameworks. Frameworks, mostly based on middle-ware like ROS[2], such as SMATCH [13]) or REIN [14] have targeted the ease of program development but the problems of boosting perception performance through more powerful method combination has received surprisingly little attention.

Existing perception systems usually consider the case where a database of trained object is used to match it with sensor data. Even more, many systems focus on individual algorithms that only work on objects with specific characteristics, e.g. point features for 3D opaque objects [15], visual keypoint descriptor based systems like MOPED [7] for textured or [16] for translucent objects. ROBOSHERLOCK is capable of incorporating all of these different frameworks, and combine their result.

VII. Conclusions

In this paper, we have presented ROBOSHERLOCK, an open source software framework for implementing perception systems for robots performing human-scale everyday manipulation tasks. We have shown how the principles of unstructured information management can be used to design and realize robot perception systems that are taskable and can scale towards broader ranges of perception tasks and improve robustness and performance by exploiting ensembles of possibly knowledge-enabled perception experts. The resulting ROBOSHERLOCK can outperform other state-of-the-art robot perception systems as they are used as components, as ROBOSHERLOCK can reason about which AEs to apply to which SOFAs, can integrate the results of the individual AEs, and employs knowledge and reasoning for interpreting perception results. We have presented the capabilities of ROBOSHERLOCK in a kitchen scenario with objects of daily use and demonstrated its applicability and transferability in a chemical experiment setting.

[1] http://plasmodic.github.com/ecto/
[2] http://ros.org

References

[1] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building Watson: An overview of the DeepQA project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2010. [Online]. Available: http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303

[2] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[3] M. Tenorth and M. Beetz, "KnowRob – Knowledge Processing for Autonomous Personal Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4261–4266.

[4] D. Nyga, F. Balint-Benczedi, and M. Beetz, "Pr2 looking at things: Ensemble learning for unstructured information processing with markov logic networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, accepted for publication.

[5] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011, pp. 1–4.

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

[7] A. Collet Romea, M. Martinez Torres, and S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284 – 1306, September 2011.

[8] L. C. Goron, Z. C. Marton, G. Lazea, and M. Beetz, "Segmenting cylindrical and box-like objects in cluttered 3D scenes," in *7th German Conference on Robotics (ROBOTIK)*, Munich, Germany, May 2012.

[9] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[10] D. Pangercic, M. Tenorth, B. Pitzer, and M. Beetz, "Semantic object maps for robotic housework - representation, acquisition and use," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

[11] M.-K. Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 179–187, February 1962.

[12] S. Gould, O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Y. Ng, and D. Koller, "The stair vision library (v2.4)," 2010, http://ai.stanford.edu/ sgould/svl.

[13] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the pr2," in *ICRA*, Shanghai, China, May 2011.

[14] M. Muja, R. B. Rusu, G. Bradski, and D. Lowe, "Rein - a fast, robust, scalable recognition infrastructure," in *ICRA*, Shanghai, China, 09/2011 2011.

[15] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point Cloud Library – Three-Dimensional Object Recognition and 6 DoF Pose Estimation," *Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, September 2012.

[16] I. Lysenkov, V. Eruhimov, and G. Bradski, "Recognition and Pose Estimation of Rigid Transparent Objects with a Kinect Sensor," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.