Pervasive 'Calm'* Perception for Autonomous Robotic Agents

Thiemo Wiedemeyer, Ferenc Bálint-Benczédi and Michael Beetz
Institute for Artificial Intelligence and TZI (Center for Computing Technologies)
Universität Bremen, Germany
{wiedemeyer, balintbe, beetz}@cs.uni-bremen.de

ABSTRACT

A major bottleneck in the realization of autonomous robotic agents performing complex manipulation tasks are the requirements that these tasks impose onto perception mechanisms. There is a strong need to scale robot perception capabilities along two dimensions: First, the variations of appearances and perceptual properties that real-world objects exhibit. Second, the variety of perceptual tasks, like categorizing and localizing, decomposing objects into their functional parts, perceiving the affordances they provide.

This paper, addresses this need by organizing perception into a two-stage process. First, a pervasive and 'calm' perceptual component runs continually and interprets the incoming image stream to form a general purpose hybrid (symbolic/sub-symbolic) belief state. This is used by the second component, the task-directed perception subsystem, to perform the respective perception tasks in a more informed way. We describe and discuss the first component and explain how it can manage realistic belief states, form a memory of past perceptual experiences, and compute valuable perceptual attributes without delaying plan execution. It does so by exploiting that perception is not a one-shot task but rather a secondary task that is pervasively and calmly performed throughout the lifetime of the robot. We show system operating on a leading-edge manipulation platform.

Categories and Subject Descriptors

I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*3D/stereo scene analysis, Perceptual reasoning, Architecture and control structures*

General Terms

Performance, Algorithms, Design

*Calm refers to the term "calm technology" introduced by Mark Weiser [29], who considers calm information processing systems as systems "that inform but don't demand our focus or attention." For us we consider a calm perception system to be one that continuously runs and provides useful perceptual information without demanding high computational resources and in particular without requiring the robotic agent to wait for the results of perception processes.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

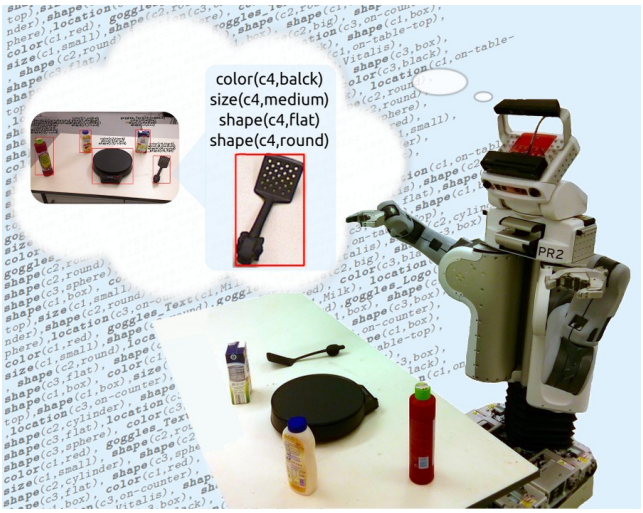

Figure 1: PR2 looking at a cooking scenario.

Keywords

robot perception; pervasive perception; task dependency; lifelong learning

1. INTRODUCTION

Autonomous mobile robots performing fetch and place tasks in a factory, a supermarket, or a household have to fetch all kinds of objects, in all kinds of scenes, and put them where they are needed. The perception tasks that such robots face are tremendously difficult. Some objects are characterized by their shape, others by their visual appearance, others are translucent. The robots must fetch objects based on descriptions and often within very cluttered scenes as for example inside of drawers and cupboards or the shelves of supermarkets where the individual objects are lined up together making it almost impossible to visually segment an object from the next one.

On the other hand, the tasks also exhibit a considerable amount of structure. The robots navigate around in their environment continuously and for long periods of time while the rate in which the environments change is typically slow. They can therefore form strong expectations about what they are going to perceive. In addition, there is a lot of temporal redundancy in the perceptual data and they often know what they will be looking for long before they can start the perception process.

The design and investigation of perception systems that aim at tackling these challenges as well as exploiting the structure of the perceptual tasks to be more efficient, robust, and exhibit better performance has received surprisingly little attention so far. There are partial solutions to some of the challenges in the areas of biological approaches to low-level vision [8] and multi-object tracking [22]. The research work of Collet et al. [9] is a notable exception who is looking into the lifelong exploration, object learning, and re-detection of textured objects. Also, autonomously driving cars have developed their own need for continual visual observation of traffic scenes and the individual entities in the scene [26].

In this paper, we propose PERCAP a pervasive 'calm' perception component of ROBOSHERLOCK [2, 4], for longterm autonomous robot manipulation. This component is to support the high-level perception system of the robot, which receives perception tasks such as *"detect the red cup on the table"*, *"find the box with the Kellogg's logo"*, etc. by pervasively collecting perception data and preparing information by interpreting the data and making logical assertions to a knowledge base, in order to facilitate the fast and robust accomplishment of requested perception tasks.

To do so, PERCAP provides the following functions:

- It estimates the partial belief state of the robot with respect to the current state of the environment,and the current poses and attributes of objects in use and provides information useful for requested perception tasks, without having any previous knowledge about the objects in the environment; and
- it maintains a perceptual memory of past perceptual data and interpretations thereof, which are also used for lifelong learning.

The main contributions of this paper are the following ones:

- configurable, high-performance preparatory perception system which computes:
  - symbolic representations and context information for simplifying perception tasks;
  - extracts sensor data of object hypotheses;
- belief state management including perceptual memory
  - object identity resolution
  - symbolic, relational model
  - reasoning with background knowledge (exploiting the closed world assumption)

Results and executable pipelines of PERCAP are publicly available on the OPENEASE [3] web-page[1], an online knowledge representation and processing service. In the remainder of the paper we proceed as follows. Section 2 and 3 describe the motivation and the system overview followed by a presentation of the low-level pervasive perception in sections 4, 5 and 6. How pervasive perception is used in task dependent robotic actions is explained in Section 7, followed by the evaluation of the validity of our approach in Section 8. We conclude by presenting state-of-the-art and future work in Section 9 and 10.

[1] `www.open-ease.org`

2. MOTIVATION AND USE CASE

Performing everyday manipulation and activities in a human environment is a sophisticated task for a robot. The main source for information in such an environment is the perception system of the robot. The perception system has to work in an open scenario and be responsive under the limitations of the restricted computation capabilities of the mobile hardware. Under these circumstances the perception system has to be taskable and has to decide which algorithms to execute at a certain point in time.

A human entering a kitchen, is constantly receiving, processing and storing information, on objects that he sees, without actively inspecting each object in detail. If he later starts a cooking task where a bowl, capable of holding 1 liter is needed, he will first look for the objects that could be possible candidates from the ones he has seen in the past, instead of starting a search.

For a robot this is a difficult task. If the perception system always detects and categorizes all the objects it sees, it would consume much of the limited processing capabilities of the mobile hardware. This could decrease the overall responsiveness. But on the other hand, if the perception system only perceives the objects of actual interest, it will have to search for them each time they are requested.

PERCAP solves this issue by constantly perceiving and memorizing the objects it sees without running any demanding high level perception algorithms all the time. It stores all collected information on the objects, including images and point clouds and creates a partial belief state of the current world that the perception system can use to quickly reply to queries or to analyze objects in more detail if needed. With our preparatory perception system applied to the above example, it is possible to lookup all perceived bowls and calculate their capacity just by using the memory, without any active search in the environment.

Another use case might be that the robot just sees the backside of an object of interest at the current position, but needs information from the front cover to identify the object. With our perception system it is possible to lookup all the viewpoints in which this object was seen to filter out frontal views. This way the object can be identified without the need to physically interact with it or to move the robot to another position.

3. SYSTEM ARCHITECTURE

As an extension to ROBOSHERLOCK, PERCAP operates according to the unstructured information management architecture (UIMA) [10]. The system is composed of a set of expert analysis engines (AEs). Some AEs detect point clusters in the RGB-D point cloud data that might correspond to objects and object groups in the operating environments. Others analyze the object (group) hypotheses and structure and interpret the regions that correspond to the objects and add inferred information as annotations for the object hypotheses.

In UIMA the basic data structure is called the Common Analysis Structure (CAS). The CAS consists of the sensor data (called the artifact), a number of SOFA s (Subjects of Analysis), which represent the object hypotheses, annotations of the SOFA s, which include the inferred information about hypotheses, and a common type system that enables the different AEs to exchange information in a common for-

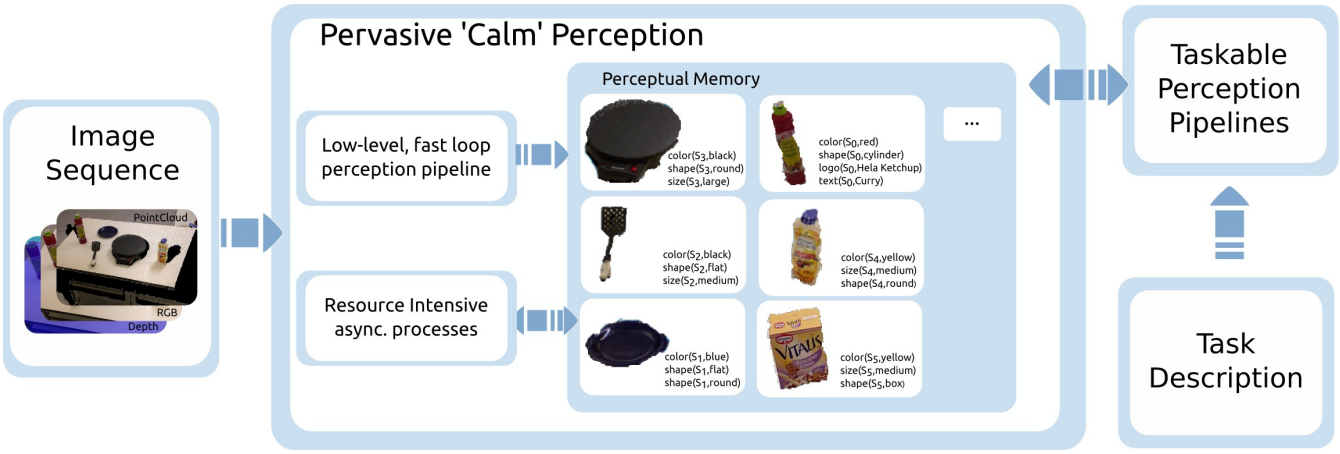

Figure 2: Architectural overview of the pervasive 'calm' perception system

mat. The types of information the SOFA s can be annotated with include the estimated size, color, texture, and shape of the hypothesized objects. More sophisticated and complex annotations are shapes, text written on the objects, logos, the identity of objects, and other kind of semantic information.

Figure 2 presents an overview of the underlying architecture of PERCAP. During task-execution a robotic agent can issue perceptual task descriptions which in turn get interpreted by a taskable perception pipeline. We consider one of two kinds of perceptual task descriptions:

1. **detect *obj-descr*** to find objects in the sensor data that satisfy the description *obj-descr* and return the detected matching *object hypotheses*. i.e.: *(an object (category spoon) (color red))*.

2. **examine *obj-hyp attributes*** asks the perception system to examine a given hypothesis *obj-hyp* in order to extract additional *attributes* requested. i.e. exact pose of the object or its 3D model.

PERCAP addresses the first kind of task descriptions. In Figure 2 a low-level perception system processes, interprets, and maintains high-volume sensor data at a rate of 5-10 Hz without slowing down the operations of the robot, and stores the generated information in a perceptual memory. It is the base pipeline that is always run and on the results of which all other pipelines build up. It is designed for robotic agents operating in household environments and uses knowledge about the environment, like semantic maps [21] and location of the robotic agent in its environment. The proposed system is continuously perceiving and memorizing low level percepts of objects it sees. The later part has two requirements: a basic perceptual pipeline for detecting objects and adding low level annotations, and a persistent perceptual memory system combined with object identity resolution capable of storing all information for each processed frame.

Objects stored in the perceptual memory are further processed by resource intensive asynchronous processes. These expert methods are computationally inexpensive for the robot, but their response times do not meet the execution-time requirements of the low-level pipeline. Examples of such kind are web services (e.g. Google Goggles or Barcoo) or computationally more expensive algorithms running on a server (integrated using ROS[2] services). In previous work [20] we reported on how some of the results produced by the annotators are used as evidences, to probabilistically infer information about objects in the robots environment. PERCAP makes use of these annotators and more to build a preliminary belief-state. A more detailed presentation of the annotators used is presented in Table 1.

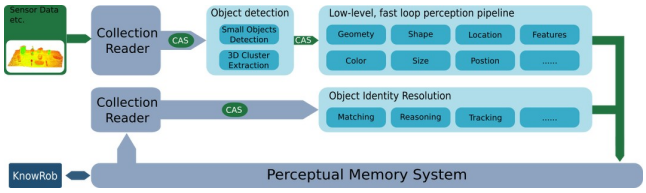

Figure 3: Overview of the low level perception system.

4. LOW-LEVEL PIPELINE

Following the UIMA paradigm each image obtained by a camera is seen as unstructured information. The low-level pipeline has to identify structures in this information. This is done by multiple algorithms (ensemble of experts) which are designed for specialized task.

Since the robot is localized in a known household environment, there are many static objects in the world, like cupboards, counters, tables. The map of the environment and the position of these static objects is known and stored in a knowledge base that is actively used to focus the attention of the perception system. In this environment objects of daily use appear either on top of tables or counter tops or inside drawers, cupboards, oven, refrigerator.

For object detection only the supporting planes of these furnitures are of interest. Therefore to reduce processing power for detecting volumetric objects, first the normals of each point of the point cloud is estimated. Then supporting planes are identified by a plane annotator. Each plane is then used to detect objects on it by euclidean clustering. For flat objects, that do not possess dominant 3D characteristics

[2] `http://ros.org`

Annotator	Process Type	Symbolic Representation	Description
Color	Fast-Loop	color(cluster, color)	returns symbolic color annotation and color histograms based on color distribution in HSV color space. Depending on the distribution, one object can have multiple symbolic color labels assigned.
Size	Fast-Loop	size(cluster, size)	labels objects into *small*,*medium* or *big* depending on the volume of the 3D bounding box normalized with the distance to the camera to the object.
Goggles	Asynchronous	logo(cluster, logo) text(cluster, text) texture(cluster, t)	sends the image region of an object hypotheses to the Google Goggles servers and interprets the answer to extract text, logo, and texture information.
FlatObject	Fast-Loop	shape(cluster, shape)	looks for additional object hypotheses in color space (e.g., cutlery . . .).
PrimShape	Fast-Loop	shape(cluster, shape)	fits lines and circles to 3D point clusters projected on to the 2D plane using RANSAC [12]. Values returned: *box, round*
SACmodel	Fast-Loop	shape(cluster, shape)	recognizes cylindrical objects and planes in 3D space. for objects the number of inliers found needs to exceed the given threshold (60% of the total points in a cluster). Value returned: *cylinder*
Semantic Location	Fast-Loop	location(cluster,location)	interprets object positions in terms of a semantic environment map [21] and returns places such as *counter tops, tables, fridges, and drawers.*
PCLFeatureExtraction	Fast-Loop	feature(cluster,<feat>)	Extracts 3D feature descriptors implemented in PCL [23] for every cluster(e.g. VFH, PFH, SHOT)
LineMod	Asynchronous	instance(cluster, category)	matches each object hypothesis to a set of object models that the robot should actively look for using the Linemod algorithm [13].
PoseAnnotation	Low-level	pose(cluster, <pose>)	estimates an oriented 3D bounding box, and annotates the clusters with the respective pose
FeatureAnnotation	Low-level	feature(cluster,<feat>)	extracts key-points and their respective key-point description. Wraps around the key-point extraction functionalities in OpenCV[7]

Table 1: Description of the annotators, the conditions under which they work and the symbolic representation they result in

a color based segmentation is applied, exploiting the color distribution of the supporting plane.

Multiple experts are run on all detected objects and add low level annotations to them, including pose, shape, 3D bounding, image features, color histograms, semantic color description, semantic relational locations, etc. Some of these low-level experts and more detail about them is shown in Table 1.

5. PERCEPTUAL MEMORY

The perceptual memory is the main service that PERCAP provides to the overall perception system. The perceptual memory stores all the objects that the robot has detected over its operation time, maintains the identity of these objects, and accumulates and updates the knowledge about these objects.

The knowledge about the objects stored in the perceptual memory includes pieces of the raw sensor data that have been used to perceive the objects, other data structures that resulted from processing these sensor data, and symbolic assertions that were produced from the expert perception routines running on the detected object hypotheses. Thus, the symbolic assertions that were generated for the detected ketchup bottle on the table (see Figure 2) include the following ones:

- color(s_0,red)
- shape(s_0,cylinder)
- logo(s_0,Hela Ketchup)
- text(s_0,Curry)

The set of assertions of all objects detected on the kitchen table can therefore be considered as a partial knowledge base of the table scenario that can be used to reason about the context of the table scene in a more informed manner. To this end, the facts are asserted into a logical (Prolog) knowledge base. Using Prolog we can encode reasoning knowledge in the form of rules that states that a cylindrical object with the logo "Hela Ketchup" and some text "Curry" could be the ketchup bottle:

category(Obj,Cat) :-
logo(Obj,Hela Ketchup), text(Obj,Curry),
shape(Obj,cylinder)
Cat is 'Ketchup'.

Another rule might state that in the context of a table set for breakfast ketchup often co-occurs with salt which can be used by the task-directed perception system to actively search for objects that could be salt containers.

Maintaining a perceptual memory is not only advantageous for guiding perception processes with background and context knowledge. It is also a resource for ensemble learning. The robotic agent can learn the co-occurrence of objects in scenes as well as the co-occurrence of objects and perceptual appearance attributes or it can learn using the raw

image data of an object how to best detect ketchup bottles on breakfast tables.

To facilitate these learning tasks the perceptual memory is realized as a "big data" data storage that collects data over time. For this purpose we choose to use MongoDB as an implementation basis for the perceptual memory. MongoDB is a good choice for storing data in a native way in an object oriented database [19]. It is the most prominent noSQL database, enables better intercommunication with other systems like KnowRob [27], and organizes the memory structure in a document-like manner.

The perceptual memory of PerCaP is implemented to be highly efficient. To this end, the stored data are rigorously typed in a structure called FeatureStructure, which can be a list, an array, or a structure of features. A feature is either of a basic type (like boolean, byte, integer, floating point, string) or is a FeatureStructure itself. In MongoDB all data is stored in the BSON (Binary JSON) format. The BSON format is object oriented and each object has multiple named elements with certain types (including integer, double, string, binary, array, another BSON object). This enables a near native 1-to-1 mapping between the UIMA FeatureStructures and the BSON objects. PerCaP applies a generic algorithm for converting from UIMA data into BSON and vice versa. The algorithm can store and load any CAS to and from the database and is independent of the type system that is being used. To improve the performance the algorithm automatically stores large lists and arrays of basic UIMA types as binary data elements in BSON.

The perceptual memory is organized into two databases: one containing the scenes and one containing the known objects. After the low-level pipeline from PerCaP is run, all information is stored as a new entry in the scenes database and the perceptual memory checks whether this could be a re-detection of an object that is already contained in the memory. This is necessary to perform perception in a more informed way. Keeping track of object identity also enables PerCaP to accumulate perceptual knowledge about objects over longer time. *Object identity resolution*, which we will detail in the next section, updates the global belief state and connects the objects from the actual scene to the ones already known.

Resource intensive asynchronous processes e.g. web based algorithms like Google Goggles or computational expansive high level perception algorithms like CAD based model fitting can then be executed on demand and detached from the low-level pipeline. These processes find the objects in the database and further analyze them. New annotations get added and updated ones replace existing annotations. The results are then again stored in the object database.

6. OBJECT IDENTITY RESOLUTION

The low-level pipeline detects objects in each frame obtained by the sensors. Each frame is only a snapshot of the current scene. By default the objects detected in one frame are detached from the ones detected in previous or future frames. To connect each of these objects to an specific object over time we use an object identity resolution that uses the low level percepts.

The object identity resolution is based on the entity resolution framework proposed by Blodow et. al. [5]. This framework uses a probabilistic first order model which considers the shape, position and a numerical similarity measurement in order to achieve a globally consistent belief state even in the presence of ambiguity and partial observability.

Each object contains a number of annotation computed by the low-level pipeline. Shape and position needed by the entity resolution framework are directly given by annotators of the low-level pipeline. For the similarity measurement we use a subgroup of other low level annotations; these are geometry, color histograms and image features. To compute a numerical similarity measurement from these annotations a distance function $dist_t(a_t, b_t) \rightarrow [0 \dots 1]$ is defined for each of the annotations. The distance function takes two annotations of the same type t as parameter and computes a normalized distance d_t. Each distance d_t is weighted by the factor w_t, which is statically defined for each type. The overall distance for an object to another is than the sum of all weighted distances normalized to $[0 \dots 1]$:

$$dist(a, b) = \frac{\sum w_t * dist_t(a_t, b_t)}{\sum w_t}$$

The weighting is applied to prioritize certain annotations that are more reliable and discriminating than others. For example shapes are similar for many objects while color histograms and image features differ much. The weight for each annotation is based on empirical results on their reliability and discrimination.

For the geometry annotation, which contains a minimal bounding box, the distance is the normalized sum of the distances in each dimension (width, depth, height) of the bounding box:

$$dist_{geo}(a, b) = \frac{dist_d(w_a, w_b) + dist_d(d_a, d_b) + dist_d(h_a, h_b)}{3}$$

The distance for each dimension is the absolute difference divided by the minimal length, limited to 1. Every difference that is greater than or equal to the doubled minimal distance is mapped to 1, everything lower is linear mapped from 0 to 1:

$$dist_d(a, b) = \min(1, \frac{|a - b|}{\min(a, b)})$$

The distance between two color histograms, which are normalized, is the sum of absolute differences per bin divided by 2:

$$dist_{color}(a, b) = \frac{\sum |a_i - b_i|}{2}$$

All image features (SIFT, ORB, FREAK) descriptors from one object are matched against the ones from the other object using a brute force matching algorithm, which gives back a minimal distance d_i for each descriptor. These distances are summed up and divided by the product of the number of descriptors n and a constant $maxDist$ which defines the maximal distance between descriptors that is taken into account. The result is limited to 1 if higher:

$$dist_{feature}(a, b) = \min(1, \frac{\sum d_i}{n * maxDist})$$

The maximal distance $maxDist$ can be defined lower than the real maximal distance between descriptors. Image descriptors can be considered as dissimilar even if they are not near the maximal distance. This increases the resolution for the similarity measurement for near descriptors and reduces the limit of dissimilarity that is inspected.

For each detected object the similarity to each known object is computed and together with their pose and shape passed to the entity resolution framework, which gives back the probabilities for each cluster belonging to which object. Based on the results the clusters are assigned to the best candidate, get added as new clusters, while objects that are not in the current scene are marked as persistent.

7. TASK SPECIFIC PERCEPTION

In our laboratory PerCaP is used on all of our robots for all the tasks they perform. This is a broad range of tasks, from pick-and-place in a kitchen environment, to DNA extraction in a chemical laboratory environment. For all these tasks the preparatory perception system is used and all task specific parts build on top of it.

We showcase the use of PerCaP on two tasks of a robotic agent performing different experiments and show how the taskable perception pipeline makes use of the preliminary belief state generated by PerCaP in order to come up with the correct answers.

Pick and Place Surgical Utensils

The first task involves a robotic agent picking up surgical utensils, and putting them in a bowl. The input image, the generated clusters, and their respective annotations are shown in Figure 4. Finding the object clusters was achieved using a color based segmentation, class labels were assigned by using a nn-classifier trained on Hu-moments[14] for the objects that were used in the experiment. Because of the flat nature of the objects the 6 DOF poses were calculated using the camera parameters and the equation of the supporting plane.

In this scenario the taskable perception system receives queries of the form:

(detect (an object
(shape (flat)), (location (on-table))
(category (C)), (pose (P))))

Figure 4: Surgical utensils on a table: Color *(left)*, SofAs *(center)* and pose *(right)*

PerCaP continuously detects the objects, calculates the Hu-moments and the pose estimate for these, adding them to the initial belief space. The taskable perception system triggers the execution of the classification in order to find the correct object labels.

Chemical experiment

In a very different scenario the robotic agent's task was to perform pipetting in a DNA extraction scenario. This involves picking up the pipette, mounting a tip on it, getting some solution from the bottle and releasing it into one of

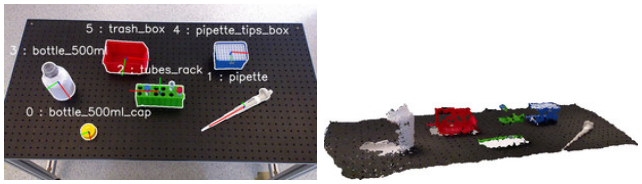

Figure 5: Pipetting scene as seen by the robot. *Left: RGB, Right: PointCloud*

the tubes found in the rack. The challenges for perception here are

1. not all objects can be perceived using RGB-D sensors (see Fig. 5), hence segmentation methods need to be combined.
2. it is not enough to detect and track the objects in the environment to execute the task. Parts of objects need to be found depending on the current sub-task, e.g. in order to get solution from the bottle, its cap needs to be removed and the opening detected.

Although this task is significantly different from picking and placing surgical utensils, the same low-level perceptual pipeline is used in order to generate the belief-state. Object hypotheses are being constantly updated in the perceptual memory and the basic characteristics of the objects are stored. During the different subtasks of pipetting, the high-level perception queries PerCaP and finds the object parts needed by the robotic agent to successfully accomplish its goal.

In both tasks executed by the robot PerCaP provides valuable information without delaying execution times while maintaining a perceptual memory that enables offline learning.

8. EXPERIMENTS AND EVALUATION

Because PerCaP does not use any models of the objects it is to encounter in the environment, a key element in the successful application is the precision of the individual experts that are used in the perception pipeline and the maintenance of a correct belief state about the objects in the environment. To this end we conducted experiments to evaluate the correctness of the individual annotators and then the reliability of the belief state.

We report the correctness of the symbolic representations returned by the annotators on a collection of scenes containing objects of daily use in a kitchen scenario as it was shown in [20], in Figure 6. We took 50 scenes with different number of objects in a scene (between 4 and 8). The objects were chosen so that they vary in defining characteristics (textured, untextured, etc.). The data was hand labeled with ground truth for each of the annotators. Correctness of the different annotators is shown in Figure 7.

Evaluation of the belief state management system was done on a sequence of images (Figure 8), with increasing complexity both in the number of objects in the environment but also in clutteredness/occlusions of these. We took 60 scenes gradually increasing the number of objects, adding and removing them, and also moving the robot around. We compared the actual number of objects introduced in the

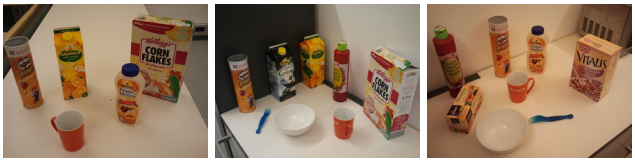

Figure 6: Example of table top scenes used to evaluate the individual annotators

Annotator	# total Annotations	# correct Annotations
Color	289	231 (79.9%)
Goggles	80	—
Prim. Shape	336	233 (69.3%)
SACmodel	38	31 (81.5%)
FlatObject	142	116 (81.6%)
Linemod	90	45 (50%)

Figure 7: Evaluation of annotators: Correctness of their annotations Shape, SACmodel and FlatObject have been aggregated since they all contribute to the "shape" predicate.

robot's environment to the number of object-hypotheses in the belief state. Note that none of the objects used in the experiments are previously known to the robot, and we solely base our identity resolution on the low-level percepts generated by PerCaP. The objects were also chosen to have more and more similarities between them as the scene's complexity grew.

# Obj	# Scenes	# Clusters	# Obj.Hyp. (O/O_{hyp})
2	10	20	2(100%)
4	13	47	5(80%)
6	17	92	8(75%)
11	20	170	19(57%)

Figure 9: Evaluation of the belief state

In Figure 9 we report the results of our experiments. In the total of 60 scenes we introduced 11 different objects resulting in a total of 329 clusters after segmentation. These clusters were assigned to a total number of 19 object hypotheses at the end of the experiment. As it can be observed the number of object hypotheses in the belief space grows with the number of objects introduced in the environment. This is due to the fact that with the increase objects and occlusion the chances of confusing them also grow, as specially since the low-level pipeline contains very basic perceptual capabilities. The relation between the objects in the real world and the object hypotheses in the robot's belief stat are also shown in Figure 10.

Since a robotic agent, during a task execution does not need to deal with an unlimited number of objects (Chapter 7), even though the initial belief space gets more and more imprecise, the number of objects that need to be examined drops significantly. Compared to a conventional pipeline in RoboSherlock this means that PerCaP will speed up the lookup for objects, because the robot can directly analyze all previously seen objects to first find the

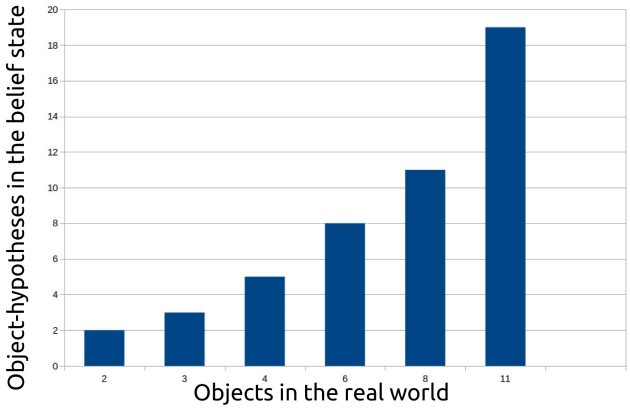

Figure 10: Relationship between objects in the robots environments and the belief-state

searched object and then look to the last known position to see if the object is still there. We experimented with different task descriptions (like the ones presented in Chapter 3 and 7) on the complete belief-state containing 19 objects and came to the conclusion that for any query where we give at least two atomic evidences for the searched object (e.g. size and color or color and shape) the system needs to analyze at most five object hypotheses. In a table-top scene like the ones in the bottom row of Figure 8, where significantly more objects are found, this helps limit our search space for the correct answer, thus speeding up processing time.

9. RELATED WORK

Current robotic perception systems mostly consider the case where a database of objects models are used in order to match them with the sensor data. Furthermore, these systems usually focus on the development of individual algorithms, that work on objects with specific characteristics e.g. point features for 3D opaque objects [1], visual key point descriptor based systems like MOPED [9] for textured or [16] for translucent objects. These also is no single algorithm that can perceive the wide range of characteristics objects can possess. However, several methods exist that handle subproblems of perception reasonably well, many of which are complementary and could be combined to boost performance.

We expect that the combining various approaches with complementary strengths can improve over the individual performance of the methods [24, 28] and make it generalize better [15]. In the application domain of machine learning this has been demonstrated through the NetFlix Prize [25]. During the challenge ensemble learning became a prominent and very successful approach, as individual teams started to join efforts.

There are many existing perception frameworks that implement stat-of-the-art task specific algorithms e.g. PCL [23], OpenCV [7], which can be used as building blocks for general robotic perception systems, but the perception tasks that a robot assistant has to accomplish go substantially beyond what is supported by the perception libraries and frameworks. Frameworks, mostly based on middleware like ROS, such as SMATCH [6])or REIN [18] have targeted the

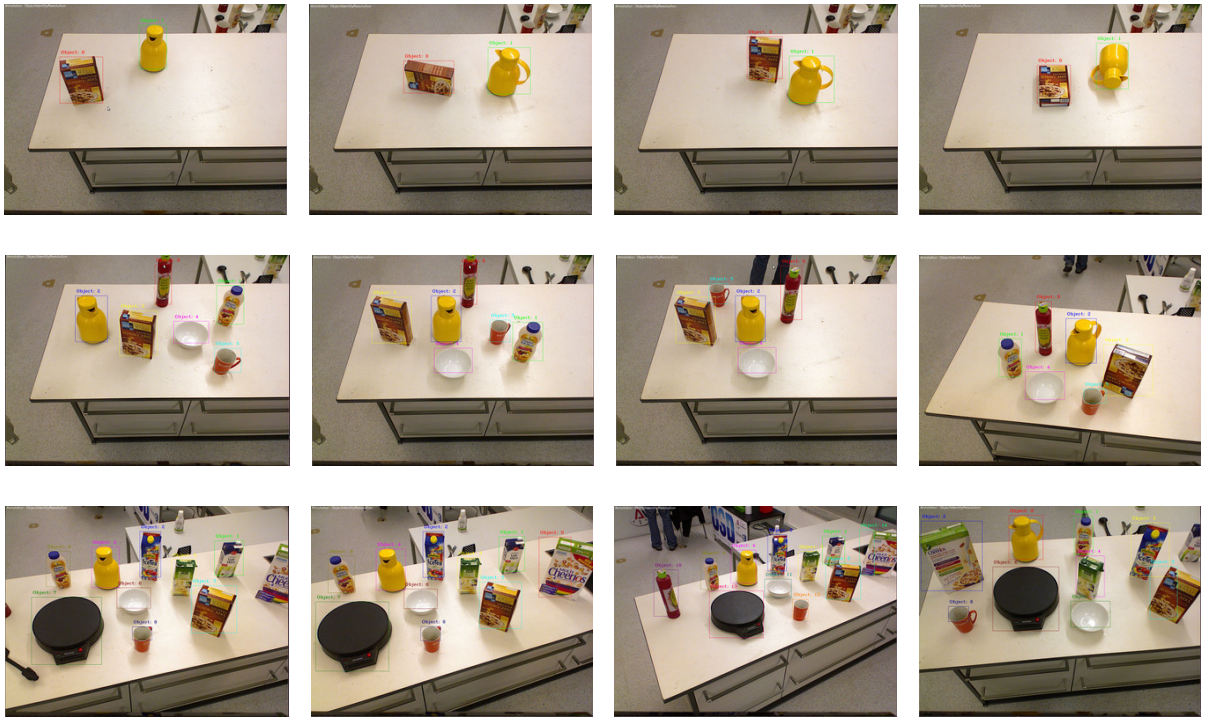

Figure 8: Images of the table top scenes used for evaluation. Complexity of the scenes grows from top down, object or robot positions change left to right

ease of program development but the problems of boosting perception performance through more powerful method combination has received surprisingly little attention.

The UIM architecture allows both for the incorporation of the ensemble learning framework from [17] and allows for steering the processing work flow in order to support active classification [11].

Our work builds on top of the existing state of the art, bringing the context of a task being executed and a perception capabilities of a mobile robotic platform closer.

10. PERSPECTIVES

Using pervasive 'calm' perception the robot collects comprehensive data and information about perception in task contexts. For every perception task it stores the relevant image parts that were processed in order to perform the task, the outcome of the perception task, and statistics about the way the task was approached and performed. The statistics keep information about what succeeded and failed and how long individual processing steps took. In addition, the memory mechanism will also propagate reliable results as "ground truth data" to low quality observations of the same object. For example, if the milk pack is the only blue object on the table and at some point the robot can read the bar code on the package, it can then label the blue object in the same scene as being milk, as long as the scene does not change. This way the robot will be able to automatically generate supervised learning tasks.

Based on the collected experiences of perception tasks, the robot can automatically learn the perception tasks that are typically performed in given manipulation actions. For example, while setting the table the robot typically looks for knives and forks in the drawer. We can learn statistics of how often perception failed until a knife or fork was found, we can also identify cases where the plan later signaled a failure "wrong object acted on". Learning about the internal operations of perception pipelines the robot can also learn which expert methods worked and which ones did not. It can also learn new experts or improve already existing ones by performing supervised learning tasks on captured images with labeled results.

Acknowledgements

This work was supported in part by the EU FP7 Projects *SAPHARI* (Grant Agreement Number 287513) and *RoboHow* (Grant Agreement Number 288533).

REFERENCES

[1] 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*, 19(3):80–91, September 2012.

[2] M. Beetz, F. Balint-Benczedi, N. Blodow, D. Nyga, T. Wiedemeyer, and Z.-C. Marton. RoboSherlock: Unstructured Information Processing for Robot Perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015. Accepted for publication.

[3] M. Beetz, M. Tenorth, and J. Winkler. Open-EASE – a knowledge processing service for robots and robotics/ai researchers. In *IEEE International*

Conference on Robotics and Automation (ICRA), Seattle, Washington, USA, 2015. Accepted for publication.

[4] N. Blodow. *Managing Belief States for Service Robots: Dynamic Scene Perception and Spatio-temporal Memory*. PhD thesis, Intelligent Autonomous Systems Group, Department of Informatics, Technische Universität München, 2014.

[5] N. Blodow, D. Jain, Z.-C. Marton, and M. Beetz. Perception and Probabilistic Anchoring for Dynamic World State Logging. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 160–166, Nashville, TN, USA, December 6-8 2010.

[6] 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.

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

[8] A. Buch, D. Kraft, J.-K. Kamarainen, H. Petersen, and N. Kruger. Pose estimation using local structure-specific shape and appearance context. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2080–2087, May 2013.

[9] A. Collet Romea, M. Martinez Torres, and S. Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research*, 30(10):1284 – 1306, September 2011.

[10] 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*, 31(3):59–79, 2010.

[11] T. Gao and D. Koller. Active classification based on value of classifier. In *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

[12] 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.

[13] 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.

[14] M.-K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, February 1962.

[15] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945 – 954, 1995.

[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.

[17] Z.-C. Marton, F. Seidel, F. Balint-Benczedi, and M. Beetz. Ensembles of Strong Learners for Multi-cue Classification. *Pattern Recognition Letters (PRL), Special Issue on Scene Understandings and Behaviours Analysis*, 2012. In press.

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

[19] T. Niemueller, G. Lakemeyer, and S. S. Srinivasa. A Generic Robot Database and its Application in Fault Analysis and Performance Evaluation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2012*, Vilamoura, Algarve, Portugal, 2012. IEEE/RAS.

[20] 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, May 31-June 7 2014.

[21] 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.

[22] Z. W. Pylyshyn and R. W. Storm. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial Vision*, pages 179–197, 1988.

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

[24] M. Sewell. Ensemble learning, 2007. Available online.

[25] J. Sill, G. Takács, L. Mackey, and D. Lin. Feature-weighted linear stacking. *CoRR*, abs/0911.0460, 2009.

[26] A. Teichman and S. Thrun. Practical object recognition in autonomous driving and beyond. In *IEEE Workshop on Advanced Robotics and its Social Impacts*, 2011.

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

[28] K. M. Ting and I. H. Witten. Stacked generalization: when does it work? In *Procs. International Joint Conference on Artificial Intelligence*, pages 866–871. Morgan Kaufmann, 1997.

[29] M. Weiser and J. S. Brown. Designing calm technology. *POWERGRID JOURNAL*, 1, 1996.