Project no.:	**027657**
Project full title:	**Perception, Action & Cognition through learning of Object-Action Complexes**
Project Acronym:	**PACO-PLUS**
Deliverable no.:	**D8.1.4**
Title of the deliverable:	**Multi-sensory object categorisation**

Contractual Date of Delivery to the CEC:	1.2.2008
Actual Date of Delivery to the CEC:	31.1.2008
Organisation name of lead contractor for this deliverable:	SDU

Author(s): Dirk Kraft, Norbert Krüger, Mila Popovic, Kai Welke, Tamim Asfour, Kai Hübner, Danica Kragic, Alex Bierbaum, Florentin Wörgötter, Nicolas Pugeault, Christopher Geib, Ron Petrick, Mark Steedman, Bernhard Hommel, Renaud Detry, Justus Piater, Rüdiger Dillmann
Participant(s): KTH, BCCN, AAU, JSI, UL, UEDIN, SDU, ULg, UniKarl

Work package contributing to the deliverable:	WP1,WP2,WP3,WP4,WP5
Nature:	R
Version:	Final
Total number of pages:	17
Start date of project:	1st Feb. 2006 **Duration:** 48 month

Project co-funded by the European Commission within the Sixth Framework Programme (2002–2006)
Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

We present our work on multi-sensorial object categorisation. The categorisation problem is embedded in a three level cognitive architecture that is realized in an embodied system in a simplified domain. The architecture consists of three hierarchically nested layers: a low-level, sensorimotor layer connecting sensory processors to motor procedures; a mid-level layer that stores object-action episodes (Object-Action Complexes or OACs) and reasons about memorised events; and a high-level symbolic planner that creates abstract action plans to be specified at lower levels. The current domain simplification addresses the use of a simple two-finger grasping device as well as the restriction to objects with 3D circles as parts. The simplification allows for the processing and use of object categories on all three levels of the processing hierarchy including planning and plan execution. Besides describing the currently realized cognitive system in the simplified domain we also present our work on the extension of the domain to general objects and more complex grasping devices.

Keyword list: Cognitive Architecture, Grounding, Learning, Object-Action Complexes

Table of Contents

1. Introduction

We have developed a three level hierarchy representation in which we perform learning, planning as well as plan execution as described in section 2 (see also Delverable 1.2.1). Based on this three level hierarchy, we have been realising embodied systems as described in this deliverable. For this, we have been following a parallel design strategy. On the one hand, we have already realised a system that computes and performs plans as well as learning in a simplified domain realising the full hierarchy (this is described in section 2). This simplified domain consists of objects having circular parts. The aim of working in this simplified domain is to realize the full hierarchy by reducing some aspects of the vision and robotic problems. This allows for treating the categorisation problem in a cognitive perspective. In parallel, we have done important steps towards extending the domain to arbitrary objects and more elaborated actions which is described in section 3.

Categorisation takes place on multiple levels. In the context of realising the full hierarchy we found it important that the categories 'objectness', 'graspability' as well as the categories open/closed and full/empty are realised on all levels, in particular that the highest planning level makes use of these categories. Furthermore, we addressed elaborated tactile sensing for shape categorisation in section 3.1 as well as aspects of efficient memorisation, active manipulation and the combination and extension of categories in section 3.

Our work is described in an accepted journal publication [E] as well as conference contributions [A, C, D, H, I] and technical reports [B, F, G].

2. Three Level Architecture

The cognitive system we have been developing consists of three hierarchically nested layers: a low-level, sensorimotor layer connecting sensory processors to motor procedures; a mid-level layer that stores object-action episodes (Object-Action Complexes or OACs) and reasons about memorised events; and a high-level symbolic planner that creates abstract action plans to be specified at lower levels (see also Delverable 1.2.1). The cognitive system works in two modes. In the first mode, the system explores its environment and learns object–action associations such as object–grasp associations or consequences of actions such as 'poking'. Furthermore, representations of objects which have been grasped become learned. In that way, the world knowledge of the system becomes extended during exploration. In the second mode, it constructs and performs plans. In both modes it is able to deal with unexpected events or errors (i.e., the system can recover from such situations) and is also able to adapt internal representations based on the different events occurring either in the exploration or plan performance mode.

This architecture is the basis for the different embodiments used in PACO-PLUS. One of the settings (used at SDU) is an embodied system consisting of an industrial 6 degrees of freedom (DoF) robot with a two finger gripper, haptic sensing by means of a force-torque sensor and a high resolution stereo system in which a preliminary attention system is realized (see figure 1 and Deliverable D4.1.2). One task we want to address is cleaning up a table, but the proposed architecture allows for more general tasks. Furthermore, in parallel to the planner's performance, the system is provided with a memory of past experiences that allows for learning processes.

The aim of the architecture is not only to solve a given task with best possible performance but also to allow for a set of cognitively rich processes in which the robot

- derives a plan to solve the given task,
- executes the plan,

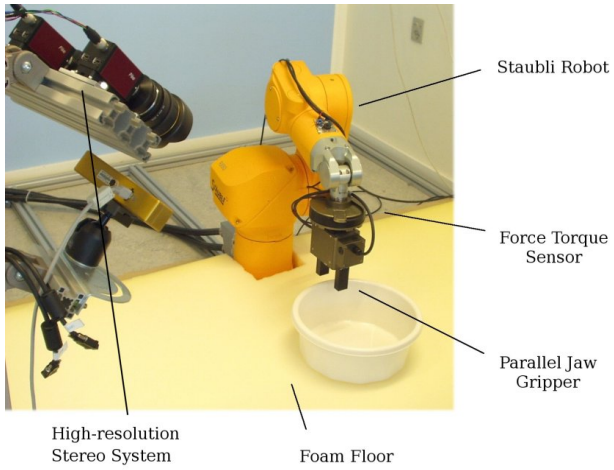


Figure 1: Embodiment of the Cognitive System

(a) (b) (c)

Figure 2: Withdrawal after the detection of a collision. (a) Approaching the object, (b) collision is detected, (c) start of withdrawal action.

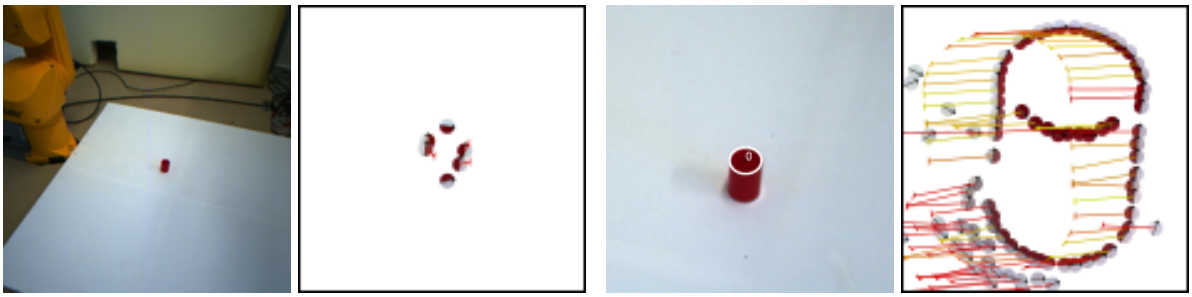
- is able to recognise and react on unexpected events during the exploration or execution of the plan on
  - either a rather low level (for example in case of collisions by withdrawal, see figure 2),
  - or on the mid–level by reinspecting the scene with a certain problem statement in mind (e.g., by looking at a certain aspect with higher resolution (see figure 3) and hence, resolving the situation within the given plan),
  - or on the high-level by a complete replanning.

In parallel, made experiences in terms of Object Action Complexes (OACs) become stored and transferred to various learning processes on a mid-level stage (see figure 4).

In its early stages, this work focuses on a limited domain: objects become represented as 3D circle and grasps become associated to these. This limitation is merely for development purposes and there is ongoing work toward the extension of the proposed system to arbitrary objects (see section 3)

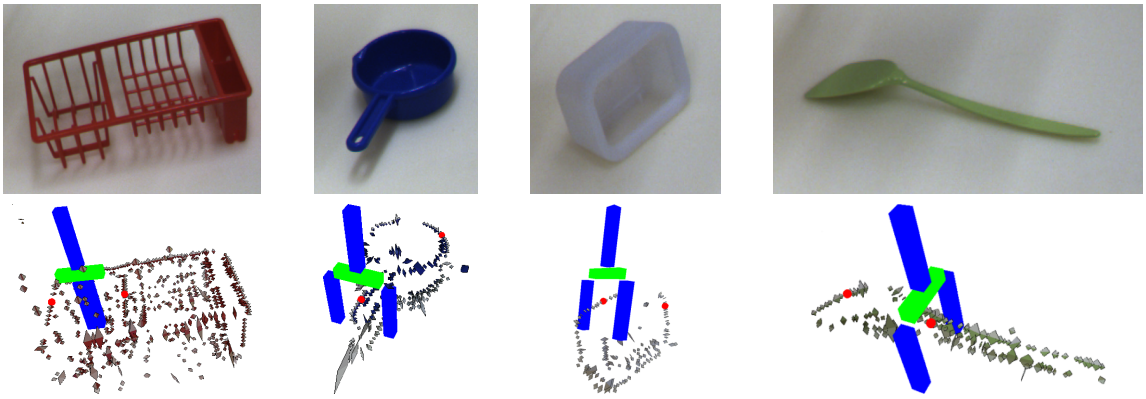
The system consists of three levels: The low, sensorimotor robot-vision level providing multi-sensorial information and action options, a mid-level on which made experiences are stored and provided for different learning processes, and on various levels as well as a planning level that generates plans based on the information provided by the robot-vision or the mid-level and which also monitors the execution of these plans. This architecture integrates several approaches and concepts from the computer sciences, artificial intelligence, and cognitive psychology, and it allows for learning and adaptation processes at all three levels.

The robot-vision level provides visual and tactile information to the higher levels in terms of scene affordances (see figure 5) as well as object identities and poses. It also is able to handle action commands, such



(a) (b)

Figure 3: Circle detection is not successful (a) because of the small number of feature descriptors extracted from a downsampled version of the high resolution images. It is successful (b) when the system focuses on the object at full resolution.



(a) (b) (c) (d)

Figure 4: Objects (top row) and an associated object representation with tested actions (bottom row). The actions are recorded after they have been executed and have a success description attached: (a,b) successful grasps, (c) unsuccessful grasp (collision with object), (d) unsuccessful grasp (object could be grasped but grasp was unstable).

as:

- apply grasp A on object B,
- start an explorative action such as 'poking', or
- shift attention to a certain location (see figure 3).

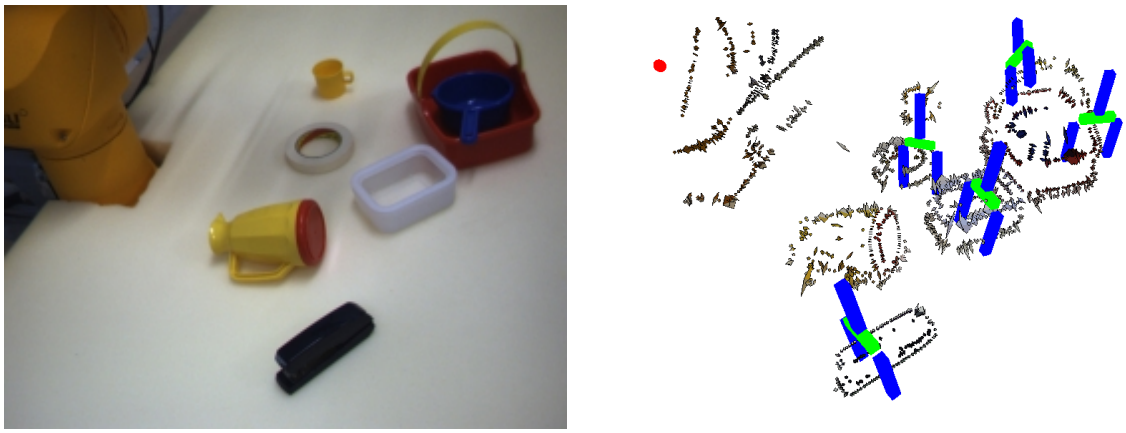
Moreover, it has control mechanisms that allow for the detection of unexpected events, events that would normally either lead to emergency stops and/or damaging the objects or the robot itself. It has pre-programmed behaviours to overcome such situations (see figure 2).

In its current (preliminary) state, the mid-level is responsible for the following tasks:

- the storage of OACs in memory, and their access by different learning processes,
- the refinement of grasping reflexes and object-action models based on the stored OACs (see figure 6),
- and the stabilisation of transient sensorial input into messages passed onto the planning level.

Figure 8 shows how an object with a circle part can be tested if it falls into the categories open or closed.

The planning level is responsible for constructing high-level, goal-oriented plans and for feeding those



(a) (b)

Figure 5: Scene information. (a) Example scene. (b) Example affordances associated to potential objects in the scene.

plans to the lower level systems for execution by the robot. To do so, the planner maintains an abstract model of the objects, properties, and actions available to the robot in the world (described in detail in [F]). The planner also receives regular updates on the state of the world from the lower levels, which it uses to monitor the success of plans being executed, and to control resensing and replanning activities in the system. The execution of a simple plan is shown in Figure 7.

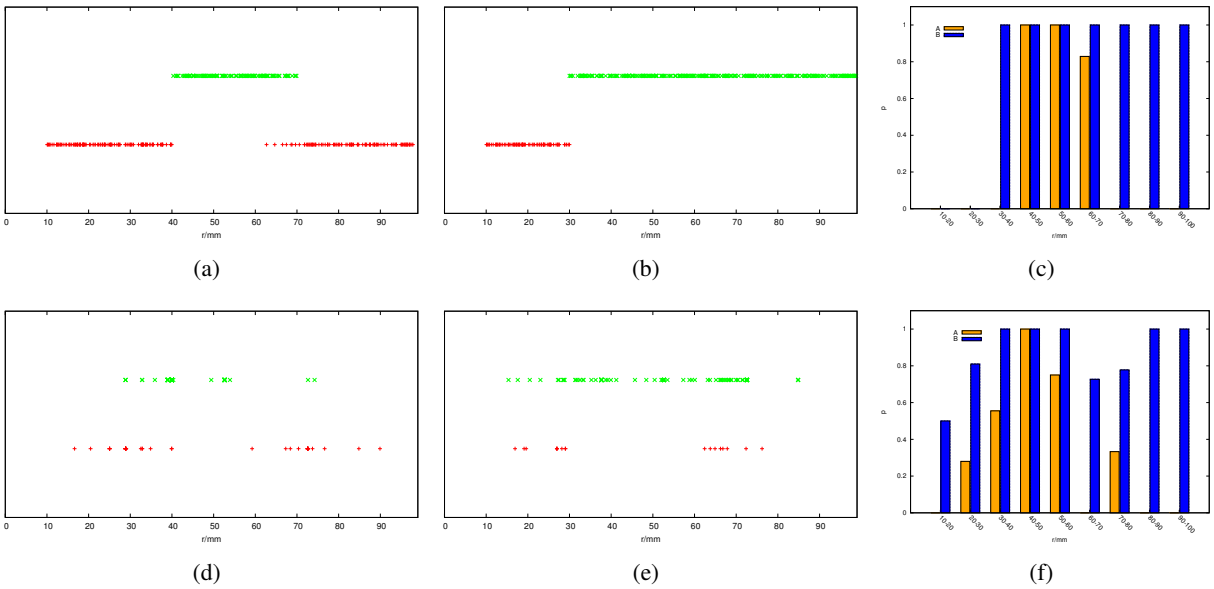
For instance, a category like "open/closed" which arises at the robot and mid levels, also has a counterpart at the planning level as a high-level property (i.e., a logical predicate). In this case, "open" is used by the planner in three different ways. First, it acts as a precondition to particular grasping operations (e.g., an object can only be grasped using Grasp Type A if it is open). Second, the planner can obtain information about the state of "open" by including a knowledge-producing action in a plan (e.g., the *sense-open* action requests information from the lower levels as to whether or not an object is "open"). Finally, "open" can be specified as part of a goal condition to be achieved by the planner (e.g., the planner can construct a plan to remove only open objects from the table). Together, the planner can use this representation to construct a plan such as:

$$\begin{aligned} &\mathit{sense\text{-}open}(\mathit{obj1}),\\ &\mathit{branch}(\mathit{open}(\mathit{obj1}))\\ &K^{+} : \mathit{graspA\text{-}table}(\mathit{obj1}), \mathit{putAway}(\mathit{obj1}).\\ &K^{-} : \mathit{nil}. \end{aligned}$$

Here, the planner first senses the openness of an object *obj1*: if it is open (the K^{+} branch) then it grasps *obj1* using Grasp Type A and clears it from the table; otherwise (the K^{-} branch), it leaves *obj1* unchanged. (We note that although the planner can reason about the outcome of an action like *sense-open*, the lower system levels must actually execute this action and return the results back to the planner.) The treatment of other lower-level properties and actions that are part of the abstracted high-level model is similar.

3. Object Learning, Object Categorisation and Pose Estimation

In its current state, the embodied system realises already the outlined three level hierarchy which is a suitable framework for our ongoing research. Our research now focusses on the extension of the system in terms of a richer embodiment (in particular on more sophisticated grasping devices), an extension to general objects and actions as well as on enriching of internal structures in the three level architecture.



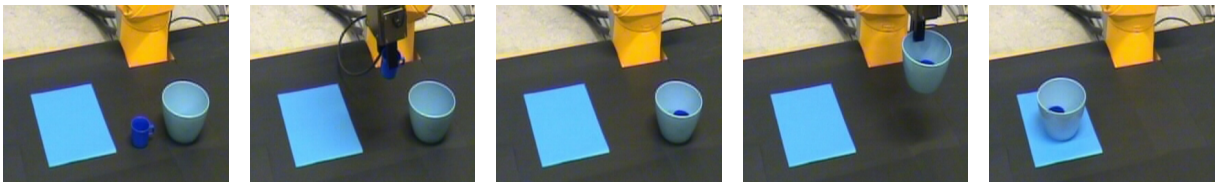
(a) (b) (c)

(d) (e) (f)

Figure 6: Grasp experiences and success distributions for two grasps associated to the circle part. The x-axis is representing the radius of the circle in mm. Grasp A represents a grasp where the gripper grasps the object with with fingers inside the circle, opening the fingers to grasp. Grasp B grasps the object on the brim of the circle. (a)-(c) show the results for artificial data, while (d)-(f) show data from real experiences. (a) and (d) show grasping experiences made for grasp type A for different radiuses while (b) and (e) show the same for grasp type B. In these diagrams the green crosses (in the upper row) represent a successfull grasp, while the red crosses (lower row) represent failures. (c) and (f) show success probabilities for the two grasp types in discrete radius bins. These can be used as a indicator which grasp to choose for an unknown grasping situation. A more detailed description of the circle grasp relationship can be found in Deliverable D4.1.2.

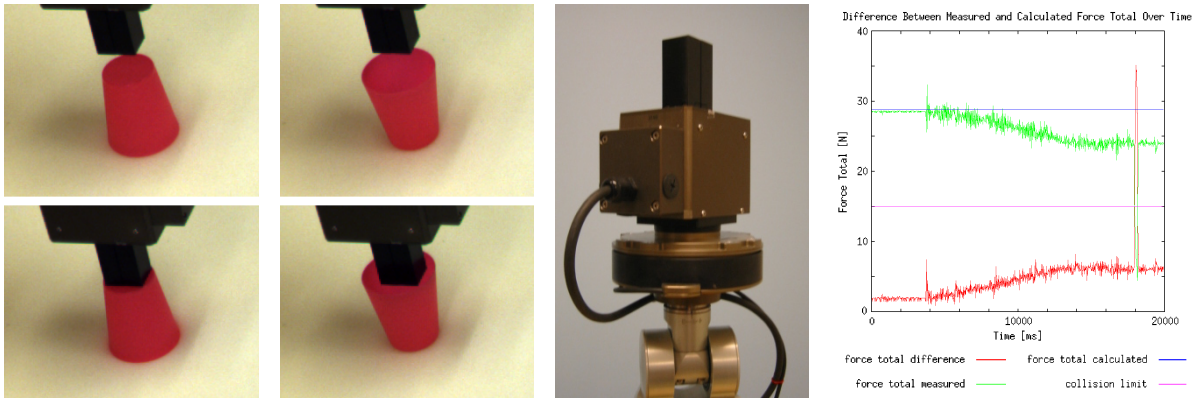
More specifically, we are currently working on the following extensions:

- The end–effector is limited to a two–finger parallel gripper and needs to be extended to a more flexible device. In this context we have been developping a finger that is fully equipped with tactile sensors (see Deliverable D4.1.2) by which we can detect categories such as openness and closedness as well as information about the weight of objects. We also equipped a five finger hand with the tactile sensors as described in section 3.1.
- The object domain is limited to objects including circles as parts and needs to be extended to arbitrary objects. The limitation concerns the learning of object representations, the association of actions as well as the recognition based on the learned representations. However, we want to stress that representing objects as a combination of parts has a number of advantages already useful in our limited system. In the context of the extension to arbitrary objects, we have
  - developed a system in which the combination of two OACs leads to the detection of the category 'objectness' as well as the extraction of object shape. This work has been published in [E].
  - improved the object shape extraction process by using Kalman filtering. This work has been described in [G] and section 3.2.
  - addressed the problem of object recognition using these extracted representations. This work has been described in [B] and is briefly described in section 3.2.2.
- The representation of learned objects need to be structures that allow for efficient storage and enable the robot to decide about what aspects of the object are relevant in a certain situation. In this context in [I] the problem of finding similar categories between object views is addressed to arrive at a more



(a) (b) (c) (d) (e)

Figure 7: Performing a Plan: (a) Initial scene. The small blue cup is *obj1*, while *obj2* stands for the light-blue bowl. The blue rectangle represents the shelf area. (b) Scene after executing *graspD-table*(*obj1*). (c) Scene after executing *putInto-object*(*obj1*, *obj2*). (d) Scene after executing *graspB-table*(*obj2*). (e) Scene after executing the final command *putAway*(*obj2*).



(a) (b) (c) (d)

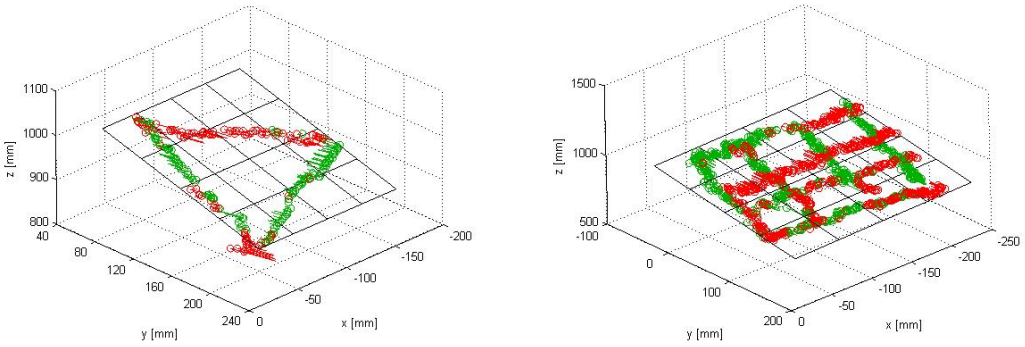
Figure 8: Open/closed detection using the force-torque sensor. The finger pokes into an potential object opening (here detected using the circle extraction described in [1]). By evaluating the force-torque sensor values it can be determined if the object is closed or open. (a) Depiction of the process without opening in the object. (b) Same for an object with opening. (c) Setup of the robot with the force-torque sensor. (d) Force graph for a collision situation. If the gravity corrected force (red line) exceeds a certain threshhold a collision is assumed, a poking action can be aborted and the object can be categorised as closed.

general representation of sets of objects (see section 3.3.1). In addition, the active use of different views of objects for object separation has been addressed in [H]. This approach is briefly described in section 3.3.2.

- A further limitation concerns the number of categories and properties being integrated into the system architecture which needs to be extended and linked to actions. We have addressed this issue in section 3.4.
- In addition, a cognitive system requires to learn the relations between objects and their affordances as well as the consequences of actions on different levels of the hierarchy. This is addressed in section 4.

3.1 Shape exploration with a five Finger Hand equipped with tactile Sensors

A two finger grasper naturally set limits to the exploration process in terms of the kind of objects that can be grasped or manipulated as well as in terms of tactile exploration. Hence, it is important to extend the embodiment to a more sophisticated manipulation device such as a five finger hand with rich tactile sensing for which also rich interactions with visual information can be realized. In this context, the sensor system for the humanoid robot hand was further developed at UniKarl. This comprised development and integration



(a) Triangle contour (b) Grid structure contour

Figure 9: Resulting point clouds for tactile contour following of planar shapes and corresponding fitted planes. Red points are situated below the plane, green points above, [A].

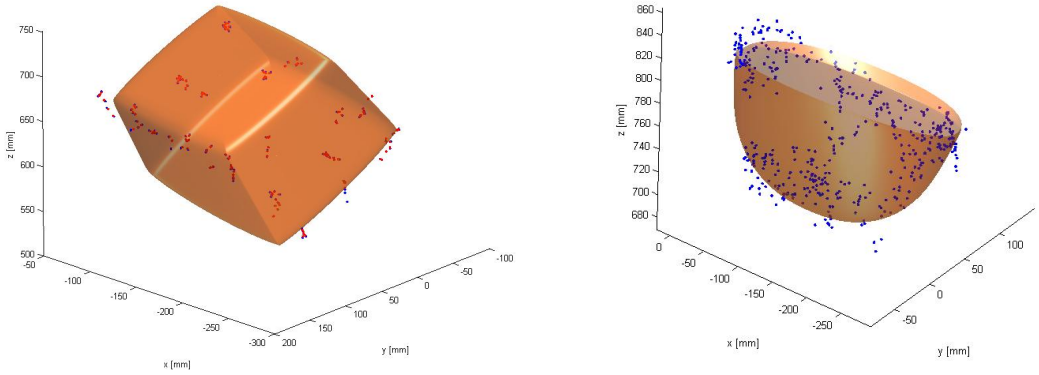


Figure 10: Surface exploration data and superquadric approximation for a box and a salad bowl.

of tactile sensor modules based on the commonly investigated technology. The details of the system are described in detail in Deliverable D1.1.1

Also, progress in processing and interpreting rich tactile sensor data was achieved. In [A] a framework for haptic exploration was presented which may be used for tactile data acquisition with both, a five-finger humanoid robot hand as well as with a human hand. The fusion of proprioceptive and tactile sensor data input acquired during haptic exploration was shown in experiments, see figure 9. The approach aims towards 3D shape recovery based on the haptic exploration data acquired by touching the object and following arbitrary trajectories on the surface.

The resulting scattered spatial point clouds of the explored objects were processed to extract geometric shape primitives using superquadrics, as shown in figure 10. The application of shape estimation techniques to sparse 3D point data lead to further investigations on how available contact normal information may be introduced in the estimation process. This is promising to stabilise the results and to establish a robust estimation process.

The hand is equipped with tactile and position sensors. Furthermore, joint torques can be derived from actuator pressure sensors. The humanoid robot hand is described in detail in Deliverable D1.1.1. This hand

will be used to investigate haptic exploratory procedures on the humanoid robot.

3.2 Work on Object Learning and Pose Estimation

Already in the Deliverable D8.1.2 we presented work on the extraction of 'objectness' as well as object shape — which we called the 'Birth of the Object — and confirmed grasping hypotheses (see figure 5 as well as [E]). We have done significant progress on this by stabilising and evaluating the initial grasping reflex for a number of complex scenes (see Deliverable D4.1.2) as well as the improvement of shape extraction as described in section 3.2.1. We also addressed the object recognition problem using the extracted representations, which is described in section 3.2.2.

3.2.1 Refinement of Learned Object Models

Categorisation, recognition, grasping as well as pose estimation profit from a precise representation of the learned objects. In this context, we have improved the extraction of the object models based on multi-modal primitives by combining a particle filtering process with a Kalman filter (for details, see [G]). The envisionned framework will make use of a Rao-Blackwellised particle filter [6] to correct estimated camera poses. Rao-Blackwellised particle filters make the assumption that visual landmarks are independent, allowing to correct landmark position using separate, small Kalman filters. This is advantageous because Kalman filtering variants increase quickly in complexity with the number of landmarks considered and are sensitive to outliers. A particularity of the approach is that the Kalman filtering does not only take place in the 3D position but also the 3D orientation space, i.e., we do Kalman filtering of the complete 3D-primitives' pose (6 DOF). This was evaluated using sequences with accurate known motion and lead to an improvement of the representation in two respects: 1) the filtering allowed to discard erroneous hypotheses and thus to reduce ambiguity; and 2) 3D-primitives were corrected over time, increasing their accuracy. Figure 11 shows the extracted representations with and without the Kalman filtering process.

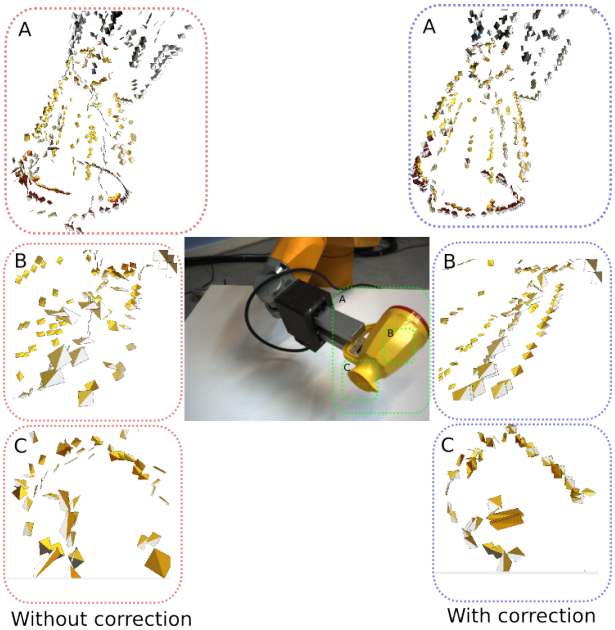


Figure 11: Illustration of the effect of Kalman based 3D-primitive correction over time. Inserts A, B, and C show details of the accumulated 3D-representation, with (right) and without (left) Kalman filtering of the primitives' pose.

3.2.2 Learning Feature Combinations for Pose Estimation

Object representations for pose estimation are based on the multi-modal primitives and are under development at ULg. The proposed representation has the form of a hierarchy of increasingly expressive object features. Bottom-level features are the most local, they correspond to the multi-modal primitives.

The hierarchy encodes probabilistic 3D relationships between neighboring features. It is learned by iteratively combining correlated features together. Instances are detected using a nonparametric belief propagation algorithm which propagates evidence through the hierarchy to infer globally consistent poses for every feature of the model. See figure 12 for an example result.

While we currently only use visual primitives as input, our method can in principle incorporate features from diverse perceptual modalities within a coherent framework. For example, vision plus haptic and proprioceptive inputs could be used simultaneously. This would produce cross-modal descriptions and cross-modal behaviours directly applicable to robotic tasks such as grasping and object manipulation. Our objective is to observe haptic and kinematic features that correlate with successful grasps, and integrate them into the feature hierarchy (see figure 4). Then, given a visual scene, grasp parameters can be suggested by probabilistic inference within the feature hierarchy.

3.3 Structuring Object Representations

In this section, we present results on processes which allow to efficiently store object representations and to apply them in the context of Object-Action Complexes. Throughout this section, objects are represented using aspect graphs [2] which cover rotational variations of objects. Sensorimotor processes that can be utilised to acquire the initial views of new objects have been developed in WP2 and will be reported in D2.1.4 [7].

3.3.1 Similarity based Object Categorisation

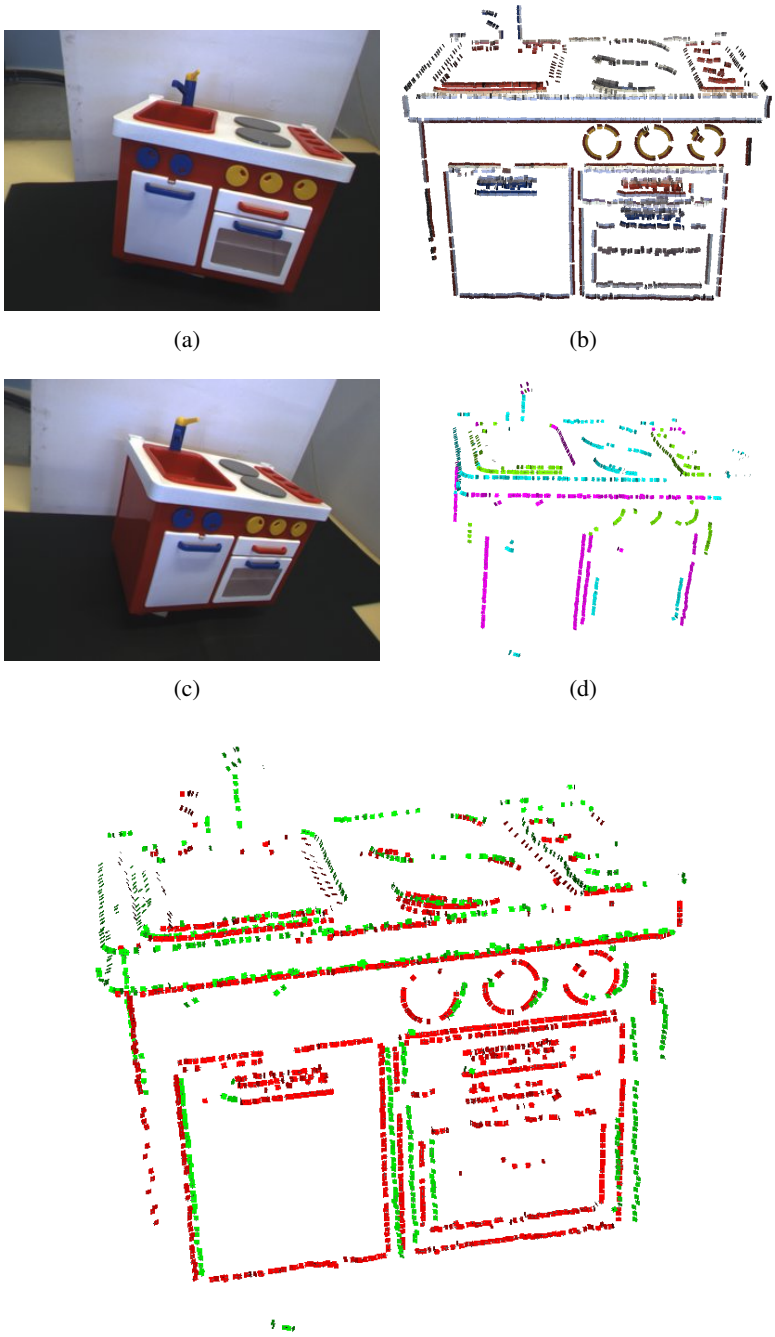
In order to store object representations in a compact way, similarities between objects and object views can be exploited. In [I] a representation scheme is introduced which allows for reduction of the storage requirements for object representation while maintaining the information about similarities between objects. This is achieved by selecting important views of objects, depending on their similarities among different views and multiple objects.

The approach consists of two stages:

- View clustering
  In the clustering step, prototypical views are extracted using unsupervised clustering techniques. We use the growing neural gas algorithm (GNG) [4] to cluster the feature space with respect to the input distribution.
- View labelling
  In the labelling step, objects are associated with the prototypical views. Each object representation references the prototypical views which are most similar to the views of the object.

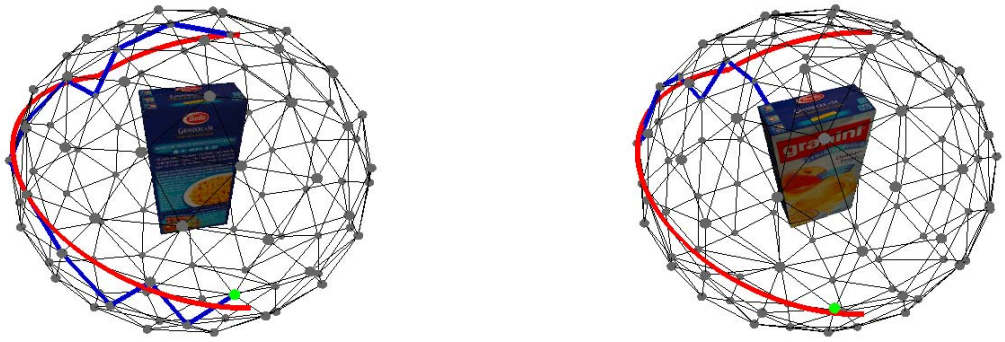
With only storing prototypical views, the storage requirement has been significantly reduced. In the experiments, 720 views of 10 objects were represented with about 20 prototypical views. The uncertainty about the currently perceived object could be reduced from 10 to 2.7 on the average.

In the experiments, we used colour histograms [3] as global descriptors of objects. However, the approach can be used without modifications for all feature descriptors which allow to determine the similarities be-



(a) (b) (c) (d) (e)

Figure 12: Learning Feature Combinations for Pose Estimation (a) Input scene used for learning the object model: toy kitchen. (b) Input scene in (a) represented by multi-modal primitives. (c) Input scene where our system will locate the toy kitchen. (d) Instantiation in (c) of the hierarchy learned from (a). Positions of the bottom-level features (higher levels are not shown). The bottom-level features correspond to input multi-modal primitives, colour indicating correspondences. Note that this representation is sparse compared to (b), which demonstrates the kind of noise our system is robust to. (e) Illustration of the pose estimation accuracy. The picture shows in green the ground-truth pose of the kitchen in scene (c) and in red its pose inferred by our system.



(a) The path hypothesis for the correct object representation approximates the controlled path.

(b) The path hypothesis for the incorrect object representation does not converge to the controlled path.

Figure 13: Object Separation

tween views. With this in mind, the approach can be applied to different visual modalities in order to further reduce the uncertainty during perception. This uncertainty can also be reduced by using active vision methods as described in the next section (Section 3.3.2).

3.3.2 Active Object Separation

Daily life objects reveal natural similarities, which cannot be resolved with the perception of a single view of an object. In [H] we present a method for the separation between objects making use of active methods and taking into account multiple object views. By actively rotating the object, i.e. having physical control over the object, the coherence between controlled action path, acquired object representations (inner models) and percepts (object views) are observed over time and allows to reject implausible path hypotheses. At the same time, the controlled path can be registered on the surface of the aspect graph [2] of each plausible object hypothesis. Once the percept is registered, a movement is generated which reveals the most separating view of the remaining object hypotheses in order to perform robust object separation.

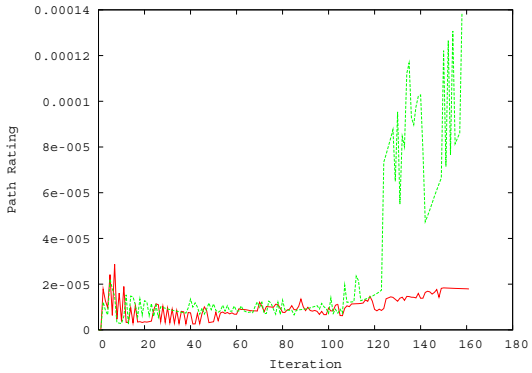
Our approach determines hypotheses for the pose of each aspect graph representation. In each iteration, the hypothesis is updated and rated on the basis of the coherence between controlled action path, object representations and current percept.

The approach uses three different measures to rate path hypotheses:

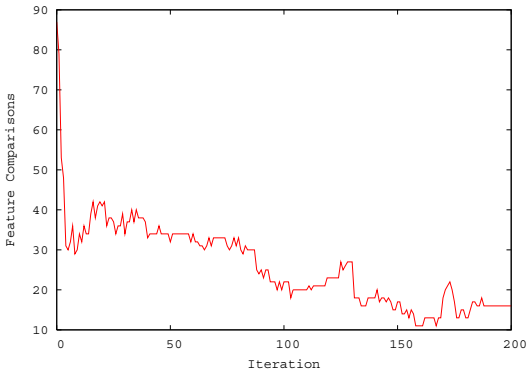
- Similarity of the current percept with the feature associated to the current view from the aspect graph as predicted by the pose hypothesis.
- Similarity of the course of the controlled path and estimated path on the surface of the aspect graph.
- Similarity of past views and the corresponding views from the object representation, determined by accumulation over time along the hypothesis path course.

Figure 13 shows the course of the controlled path (red) and the best path hypothesis (blue) for two box-shaped objects which have a different texture on one side. The left object was perceived during a separation task. The controlled path is approximated well by the path hypothesis for the correct object (Figure 13(a)). In contrast, the path hypothesis of the incorrect object is not coherent with the controlled path (Figure 13(b)). The rating of both paths is shown in Figure 14(a). Once an object view is perceived and it differs from the inner model, the patch rating drops. The rating can be deployed to separate between different objects.

Figure 14(b) shows the number of feature comparisons required in the course of the approach. Compared to a brute-force approach, only few comparisons have to be performed.



(a) The path rating measure is shown for two object hypotheses with different backsides. Once the backside is revealed through the control movement, the rating of the invalid hypothesis increases.

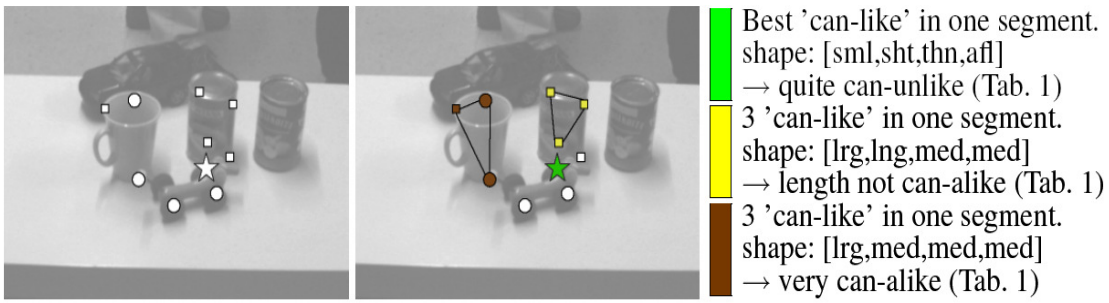


(b) Number of feature comparisons required in relation to the iterations of the algorithm. With convergence of path hypotheses the number of required comparisons decreases.

Figure 14: Experimental results using the proposed approach.

3.4 Using Visual and Shape Attributes for Object Categorisation

In [C], we present a visual subsystem in which we integrated both visual and shape attributes towards the concept of OACs. The attribute determination is part of a more general vision system which can support also other applications, as briefly proposed in the paper. For the learning of OACs, we focus on the issue of meaningful attributes that constitute an object as opposed to a thing. While finding complex attributes like hollowness or emptiness is hard using vision only, we start by collecting very basic visual 2D and 3D attributes. The presented system computes both visual (from attention) and shape (from 3D segmentation) attributes, as listed in Table 1, combines and uses their advantages in a supplementing manner and also provides a fundament of attributes that can serve several higher-level learning and planning processes. Combining pure visual attributes with shape attributes has practically been exemplified by validating visual hypotheses according to shape attributes. Hereby, it is possible to neglect wrong hypotheses, to cluster and affirm the good ones (see Figure 15), or even to distinguish between a car-like object and the plain image of it.



"Find the 'mango can'-like."

Figure 15: Left: A set of best hypotheses for the described task (2D visual attention). Right: shape attribute information (3D segmentation) allows for grouping and validating of hypotheses by comparison of qualitative shape attributes.

Object	Visual Attributes		Shape Attributes			
	Color	Orientation	Area	Length	Width	Height
	R G B Y	0° 45° 90° 135°	tin sml lrg hug	sht med lng vln	thn med wid vwd	afl med hig vhg
Car (4x)	(12,5,40,10)					
Dog (3x)	(35,22,11,13)					
Giraffe (2x)	(64,1,24,48)					
Mango (6x)	(12,60,21,23)					
Mug (2x)	(57,83,29,113)					
Peach (4x)	(13,8,15,12)					
Sugar (2x)	(28,6,2,14)					

Table 1: Attribution distributions of 7 test objects. In brackets, each object carries the number of its appearance along the test sequence of 6 scenes.

Besides the different attribute detectors, the framework consists of a server database which holds a set of attribute classes (e.g. height, size, colour) and corresponding attribute instances (e.g. small, medium, large for size). One can change or extend this very simply in case that new attribute detectors (e.g. for mass) become available. An agent (in our case it is only the vision system client) can access the server, ask for available attributes classes, as also insert or request a perceived attribution, i.e. an object. The current system server is able to notice and reply if this object and its attribution, respectively, has been seen before. However, as we have not yet introduced manipulative capabilities and feedback in practice, the system is not yet able to connect and learn actions from attributes. However, it offers a fundamental technique to produce the necessary attributes for such learning issues.

Categories of objects will thereby not be built by visual appearance only, but by the actions an agent can perform and the attributes it can perceive. The core of the OAC concept is constituting objects from a set of attributes, which can be manifold in type (e.g. colour, shape, mass, material), to manipulative actions. Including manipulation attributes, e.g. hollowness or weight, by interacting with a thing, is one issue of future work.

4. Learning Action Effects

Although the planner is a powerful tool for controlling the robot's actions, it relies on an accurate model of the dynamics of the world in which the robot will operate. Using state information provided by the lower levels, and returned to the high level, the planner can improve its domain model (and thus, the quality of its plans) by applying machine learning techniques. In particular, we are investigating an approach based on a kernel perceptron learning model, where action and state information is encoded in a compact vector representation as input to the learning mechanism, and resulting state changes are produced as output. Currently, our approach can only be used to learn STRIPS/LDEC-style action effects. However, we are in the process of adapting it to action preconditions. Empirical results indicate efficient training and prediction times, with low average error rates ($< 3\%$). Details of this work can be found in [5].

5. Links to other Workpackages

Deliverable D8.1.4 is linked to and makes use of work made in a number of workpackages. It is linked to the software and hardware integration issues dealt with in WP1. There are potential links to be exploited in terms of grasp evaluation and optimal actions for object learning in WP2. In Deliverable D8.1.1 a number of sub-modules are used that have been developed in WP4 (see Deliverable D4.1.2) and the integration with the higher level planning system (see WP5).

Attached Papers

[A] A. Bierbaum, K. Welke, D. Burger, T. Asfour, and R. Dillmann. Haptic exploration for 3D shape reconstruction using five–finger hands. In *IEEE-RAS International Conference on Humanoids (Humanoids)*, 2007.

[B] R. Detry, N. Pugeault, and J. H. Piater. Probabilistic pose recovery using learned hierarchical object models. Technical Report 2008-01-25, INTELSIG, ULg, 2008.

[C] K. Huebner, M. Björkman, B. Rasolzadeh, M. Schmidt, and D. Kragic. Integration of visual and shape attributes for object action complexes. In *International Conference on Computer Vision Systems (ICVS)*, 2008. (accepted).

[D] D. Kraft, E. Başeski, M. Popović, A. M. Batog, A. Kjær-Nielsen, N. Krüger, R. Petrick, C. Geib, N. Pugeault, M. Steedman, T. Asfour, R. Dillmann, S. Kalkan, F. Wörgötter, and B. Hommel. Exploration and planning in a three level cognitive architecture. In *International Conference on Cognitive Systems (CogSys)*, 2008. (submitted).

[E] D. Kraft, N. Pugeault, E. Başeski, M. Popović, D. Kragic, S. Kalkan, F. Wörgötter, and N. Krüger. Birth of the object: Detection of objectness and extraction of object shape through object action complexes. *Special Issue on "Cognitive Humanoid Robots" of the International Journal of Humanoid Robotics*, 2008. (accepted to).

[F] R. Petrick, C. Geib, and M. Steedman. A scenario for integrating low-level robot/vision, mid-level memory, and high-level planning with sensing. Technical report, 2008.

[G] N. Pugeault, F. Wörgötter, and N. Krüger. Object model learning using bayesian filtering. Technical Report 2008 – 3, Robotics Group, Maersk Institute, University of Southern Denmark, 2008.

[H] K. Welke, T. Asfour, and R. Dillmann. Object separation using active methods and multi-view representations. In *Proceedings of the IEEE Int. Conf. on on Robotics and Automation (ICRA 2008)*, 2008. (accepted).

[I] K. Welke, E. Oztop, G. Cheng, and R. Dillmann. Exploiting similarities for robot perception. In *Proceedings of the IEEE Int. Conf. on Intelligent Robot Systems (IROS 2007)*, 2007.

References

[1] E. Başeski, D. Kraft, and N. Krüger. A hierarchical 3D circle detection algorithm applied in a grasping scenario. Technical Report 2008–2, Robotics Group, Maersk Institute, University of Southern Denmark, 2008.

[2] C. M. Cyr and B. B. Kimia. A similarity-based aspect-graph approach to 3D object recognition. *Int. J. Comput. Vision*, 57(1):5–22, 2004.

[3] S. Ekvall and D. Kragic. Receptive field cooccurrence histograms for object detection. In *Proceedings of the IEEE/RSJ International Conference Intelligent Robots and Systems*, 2005.

[4] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, volume 7, 1995.

[5] K. Mourão, R. Petrick, and M. Steedman. Using kernel perceptrons to learn action effects for planning. In *International Conference on Cognitive Systems*, 2008. (submitted to).

[6] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*, volume 11, 1999.

[7] A. Ude, D. Omrčen, and G. Cheng. Making humanoid object learning and recognition an active process. *Special Issue on "Cognitive Humanoid Robots" of the International Journal of Humanoid Robotics*, 2008. (to appear).

Haptic Exploration for 3D Shape Reconstruction using Five-Finger Hands

A. Bierbaum, K. Welke, D. Burger, T. Asfour, and R. Dillmann
University of Karlsruhe
Institute for Computer Science and Engineering (CSE/IAIM)
Haid-und-Neu-Str. 7, 76131 Karlsruhe, Germany
Email: {bierbaum,welke,asfour,dillmann}@ira.uka.de

***Abstract*—In order for humanoid robots to enter human-centered environments, it is indispensable to equip them with the ability to recognize and classify objects in such an environment. A promising way to acquire the object models necessary for object manipulation appears in the supplement of the information gathered by computer vision techniques with data from haptic exploration. In this paper we present a framework for haptic exploration which is intended for use with both, a five-finger humanoid robot hand as well as with a human hand. We describe experiments and results on haptic exploration and shape estimation of 2D and 3D objects by the human hand. Volumetric shape data is acquired by a human operator hand using a data glove. The exploring human hand is located by a stereo camera system, whereas the finger configuration is calculated from the glove data.**

I. Introduction

In humans, different types of haptic exploratory procedures (EPs) for perceiving texture, weight, rigidity, contact, size and the exact shape of a touched object are known [1]. These EPs require the exploring agent to initiate contact with the object and are therefore also referred to as active touch sensing.

In this paper, the contour following EP for shape recovery is subject of interest. A volumetric object model composed this way delivers a rather high amount of information for discriminating between objects. Also, volumetric object data is most suitable for supplementing and verifying geometric information in multimodal object representations.

Several approaches have been proposed for acquiring object shape information by robots through haptic exploration. An early, comprising experimental setup was presented in [2], where a dextrous robot hand was used in conjunction with a manipulator arm. The hand probed contact by enclosing the test objects at predefined positions and evaluating joint angle and force readings. The resulting sparse point clouds were fitted to superquadric models defined by a set of parameters describing shape and pose.

In addition to the contact locations, the contact normal information gathered during haptic exploration was used in [3]. Instead of a superquadric model here a polyhedral model was chosen as volumetric object representation. For object recognition the Euclidian distances of the polyhedral surface points to the borders of a surrounding cubic workspace box were measured at equidistant coordinates and matched to those of synthetic models. This approach was evaluated only in simulation.

The basic kinematics of contact and its application in contour following are presented thoroughly in [4]. Moreover, several procedures for haptic exploration of object features have been investigated in [5], [6]. Other approaches in active contact sensing concentrate on the detection of local surface features [7].

In our approach, a framework has been developed that allows haptic exploration of unknown objects for recovery of the exact global shape using different types of manipulators equipped with contact sensing devices. This separates our work from approaches involving model based pose estimation of known objects as in [8].

As we are interested to integrate the framework as basis for haptic exploration in our humanoid robot system [9] which is equipped with two five-finger human-like and human-sized hands, we focus on the application of exploring with five-finger hands. In particular, the developed framework allows us to use the human hand of an operator as exploring manipulator by deploying a data glove with attached tactile sensors. This gives us the opportunity to already investigate the EPs of interest until the humanoid robot hand is fully integrated into our robot. It also provides rich possibilities for immediate comparison of haptic exploration results by a human versus a humanoid robot hand.

Our aim is to establish a robust exploratory process for 3D shape reconstruction and modeling which can be performed in an unstructured environment, while only providing observability of the hand pose, the finger configuration and the contact information. Currently we are limited by the constraint that the object being explored must remain in a fixed pose.

As modeling primitive we chose an extended superquadric function as in [2] which can represent a variety of cubical and spherical geometries. In the context of grasp planning this type of representation has just recently been investigated for modeling physical objects by superquadric decomposition [10]. Yet, in this paper we only address modeling of basic superquadric shapes. Also, we can not give results related to non-convex objects at this stage of our work, as those are not reflected by the chosen model, although the exploration process itself is not limited to convex objects.

This paper is organized as follows. In the next section

the relevant details and components of our system for haptic exploration focusing on object shape recovery are described. This includes a description of the human hand model and visual tracking of the operator's hand. In section III we present an evaluation of the system in terms of haptic exploration on real world objects. Finally we give a conclusion and an outlook on our future work in sectionIV.

II. System description

Figure 1 gives a system overview with the components involved in acquisition of contact points during contour following EP.

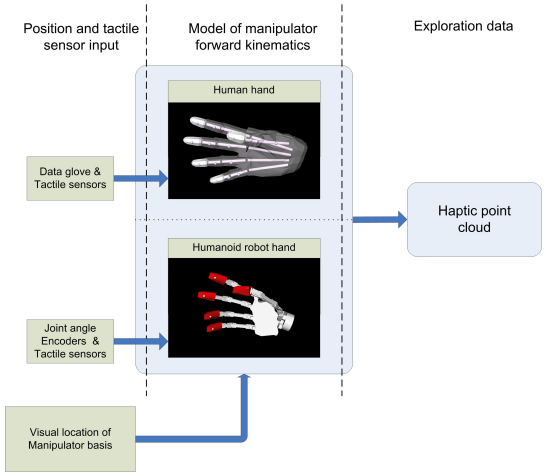


Fig. 1. System for acquisition of object shape data from haptic exploration using a human or a humanoid robot hand as exploring manipulator.

During haptic exploration with the human hand, the subject wears a data glove that serves as an input device for calculating the joint angle configuration of the hand. The data glove we use is equipped with binary micro switches at the distal phalanges. When touching an object, the switches are actuated as local contact force exceeds a given threshold. During actuation the clicking of the switch also provides the operator with a mechanical sensation, that helps to control the contact pressure during exploration. The data glove is made of stretch fabric and uses 23 resistive bend sensors that provide measurement data of all finger joint angle positions and the orientation of the palm.

Before starting exploration the operator needs to calibrate the data glove sensors with the forward kinematics of the underlying hand model. Subject to calibration are abduction/adduction and flexion/extension of all fingers, curvature of the palm and the thumb motion.

A linear relation is used for projecting glove sensor readings to joint angles. Calibration is accomplished by engaging positions which result in minimum and maximum sensor response in a separate calibration procedure.

Wrist position and orientation are determined in the reference frame of the camera coordinate system as described in section II-B. During exploration the subject visually guides the finger tips following desired paths. When the micro switch is actuated by touch, the current location of the sensor in the global reference frame is registered as a point in the 3D object point cloud. The sensor position is calculated using the forward kinematics of the hand model as described in the following section.

For exploration with our humanoid robot platform Armar-III we may later use a model for the forward kinematics of the robot hand as presented in [11]. The robot is equipped with an advanced version of this hand with joint angle encoders attached to all controllable degrees of freedom (DoF) and tactile sensors at the fingertips.

The data acquired as 3D point coordinates in the haptic point cloud set is finally used for performing a superquadric model estimation.

A. Human hand model

The forward kinematics of the human hand must be modeled accurately to transform the coordinates of the tactile sensor locations gathered from the data glove sensor readings to a global reference frame in which the resulting 3D contact point cloud is accumulated. Furthermore, the model is required to cover the entire common configuration space of the human hand and the data glove, so that the human operator is preferably not restricted in the choice of hand movements that can be projected to the model.

A complex hand model deploying 27 DoFs was introduced in [12] and used in several studies requiring exact models ([13], [14]) for hand pose estimation from sensor input. The Carpometacarpals joints (CMC) were fixed, assuming the palm to be a rigid part of the hand. The fingers were modeled as serial kinematic chains, attached to the palm at the metacarpophalangeal joints (MCPs). Interphalangeal joint (IP), distal interphalangeal joints (DIP) and proximal interphalangeal joints (PIP) have one DoF for flexion and extension. All MCPs joints have two DoFs, one for flexion and extension and one for abduction and adduction. The CMC joint of the thumb is modeled as a saddle joint. Several variants of this model exist in literature (see [15], [16]).

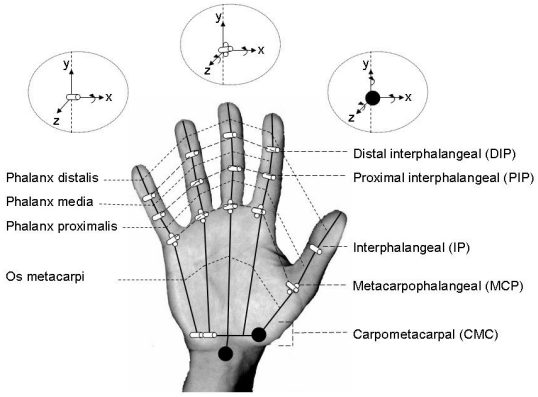


Fig. 2. The hand model for haptic exploration by a human operator wearing a data glove.

The hand model which we use in the presented framework is shown in Fig. 2. We have added two modifications to the basic model to improve the representation of real human hand kinematics. The first modification affects the modeling of the thumb's CMC joint. Following [16], the first metacarpal of the thumb performs a constrained rotation around a third orthogonal axis in the CMC joint, which contrasts the CMCs joint model as a two DoF saddle joint. For reasons of simplicity we model this joint as a three DoF joint.

The second modification is to overcome the inadequate representation of the palm as a rigid body. As we want to incorporate the distal phalanges of the ring and little finger in the exploration process, we have extended the model by adding one DoF at the CMCs of these fingers respectively. By doing this the ability of the palm is reflected to fold and curve, when the little finger is moved towards the thumb across the palms inner side [17]. It is important to model this behavior as a human operator will occasionally utilize these types of movement when touching an object with the whole hand.

The resulting hand model consists of 26 DoFs. The four fingers have 4 DoFs each at the DIP, 4 DoFs at the PIP and 8 DoFs at the MCPs. The thumb is modeled with 1 DOF at its IP, its MCP is modeled with 2 DoFs and its CMC, as mentioned before, with a 3 DoF joint. Additionally we model the palm with 2 DoFs representing the CMCs of the little and ring fingers and add 2 DoFs for the wrist movement.

We have used the Open Inventor™[1] standard for constructing the hand model. This 3D modeling package allows the implementation of local reference frames as described before in a transparent way.

B. Wrist tracking

As mentioned earlier, the absolute position and the orientation of the data glove are determined using vision. In order to track the data glove in a robust manner, we use a marker bracelet which is attached to the wrist of the subject wearing the data glove and is tracked using a stereo camera system. Figure 3 shows the marker bracelet used for our experiments. The bracelet comprises twelve red markers for tracking and one green marker for initialization. All markers are printed on yellow background. The wrist localization consists of two phases. In the initialization phase, the pose of the wrist is estimated without the knowledge of past poses. Once an initial pose has been calculated, a particle filter approach is used to track the pose of the wrist.

For the initialization, the relevant markers are identified by HSV color segmentation performed in the current scene. Only green and red markers inside yellow areas are considered for color segmentation. With the calibration of the stereo camera system, the 3D coordinates of the green marker and the second and third closest red markers are calculated. Since these markers lie all in the same plane, the plane normal can be calculated from this information. With the radius of the bracelet known, the center of the circle described by the three considered markers can be calculated. The x-axis of the coordinate system (denoted red) can be derived from plane normal and center. The y-axis (denoted green) is calculated from the difference of the green marker and the center. The z-axis is calculated as the cross product of x- and y-axis.

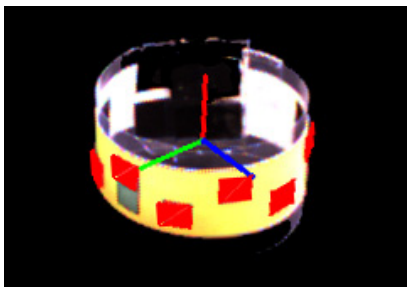


Fig. 3. Bracelet and coordinate system.

After initialization, the bracelet is tracked using a particle filter [18] based on a model of the bracelet comprising all 12 red markers. The configuration of the model is defined by the 6D pose of the band. In each iteration of the particle filter algorithm 100 new configurations are generated using a gaussian distribution with variance $\sigma^2 = 0.45$. Furthermore the movement of the model is estimated by taking into account the movement in the previous frame. In order to retrieve the estimated 6D pose of the wrist band, the visible part of the model is projected into both camera images using the camera calibration. To validate each particle, a weighting function is used which compares the segmentation mask for the red color with the model. In order to derive a weighting function for each configuration, we count all red pixels inside yellow regions f and all red pixels, which overlap with the projected model m. The probability for each particle z and the current images i can then be formulated in the following way:

$$p(z|i) \propto exp\left(\lambda * \frac{m}{f}\right)$$

where λ defines the sector of the exponential function which is used. After all particles have been weighted according to this equation the 6D pose is estimated by the weighted sum of all configurations.

C. Superquadric fitting for shape estimation

The concept of superquadrics has been introduced as a family of parametric 3D shapes in [19], among which the superellipsoid has become the most popular one and therefore is often termed as superquadric, a convention we will preserve here.

A superquadric centered in the origin and with its axes aligned to the x, y, z coordinate axes can be described with the following parametric equation

$$\chi(\eta,\omega) = \begin{pmatrix} a_1 \cos^{\varepsilon_1}(\eta) \cos^{\varepsilon_2}(\omega) \\ a_2 \cos^{\varepsilon_1}(\eta) \sin^{\varepsilon_2}(\omega) \\ a_3 \sin^{\varepsilon_1}(\eta) \end{pmatrix} \quad .$$

[1] http://oss.sgi.com/projects/inventor/

The parameters a_1, a_2, a_3 describe the extent of the superquadric along each axis. The exponents ε_1, ε_2 ϵ $[0..2]$ produce a variety of convex shapes and describe the shaping characteristics from cubic to round in x and y directions. This way different 3D primitive shapes can be modeled, e.g. boxes $(\varepsilon_1, \varepsilon_2 \approx 0)$, cylinders $(\varepsilon_1 = 1, \varepsilon_2 \approx 0)$ and ellipsoids $(\varepsilon_2 = 1)$.

To locate the superquadric arbitrarily in space, we further introduce a rotational matrix $\mathbf{R}$ and a translation vector $\mathbf{x_0}$, which add 6 more parameters to our model.

As superellipsoids are restricted to symmetric shapes only, we also add deformation parameters $\{t_x, ty$ ϵ $[-1..1]\}$ for modeling tapering in z direction as described in [20]. This enables our model to also represent wedge resembling shapes. Applying a scaled tapering deformation function

$$D_t(x,y,z) = \begin{pmatrix} t_x \frac{z}{a_3} + 1 \\ t_y \frac{z}{a_3} + 1 \\ 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

we finally get the model function

$$\mathbf{m} = \mathbf{R} D_t(\chi(\eta, \omega)) + \mathbf{x_0} \quad .$$

To estimate the 11 parameters of our superquadric model from the 3D contact point data we use the Levenberg-Marquardt non-linear least-squares algorithm [21] to minimize the radial Eucledian distance d between the data points and the superquadric surface

$$d = \|\mathbf{x}\| \left(1 - \frac{1}{F(\mathbf{x})}\right) \quad .$$

as proposed in [22]. Here, $F(\mathbf{x})$ is the inside-outside function of the superquadric, which has a value of 1 for points $\mathbf{x} = (x, y, z)^\top$ on the surface of the superquadric, while points inside result in $F < 1$ and points outside result in $F > 1$.

III. Experiments for object shape recovery

For evaluation of the system described above we have performed experiments related to the exploration by a human subject. The experimental setup hereby is shown in Fig. 4.

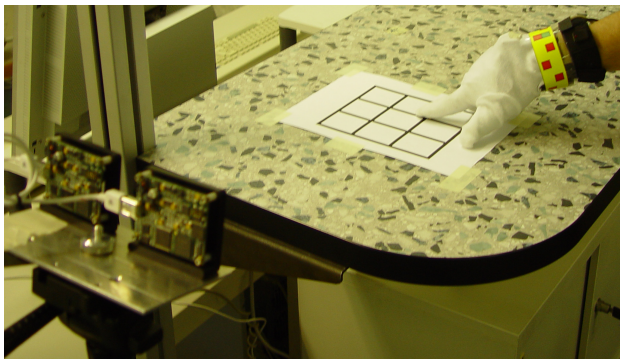


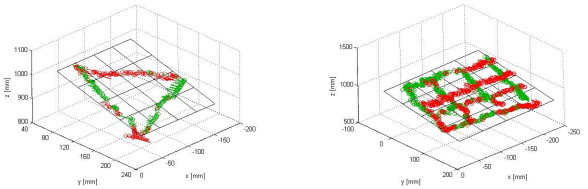
Fig. 4. Experimental setup. Here a planar grid structure is subject to contour following.

The region viewable by the stereo camera system defines the workspace in which the objects to be explored have to reside. The human operator wears a data glove with the wrist marker attached and performs a contour following EP with the tip of the index finger upon the object, i.e. the human operator visually guides the contact sensor along the contours. As mentioned earlier, the micro switch for contact sensing is also located at the distal phalanx. During the EP the object is fixed within the workspace. The human operator is allowed to move hand and fingers arbitrarily within the workspace as long as the marker bracelet may be visually detected and localized. In case the marker localization fails, the subject needs to move the hand until it can be detected again.

A. 2D contour following in 3D space

As an initial experiment we chose to follow the visible contours of a planar structure to verify whether the exploration system delivers length and angle preserving point clouds. These properties were inspected visually from the resulting point cloud data. We calculated the PCA for all points in the point cloud for approximation of the plane the structures are located in. Further, we determined the standard deviation of the point locations in respect to this plane.

As planar shapes we chose a circle, an isoceles triangle and a 3×3 grid. The edge length of the bounding box for each of these shapes was set to $160mm$. The subject followed the contours of a printout of the respective shape with the index finger. Resulting point clouds for triangle and grid contours are shown in Fig. 5. The figures show that the contours of the test shapes are situated in different planes, which originates from a change in the position of the camera system between the two explorations. The duration of the exploration process was 40 seconds for the circle and triangle shapes and 2 minutes for the grid structure. Exploration speed was mainly limited by the performance of the wrist tracking algorithm.



(a) Triangle contour (b) Grid structure contour

Fig. 5. Resulting point clouds for tactile contour following of planar shapes and corresponding fitted planes. Red points are situated below the plane, green points above.

During circle contour following 245 contact points were acquired, the standard deviation to the fitted plane was calculated to $\sigma = 5.37mm$. For the triangle contour following exploration 259 data points were acquired with $\sigma = 6.02mm$. For the grid exploration finally, 1387 data points were acquired with $\sigma = 5.86mm$.

B. Edge tracking of a 3D object

In this experiment the subject had to follow the edges of a rectangular box situated on a table in the workspace. The exploration procedure delivered 1334 data points, the resulting

point cloud is depicted in Fig. 6. The box dimensions were $150 \times 50 \times 120mm$ and therefore in the range of the dimension of the human hand itself.

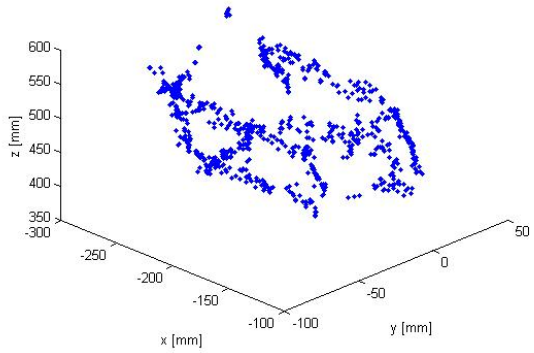


Fig. 6. Resulting point cloud for haptic contour following of a rectangular box.

It is not possible to directly fit exploration data to a superellipsoid which comprises only points from the edges of an object as the data is ambiguous in describing the adjacent surfaces without having additional surface normal information. Yet, we wanted to proof the systems capability to generate contiguous 3D point clouds of real world objects.

C. Arbitrary touch exploration and superquadric fitting of a 3D object

In a further experiment the exploring subject acquired contact coordinate information by arbitrarily touching all reachable surfaces of the objects while no preference was given to the edges. For exploration a different box and an upside-down placed salad bowl were chosen. As all fingers were involved the exploration process could be performed within less than a minute while acquiring still enough data points. During the exploration it was considered not to acquire too many data points as this significantly extends the amount of calculation time for the superquadric estimate. From the acquired data approximating superquadric representations were successfully estimated, as shown in figure 7.

The estimation algorithm could also handle incomplete surface point data as in case of the box, where it was not possible to acquire point data from the bottom side on which it was situated. The acquired data set comprised 176 points.

For the salad bowl the resulting data set comprised 472 points. As the superquadric function involved naturally only describes convex shapes the approximation of the salad bowl was rendered in the figure only in the representative range $\omega \, \epsilon \, [-\pi..0]$, which describes one half of the superellipsoid.

For the objects explored the extension coefficients $a_{1,2,3}$ of the superquadric model were in good correspondence to the dimensions of the real object.

It can be seen in the result plots, especially figure 6 and 7, that the haptic point cloud exhibits basic noise and outliers

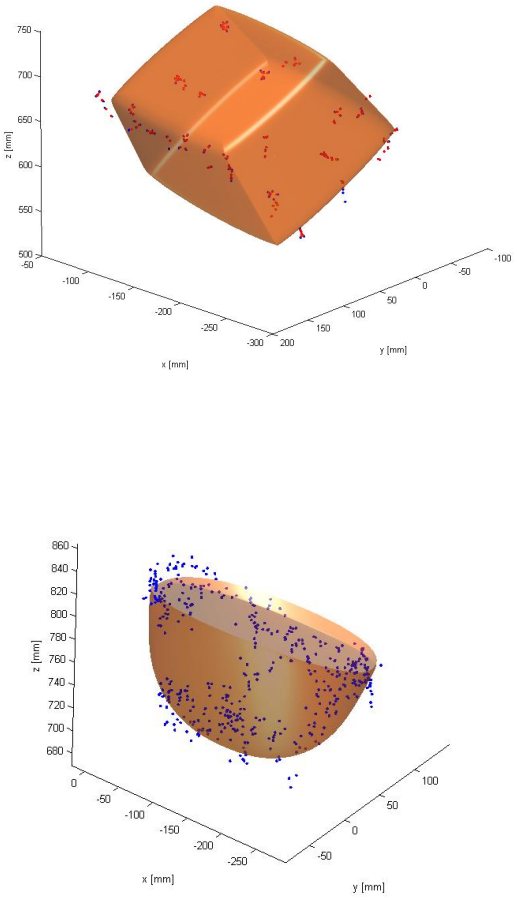


Fig. 7. Surface exploration data and superquadric approximation for a box and a salad bowl.

in some regions. From these experiments we found several reasons to affect the quality of the resulting point clouds:

1) The parametrization of our human hand model introduces a static error which could be minimized by adapting the model to the data glove's basic dimensions. Yet, hands of different individuals lead to irregular stretching of the glove fabric and occasionally to misalignment between finger joints and strain gauge sensors. This introduces a static model error which is not covered by the calibration process. We decided not to address this issue as our final goal is the operation of the system with a humanoid robot hand, where these problems cease to apply.
2) The measurement signal of the glove's strain gauge sensors is naturally afflicted with background noise which limits the resolution of physical features. We also expect this type of measurement noise for the joint angle sensors of our robot hand.
3) Beside the above, the pose data from visual tracking is superimposed by anisotropic interference as the estima-

tion of the wrists z-coordinate shows a significant higher amount of noise level than for the x- and y-coordinates, which is natural for stereo-camera systems. Also, some noise arises in the pose estimation, as the bracelet still has some clearance to move when worn by an individual. Further, estimation uncertainty increases if only a low number of markers can be detected in the scene, e.g. due to lighting conditions. The latter problems will become diminished when using a robot arm for exploration, as we get additional pose information from joint angle sensors and attach the marker band in a stable position. This can be used in conjunction with the visual tracking system using data fusion.

IV. Conclusion and Future Work

In this paper we presented a framework for acquiring and estimating volumetric object models via haptic exploration by contour following. In a first evaluation results for haptic exploration of 2D and 3D shapes by a human subject wearing a data glove in an unstructured environment were described.

It could be shown that the underlying human hand model and the data acquisition process is sufficiently precise enough to acquire contact position data when exploring the shape of objects having the magnitude of dimension as the human hand does, while the exploration process could be performed using the modeled set of degrees of freedom of the human fingers. Further we could demonstrate fitting the resulting contact data to superquadric shapes.

As a next step we will address the transfer of the haptic exploration framework to our humanoid robot platform. The platform already incorporates the same stereo vision system as used for the experiments described in this paper. A tactile sensor system for the robot hand has been developed that provides more information over the binary switches deployed with exploration by a human.

Hence, the focus of our work will move to the development of autonomous and robust visually guided haptic exploration strategies for shape recovery by a humanoid robot with five-finger hands.

Acknowledgement

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission and the German Humanoid Research project (SFB 588) funded by the German Research Foundation (DFG: Deutsche Forschungsgemeinschaft).

References

[1] Susan J. Lederman and Roberta L. Klatzky, "Hand movements: A window into haptic object recognition," *Cognitive Psychology*, vol. 19, no. 3, pp. 342–368, 1987.

[2] P.K. Allen and K.S. Roberts, "Haptic object recognition using a multi-fingered dextrous hand," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, 14-19 May 1989, pp. 342–347 vol.1.

[3] S. Caselli, C. Magnanini, and F. Zanichelli, "Haptic object recognition with a dextrous hand based on volumetric shape representations," in *Multisensor Fusion and Integration for Intelligent Systems, 1994. IEEE International Conference on MFI '94.*, 2-5 Oct. 1994, pp. 280–287.

[4] David J. Montana, "The kinematics of contact and grasp," *International Journal of Robotics Research*, vol. 7, no. 3, pp. 17 – 32, June 1988.

[5] P.K. Allen, "Mapping haptic exploratory procedures to multiple shape representations," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 13-18 May 1990, vol. 3, pp. 1679–1684.

[6] N. Chen, H. Zhang, and R. Rink, "Edge tracking using tactile servo," in *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, 5-9 Aug. 1995, vol. 2, pp. 84–89.

[7] A.M. Okamura and M.R. Cutkosky, "Haptic exploration of fine surface features," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 10-15 May 1999, vol. 4, pp. 2930–2936.

[8] A. Petrovskaya, O. Khatib, S. Thrun, and A.Y. Ng, "Bayesian estimation for autonomous object manipulation based on tactile sensors," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 15-19, 2006, pp. 707–714.

[9] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "Armar-iii: An integrated humanoid platform for sensory-motor control," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, Dec. 2006, pp. 169–175.

[10] Corey Goldfeder, Peter K. Allen, Claire Lackner, and Raphael Pelossof, "Grasp planning via decomposition trees," in *Robotics and Automation, 2007 IEEE International Conference on*, 10-14 April 2007, pp. 4679–4684.

[11] A. Kargov, T. Asfour, C. Pylatiuk, R. Oberle, H. Klosek, S. Schulz, K. Regenstein, G. Bretthauer, and R. Dillmann, "Development of an anthropomorphic hand for a mobile assistive robot," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, 28 June-1 July 2005, pp. 182–186.

[12]] J. Lee and T. Kunii, "Constraint-based hand animation," in *Models and Techniques in Computer Animation*, Tokyo, 1993, pp. 110–127, Springer, Tokyo.

[13] Y. Yasumuro, Qian Chen, and K. Chihara, "3d modeling of human hand with motion constraints," in *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, 12-15 May 1997, pp. 275–282.

[14] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, and X. Twombly, "A review on vision-based full dof hand motion estimation," in *Computer Vision and Pattern Recognition, 2005 IEEE Computer Society Conference on*, 20-26 June 2005, vol. 3, pp. 75–75.

[15] M. Bray, E. Koller-Meier, and L. Van Gool, "Smart particle filtering for 3d hand tracking," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, 17-19 May 2004, pp. 675–680.

[16] B. Buchholz and T.J. Armstrong, "A kinematic model of the human hand to evaluate its prehensile capabilities," *Journal of Biomechanics*, vol. 25, no. 2, pp. 149–162, 1992.

[17] J.J. Kuch and T.S. Huang, "Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, 20-23 June 1995, pp. 666–671.

[18] Michael Isard and Andrew Blake, "ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework," *Lecture Notes in Computer Science*, vol. 1406, pp. 893–908, 1998.

[19] A.H. Barr, "Superquadrics and angle-preserving transformations," *Computer Graphics and Applications, IEEE*, vol. 1, no. 1, pp. 11–23, Jan 1981.

[20] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: the case for superquadrics with global deformations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 2, pp. 131–147, Feb. 1990.

[21] Donald W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[22] A.D. Gross and T.E. Boult, "Error of fit measures for recovering parametric solids," in *Computer Vision., 1988. Second International Conference on*, December 5-8, 1988, pp. 690–694.

Probabilistic Pose Recovery
Using Learned Hierarchical Object Models

Renaud Detry
renaud.detry@ulg.ac.be
Montefiore Institute
University of Liège
Belgium

Nicolas Pugeault
npugeaul@inf.ed.ac.uk
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
Odense, Denmark;
School of Informatics
Univ. of Edinburgh
United Kingdom

Justus Piater
Justus.Piater@ULg.ac.be
Montefiore Institute
University of Liège
Belgium

INTELSIG Technical Report 2008-01-25

Department of Electrical Engineering and Computer Science
University of Liège

Probabilistic Pose Recovery Using Learned Hierarchical Object Models

Renaud Detry[1], Nicolas Pugeault[2], and Justus Piater[1]

[1] Université de Liège, Liège, Belgium,
Renaud.Detry@ULg.ac.be, Justus.Piater@ULg.ac.be
[2] The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark;
The University of Edinburgh, Edinburgh, Scotland, UK;
npugeaul@inf.ed.ac.uk

Abstract. This paper presents a probabilistic representation for 3D objects, and details the mechanism of inferring the pose of real-world objects from vision. Our object model has the form of a hierarchy of increasingly expressive 3D features, and allows probabilistic encoding of 3D relations between these. Features at the bottom of the hierarchy are bound to local visual perceptions. While we currently only use visual features, our method can in principle incorporate features from diverse modalities within a coherent framework. Model instances are detected using a Nonparametric Belief Propagation algorithm which propagates evidence through the hierarchy to infer globally consistent poses for every feature of the model. We present an importance-sampling mechanism for belief updates that is critical for efficient and precise propagation. We finally present a series of pose estimation experiments on real objects, along with quantitative performance evaluation.

1 Introduction

Representations of objects as configurations of parts have many potential advantages. Part-based representations are more robust to occlusions and viewpoint changes than global representations, and spatial configurations increase their expressiveness. Moreover, they not only allow bottom-up inference of object parameters based on features detected in images, but also top-down inference of image-space appearance based on object parameters.

The advantages of visual part-based representations naturally extend to multisensory cases. For example, haptic and proprioceptive information won't directly relate to an object as a whole. Instead, they typically emerge from specific grasps, on specific parts of the object. Part-based representation offer a neat way to *locally* encode cross-modal descriptions that emphasise the relations bewteen the different types of percepts.

We are currently developing a framework for object representation that combines local appearance and 3D spatial relationships, along with mechanisms for unsupervised learning and probabilistic inference of the model. We emphasize

that we are not developing an object classification framework, which would imply concentrating on specific aspects that separate a number of object types. Instead, we intend to develop object-centric representations that lend themselves to more application than mere classification (e.g. manipulation).

Our model has the form of a hierarchy. Features at the bottom of the hierarchy are bound to local visual perceptions. Pairs of features that present strong geometric correlation are iteratively grouped into higher-level meta-features that encode probabilistic relative spatial relationships between their children.

The features organized in our hierarchies are not specially restricted to one input modality. We currently work with visual input only, but our model is intended to unite different types of perceptual information, e.g. vision plus haptic and proprioceptive inputs simultaneously. This would produce cross-modal descriptions and cross-modal behaviors directly applicable to robotic tasks such as grasping and object manipulation, as a grasp strategy may be linked directly to visual features that predict its applicability.

In previous work [1], we presented a learning method, and we gave an overview of inference mechanisms. Inference is responsible for propagating local evidence to top-level features, leading to one or more consistent scene interpretations; Propagation was carried out following a straightforward Nonparametric Belief Propagation [8] scheme, which allowed pose recovery on artificial objects. In this paper, we present in greater details an evolution of these inference mechanisms, along with practical considerations. We added the importance-sampling (IS) message product suggested by Ihler et al. [2], and extended it to a two-level IS sampling of *implicit* message products which is now applicable for pose estimation on real-world objects, as presented in Section 5.

2 Hierarchical Model

Our object model consists of a set of generic *features* organized in a hierarchy. Features that form the bottom level of the hierarchy, referred to as *primitive features*, are bound to visual observations. The rest of the features are *meta-features* which embody spatial configurations of more elementary features, either meta or primitive. Thus, a meta-feature incarnates the relative configuration of two features from a lower level of the hierarchy.

A feature can intuitively be associated to a "part" of an object, i.e. a generic component instantiated once or several times during a "mental reconstruction" of the object. At the bottom of the hierarchy, primitive features correspond to local parts that each may have many *instances* in the object. Climbing up the hierarchy, meta-features correspond to increasingly complex parts defined in terms of constellations of lower parts. Eventually, parts become complex enough to satisfactorily represent the whole object. Figure 1 shows a didactic example of a hierarchy for a bike. The bike is the composition of *frame* and *wheel* features. A wheel is composed of pieces of tire and spokes. The generic piece of tire at the bottom of the hierarchy is a primitive feature; the pieces of tire squared in green in the scene (Figure 2) are instances of that primitive feature.

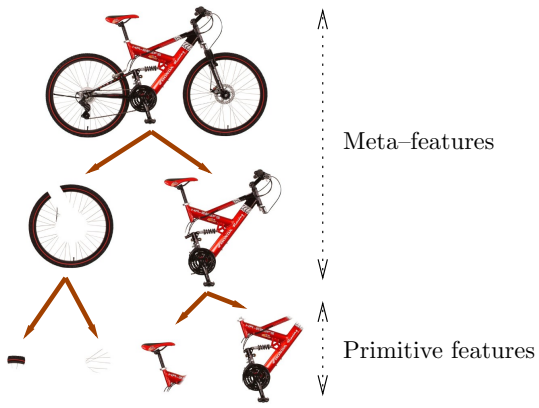


Fig. 1. A didactic example of a hierarchical model of a bike.

Fig. 2. Instances of the generic piece-of-tire primitive feature in the bike scene.

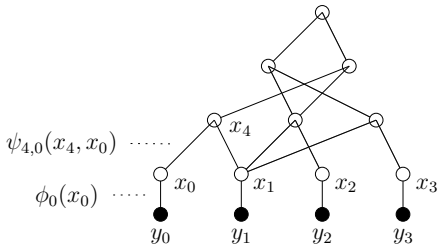


Fig. 3. A Pairwise Markov Random Field representing a feature hierarchy. Features correspond to hidden variables (white). Observed variables (black) correspond to observations, bound to bottom-level primitive features.

At the bottom of the hierarchy, primitive features are tagged with an appearance descriptor called a *codebook vector*. The set of all codebook vectors forms a *codebook* that binds the object model to the feature observations, by associating observations to primitive features.

In summary, information about an object is stored within the model in the three following forms:

i. the topology of the hierarchy,
ii. the relationships between related features,
iii. the codebook vectors annotating bottom-level features.

Formally, the hierarchy is implemented using a Pairwise Markov Random Field (see Figure 3). Features are associated to hidden nodes (white in Figure 3), and the structure of the hierarchy is reflected by the edge pattern between them. Each meta-feature is thus linked to its two child features. Observed variables y_i of the random field stand for observations.

When a model is associated to a particular scene (during construction or instantiation), features are associated to corresponding instances in that scene. The correspondence between a feature i and its instances is represented by a probability density over the pose space $SE(3) = \mathbb{R}^3 \times SO(3)$ represented by a random variable x_i.

As noted above, a meta-feature encodes the relationship between its two children. However, the graph records this information in a slightly different but equivalent way: instead of recording the relationship between the two child features, the graph records the two relationships between the meta-feature and each of its children. The relationship between a meta-feature i and one of its children j is parametrized by a *compatibility potential function* $\psi_{ij}(x_i, x_j)$ associated to the edge e_{ij}. A compatibility potential specifies, for any given pair of poses of the features it links, the probability of finding that particular configuration for these two features. We only consider rigid-body relationships. Moreover, relationships are *relative* spatial configurations. Compatibility potentials can thus be represented by a probability density over the feature–to–feature transformation space $SE(3)$.

Compatibility potentials allow relationship distributions to have multiple modes. In the bike model, let us consider the meta-feature that represents a generic wheel. There are two wheels in the picture; two instances of the wheel feature will be used in a mental reconstruction of the bike. Hence, the compatibility potential between the *wheel* feature and the *bike* feature will be dense around two modes, one corresponding to the transformation between the bike and the front wheel ("the front wheel is on the right side of the bike"), the other between the bike and the rear wheel ("the rear wheel is on the left side of the bike").

Finally, the statistical dependency between a hidden variable x_i and its observed variable y_i is parametrized by an *observation potential* $\phi_i(x_i)$, also referred to as *evidence* for x_i, which corresponds to the spatial distribution of x_i's observations.

The term *primitive feature instance* formally refers to a random draw from a primitive feature distribution. While a primitive feature instance often corresponds to an observation, observations enter into the graphical model merely as prior knowledge. Primitive feature instances result from inference; they depend on observations *and* on all features of the hierarchy. Owing to inference mechanisms presented in the next paragraph, if an observation is discarded (e.g. occluded), a primitive feature instance may nevertheless appear at its place.

3 Inference

Model instantiation is the process of detecting instances of an object model in a scene. It provides pose densities for all features of the model, indicating where the learned object is likely to be present. Instantiating a model in a scene amounts to inferring posterior marginal densities for all features of the hierarchy. Thus, once priors (observation potentials, evidence) have been defined, instantiation can be achieved by any applicable inference algorithms. We currently use a Belief Propagation algorithm, which we describe below in detail.

For primitive features, evidence is estimated from feature observations. Observations are classified according to the primitive feature codebook; for each primitive feature i, its observation potential $\phi_i(x_i)$ is estimated from observations that are associated to the i^{th} codebook vector. For meta-features, evidence is uniform.

3.1 Belief Propagation

Graphical models are a convenient substrate of sophisticated *inference algorithms*, i.e. algorithms for efficient computation of statistical quantities. An efficient inference algorithm is essential to the hierarchical model, for it provides the mechanism that will let features communicate and propagate information.

Our inference algorithm of choice is currently the Belief Propagation algorithm (BP) [7,9,3]. Belief Propagation is based on incremental updates of

marginal probability estimates, referred to as *beliefs*. The belief at feature i is denoted by

$$b(x_i) \approx \mathbf{p}(x_i|y) = \int ... \int \mathbf{p}(x_1, ..., x_N|y)\, dx_1 ... dx_{i-1} dx_{i+1} ... dx_N$$

where y stands for the set of observations. During the execution of the algorithm, *messages* are exchanged between neighboring features (hidden nodes). A message that feature i sends to feature j is denoted by $m_{ij}(x_j)$, and contains feature i's belief about the state of feature j. In other words, $m_{ij}(x_j)$ is a real positive function proportional to feature i's belief about the plausibility of finding feature j in pose x_j. Messages are exchanged until all beliefs converge, i.e. until all messages that a node receives predict a similar state.

At any time during the execution of the algorithm, the current pose belief (or marginal probability estimate) for feature i is the normalized product of the local evidence and all incoming messages, as

$$b_i(x_i) = \frac{1}{Z}\phi_i(x_i) \prod_{j \in \text{neighbors}(i)} m_{ji}(x_i), \tag{1}$$

where Z is a normalizing constant. To prepare a message for feature j, feature i starts by computing a "local pose belief estimate", as the product of the local evidence and all incoming messages *but* the one that comes from j. This product is then multiplied with the compatibility potential of i and j, and marginalized over x_i. The complete message expression is

$$m_{ij}(x_j) = \int \psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k \in \text{neighbors}(i) \setminus j} m_{ki}(x_i) dx_i. \tag{2}$$

As we see, the computation of a message doesn't directly involve the complete local belief (1). In general, the explicit belief for each node is computed only once, after all desirable messages have been exchanged.

When BP is finished, collected evidence has been propagated from primitive features to the top of the hierarchy, permitting inference of marginal pose densities at top-level features. Furthermore, regardless of the propagation scheme (message update order), the iterative aspect of the message passing algorithm ensures that global belief about the object pose – concentrated at the top nodes – has at some point been propagated back down the hierarchy, reinforcing globally consistent evidence and permitting the inference of occluded features. While there is no theoretical proof of BP convergence for loopy graphs, empirical success has been demonstrated in many situations.

3.2 Nonparametric Representation

We opted for a nonparametric approach to probability density representation for all entities of the model, i.e. random variable and functions of random variables, including potentials, messages, and evidence. A density is simply represented by

a set of (possibly weighted) particles; the local density of these particles in a given region is proportional to the actual probabilistic density in that region. The number of particles supporting a density is fixed, and will be denoted by n. Whenever a density has to be evaluated, traditional kernel density estimation methods can be used. Compared to usual parametric approaches that involve a limited number of parametrized kernels, a nonparametric approach eliminates problems like fitting of mixtures or the choice of a number of components. Also, no assumption concerning the shape of the density has to be made.

Particles live in the Special Euclidean Space $SE(3)$. The location/translation component is parametrized by a 3–vector. For the orientation/rotation component it was decided to prefer quaternions over rotation matrices, because they provide a well-suited formalism for the manipulation of rotations such as composition or metric definition [6,4].

3.3 Nonparametric Belief Propagation

For inference, we use a variant of BP, Nonparametric Belief Propagation (NBP), an algorithm for BP message update (2) in the particular case of continuous, non-Gaussian potentials [8]. The underlying method is an extension of particle filtering; the representational approach is thus nonparametric and fits our model very well.

NBP is easier to explain if we decompose the analytical message expression (2) into two steps:

1. Computation of the local belief estimate
$$\beta_{ts}(x_t) = \phi_t(x_t) \prod_{i \in N(t) \setminus s} m_{it}(x_t) \tag{3}$$
2. Combination of β_{ts} with the compatibility function ψ_{ts}, and marginalisation over x_t
$$m_{ts}(x_s) = \int \psi_{ts}(x_t, x_s) \beta_{ts}(x_t) dx_t \tag{4}$$

NBP forms a message by first sampling from the product (3) to collect a non-parametric representation for $\beta_{ts}(x_t)$, it then samples from the integral (4) to collect a non-parametric representation for $m_{ts}(x_s)$. These two operations are executed alternately: transform local estimate to form a message, merge messages to form a local estimate, etc...

Sampling from the product (3) is conceptually straightforward. Using Gaussian kernel density estimation, each factor (messages and evidence) can be represented by a weighted sum of n Gaussians. The product of a series of Gaussians is also a Gaussian, and the parameters (mean, variance, weight) of the product Gaussian can easily be computed from the parameters of the factor Gaussians. Hence, letting $d = (N(t) - 1) + 1$ denote the number of factors in the product (3), $\beta_{ts}(x_t)$ can be expressed as a weighted sum of n^d Gaussians [8]. A nonparametric representation for $\beta_{ts}(x_t)$ can thus be constructed by sampling from a

mixture of n^d Gaussians, which amounts to repetitively selecting one Gaussian at random and taking a random sample from it. The computational cost of this exhaustive approach is $O(n^d)$. Clearly, exhaustive product implementations will suffer from overly long computation times.

The second phase of the NBP message construction computes an approximation for the integral (4). This stochastic integration requires the potential $\psi_{ts}(x_t, x_s)$ to be decomposed into its marginal influence on x_t and the conditional compatibility it defines between x_t and x_s. In our case however, potentials only depend on the difference between their arguments; the marginal influence is a constant and can be ignored. The stochastic integration is simply completed by propagating to s a series of samples $\hat{x}_t^{(i)}$ from $\beta_{ts}(x_t)$, which can be achieved by sampling from $\psi_{ts}(\hat{x}_t^{(i)}, x_s)$ for each $\hat{x}_t^{(i)}$.

The computation bottleneck of NBP clearly lies in the message product. Multiple improvements over the exhaustive product have been suggested [2], one of these being Importance Sampling (IS). Sampling from a product with IS works by selecting repetitively one of the factors as a proposal distribution, the rest of the factors providing an importance weight. IS produces n samples from a product of d factors in $O(dn^2)$ time. From here on, we will consider that the number of neighbors a node may have is bounded and typically low, and ignore it in complexity statements. IS thus produces n samples from a product of d factors in $O(n^2)$ time.

3.4 Density Resolution

As explained above, *A message that feature i sends to feature j – denoted by $m_{ij}(x_j)$ – contains feature i's belief about the state of feature j.* Feature i is likely to posses a rather inaccurate local estimate, at least at the beginning of propagation. Additionally, transforming the local estimate with ψ_{ij} introduces additional noise. As a general observation, messages will often carry a wide space of possible states. The job of message product is to extract sections that overlap between incoming messages, and discard sections that do not "agree". The portion of a message that will actually represent good prediction is generally expected to be small. For that reason, one may benefit from using larger particle sets for messages than for local estimates (and potentials). We will thus denote by n the number of particles composing a message, and by m the number of particles for other distributions. Typically, we use $m = \sqrt{n}$.

The previous remark has no incidence on the performance of an exhaustive product. However, sampling a product with IS would be down to $O(mn)$ time.

3.5 Importance Sampling On Implicit Message Products

Let us now turn to the propagation equation (2), which we *analytically* decomposed into (3) and (4). We explained that NBP implements BP by *physically* performing the same decomposition, i.e. computing explicit nonparametric representations for messages and local estimates alternately. In this section, we

propose a somewhat different implementation, in which explicit representations are only computed for local estimates.

Let us assume we are in the process of sampling from $\beta_{ts}(x_t)$, to merge all incoming information but that from s. We choose one incoming message $m_{ut}(x_t)$ at random as IS proposal density and take a sample $\hat{x}^{(1)}_{t\leftarrow u}$ from it. We then compute an importance weight as

$$w^{(1)}_{t\leftarrow u} = \phi_t(\hat{x}^{(1)}_{t\leftarrow u}) \prod_{i\in N(t)\setminus\{s,u\}} m_{it}(\hat{x}^{(1)}_{t\leftarrow u}). \tag{5}$$

This operation is $O(n)$.

One can notice though that both these operations don't actually need an explicit expression for incoming messages. Producing $\hat{x}^{(1)}_{t\leftarrow u}$ from $\beta_{ut}(x_t)$ and $\psi_{ut}(x_u, x_t)$ is $O(1)$. In turn, Expression (5) can be rewritten

$$w^{(1)}_{t\leftarrow u} = \phi_t(\hat{x}_{t\leftarrow u}) \prod_{i\in N(t)\setminus\{s,u\}} \int \psi_{ts}(x_i, \hat{x}^{(1)}_{t\leftarrow u})\beta_{it}(x_i)dx_i. \tag{6}$$

Evaluating $\phi_t(\hat{x}_{t\leftarrow u})$ is $O(m)$. Evaluating each integral is achieved by sampling p times $\hat{x}^{(k)}_i$ from either $\psi_{ts}(x_i, \hat{x}^{(1)}_{t\leftarrow u})$ or $\beta_{it}(x_i)$ and evaluating $\beta_{it}(\hat{x}^{(k)}_i)$ or $\psi_{ts}(\hat{x}^{(k)}_i, \hat{x}^{(1)}_{t\leftarrow u})$ respectively, and taking the average result. To achieve the same resolution as in (5), p should be equal to m. The time complexity of (6) can be thought of as $O(m^2)$, or $O(n)$, i.e. the same as (5). The memory impact is $O(m)$ instead of $O(n)$ though, at the price of more potential evaluations.

The advantage of implicit messages is twofold. Fist, one can use *a priori* knowledge on the shape of potential and node distributions to decide between sampling from $\psi_{ts}(x_i, \hat{x}^{(1)}_{t\leftarrow u})$ and evaluating $\beta_{it}(\hat{x}^{(k)}_i)$, or going the other way around. Second, p can be chosen different than m; this is exploited in the next paragraph.

One weakness of IS is that it cannot intrinsically concentrate its attention on the modes of the product [2], which is an issue if, as is often the case, messages present many irrelevant modes. The previous text allows for a solution to that issue. One can imagine a two-level IS, in which we first compute an approximate representation for $\beta_{ts}(x_t)$ with a low p, then use it as proposal distribution for a second IS that will be geared towards relevant modes.

3.6 Practical considerations

Ihler et al. draw a lot of attention towards *kd*-Trees [2], which provide logarithmic access to elements enclosed in a given region of $\mathbb{R}^k$. We are less enthusiastic about them, for several reasons.

1. They are expensive to build.
2. The population size n for which *kd*-Trees become faster than linear search grows as k grows. For $k = 3$, n is at the very least of the order of 1000–10000. Moreover, our particle space is $SE(3)$, which includes the non-Euclidean subspace $SO(3)$, making for even more difficult search.

3. Computational details need to be kept in mind, too. For instance, m elements may very well fit in processor cache, but m^2 will continuously trigger cache misses. This of course is very dependant on the hardware, but it may motivate an implementation like (6), instead of (5) with a *kd*-Tree.
4. Finally, *kd*-Trees, and in particular the dual trees [2], represent a considerable implementation effort.

4 Object Pose Estimation

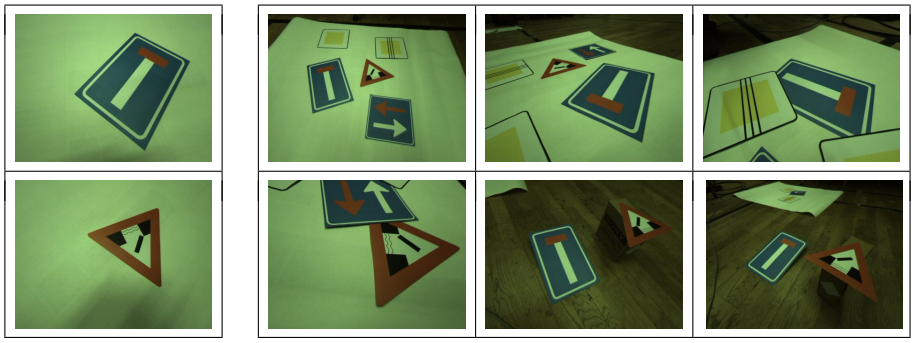
Features at the top of an object model represent the whole object, and they will present relatively concentrated densities that are unimodal if exactly one instance of this object is present in the scene. These densities can be used to estimate the object pose. Let us consider a model for a given object, and a pair of scenes where the object appears. In the first scene, the object is in a reference pose. In the second scene, the pose of the object is unknown. The application of our method to estimate the pose of the object in the second scene goes as follows:

1. Instantiate the object model in the reference scene. For every top-level feature i of the instantiated graph, compute a *reference aggregate feature pose* π_1^i from its unimodal density.
   Instantiating the model in a reference scene is necessary because even though the top-level features all represent the whole object, they come from different recursive combinations of features of various poses.
2. Instantiate the object model in the unknown scene. For every top feature of that graph, compute an *aggregate feature pose* π_2^i.
3. For all top level features i, the transformations from π_1^i to π_2^i should be very similar; let us denote the mean transformation by t. This transformation corresponds to the rigid body motion between the pose of the object in the first scene and its pose in the second scene. Since the first scene is a reference pose, t is the pose of the object in the second scene.

A prominent aspect of this procedure is its ability to recover an object pose without explicit point-to-point correspondences. The estimated pose emerges from a negotiation involving all available data.

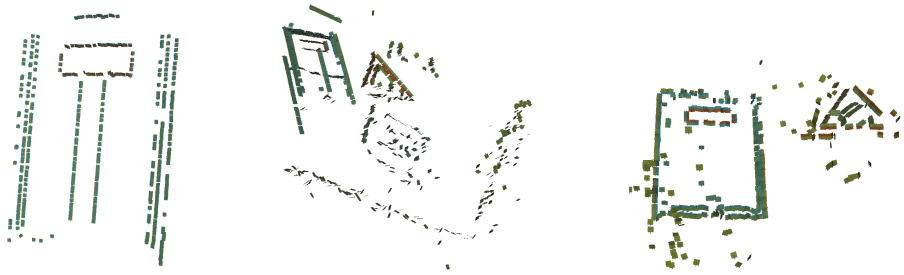
5 Experiments

We ran pose estimation experiments on two objects, presented in Figure 4(a), for a series of scenes, presented in Figure 4(b). Input 3D features for the bottom levels (primitive feature observations) are provided by an early-cognitive-vision (ECV) system [5], which extracts 3D primitives from stereo views of a scene. The quality of such ECV representations vary in function of different parameters. Figure 5 illustrates the ECV primitives for certain scenes. Figure 5(a) is extracted from the first image of Figure 4(a). Figures 5(b) and 5(c) are extracted from the two images of the second column of Figure 4(b).



(a) Training (b) Evaluation

Fig. 4. Input imagery. For each stereo pair, only the left image is presented. Effective resolution is 1280×960 pixels.



(a) (b) (c)

Fig. 5. ECV primitives.

In the next paragraphs, we go through the procedure of a pose estimation experiment. First, a model is learned from one set of observations of an object of interest (the reference scene), like those in Figure 5(a). A hierarchy is built up to n levels, we instantiate the model in the reference scene, and compute a reference aggregate feature pose π_1^i for every top feature i of the model.

We are then ready to estimate the pose of our object in a novel scene. We initialize primitive-feature evidence of the model with observations from the scene, e.g. those in Figure 5(b). Evidence is propagated through the hierarchy, and we can eventually estimate the top-feature poses. Since the object of interest is present only once in the noisy scene, top level features should, after instantiation, present unimodal densities; we can safely compute a mean pose π_2^i for each of them.

Finally, we compute the transformation t_i between π_1^i and π_2^i for every top feature i. As noted in Section 4, all t_i are very similar. Let us denote the mean transformation by t, which corresponds to the *estimated* rigid body motion between the pose of the object in the reference scene, and its pose in the noisy scene.

We evaluate the quality of transformation estimates by transforming the ECV primitives from the training scene with t, and superimposing the result to the current scene. We then see if shapes match. (Note for PACO review: We will do quantitative evaluation on t for the final version.) Pose estimation worked for both traffic signs (dead-end, opening bridge) in all scenes of Figure 4. The worst estimate, presented at Figure 6(a), corresponds to the dead-end signal pose estimation in the last image. This is however one of the most difficult scenes: it has a brown background, thus changing the outside color of ECV primitives on the traffic sign contours. This induces wrong associations of observations to primitive features, and makes for harder inference. Estimation is still quite accurate given the difficulty of the scene. Other typical estimates are presented at Figure 6. In particular, 6(c) shows a good result despite occlusion.

6 Conclusion

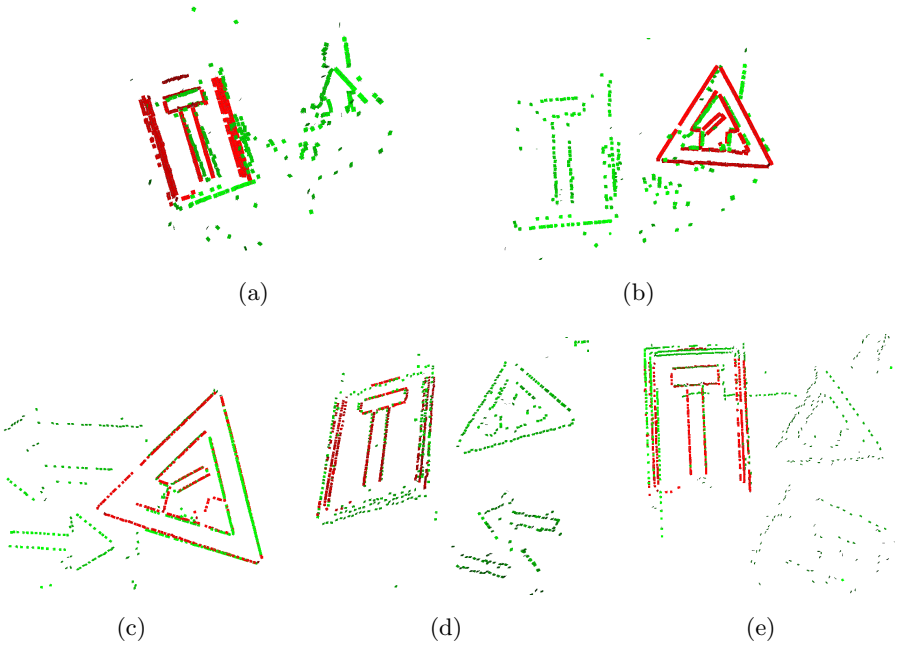
Acknowledgment

This work was supported by the Belgian National Fund for Scientific Research (FNRS) and the EU Cognitive Systems project PACO-PLUS (IST-FP6-IP-027657).

The authors thank Norbert Krüger and Nicolas Pugeault for providing MoInS data and for fruitful discussions.

References

1. Renaud Detry and Justus H. Piater. Hierarchical integration of local 3D features for probabilistic pose recovery. In *Robot Manipulation: Sensing and Adapting to the Real World (Workshop at Robotics, Science and Systems)*, 2007.



(a) (b) (c) (d) (e)

Fig. 6. Estimates.

2. Alexander T. Ihler, Erik B. Sudderth, William T. Freeman, and Alan S. Willsky. Efficient multiscale sampling from products of Gaussian mixtures. In *Neural Information Processing Systems*, 2003.
3. Michael I. Jordan and Yair Weiss. Graphical models: Probabilistic inference. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks, 2nd edition*. MIT Press, 2002.
4. Charles F. F. Karney. Quaternions in molecular modeling. *J. Mol. Graph. Mod.*, 25, 2006.
5. Norbert Krüger and Florentin Wörgötter. Multi-modal primitives as functional models of hyper-columns and their use for contextual integration. In Massimo De Gregorio, Vito Di Maio, Maria Frucci, and Carlo Musio, editors, *BVAI*, volume 3704 of *Lecture Notes in Computer Science*, pages 157–166. Springer, 2005.
6. James Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *Proc. 2004 IEEE Int'l Conf. on Robotics and Automation (ICRA 2004)*. IEEE, May 2004.
7. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
8. Erik B. Sudderth, Alexander T. Ihler, William T. Freeman, and Alan S. Willsky. Nonparametric belief propagation. *cvpr*, 01:605, 2003.
9. Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. Technical report, Mitsubishi Electric Research Laboratories, 2002.

Integration of Visual and Shape Attributes for Object Action Complexes

Kai Huebner, Mårten Björkman, Babak Rasolzadeh, Martina Schmidt, and Danica Kragic

Kungliga Tekniska Högskolan, Computer Vision & Active Perception Lab.,
S-100 44 Stockholm, Sweden,
{khubner,celle,babak2,schm,danik}@kth.se,
WWW home page: http://www.csc.kth.se/cvap

Abstract. Our work is oriented towards the idea of developing cognitive capabilities in artificial systems through Object Action Complexes (OACs) [10]. The theory comes up with the claim that objects and actions are inseparably intertwined. Categories of objects are not built by visual appearance only, as very common in computer vision, but by the actions an agent can perform and by attributes perceivable. The core of the OAC concept is constituting objects from a set of attributes, which can be manifold in type (e.g. color, shape, mass, material), to actions. This twofold of attributes and actions provides the base for categories. The work presented here is embedded in the development of an extensible system for providing and evolving attributes, beginning with attributes extractable from visual data.

Keywords: Cognitive Vision, Object Action Complexes, Object Attribution
Track: Cognitive Vision

1 Introduction

In cognitive systems, like we want robots to become, representation of objects plays a major role. A robot's local world is built by objects that are ought to be recognized, classified, interpreted or manipulated. Though also *things*, as untreated basic sensory features, might help for some of these tasks, the semantic representation of an *object* seems to be more intuitive. Nevertheless, the question arises what makes an object an object, what makes Peter's cup being Peter's cup? There has been plenty of research on this issue, most of which concentrates on example-based recognition of objects by learned features, may they be visual or shape-based. In such systems, Peter's cup has been shown to the robot and can thus be recognized again. However, this does not make the robot identify arbitrary cups it has never seen before.

Due to this demerit of model-based recognition, another approach which is focussed on the functionalities or affordances of objects is motivated. Peter's cup is solid, it can stand stable and it is hollow so it can keep coffee, and is mainly used for filling or drinking. Maybe each other object that holds the same attributes can also be used as a 'filling device' or 'drinking device', which humans might name a cup. However, the 'filling device' property is alone more general and allows Peter to put in flowers, which would make one name it a vase instead. This line of argument from objects to actions is formalized into an upcoming concept, the Object Action Complexes (OACs). We refer to [10] on the theory of OACs, while we here present a system which is able provide a variety of attributes from visual sensory input to support OACs. We link to several fields of research that relate to our work.

1.1 Cognitive Vision Systems

One of the major requirements of a cognitive robot is to continuously acquire perceptual information to successfully execute mobility and manipulation tasks, [9, 12, 15]. The most effective way of performing this is if it occurs in the context of a specific task. This was, for a long time, and still is the major way of thinking in the field of robotics. Focus is usually put on the on task-specific aspects when processing sensor data which may reduce the overall computational cost as well as add to the system robustness. However, in most cases this leads to the development of special-purpose systems that are neither scalable nor flexible. Thus, even if significant progress has been achieved, from the view of developing general system able to perform various tasks in domestic environments, research on autonomous manipulation is still in its embriotic stage.

In this work, we treat the development of active vision paradigms and their relation of how to exploit both kinematic and dynamic regularities of the environment. Early work recognized that a robot has the potential to examine its world using causality, by performing probing actions and learning from the response [11]. Visual cues were used to determine what parts of the environment were physically coherent through interplay of objects, actions and imitations. Our interest is very similar, but geared to the development of a more advanced vision system necessary for such an application.

[14] examines the problem of object discovery defined as autonomous acquisition of object models, using a combination of shape, appearance and motion. The authors discuss that object discovery is complicated due to the lack of a clear definition of

what constitutes an object. They state that rather than trying for an all-encompassing definition of an object that would be difficult or impossible to apply, a robot should use a definition that identifies objects useful for it. From the perspective of the object-fetching robot, useful objects would be structures that can be picked up and carried. Similar line of thinking is pursued in our work, while we go one step forward by also extracting a set of object attributes that can be used for manipulation purposes or further learning of object properties later on.

Such a system has been presented in [13] with an objective of learning visual concepts. The main goal is to learn associations between automatically extracted visual features and words describing the scene in an open-ended, continuous manner. Unfortunately, the vision system is very simple and the experimental evaluation is performed with homogeneous objects of simple and easy distinguishable shapes.

In relation to representation of object properties, there is a close connection to *anchoring*, [8], that connects, inside an artificial system, the symbol-level and signal-level representations of the same physical object. Although some nice ideas about the representations are proposed, there is no attempt of developing the underlying vision system necessary for extraction of symbols.

2 From Visual Sensors to Attributes

Our original system was purely top-down driven, with top-down information given in terms of visual search tasks [1]. These tasks were represented as precomputed models, typically one model for each possible requested object. Given a task the system scanned the environment for suitable new fixation points and at each visited such point the attended region was compared to the model corresponding to the task. Due to its dependence on precomputed models, the original system had a number of key weaknesses. It could not generalize beyond the scope of the models and it was unable to explore the environment so as to learn new models. The aim of the work presented here is to go beyond these limitations and open up for scenarios in which all objects are not necessarily known beforehand. The attentional system in Section 2.1 is based on top-down as well as bottom-up cues and can be tuned towards either exploration or visual search. With attention driven by generalizable attributes as explained in Section 2.3, rather than models of known objects, tasks can be expressed in more general terms.

As already stated, our work is oriented towards the idea of Object Action Complexes (OACs). The theory comes up with the claim that objects and actions are inseparably intertwined. Categories of objects are not built by visual appearance only, as very common in computer vision, but by the actions an agent can perform, as also by attributes linked to them. Such attributes can be various in type, e.g. color, shape and mass. While color might be perceived by visual processes only, interaction greatly supports the recognition of shape by vision, e.g. in case of hollowness. Mass probably is an attribute that is preceivable by strong interaction with the object only. On the other hand, attributes can be a result of higher-level reasoning, e.g. that the cup is full or empty.

The core of the OAC concept is connecting objects from a set of attributes to actions. Peter's cup will not only be Peter's cup, because the color or the texture of the specific instance has been learned, but because the concept of attributes (e.g. *solid*, *hollow*) shall

be learned to be connected to a set of actions (e.g. *pick up*, *fill*, *drink*). This twofold of attributes and actions provides the base for categories, where even bins or vases might be seen as a cup in terms of the actions one can perform with them. Our aim here is the development of an extensible system for providing and evolving early Object Action Complexes, beginning with attributes extractable from visual data in Section 2.1. After this, basic shape primitives are introduced by 3D segmentation in Section 2.2.

2.1 Visual Attributes from Attentional Cues

It has been suggested that objects present in a scene possess a certain intrinsic ranking or "interestingness" with regards to that scene. For a visual observer this means that a dynamic combination of both top-down (task-dependent) and bottom-up (scene-dependent) control is available for selecting and attending regions in the scene [1]. Salience-based models in computational attention became largely popular after the work of Koch and Ullman [2]. A topic of research has been in explaining, with a computational model, how the top-down mechanism works. Recent work in the field include models where top-down modulations of attention are learnt with an ART-network [3]. Another attempt has been to train a bottom-up attention model towards detection of particular target objects by displaying these objects in different backgrounds [4].

Our system uses a model similar to the VOCUS-model [5] which has a top-down tuned saliency map that can be "controlled" through a set of weights. Weights are applied to each feature and conspicuity map used in the computation of the salience map. Four broadly tuned color channels (R, G, B and Y) calculated as in Itti's NVT model [6], an intensity map, and four orientation maps, computed by Gabor filters, are weighted individually. Following the original version, we create scale-space pyramids for all these 9 maps and form conventional center-surround differences by across-scale-subtraction, followed by normalization. This leads to the final conspicuity maps for intensity, color and orientation. As a final set of weight parameters we introduce one weight for each of these maps, constructing the final modulated top-down saliency map.

The purpose of the attentional system is twofold. First of all, attention is used to control the stereo head so that objects of interest are placed in the center of the visual field. The second purpose is to derive visual attributes to describe objects in scene, objects that can later be revisited and possibly manipulated. In the current version of the system, the visual attributes of an object are represented by the weights that make this object stand out in the top-down salience map (see Fig. 1). From previous searches for the object, weights are optimized through gradient descent, with the influence of context modelled using a neural network [1].

2.2 Shape Attribution from 3D Segmentation

3D Segmentation without Table Plane Assumption. Image data needs to be grouped into regions corresponding to possible objects in the 3d scene, for shape attributes to be extracted and manipulation performed. The attentional system, as described in Section 2.1, does no such grouping, it only provides hypotheses of where such objects may be located in image space. However, from binocular disparities the extent of hypotheses can be detemined also in 3d space. Disparities can be considered as measurements in 3d

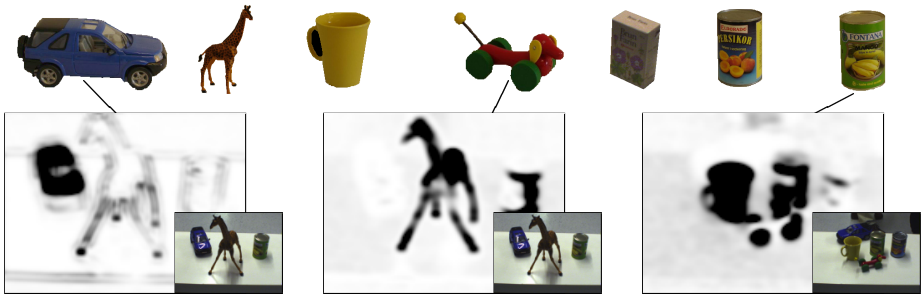


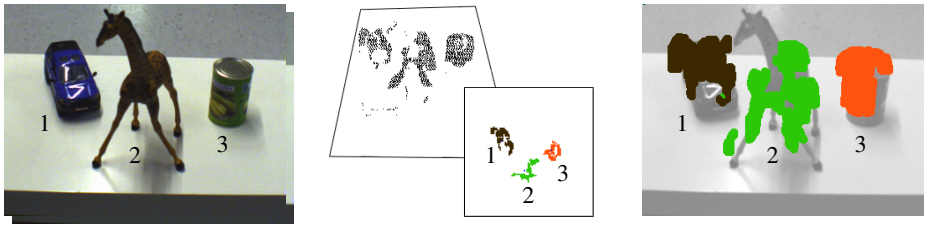
Fig. 1. Top row: Objects in use: car, giraffe, mug, dog, sugar box, peach can and mango can. Bottom row: Three exemplary visual attribute searches depicted by their saliency maps.

space, clustered around points of likely objects. To find such clusters we apply a kernel-based density maximization method, known as Mean Shift [7]. Clustering is done in image and disparity space, using a 3d Gaussian kernel with a size corresponding to the typical 3d size of objects that can be manipulated. The maximization scheme is iterative and relies on initial center point estimates. As such estimates we use the hypotheses from the attentional system. Examples of segmentation results using this approach can be seen in the second line of Fig. 3.

The above mentioned approach has a number of weaknesses that tend to complicate the extraction of shape attributes. First of all, the approximate size of objects has to be known. Elongated objects tend to be broken up into parts, typically on either side of the object. The most important weakness, however, is the fact that an object can not be reliably segregated from the surface it is placed on, if there is no evidence supporting such a segregation. Without any additional assumption on surface shape or appearence there is no way of telling the surface from the object. However, in many practical scenarios it might be known to the robotic system, that objects of interest can in fact be expected to be located on flat surfaces, such as table tops.

3D Segmentation with Table Plane Assumption. As an alternative approach we test a parallel solution, i.e. segmentation is done independently of the attentional system. The dominant plane in the image is the table top. Using a well-textured surface, it is possible to find the main plane and cut it with a 3D version of the Hough transform. Since the Hough transform requires relatively long computation time of a few seconds, we assume in this scenario that the setup, i.e. camera and table position, does not change. The plane has therefore only to be computed once and can be re-used afterwards. An additional advantage of this solution in contrast to online computation of the dominant plane is that 3D information of objects which greatly occlude the table will not effect the plane clipping. Following the table assumption the 3D points are mapped onto a 2d grid to easily find segments and basic attributes.

The result of transformation and clipping on the scenario given in Fig. 2(a) can be seen in Fig. 2(b). The segmentation of objects is computed on the 2d grid with a simple region growing algorithm grouping pixels into larger regions by expanding them bottom up. The recursive algorithm uses an 8-neighborhood on the binary 2d grid. The



(a) Stereo images (b) 3D points (c) 2d segmentation (d) Reprojection

Fig. 2. Segmentation under the table plane assumption. Disparity information from the stereo images (a) produces 3D points (b). Having defined the dominant plane, the points can be projected onto this plane, where distinctive segments are computed (c) and reprojected to the image (d).

procedure is depicted in Fig. 2(c). Since the grid is thereby segmented, simple shape-based attributes of each segment can be determined and the segments reprojected to 3D points or to the image plane (see Fig. 2(d)). Note that dilation has been applied for the reprojected segments for the later application of point-based object hypotheses verification.

2.3 Attribution of Segments

Each of the produced segments is just a *thing* according to our definition, as the step to an *object* longs for semantics. One way to identify the semantics of a thing in order to derive an object is to link attributes to it. Attributes can be divided in two different groups, which are named intrinsic and extrinsic. Intrinsic attributes are object-centered and thereby theoretically viewpoint-independent (e.g. physical size, color, mass). Extrinsic attributes describe the viewpoint-dependent configuration of an object (e.g. position, orientation), which mostly is measured in the quantitative domain. In our system, the basic intrinsic attributes of covered area, length (along the dominant axis), width (perpendicular to the dominant axis) and height can be qualitatively determined for each segment. The discretization, i.e. if an object is *small* or *large* in size, is adapted to our table manipulation scenario at hand. Additionally, the centroid position of a segment is calculated and its 3D point cloud kept available for further application, e.g. shape approximation and grasping.

3 Experimental Evaluation

Experiments are performed on a Yorick stereo head with four degrees of freedom: neck pan and tilt, and an additional tilt for each camera. The cameras used are 1.3 Mpixel cameras from Allied Vision. The head is controlled so that the cameras are always fixating on something in the center of view. Given commands from the attentional system, the fixation point can be changed through rapid gaze shifts, a process that takes about half a second. Since the extrinsic camera parameters are constantly in change, camera calibration is done on-line. The work presented here is integrated into an existing software system, modularized and containing modules for frame grabbing, camera calibration, binocular disparities, attention, foveated segmentation, recognition and pose

estimation. Modules are implemented as CORBA processes that run concurrently, each module at a different speed. Using a 2.6 GHz dual dual-core Opteron machine, cameras are calibrated and disparities computed at a rate of 25 Hz, while foveated segmentation and recognition is done only upon request.

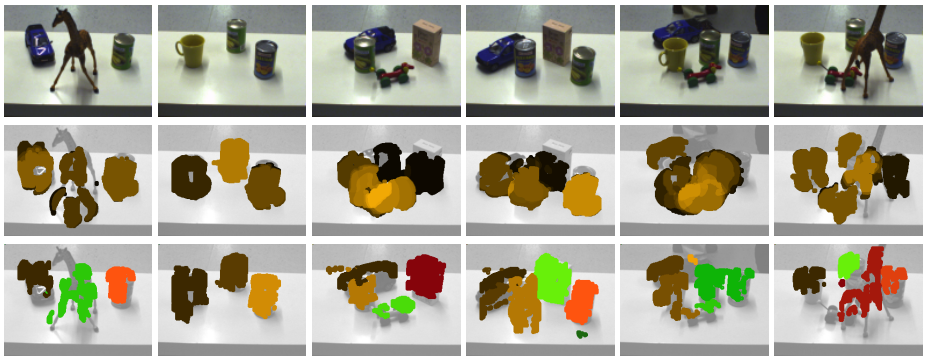
Based on the hypotheses produced by the attentional system from Section 2.1, we will validate these by introducing the segmentation information. As discussed in Section 2.2, segmenting with the table plane assumption can be made in parallel and initially independent of the attentional system. Results of six different scenarios of the segmentation are presented in Fig. 3. The plane was previously computed by using a textured table top in order to apply table clipping throughout the experiments. However, we removed the textured top to provide a clearer comparison of the two segmentation techniques we evaluated.

Although distortion and uncertainty in the disparity calculation clearly influence the results, it can be seen that segmentation in terms of 3D distinction of the objects in the scene works well in general. The errors arising, like oversegmentation (Fig. 3d), undersegmentation (Fig. 3e) or occlusion in height due to the vertical projection (Fig. 3f), can partially be solved by the attribution issues shown in Fig. 4. While the segmentation presented in Fig. 3 (bottom row) depicts segments reprojected to the image plane, the important step inbetween has been the segmentation in the $2\frac{1}{2}$d table plane space. Here, also the attribution of segments, as presented in Section 2.3, takes place. As a simple and neat set, we only use the intrinsic attributes aligned in Tab. 1. The table also shows the distribution on how segments have been attributed along the sample scenarios. Note that the identification of a segment as a specific object has been performed manually to establish the distribution. Though the number of attributes is sparse and the quantification into four levels per attribute is coarse, one can detect differences and similarities of objects. While the car and the dog are mostly attributed *almost flat* (afl) in height, the cans and the mug are usually *medium high* (med) and the giraffe and the sugar box *high* (high). Also note that the mango can, the mug and the peach can are attributed almost alike. This is quite reasonable, as they are very similar in the rough shape domain that we span with the four attributes.

3.1 Validation

We can now combine results of both visual and shape attribution. One the one hand, this set gives an extensible base for attribution of things to make them objects. The framework that takes care of the management of attribute sets and connecting them to actions is an extensible system. On the other hand, we can show in our practical experiment how an extended set of attributes improves the results of object interpretation. Three examples of such validations are shown in Fig. 4.

In Fig. 4(a) we focus on the visual attribute “car-like”, i.e. on hypotheses that the attentional system rates similar to the car model in terms of color and gradients. As it can be seen, the top-ranked hypotheses are on the car in the image (1st column). However, combining this result with the segmentation, hypotheses can be grouped and also neglected (2nd column). Finally, five hypotheses fall in one segment. Comparing the shape attributes of this segment to the distribution shown in Tab. 1 clearly affirms that this segment corresponds to a car-like object. The same process results in a negative



(a) (b) (c) (d) (e) (f)

Fig. 3. Segmentations of sample scenarios (best viewed in color). The original images are shown in the first row. the second row shows results using the Mean-Shift segmentation, the botton row those using the table plane assumption (see Section 2.2). In the latter, (a) and (b) seem well segmented and in (c) there ist just some noise at the table edge. Problems arise for (d)-(f): (d) two segments for the car, (e) one segment for two cans, and (f) the unnoticed dog below the giraffe.

answer for example Fig. 4(b). We look for a "dog-like" object, though there is no dog in the image. While the attention returns ten hypotheses for this search, the shape attribute check clearly neglects that the only segmented area is "dog-like". In Fig. 4(c), the process returns three selected segments. The first one with the strongest hypothesis from the visual attributes is declined by shape attributes again. Both the other segments are very similar and only differ in one shape attribute. However, the interesting result is that there are two objects, both the mango can and the mug, which are very "mango-can"-like. If one would aim at distinguishing between those, this might be approached by new attributes (e.g. more detailed shape and hollowness). On the other hand, both objects are truely can-like in terms of color and shape and would fall in one category of actions performable on them.

4 Conclusion

In this work, we presented a subsystem in which we integrated both visual and shape attributes into the concept of OACs. The overall vision system in which we embed our attribute determination is more general and supports also other applications, as proposed in the Chapter 2.1. Here, we focus on the issue of meaningful attributes that constituted an object as opposed to a thing. The collection of the basic attributes used here is extensible. While finding complex visual attributes like hollowness or emptiness is hard to do, the attributes used in our experiments are computed by the system. In future work, we will also include manipulation attributes, e.g. hollowness or weight, by interacting with a thing.

Besides the buildup of this system, its improvement by combining pure visual attributes with shape attributes has been exemplified. In our experiment visual hypotheses were checked and tested according to shape attributes. Hereby, it is possible to neglect

wrong hypotheses, to cluster and affirm the good ones, or even to distinguish between a car-like object and the plain image of it.

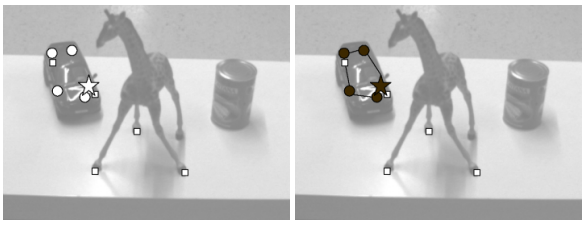
We also compared two types of 3D segmentation techniques for shape attribute generation. The specificity of our system towards table set scenarios with high camera view on the objects supports the table plane assumption. In particular, it has the advantages of a reference system (which is the table) and the hereby introduced spatial arrangement, as most objects are placed next to another than on one another on a table. If this assumption does not hold, manipulation might help by picking up or moving something around. Though the system was kept fixed in combination with an a-priori table plane detection here, our future goal is to dissolve this constraint by a 3D acceleration sensor on the head, linking the vertical gravity vector to mostly horizontal table planes.

References

1. S. Claus and R. Reindeer. A Fake Reference for ICVS Double-Blind Review.
2. C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry, *Human Neurobiology*, 4, pp. 219-227, 1985.
3. S.B. Choi, S.W. Ban, and M. Lee. Biologically motivated visual attention system using bottom-up saliency map and top-down inhibition, *Neural Information Processing - Letters and Review*, 2, 2004.
4. V. Navalpakkam and L. Itti. Sharing Resources: Buy Attention, Get Recognition, in *Int. Workshop Attention and Performance in Computer Vision*, Graz, Austria, Jul 2003.
5. S. Frintrop. *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*, Springer, Lecture Notes in Computer Science, 3899, 2006.
6. L. Itti, C. Koch and E. Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis, *Trans. on Pattern Analysis and Machine Intelligence*, 20, pp. 1254-1259, 1998.
7. D. Comaniciu and P. Meer. Mean Shift: A Robust Approach toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
8. S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003. Special issue on perceptual anchoring.
9. A. Edsinger and C.C. Kemp. Manipulation in Human Environments. In *IEEE/RSJ International Conference on Humanoid Robotics*, Beijing, China, October 2006.
10. C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krüger, and F. Wörgötter. Object Action Complexes as an Interface for Planning and Robot Control. In *IEEE RAS International Conference on Humanoid Robots*, 2006.
11. G. Metta and P. Fitzpatrick. Better vision through experimental manipulation.
12. E. S. Neo, T. Sakaguchi, K. Yokoi, Y. Kawai, and K. Maruyama. Operating Humanoid Robots in Human Environments. In *Workshop on Manipulation for Human Environments, Robotics: Science and Systems*, 2006.
13. D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hewes. A system for continuous learning of visual concepts. In *International Conference on Computer Vision Systems ICVS 2007*, Bielefeld, Germany, March 2007.
14. T. Southey and J.J. Little. Object discovery using motion, appearance and shape. In *AAAI Cognitive Robotics Workshop*, 2006.
15. M. Sutton, L. Stark, and K. Bowyer. Function from visual analysis and physical interaction: a methodology for recognition of generic classes of objects.

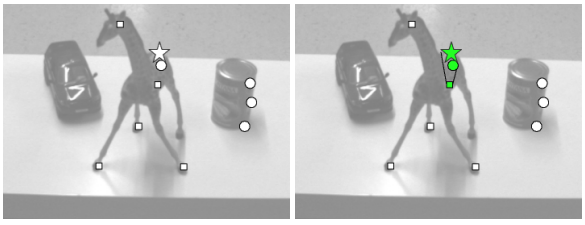
Table 1. Attribution distributions of our 7 test objects (see Fig. 1). Most meaningful is probably distribution of color and shape attributes. For the latter, checking the identity of the objects has been done manually over our 6 scenes (see Fig. 3); in brackets, each object carries the number of appearance along the sequence.

Object	Visual Attributes		Shape Attributes			
	Color	Orientation	Area	Length	Width	Height
	R G B Y	0° 45° 90° 135°	tin sml lrg hug	sht med lng vln	thn med wid vwd	afl med hig vhg
Car (4x)	(12,5,40,10)					
Dog (3x)	(35,22,11,13)					
Giraffe (2x)	(64,1,24,48)					
Mango (6x)	(12,60,21,23)					
Mug (2x)	(57,83,29,113)					
Peach (4x)	(13,8,15,12)					
Sugar (2x)	(28,6,2,14)					



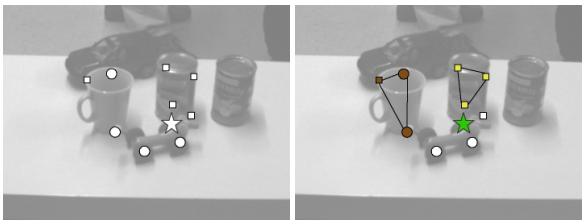
5 best 'car-like' in one segment.
shape: [lrg,med,med,afl]
→ very car-like (Tab. 1)

(a) *"Find the 'car'-like." (source Fig. 3(a))*



3 'dog-like' in one segment.
shape: [lrg,med,med,hig]
→ very dog-unlike (Tab. 1)

(b) *"Find the 'dog'-like." (source Fig. 3(a))*



Best 'can-like' in one segment.
shape: [sml,sht,thn,afl]
→ quite can-unlike (Tab. 1)

3 'can-like' in one segment.
shape: [lrg,lng,med,med]
→ length not can-alike (Tab. 1)

3 'can-like' in one segment.
shape: [lrg,med,med,med]
→ very can-alike (Tab. 1)

(c) *"Find the 'mango can'-like." (source Fig. 3(e))*

Fig. 4. Top-level tasks corresponding to Fig. 1 (continued). Left: The 10 best hypotheses (star = best hypothesis, circles = hypotheses ranked 2-5, small squares = hypotheses ranked 6-10). Right: visual and shape attribute information is merged (connected and colored hypotheses).

Exploration and Planning in a three Level Cognitive Architecture

D. Kraft, E. Başeski, M. Popović,
A. M. Batog, A. Kjær-Nielsen and N. Krüger
University of Southern Denmark, Denmark
{kraft,emre,akn,norbert}@mmmi.sdu.dk,
{mipop05,anbat06}@student.sdu.dk

R. Petrick, C. Geib, N. Pugeault and M. Steedman
University of Edinburgh, United Kingdom
{R.Petrick,C.Geib,npugeaul,steedman}@ed.ac.uk

T. Asfour and R. Dillmann
University of Karlsruhe (TH), Germany
{asfour,dillmann}@ira.uka.de

S. Kalkan and F. Wörgötter
University of Göttingen, Germany
{sinan, worgott}@bccn-goettingen.de

B. Hommel
Leiden University, The Netherlands
hommel@fsw.leidenuniv.nl

***Abstract*— In this work, we describe an embodied cognitive system based on a three level architecture covering a sensorimotor layer, a mid-level layer that stores and reasons about object-action episodes, and a high-level symbolic planner that creates abstract action plans to be realised and further specified at lower levels. The system works in two modes, exploration and execution of plans, that both make use of the very same architecture. We give results of different sub-processes as well as their interaction. In particular, we describe the generation and execution of plans as well as different learning processes taking place either independently of or in parallel to the execution of plans.**

I. Introduction

In this paper, we describe a cognitive system that consists of three hierarchically nested layers: a low-level, sensorimotor layer connecting sensory processors to motor procedures; a mid-level layer that stores object-action episodes (Object-Action Complexes or OACs) and reasons about memorised events; and a high-level symbolic planner that creates abstract action plans to be specified at lower levels. The cognitive system works in two modes. In the first mode the system explores its environment and learns object–action associations. In the second mode, it constructs and performs plans. In both modes it is able to deal with unexpected events or errors (i.e., the system can recover from such situations) and is also able to learn from the different events occurring either in the exploration or plan performance mode.

We introduce a software architecture and its application in a concrete embodied system consisting of an industrial 6 degrees of freedom (DoF) robot with a two finger grasper, haptic sensing by means of a force-torque sensor and a high resolution stereo system in which a preliminary attention system is realized (see Fig. 1). The task we want to address is cleaning up a table, but the proposed architecture allows for more general tasks. Furthermore, in parallel to the planner's performance, the system is provided with a memory of past experiences that allows for learning processes.

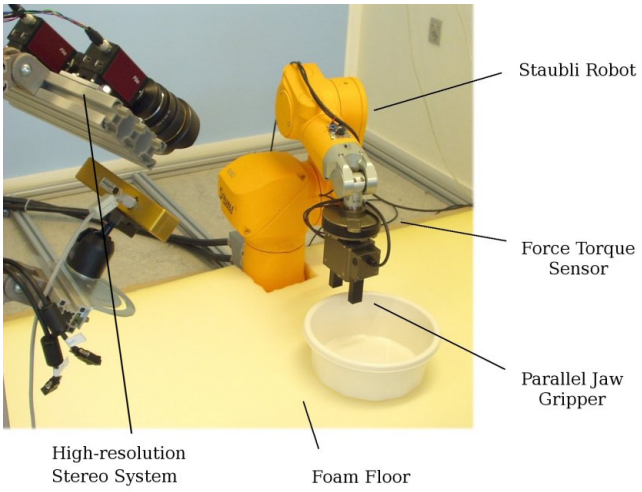


Fig. 1. Embodiment of the Cognitive System

The aim of the architecture is not only to solve the given tasks with best possible performance but to allow for a set of cognitively rich processes in which the robot

- derives a plan to solve the given task,
- executes the plan,
- is able to recognise and react on unexpected events during the exploration or execution of the plan on
  - either a rather low level (for example in case of collisions by withdrawal, see Fig. 2),
  - or on the mid–level by reinspecting the scene with a certain problem statement in mind (e..g., by looking at a certain aspect with higher resolution (see Fig. 3) and hence, resolving the situation within the given plan),
  - or on the high-level by a complete replanning.

In parallel, made experiences in terms of Object Action Complexes (OACs) [1] become stored and transferred to

various learning processes on a mid-level stage.

In its early stages, this work focuses on a limited domain: objects become represented as 3D circle and grasps become associated to these (see Fig. 4). This limitation is merely for development purposes and there is ongoing work toward the extension of the proposed system to arbitrary objects (see [2] and [3]) The only fundamental restrictions of the proposed architecture stem from limitations of the vision (namely, pose estimation of arbitrary objects [3]) and grasping modules (the grasping device only allows to grasp a limited range of objects). We briefly report current progresses on the extension of the initially limited domain.

(a) (b) (c)

Fig. 2. Withdrawal after the detection of a collision. (a) Approaching the object, (b) a collision is detected, (c) start of withdrawal action.

It is not possible here to give a full comparison to other cognitive architectures (for this, see, e.g., [4], [5]). However, we would like to mention as one particularity of the introduced architecture, that planning and exploration are two modes that perform within the same architecture. In this sense, our work is related to classical planning approaches (see, e.g, [6]). It goes beyond these by allowing in parallel for learning on the different levels of the architecture. Since planning and exploration is very much connected, we also extend on approaches focusing on the learning and developping aspect (as done, e.g., in [7]) by realizing also higher level mechanism such as planning in our cognitive architecture.

The paper is structured as follows: section II introduces the system architecture; section III, describes the system's embodiment and visual representation; section IV describes the mid-level and section V the planning level of our architecture.

II. ARCHITECTURE

The system consists of three levels: The low, sensorimotor robot-vision level providing multi-sensorial information and action options, a mid-level on which made experiences are stored and provided for different learning processes on various levels as well as a planning level that generates plans based on the information provided by the robot-vision or the mid-level and which also monitors the execution of these plans. This architecture integrates several approaches and concepts from the computer sciences, artificial intelligence, and cognitive psychology, and it allows for learning and adaptation processes at all three levels.

The robot-vision level provides visual and tactile information to the higher levels and handle action commands, such as:

- apply grasp A on object B,
- start an explorative action such as 'poking', or
- shift attention to a certain location.

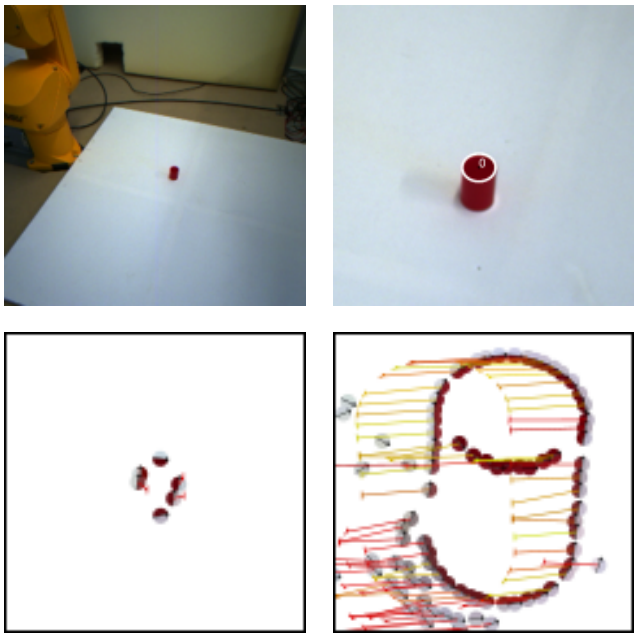
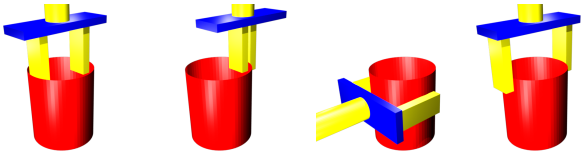


Fig. 3. Circle detection is not successful (left) because of the small number of feature descriptors extracted from a downsampled version of the high resolution images. It is successful (right) when the system focuses on the object at full resolution.

Moreover, it has control mechanisms that allow for the detection of unexpected events, events that would normally either lead to emergency stops and/or damaging the objects or the robot itself. It has pre-programmed behaviours to overcome such situations (see Fig. 2).



A B C D

Fig. 4. Grasp types available in the example domain. They are defined object centric and based on the upper circle.

In its current (preliminary) state, the mid-level is responsible for the following tasks:

- the storage of OACs in memory, and their access by different learning processes,
- the refinement of grasping reflexes and object-action models based on the stored OACs,
- the stabilisation of transient sensorial input into messages passed onto the planning level.

The planning level is responsible for constructing high-level, goal-oriented plans and for feeding these plans to the lower system levels for execution by the robot. To do so, the planning system maintains an abstract model of the objects, properties, and actions available to the robot in the world. The planner also receives regular updates on the state of the

world from the lower levels, which it uses to monitor the success of plans being executed, and to control resensing and replanning activities in the system.

III. Embodiment and Visual Representation

This section presents the robot-vision level of the architecture. The system's embodiment is described in section III-A. The visual representation used for object localisation, learning, and the generation of grasping affordances is briefly described in section III-B. Grasp part associations used in planning and exploration are described in section III-C.

A. Embodiment

To enable interactions with the real world, we use a robot-vision system with the components shown in Fig. 1. The components are:

- The workspace is observed using a 2024×2024 pixels high resolution camera system. In normal working mode downsampled images of the whole scene are used, but it is also possible to get a full resolution image of a region of interest. The camera system is calibrated relatively to the robot.
- A 6-DoF industrial robot is used, together with a two fingers grasper. This enables the system to grasp objects at different locations in the scene.
- A 6-DoF force-torque sensor mounted between robot and grasper allows for the measurement of forces at the wrist. It is used to detect collisions and to back off when a collision is detected (see Fig. 2). To limit the build up of high forces during the reaction time of the system a foam layer was placed on the floor of the scene.

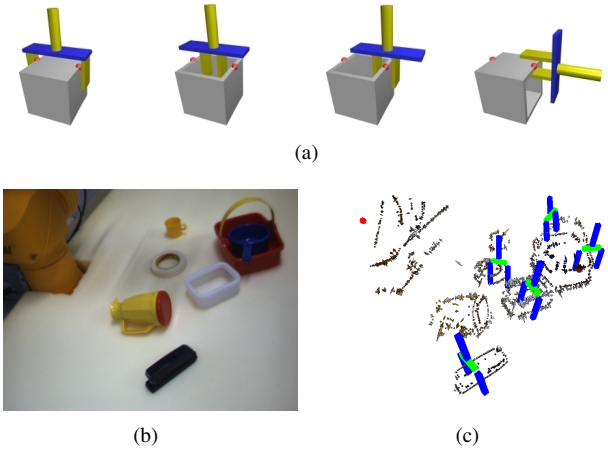
B. Representation of visual information

This work uses the hierarchical representation presented in [8]. An example is presented in Figure 5, which shows what kind of information is processed on the different representation levels. At the lowest level of the hierarchy is the image's pixel RGB values (Fig. 5(a)). The second level processes the results of local filtering operations (Fig. 5(b)), that give rise to the multi-modal 2D primitives at the third level (Fig. 5(c)). This third level processes not only the 2D primitives, but also 2D contours (Fig. 5(d)) created using perceptual organisation (see [9]). The last level contains 3D primitives and 3D contours (Fig. 5(e-f)) created using stereopsis on a pair of the previous level's representations. Circles are detected using the algorithm presented in [10], that is building up on top of the above mentioned representation.

C. Grasp-Part Association

Objects that share common features (parts) often afford the same actions (e.g., an object with an handle can be picked up by grasping it through the handle). We exploit these common parts to initiate learning by transferring the previous experience with one object to another.

In [11], coplanar line-segments are used to predict different grasp types — Fig. 6 shows more recent results on this approach. This can be either seen as a complex reflex — used for generating ground truth for grasp learning — or as a basic form of part-action association. The circle-grasp relation used in the scenario described herein uses a more complex part (the circle). Currently, the associated grasps/part associations are predefined. However, we aim at a learning of these associations from experience. A first step in this direction is described in section IV-A.



(a) (b) (c)

Fig. 6. Coplanar primitives can predict a grasp (see [11]). (a) Different grasp types based on coplanar primitives (marked as red dots). (b) Example scene. (c) The (automaticly selected) best five grasping hypotheses in the example scene.

IV. Mid-level

The mid-level is responsible for a number of tasks (from which some are not yet or only rather naively implemented). In our system, on the mid-level the storage of information used for additional learning processes (as well as the learning as such) is organised (section IV-A and section IV-B). Also the temporal consistency of the permanently varying information on the raw signal level which is required by the planning level is provided (section IV-C). It is partly motivated by the idea that perceptual events and actions are cognitively represented and integrated in a common store [12]. Apart from organising the storage of information (sections IV-A and IV-B) the mid-level also provides episodic pointers to perceptually available objects (i.e., a kind of working memory: section IV-C).

A. Refinement of Grasping Strategy

The grasping affordances associated to the parts (see section III-C) is initially hardwired and, in case of success, give rise to an additional object learning process that requires physical control over the object as achieved after a successful grasp (this is described in section IV-B). Moreover, the grasping behaviour linked to these part affordances generates labelled data that can be used for further learning processes: Since for each attempted grasp success or failure can be measured by the distance between the hand's two fingers after a lifting operation, a large number of training data becomes generated.

In case of the "circle-reflex", it is a priori unclear which grasp type can be used for which circle radius. Fig. 7(d)

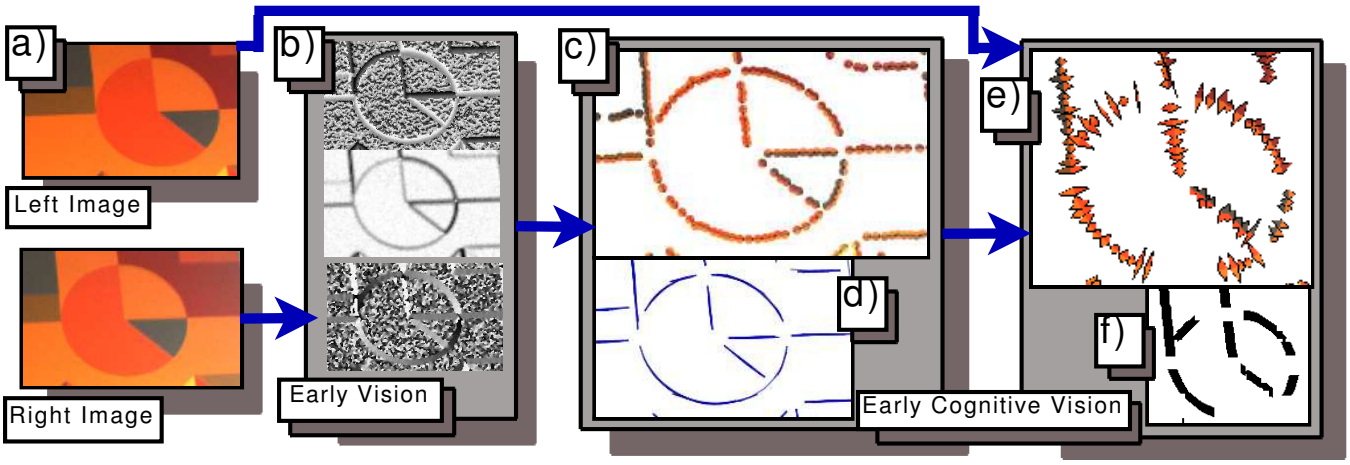


Fig. 5. The different types of visual information processed by the hierarchy of representations **(a)** Original image, **(b)** Filtering results, **(c)** 2D primitives, **(d)** 2D contours, **(e)** 3D primitives, **(f)** 3D contours.

shows distribution of successes and failures for grasp type A generated over a large number of trials. The distributions in real and artificial scenarios are rather different (as shown in Fig. 7(a)). This is because grasps in a real scenario might fail, due to external reasons such as collision before the grasp, or imprecision in the 3D circle reconstruction. Also, grasps might be successful by accident, e.g., when a planned grasp of type A is in reality of another type. Nevertheless, the experiences give valuable information and can be translated into likelihoods of success depending on the radius (see Fig. 7(f)) that can then be used by higher level processes to decide what grasp type should be selected. Note that learning can also be applied in the more difficult case of the coplanarity-based grasping reflex, since success can be measured in the very same way. This, however, requires a much more complex learning framework (see [3]).

B. Birth of the Object

In addition to the refinement of pre-wired reflex behaviours, the exploration behaviour allows the proposed system to learn a three-dimensional visual representation of shape. In [2] we have presented an algorithm that, based on the initial grasping reflexes and on the knowledge of the robot's arm motion, learn models of 3D objects' shape.

A successful grasp on an object endows the robot with control over this object. The system will then manipulate the object to visually inspect it from a variety of viewpoints. The object's motion during this inspection phase is known to be the same as the robot's arm's. This knowledge of the object's motion allows to infer predictions on the visual representation extracted at later stages. Tracking objects visual representations allows to improve the internal representation of their shape, in three respects:

- Because only the object moves as predicted by the robot's arm motion, tracking of visual primitives allows to segment the object from the rest of the scene;
- The integration in the same coordinate system of visual information gathered from multiple viewpoints allows to generate a representation of the object's full 3D shape;
- Tracking aspect's of the object representation allows to reduce inaccuracy in the shape representation.

The result of this process is a full 3D representation of the object shape, with an associated knowledge of the grasp that was successful.

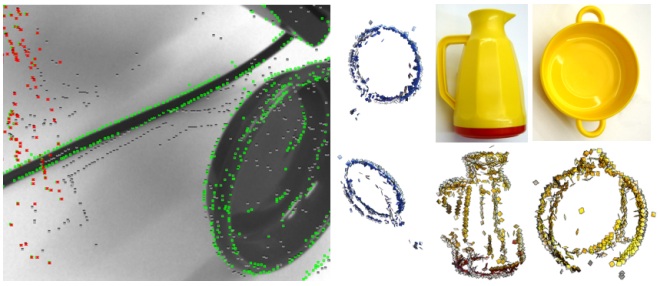


Fig. 8. Birth of the Object. On the left hand side, the dots on the image shows the predicted structures. Spurious primitives and parts of the background are not confirmed by the image, and are shown in grey. The confirmed predictions are shown in green. The middle shows the shape model learned from this object. The right hand side shows the shape model learned from tow other objects. Note that the gap in the shape models correspond to where the robot's hand held the objects.

Fig. 8 shows the results of object learning. Once the object's shape is known to a satisfying level, the planner (see section V) is informed of the new object discovery and of the associated grasp.

Current work endeavour to use the acquired representation for object recognition and pose estimation, therefore extending the simplified 'circle scenario' into a more general object context.

C. Temporal Consistency

The information provided by the robot-vision layer is intrinsically noisy. This leads to

- phantom objects appearing (i.e., false positives),
- objects present in the scene not being detected,
- object labels not being consistent over time.

While the first two cases are not frequent, the third case is a constant problem. The planning layer needs accurate state

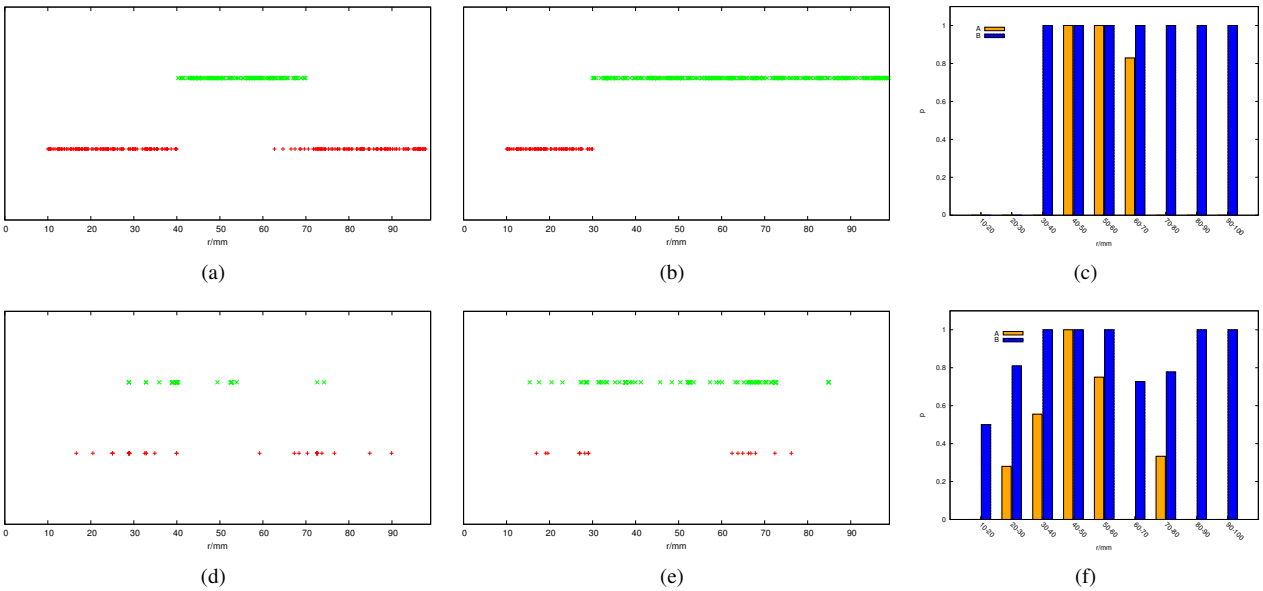


Fig. 7. Grasp experiences and success distributions. (a)-(c) show the results for artificial data, while (d)-(f) show data from real experiences. (a) and (d) show grasping experiences made for grasp type A for different radiuses while (b) and (e) show the same for grasp type B. In these diagrams the green crosses (in the upper row) represent a successfull grasp, while the red crosses (lower row) represent failures. (c) and (f) show success probabilities for the two grasp in discrete radius bins. These can be used as a indicator which grasp to choose for a unknown grasping situation.

information, therefore these problems need to be corrected before the state information is sent to the planner.

These problems become solved by matching the last know state with the sensed information. Objects that are detected in the sensory data that are close in position, orientation, and radius to one object in the last known state are assumed to be the same. This solves the object labelling problem. When no object in the old scene can be matched to a new object, the new object is considered wrong. Undetected objects are treated in the same way. This part of the system is still in a very premature mode, and future work has to address complex issues such as object permanency [13] under occlusions or accidental displacements of objects under unplanned contacts of objects and robot.

V. PLANNING

The planning component of the system is responsible for constructing action plans that direct the behaviour of the robot in order to achieve a specified set of high-level goals. (For instance, in our example scenario, a plan might be constructed to clear all the open objects from the table.) The planning system consists of three main parts: the *high-level domain model*, the *planner* itself (in this case, the PKS planner [14], [15]), and the *plan execution monitor*. Together, these components control all high-level representations and reasoning in the system, and are connected to the rest of the system through a communication architecture which controls the flow of information to and from the planning level.

A. High-level domain model

The high-level domain model consists of a formal representation of the robot's world, described in a STRIPS-like [16] language used as input to the planner. In particular, the domain model specifies the objects and properties in the world, and the actions available to the robot.

1) Objects: Objects in the high-level domain model are simply labels (e.g., strings) that denote actual objects in the real world. (E.g., *obj1* may represent a particular red block on the table in the robot's workspace.) Object names do not typically change over time and, thus, always refer to the same world-level objects. As a result, planning-level objects also act as indices to the actual real-world object information stored at the robot and memory levels, and can be used by all system levels to refer to an object uniquely.

2) Properties: Properties in the domain model are specified by predicates and functions that denote particular qualities of the world, robot, and objects. High-level properties are typically quite abstract and can correspond to combinations of concepts available at the lower levels. For instance, we define the following properties in our example scenario:

- *open*(x) - object x is open,
- *gripperempty* - the robot's gripper is empty,
- *ingripper*(x) - the robot is holding x in its gripper,
- *ontable*(x) - object x is on the table,
- *onshelf*(x) - object x is on the shelf,
- *isin*(x, y) - object x is stacked in object y,
- *clear*(x) - no object is stacked in object x,
- *instack*(x, y) - object x is in a stack with y at its base,
- *radius*$(x) = y$ - the radius of object x is y,
- *shelfspace* $= x$ - there are x empty shelf spaces.

In this case, *gripperempty* closely corresponds to a sensor that detects whether the gripper can be closed without contact, while *ontable* requires a conjunction of data from the visual sensors concerning object positions. Parametrized properties can also be instantiated by specific domain objects. Thus, *ontable*(*obj1*) means "object *obj1* is on the table" and

ingripper(*obj2*) means "object *obj2* is in the gripper."

3) Actions: Actions in the domain model represent high-level counterparts to some of the motor programs available at the robot level. Unlike low-level motor programs, however, high-level actions are modelled with a high degree of abstraction and incorporate state-specific elements into their definitions and operation. For instance, the following high-level actions are defined in our example domain:

- *graspA-table*(x) - grasp x from the table using grasp A,
- *graspA-stack*(x) - grasp x from a stack using grasp A,
- *graspB-table*(x) - grasp x from the table using grasp B,
- *graspC-table*(x) - grasp x from the table using grasp C,
- *graspD-table*(x) - grasp x from the table using grasp D,
- *putInto-object*(x, y) - put x into an object y on the table,
- *putInto-stack*(x, y) - put x into y at the top of a stack,
- *putAway*(x) - put object x away on the shelf,
- *sense-open*(x) - determine whether x is open or not.

These actions do not require 3D coordinates, joint angles, or similar real-valued parameters. Instead, they are defined in an *object-centric* manner, with parameters x and y that can be instantiated with specific objects.

Two types of actions are available to the planner: *physical actions*, that change the state of the world; and *sensing actions*, that observe the state of the world. (As we'll see, an action's type can affect the structure of a generated plan.)

The physical actions in our example domain include actions for manipulating objects in the world. The first five actions indicate the possible grasping options and correspond to the four types of grasps available to the robot (see Fig. 4). Grasp A is divided into two separate actions that account for different object configurations (i.e., an object on the table versus an object at the top of a stack). This avoids the need for conditional action effects in our representation. The two *putInto* actions similarly model different object destinations. The *putAway* action is a generic operation for moving a grasped object to a location on the shelf.

The example domain also includes a high-level sensing action, *sense-open*. This action provides the planner with specific information about an object's state (i.e., its "openness"), without intentionally changing the object's state. At the robot/vision level, this action will ultimately be executed as either a physical test, such as poking an object to check its concavity; or a visual test, such as focusing on the object at a higher resolution. The mid-level memory is responsible for refining *sense-open* actions into robot/vision operations that are appropriate for the given object and context.

B. PKS planner

High-level plans are built using PKS ("Planning with Knowledge and Sensing") [14], [15], a planner that can operate in the presence of incomplete information and sensing actions. Unlike traditional approaches, PKS operates at the "knowledge level" of abstraction, by modelling an agent's knowledge state. By doing so, PKS can reason efficiently about certain types of knowledge, and make effective use of non-propositional features, like functions, which often arise in real-world planning scenarios.

TABLE I
EXAMPLES OF PKS ACTIONS IN THE TABLE CLEARING TASK

Action	Preconditions	Effects
graspA-table(x)	K(*clear*(x)) K(*gripperempty*) K(*ontable*(x)) K(*radius*$(x) \geq$ *minA*) K(*radius*$(x) \leq$ *maxA*)	*add*(K_f, *ingripper*(x)) *add*(K_f, $\neg$*gripperempty*) *add*(K_f, $\neg$*ontable*(x))
sense-open(x)	$\neg K_w$(*open*(x)) K(*ontable*(x))	*add*(K_w, *open*(x))

PKS is based on a generalisation of STRIPS. In STRIPS, a single database is used to represent the world state. Actions update this database in a way that corresponds to their effects on the world. In PKS, the planner's knowledge state is represented by five databases, each of which stores a particular type of knowledge. Actions are described in terms of the changes they make to the databases and, thus, to the planner's knowledge. Table I shows two PKS actions, *graspA-table* and *sense-open*. (K_f is a database that models knowledge of simple facts, while K_w is a specialised database that stores the results of sensing actions that return binary information.)

Two different types of plans can be built in PKS: *linear plans*, which are sequences of actions; and *conditional plans* that contain branches resulting from the inclusion of sensing actions. For instance, using the example domain model for the table clearing task, PKS can construct the simple plan:

graspD-table(*obj1*), *putInto-object*(*obj1*, *obj2*),
graspB-table(*obj2*), *putAway*(*obj2*),

to put an object *obj1* into another object *obj2* before grasping the stack of objects and removing them to the shelf. In this case, the plan is linear since it only contains physical actions.

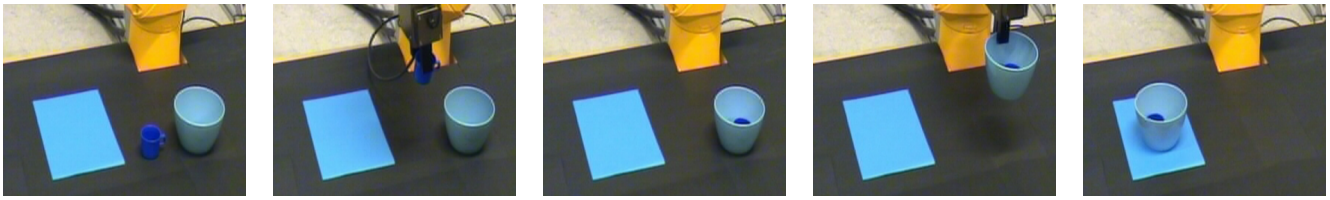
When a plan contains sensing actions, PKS can reason about the possible outcomes of the sensing by adding conditional branches to the plan. For instance, if PKS is given the goal of removing the "open" objects from the table, but does not know whether an object *obj1* is open or not, then it can construct the conditional plan:

sense-open(*obj1*),
branch(*open*(*obj1*))
K^+ : *graspA-table*(*obj1*), *putAway*(*obj1*)
K^- : *nil*,

This plan first senses the truth value of the predicate *open*(*obj1*) and then branches on the two possible outcomes. When *open*(*obj1*) is true (the K^+ branch), *obj1* is grasped and put away; when *open*(*obj1*) is false (the K^- branch), no further action is taken.

C. Plan construction, execution, and monitoring

The planning level interacts with the rest of the system to construct and execute plans. For instance, the initial world state–forming the planner's initial knowledge state–is supplied to the planner from the robot/vision system. Once the planner has such information it constructs a high-level



(a) (b) (c) (d) (e)

Fig. 9. Performing of a Plan: (a) Initial scene. The small blue cup is *obj1*, while *obj2* stands for the light-blue bowl. The blue rectangle represents the shelf area. (b) Scene after executing *graspD-table*(*obj1*). (c) Scene after executing *putInto-object*(*obj1*, *obj2*). (d) Scene after executing *graspB-table*(*obj2*). (e) Scene after executing the final command *putAway*(*obj2*).

plan and feeds it to the robot, one action at a time, upon request from the lower levels. The planner can also send complete plan structures to the mid-level memory, to help it better direct the execution of robot-level actions. Upon action completion, the lower levels inform the planner as to any changes made to the world state, allowing the cycle of plan execution to continue until the end of the plan.

A vital component in this architecture is the plan execution monitor, which assesses action failure and unexpected state information in order to control replanning and resensing activities. In particular, the difference between predicted and actual state information is used to decide between (i) continuing the execution of an existing plan, (ii) asking the vision system to resense a portion of a scene at a higher resolution in the hope of producing a more detailed state report, and (iii) replanning from the unexpected state.

The plan execution monitor also has the added task of managing the execution of plans with conditional branches, resulting from the inclusion of sensing actions (such as *sense-open*)). When a sensing action is executed at the robot level, the results of the sensing will be passed to the planning level as part of a state update. Using this information, the plan execution monitor can then decide which branch of a plan it should follow, and feed the correct sequence of actions to the lower levels. If such information is unavailable, resensing or replanning is triggered as above.

D. Performing of a Plan

Fig. 9 shows the execution of the plan mentioned in section V-B which is repeated here:

$$\textit{graspD-table}(\textit{obj1}), \textit{putInto-object}(\textit{obj1}, \textit{obj2}),$$
$$\textit{graspB-table}(\textit{obj2}), \textit{putAway}(\textit{obj2}).$$

This is a linear plan and therefore does not contain any sensing actions. It is constructed based on the objects in the initial scene.

VI. Discussion

We have introduced a three level cognitive architecture that is used in an embodied system to execute plans. Learning at different levels takes place in parallel. It also allows for fine-tuning as well as the extension of the application domain in which planning as well as the execution of plans can be performed. We are aware that some components of the introduced system are still in a premature state. However, we think that the outlined interaction between the different levels and sub-modules shows the potential of a developping cognitive system.

Acknowledgement

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

[1] "Pacoplus: Perception, action and cognition through learning of object-action complexes," *IST-FP6-IP-027657, Integrated Project*, 2006-2010.

[2] N. Pugeault, E. Baseski, D. Kraft, F. Wörgötter, and N. Krüger, "Extraction of multi–modal object representations in a robot vision system," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2007.

[3] R. Detry and J. Piater, "Hierarchical integration of local 3d features for probabilistic pose recovery," *Robot Manipulation: Sensing and Adapting to the Real World, 2007 (Workshop at Robotics, Science and Systems)*, 2007.

[4] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," Computational Learning Laboratory, CSLI, Stanford University, CA, Tech. Rep., 2006.

[5] R. Rolfe and B. Haugh, "Integrated Cognition: A Proposed Definition of Ingredients, A Survey of Architectures, and an Example Architecture," Institute for Defense Analyses, Tech. Rep., 2004.

[6] S. Hutchinson and A. Kak, "Extending the classical AI planning paradigm to robotic assemblyplanning," in *IEEE Int. Conf on Robotics and Automation*, 1990.

[7] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning About Objects Through Action - Initial Steps Towards Artificial Cognition," in *IEEE Int. Conf on Robotics and Automation*, 2003, pp. 3140–3145.

[8] N. Krüger, M. Lappe, and F. Wörgötter, "Biologically motivated multi-modal processing of visual primitives," *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour*, vol. 1, no. 5, pp. 417–428, 2004.

[9] N. Pugeault, F. Wörgötter, and N. Krüger, "Multi-modal scene reconstruction using perceptual grouping constraints," in *Proc. IEEE Workshop on Perceptual Organization in Computer Vision (in conjunction with CVPR'06)*, 2006.

[10] E. Başeski, D. Kraft, and N. Krüger, "A Hierarchical 3D Circle Detection Algorithm Applied in a Grasping Scenario," in *(submitted to) The 6th International Conference on Computer Vision Systems Conference Paper*, 2008.

[11] D. Aarno, J. Sommerfeld, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft, and N. Krüger, "Early reactive grasping with second order 3d feature relations," *IEEE International Conference on Robotics and Automation (ICRA), Workshop: From features to actions - Unifying perspectives in computational and robot vision*, 2007.

[12] B. Hommel, J. Müsseler, G. Aschersleben, and W. Prinz, "The theory of event coding (tec): A framework for perception and action planning," *Behavioral and Brain Sciences*, vol. 24, pp. 849–878, 2001.

[13] J. Piaget, *The psychology of intelligence*, 1976.

[14] R. P. A. Petrick and F. Bacchus, "A knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002)*. AAAI Press, 2002, pp. 212–221.

[15] ——, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*. AAAI Press, 2004, pp. 2–11.

[16] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, pp. 189–208, 1971.

International Journal of Humanoid Robotics

Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes

Dirk Kraft
University of Southern Denmark, Odense, Denmark, kraft@mmmi.sdu.dk

Nicolas Pugeault
University of Edinburgh, Edinburgh, UK and
University of Southern Denmark, Odense, Denmark, npugeaul@inf.ed.ac.uk

Emre Başeski, Mila Popović
University of Southern Denmark, Odense, Denmark
emre@mmmi.sdu.dk, mipop05@student.sdu.dk

Danica Kragić
Royal Institute of Technology, Stockholm, Sweden, dani@kth.se

Sinan Kalkan, Florentin Wörgötter
BCCN, University of Göttingen, Göttingen, Germany
{sinan,worgott}@bccn-goettingen.de

Norbert Krüger
University of Southern Denmark, Odense, Denmark, norbert@mmmi.sdu.dk

We describe a process in which the segmentation of objects as well as the extraction of the object shape becomes realized through active exploration of a robot vision system. In the exploration process, two behavioral modules that link robot actions to the visual and haptic perception of objects interact. First, by making use of an object independent grasping mechanism, physical control over potential objects can be gained. Having evaluated the initial grasping mechanism as being successful, a second behavior extracts the object shape by making use of prediction based on the motion induced by the robot. This also leads to the concept of an 'object' as a set of features that change predictably over different frames.

The system is equipped with a certain degree of generic prior knowledge about the world in terms of a sophisticated visual feature extraction process in an early cognitive vision system, knowledge about its own embodiment as well as knowledge about geometric relationships such as rigid body motion. This prior knowledge allows for the extraction of representations that are semantically richer compared to many other approaches.

Keywords: Early Cognitive Vision, Grasping, Exploration

1. Introduction

According to Gibson[1] an object is characterized by three properties: It

O1 has a certain minimal and maximal size related to the body of an agent,
O2 shows temporal stability, and
O3 is manipulatable by the agent.

Note that all these three properties are defined in relation to the agent (even temporal stability (O2) is relative to the agents lifetime span). Hence, no general agent independent criterion can be given. For an adult, a sofa certainly fulfills all three properties but for a fly, a sofa is more a surface than an object.

The detection of 'objectness' according to the three properties described above is not a trivial task. When observing a scene, usually in a visual system, a number of local features become extracted for which it is unclear whether and to which object they correspond to. Actually, property O3 can only be tested by acting on the scene in case that no prior object knowledge is available.

In many artificial systems, in particular in the context of robotics, the object shape is given by a CAD representation a priori and is then used for object identification and pose estimation (see, e.g., Lowe[2]). However, CAD representations are not available in a general context and hence for any cognitive system, it is an important prerequisite that it is able to learn object representations from experience.

In this paper, we address both problems: We introduce a procedure in which the objectness becomes detected based on the three Gibsonian criteria mentioned above. In addition, the object shape becomes extracted by making use of the coherence of motion induced by the agent after having achieved physical control over something that might turn out to become an object.

Our approach is making use of the concept of Object Action Complexes (OACs) where we assume that objects and actions (here the 'grasping action' and controlled object movement) are inseparably intertwined. Hence, the intention of performing a grasp, the actual attempt to grasp and the evaluation of its success as well as a controlled movement of the object in case of a successful grasp will let the 'objectness' as well as a representation of the object's shape emerge as the consequence of the actions of the cognitive agent[a].

It is worth noting that both aspects, achieving physical control over a *thing*[b] as well as the extraction of object shape is based on a significant amount of prior knowledge, which however is much more generic than a CAD model of an object. More specifically, this prior consists of the system's knowledge about

[a]We note that this extends the notion of 'affordances' by Gibson. According to Gibson: Objects afford actions. While this remains true, it is also — in our hands — the case that an action defines an object. For example the action of drinking defines a cup, where the action of 'placing on top' makes the same (!) thing a pedestal (an upside down cup).
[b]We denote with 'thing' something that causes the extraction of a visual feature but which is not yet characterized as an object since it could be for example also something fixed in the workspace of the robot and hence does not fulfill condition O3.

1) its own body in terms of the shape, the degrees of freedom and the current joint configuration of the robot arm as well as the relative position of the stereo camera system and the robot co-ordinate system,
2) a developed early cognitive system[3,4] that extracts local multi-modal symbolic descriptors (see Fig. 1(a–e)), in the following called primitives, and relations defined upon these primitives expressing statistical and deterministic properties of visual information (see Fig. 2).
3) two behavior modules in terms of two OACs:

   B1 An object independent 'grasping reflex' leads in some cases to successful grasping of potential objects (Fig. 1e shows the end-effector's pose for one successful grasp). Note that here it is less important to have a high success-rate of grasping attempts but that is is more important that a success is actually *measurable* and that it then triggers a second exploration mechanism (see B2).
   The 'grasping reflex' is based on three semantic relations defined within the early cognitive vision system: First, co-planarity of descriptors indicate surfaces and by that possible grasping options. The co-planarity relation is enhanced by a co-linearity and co-colority relation to further enhance the success rate of the 'grasping reflex'.

   B2 After a successful grasp an accumulation module explores the object by looking at different views of the object (see Fig. 1(f,g)) and accumulating this information to determine the objectness of the thing as well as to extract the shape of the object (Fig. 1(h)). This accumulation module is based on prediction based on a rigid body motion relation between primitives. Having gained physical control over an object by the grasping reflex allows for inducing a rigid body motion on the object and by that the object (its objectness as well as its shape) can be characterized by the set of visual descriptors changing according to the induced motion.

The idea of taking advantage of active components for vision is in the spirit of active vision research[5,6]. The grounding of vision in cognitive agents has been addressed for example by a number of groups in the context of grasping[7,8] as well as robot navigation[9].

The work of Fitzpatrick and Metta[7] is the most related one to our approach since the overall goal as well as the hardware set up is similar: Finding out about the relations of actions and objects by exploration using a stereo system combined with a grasping device. We see the main distinguishing feature of this work to our approach in the amount of pre-structure we use. For example, we assume a much more sophisticated vision system that covers multiple visual modalities in a condensed form as well as visual relations defined upon them. This allows us to operate in a highly structured feature space where, instead of pixel-wise representations, we can operate on local symbols for which we can predict changes not only of position but also other feature attributes such as orientation and color. Furthermore, the use

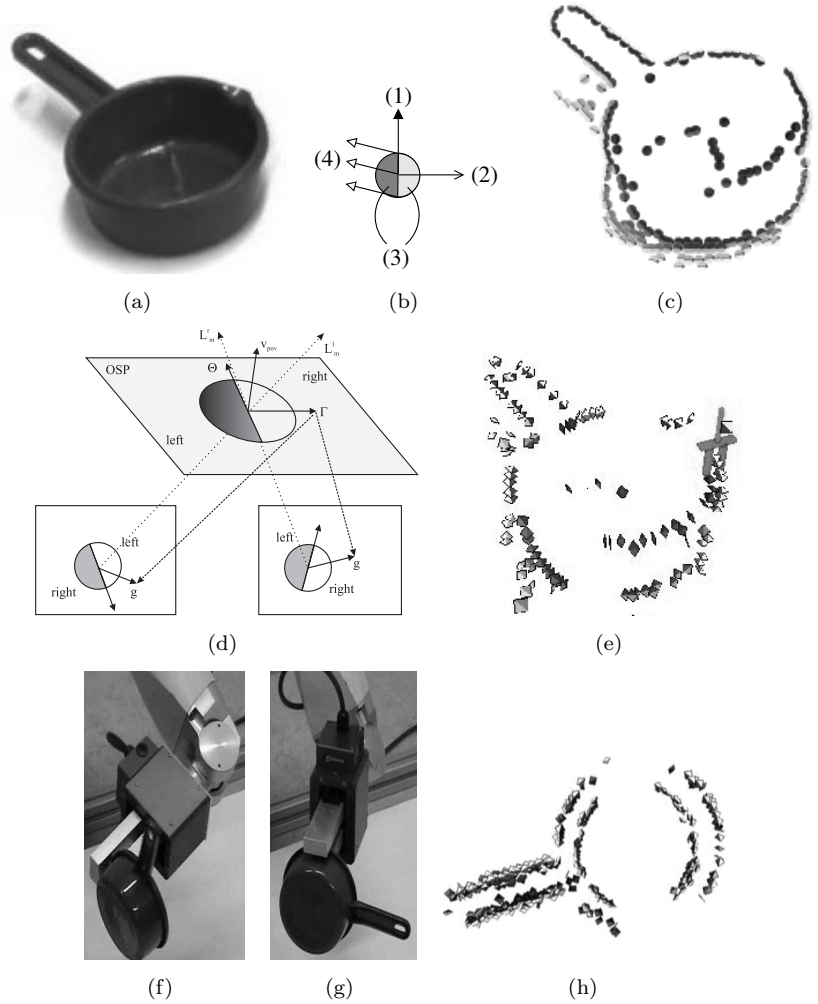


Fig. 1. Overview of the system. (a) Image of the scene as viewed by the left camera at the first frame. (b) Symbolic representation of a primitive wherein (1) shows the orientation, (2) the phase, (3) the color, (4) the optic flow of the primitive. (c) 2D primitives extracted at one object in the scene from (a). (d) Illustration of the reconstruction of a 3D primitive from a stereo pair of 2D primitives. (e) 3D primitives reconstructed from the scene and one grasping hypothesis. (f–g) Two views of robot rotating the grasped object to build its 3D representation. (h) The learned 3D representation of the object.

of a very precise industrial robot allows for a precise generation of changes exploited for the extraction of the 3D shape of the object.

It is not clear what exact prior knowledge can be assumed in the human system. However, there exist strong indications for an innate concept of 3D space as well as for sophisticated feature extraction mechanisms being in place very early

in visual experience. For a discussion of this issue see for example Kellmann and Arterberry[10]. The question of prior knowledge in the context of depth perception and possible consequences for the design of artificial systems is described in Krüger and Wörgötter[11].

Similar to Fitzpatrick and Metta[7], we assume first 'reflex-like' actions that trigger exploration. However, since in our system the robot knows about its body and the 3D geometry of the world and since the arm can be controlled more precisely, these reflexes can make use of more complex visual events. As a consequence we can make use of having physical control over the object and therefore extract rather precise 3D information (in addition to the appearance based information coded in the primitives).

Modayil and Kuipers[9] addressed the problem of detection of objectness and the extraction of object shape in the context of a mobile robot using laser information. Here also motion information (in terms of the odometry of the mobile robot) is used to formulate predictions. In this way, they were able to extract a top-view of the 3D shape of the object however only in terms of geometric information and only in terms of a 2D projection to the ground floor.

The paper is organized as following: In Section 2 the early cognitive vision system is briefly described. In Section 3 and 4 we give a description of the two sub-modules, i.e., the grasping reflex and the accumulation scheme. Sub-aspects of the work have been presented at two workshops[12,13].

2. An Early Cognitive Vision System

In this section, we introduce the visual system in which the detection of 'objectness' as well as the acquisition of the object representation is taking place. The system is characterized by rather structured prior knowledge: First, a scene representation is computed in terms of local symbolic descriptors (in the following called primitives) covering different visual modalities as well as 2D and 3D aspects of visual data (Section 2.1). Second, there are relations defined upon the symbolic descriptors that cover spatial and temporal dependencies as briefly described in Section 2.2. It is only the use of this prior knowledge that allows for the formulation of the two OACs described in Sections 3 and 4.

2.1. *Multi-modal primitives as local scene descriptors*

In this work we use local, multi-modal contour descriptors hereafter called *primitives*[3,4] (see Fig. 1). These primitives give a semantically meaningful description of a local image patch in terms of position as well as the visual modalities orientation, color and phase. The importance of such a semantic grounding of features for a general purpose vision front-end, and the relevance of edge-like structures for this purposes was discussed, e.g., by Elder[14].

The primitives are extracted sparsely at locations in the image which are most likely to contain edges. The sparseness is assured using a classical winner-take-all

operation, ensuring that the generative patches of the primitives do not overlap. Each primitive encodes the image information contained by a local image patch. Multi-modal information is gathered from this image patch, including the position $\boldsymbol{x}$ of the center of the patch, the orientation θ of the edge, the phase ω of the signal at this point, the color $\boldsymbol{c}$ sampled over the image patch on both sides of the edge, the local optical flow $\boldsymbol{f}$ and the size of the patch ρ. Consequently a local image patch is described by the following multi-modal vector:

$$\pi = (\boldsymbol{x}, \theta, \omega, \boldsymbol{c}, \boldsymbol{f}, \rho)^T, \tag{1}$$

that we will name *2D primitive* in the following. The primitive extraction process is illustrated in Fig. 1.

In a stereo scenario, *3D primitives* can be computed from correspondences of 2D primitives (Fig. 1)

$$\Pi = (\boldsymbol{X}, \boldsymbol{\Theta}, \Omega, \boldsymbol{C})^T, \tag{2}$$

where $\boldsymbol{X}$ is the position in space, $\boldsymbol{\Theta}$ is the 3D orientation, Ω is the phase of the contour, and $\boldsymbol{C}$ is the color on both sides of the contour. For details see Pugeault[15].

2.2. *Perceptual relations between primitives*

The sparseness of the primitives allows for the formulation of four *structural relations* between primitives that are crucial in our context since they allow us to relate feature constellations to grasping actions (in the first OAC in Section 3) or visual percepts in consecutive frames (in the second OAC described in Section 4). See Kalkan et al.[16] for more details.

Co-planarity: Two spatial primitives Π_i and Π_j are co-planar iff their orientation vectors lie on the same plane. The co-planarity relation is illustrated in Fig. 2(b). In the context of the grasping reflex described in Section 3, grasping actions become associated to the plane spanned by co-planar primitives.

Collinear grouping (i.e., collinearity): Two 3D primitives Π_i and Π_j are collinear (i.e., part of the same group) iff they are part of the same contour. Due to uncertainty in the 3D reconstruction process, in this work, the collinearity of two spatial primitives Π_i and Π_j is computed using their 2D projections π_i and π_j. Collinearity of two primitives is illustrated in Fig. 2(a).

Co-colority: Two spatial primitives Π_i and Π_j are co-color iff their parts that face each other have the same color. In the same way as collinearity, co-colority of two spatial primitives Π_i and Π_j is computed using their 2D projections π_i and π_j. In Fig. 2(c) a pair of co-color and non co-color primitives are shown. Testing for collinearity and co-colority help to reduce the number of generated grasping hypotheses (see Section 3.2).

Rigid body motion: The change of position and orientation induced by a rigid body motion between two frames at time t and $t+1$ ($\Pi^{t+1} = \mathrm{RBM}(\Pi^t)$) can be computed analytically[17], phase and color can be approximated to be constant.

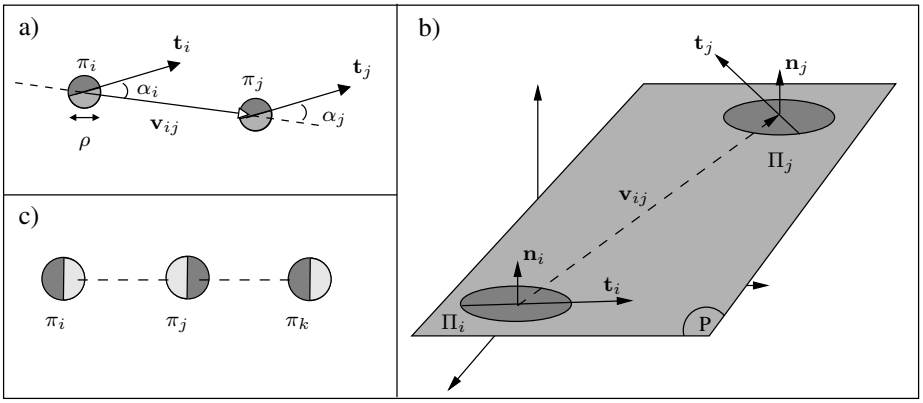


Fig. 2. Illustration of the relations between a pair of primitives. (a) Collinearity of two 2D primitives π_i and π_j. (b) Co-planarity of two 3D primitives Π_i and Π_j. (c) Co-colority of three 2D primitives π_i, π_j and π_k. In this case, π_i and π_j are co-color, so are π_i and π_k; however, π_j and π_k are not co-color.

3. Grasping Reflex

In this section, we describe the first OAC that leads to a physical control over objects. Note that a high success rate is not important in this context, but more that the success can be evaluated by haptic feedback which then gives indications to proceed with another OAC described in Section 4.

3.1. *Elementary grasping actions associated to co-planar primitives*

Coplanar relationships between visual primitives suggest different graspable planes. Fig. 3(a) shows a set of spatial primitives on two different contours l_i and l_j with co-planarity, co-colority and collinearity relations.

Four elementary grasping action (EGA) types will be considered as shown in Fig. 3(b-e). EGA type 1 (EGA1) is a 'pinch' grasp on a thin edge like structure with approach direction along the surface normal of the plane spanned by the primitives. EGA type 2 (EGA2) is an 'inverted' grasp using the inside of two edges with approach along the surface normal. EGA type 3 (EGA3) is a 'pinch' grasp on a single edge with approach direction perpendicular to the surface normal. EGA type 4 (EGA4) is a wide grasp making contact on two separate edges with approach direction along the surface normal.

EGAs are parameterized by their final pose (position and orientation) and the initial gripper configuration. For the simple parallel jaw gripper, an EGA will thus be defined by seven parameters: EGA$(x, y, z, k, l, m, \delta)$ where $\mathbf{p} = [x, y, z]$ is the position of the gripper 'center' according to Fig. 3(f); k, l, m are the roll, pitch and yaw angles of the vector $\mathbf{n}$; and δ is the gripper opening, see Fig. 3(f). Note that the gripper 'center' is placed in the 'middle' of the gripper.

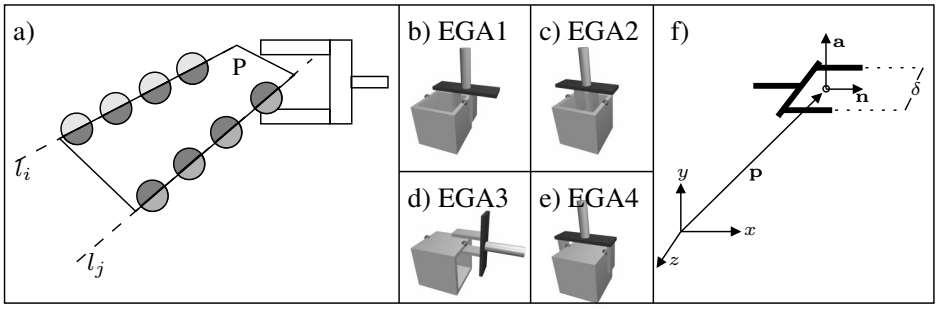


Fig. 3. (a) A set of spatial primitives on two different contours l_i and l_j that have co-planarity, co-colority and collinearity relations; a plane P defined by the co-planarity of the spatial primitives and an example grasp suggested by the plane. (b–e) Elementary grasping action types, EGA1, EGA2, EGA3 and EGA4 respectively. Please note, besides the two primitives (marked as spheres) defining the concrete EGA only the surfaces touched by the gripper are needed for the grasp to be successful. The cube form given here is used to illustrate the differences in the different EGA types and their applicability. (f) Parameterization of EGAs.

These grasp parameters are computed from coplanar pairs of 3D-primitives. Let $\Gamma = \{\Pi_1, \Pi_2\}$ be a primitive pair for which the coplanar relationship is fulfilled. Let Γ_i be the i-th pair and $\mathbf{p}$ the plane defined by the coplanar relationship of the primitives of Γ_i. Let $\Lambda(\Pi)$ be the position of Π and $\Theta(\Pi)$ be the orientation of Π. The parameterization of the EGAs is given with the gripper normal $\mathbf{n}$ and the normal $\mathbf{a}$ of the surface between the two fingers as illustrated in Fig. 3(f). From this, the yaw, pitch and roll angles can be easily computed. For example for EGA1, there will be two possible parameter sets given the primitive pair $\Gamma = \{\Pi_1, \Pi_2\}$. The parameterization is as follows:

$$\begin{aligned} \mathbf{p}_{\text{gripper}} &= \Lambda(\Pi_i), \\ \mathbf{n} &= \nabla(\mathbf{p}), \\ \mathbf{a} &= \mathbf{perp}_{\mathbf{n}}(\Theta(\Pi_i))/ \parallel \mathbf{perp}_{\mathbf{n}}(\Theta(\Pi_i)) \parallel \text{ for } i = 1, 2, \end{aligned} \tag{3}$$

where $\nabla(\mathbf{p})$ is the normal of the plane $\mathbf{p}$ and $\mathbf{perp}_{\mathbf{u}}(\mathbf{a})$ is the projection of $\mathbf{a}$ perpendicular to $\mathbf{u}$. The details of how the other EGAs are computed can be found in Aarno et al.[12].

The main motivation for choosing these grasps is that they represent the simplest possible two fingered grasps humans commonly use which can also be simulated on our robot system. The result of applying the EGAs can be evaluated by the information given by the gripper (Schunk, PT-AP 70) which gives the distance between the two jaws at each instance of time.

For EGA1, EGA3 and EGA4, a failed grasp can be detected by the fact that the gripper is completely closed. For EGA1 and EGA3, the expected grasp is a pinch type grasp, i.e. narrow. Therefore, they can also 'fail' if the gripper comes to a halt too early. EGA2 fails if the gripper is fully opened, meaning that no contact was

made with the object. If none of the above situations is encountered the EGA is considered to be successful.

3.2. *Limiting the number of actions*

For a typical scene, the number of coplanar pairs of primitives is in the order of $10^3 - 10^4$. Given that each coplanar relationship gives rise to six different grasps from the four different categories, it is obvious that the number of suggested actions must be further constrained. In addition, there exist many coplanar pairs of primitives affording similar grasps.

To overcome some of the above problems, we make use of the structural richness of the primitives. First, their embedding into collinear groups naturally clusters the grasping hypotheses into sets of redundant grasps from which only one needs to be tested. Furthermore, co-colority, gives an additional hypothesis for a potential grasp. Aarno et al.[12] quantified the reduction in EGA hypotheses using collinearity and co-colority in a simulation environment, showing that the number of EGAs can be reduced systematically.

3.3. *Experimental evaluation*

To evaluate the grasping reflex we made experiments within the simulation environment GraspIt[18] and with a real scene. In the GraspIt environment, we evaluated success rates in scenes of different complexity (see Fig. 4(a–d) for a number of successful grasps on two scenes). The success rate was dependent on the scene complexity, ranging from appr. 90%[c] in the case of a simple plane (see Fig. 4(a,b)), to around 25% for scenes of larger complexity (Fig. 4(c,d)).

We then evaluated the exploration strategy on a real scene (see Fig. 4(e)). After reconstructing 3D-primitives from stereo images (Fig. 4(h)), 912 EGAs were generated. However, in a real set-up there are additional constraints such as the definition of a region of interest[d] and the fact that not all EGAs are actually performable due to limited workspace[e]. In addition, grasps leading to collisions with the floor or the wall need to be eliminated. Table 1 shows the effects of the reductions.

In a full exploration sequence, the system attempts to perform one of the 50 remaining EGAs. A failure to grasp an object generally causes changes in the scene, and the whole sequence of capturing images, generating and reducing EGAs would

[c]A success is counted when one of the six EGAs (two instantiations of each EGA1 and EGA3, one of EGA2 and EGA4) generated by a primitive pair has been performed successfully

[d]The region of interest serves two purposes: 1. It represents a computationally cheap way to remove EGAs that would be reduced by the later, more expensive reachability check. 2. It prevents grasping attempts in regions in which no objects should be placed. In our concrete setup, the region of interest is defined as a cube in front of the robot.

[e]Note that the workspace needs to be defined in terms of a 6D pose and that even when a 3D point is reachable, it is not certain that the desired end-effector orientation can be achieved at this point.

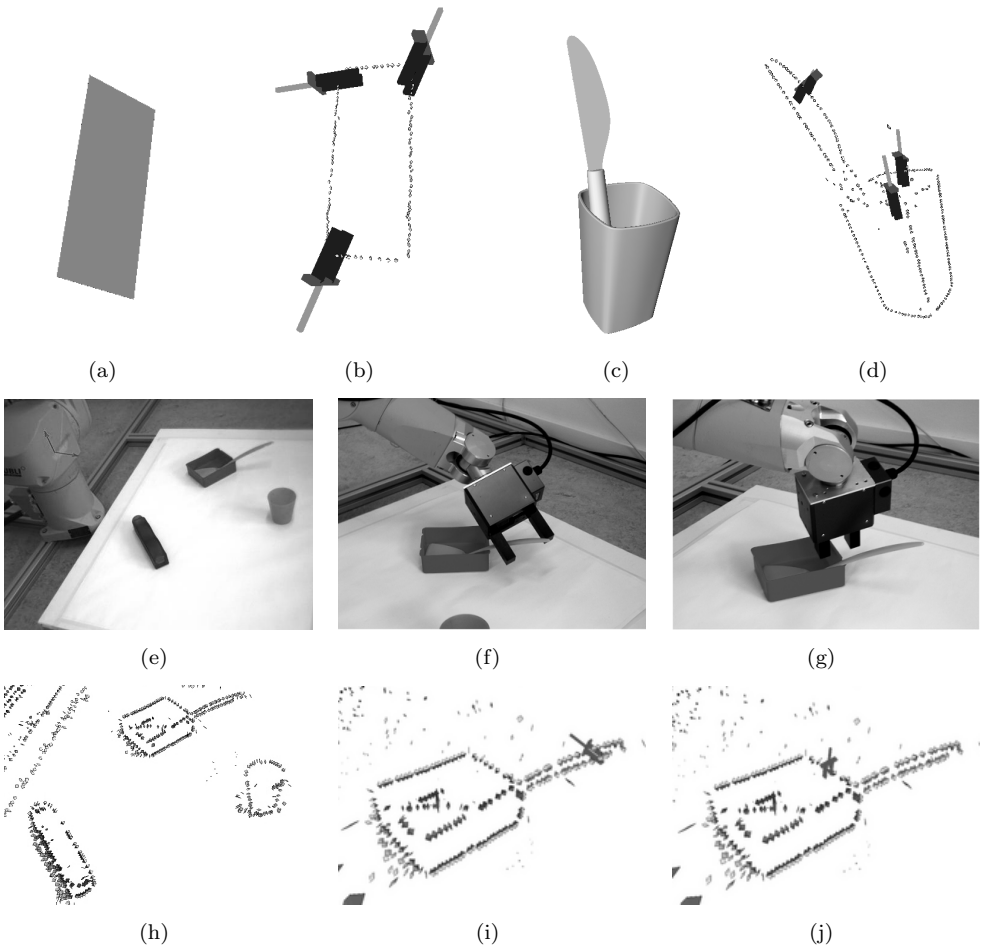


Fig. 4. Evaluation of the grasping reflex in simulation (a–d) and in a real robot environment (e–j). (a,c) Artificial grasping scenes. (b,d) Selected grasping hypotheses generated for the scenes shown in (a,c). (e) The view from the left camera for the real environment. Origin and orientation of the world coordinate frame are illustrated in top left corner. (h) Extracted 3D primitives for the whole scene (see (e)) displayed in the visualization environment. (f,g) The robot arm executing successful grasps. (i,j) The grasps from (f,g) shown in the 3D visualization environment (enlarged).

Table 1. The results of applying reductions to the initial set of EGAs.

Reductions:	Initial number of EGAs	remaining	relative reduction
to region of interest	912	228	75.0%
to reachable configurations	123	105	53.9%
collision free (floor)	105	50	52.4%

be repeated. However, for the purpose of evaluating the whole set of proposed EGAs for a single scene, the objects in this experiment were placed at their original position after each attempted grasp.

In the specific scenario shown in Fig. 4(e), three out of the four objects could be grasped by the reflex. Out of 50 grasps, 7 lead to physical control over objects. In one case, the contact area was too small, leading to an unstable grip, and the accumulation module (see Section 4.1) could not be applied.

4. Detection of Objectness and Object Shape

Having achieved physical control over an object, measured by the distance between the gripper's jaws after closing or opening (in case of EGA2), a second OAC is triggered that makes use of the additional capability of the agent: actively manipulating the object.

If the object's motion within the scene is known, then the relation between this object's features in two subsequent frames becomes deterministic (excluding the usual problems of occlusion, sampling, etc.). This means that a 3D-primitive that is present in one frame is subject to a transformation that is fully determined by the object's motion: generally a change of 3D position and 3D orientation.[f] If we assume that the motion between consecutive frames is reasonably small then a contour will not appear or disappear unpredictably, but will have a life-span in the representation, between the moment it enters the field of view and the moment it leaves it. Assuming having a fully calibrated system and having physical control over the object (as gained by the first OAC described in Section 3), we can compute the 3D-primitives' change in camera coordinates.

These predictions are relevant in different contexts:

Establishment of objectness: The objectness of a set of features is characterized by the fact that they all move according to the robot's motion. This property is discussed in the context of a grounded AI planning system in Geib et al.[19].

Segmentation: The system segments the object from the rest of the scene using its predicted motion.

Disambiguation: Erroneous 3D-primitives can be characterized (and eliminated) by inconsistent motion according to the predictions (see also Krüger et al.[20]).

Learning the object model: A full 3D model of the object can be extracted by merging different $2\frac{1}{2}$D views created by the motion of the end effector.

[f] We neglect the effects of lighting and reflection, and assume that phase and color stay constant.

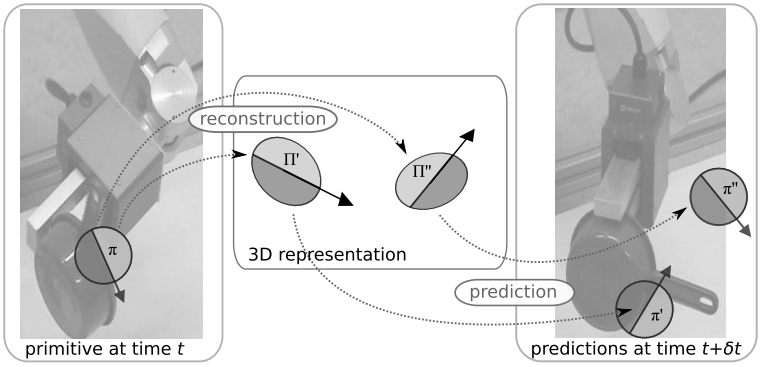


Fig. 5. Example of the accumulation of a primitive (see text).

4.1. *Making predictions from the robot motion*

If we consider a 3D-primitive $\Pi_i^t \in \boldsymbol{\mathcal{S}}_t$ describing an object's contour at time instant t, and assume that the object's motion is known between the two instants t and $t + \Delta t$, then we can predict this primitive's position at time $t + \Delta t$.

The projection of 3D-primitives to the image domain predicts where 2D-primitives should be extracted from each camera's image at time $t + \Delta t$. It is then possible to assess the correctness of a reconstructed 3D-primitive by how reliably it is confirmed by subsequently extracted 2D-primitives.

This prediction/verification process is illustrated in Fig. 5. The left image is taken from a scene at time t; the right image is taken from the same scene, at a later time $t + \delta t$. Assuming that a primitive π is extracted at time t, and lead to two distinct, mutually exclusive, putative 3D reconstructions Π' and Π''. If the object that π describes is subjected to a known motion $M_{t\rightarrow t+\delta t}$, then this motion knowledge allows for making predictions on where in the image this primitive should manifest itself at time $t+\delta t$. Mutually exclusive putative 3D-primitives (Π' and Π'') will lead to distinct predictions (π' and π''). When confronting these predictions with the new image at time $t + \delta t$, and the primitives extracted from it, it becomes apparent that π' is confirmed by the newly available information whereas π'' is contradicted, thereby revealing the erroneousness of the hypothesis Π''. Therefore, Π'' is discarded from the representation and thus the ambiguity is reduced.

We then propose to use these predictions to re-evaluate 3D-primitives' confidence depending on their resilience over time. This is justified by the continuity assumption, which states that 1) scene's objects or contours should not appear and disappear abruptly from the field of view (FoV) but move in and out gracefully according to the estimated ego-motion; and 2) a contour's position and orientation at any point in time is fully determined by the knowledge of its position at a previous instant in time and of its motion since.

Consider a primitive Π_i, predicting a primitive $\hat{\Pi}_i^t$ at time t. We write the fact

that this prediction is confirmed by the images at time time t as $\mu_t(\hat{\Pi}_i) = 1$; and the fact that it is not confirmed (i.e., there is no 2D-primitive extracted at time t that is similar to the projection of $\hat{\Pi}_i^t$ on the image plane) as $\mu_t(\hat{\Pi}_i) = 0$. By extension, we code the resilience a primitive Π_i, from its apparition at time 0 until time t as the binary vector:

$$\boldsymbol{\mu}(\Pi_i) = \left(\mu_t(\hat{\Pi}_i), \mu_{t-1}(\hat{\Pi}_i), \cdots, \mu_0(\hat{\Pi}_i)\right)^T. \tag{4}$$

We then apply Bayes formula to evaluate the posterior likelihood that a 3D-primitive is correct knowing its resilience vector:

$$p\left(\Pi_i|\boldsymbol{\mu}(\Pi_i)\right) = \frac{p\left(\boldsymbol{\mu}(\Pi_i)|\Pi\right)p\left(\Pi\right)}{p\left(\boldsymbol{\mu}(\Pi_i)|\Pi\right)p\left(\Pi\right) + p\left(\boldsymbol{\mu}(\Pi_i)|\bar{\Pi}\right)p\left(\bar{\Pi}\right)}. \tag{5}$$

In this formula, Π and $\bar{\Pi}$ are correct and erroneous primitives, respectively. The quantities $p\left(\Pi\right)$ and $p\left(\bar{\Pi}\right)$ are the prior likelihoods for a 3D-primitive to be correct and erroneous. The quantity $p\left(\boldsymbol{\mu}(\hat{\Pi}_i)|\Pi\right)$ (resp. $p\left(\boldsymbol{\mu}(\Pi_i)|\bar{\Pi}\right)$) expresses the probability of occurrence of a resilience vector $\boldsymbol{\mu}(\Pi_i)$ for a correct (resp. erroneous) primitive Π_i.

Furthermore, if we assume independence between the matches $\mu_t(\hat{\Pi}_i)$, then for a primitive Π_i that exists since n iterations and has been matched successfully m times, we have the following relation:

$$\begin{aligned} p\left(\boldsymbol{\mu}(\hat{\Pi}_i)|\Pi\right) &= \textstyle\prod_t p\left(\mu_t(\hat{\Pi}_i)|\Pi\right) \\ &= p\left(\mu_t(\hat{\Pi}_i) = 1|\Pi\right)^m p\left(\mu_t(\hat{\Pi}_i) = 0|\Pi\right)^{n-m}. \end{aligned} \tag{6}$$

In this case the probabilities for μ_t are equiprobable for all t, and therefore if we define the quantities $\alpha = p\left(\Pi\right)$, $\beta = p\left(\mu_t(\hat{\Pi}) = 1|\Pi\right)$ and $\gamma = p\left(\mu_t(\hat{\Pi}) = 1|\bar{\Pi}\right)$ then we can rewrite Eq. (5) as follows:

$$p\left(\Pi_i|\bar{\boldsymbol{\mu}}(\hat{\Pi}_i)\right) = \frac{\beta^m(1-\beta)^{n-m}\alpha}{\beta^m(1-\beta)^{n-m}\alpha + \gamma^m(1-\gamma)^{n-m}(1-\alpha)}. \tag{7}$$

We measured these prior and conditional probabilities using a video sequence with known motion and depth ground truth obtained via range scanner. We found values of $\alpha = 0.46$, $\beta = 0.83$ and $\gamma = 0.41$. This means that, in these examples, the prior likelihood for a stereo hypothesis to be correct is 46%, the likelihood for a correct hypothesis to be confirmed is 83% whereas for an erroneous hypothesis it is of 41%. These probabilities show that Bayesian inference can be used to identify correct correspondences from erroneous ones. To stabilize the process, we will only consider the n first frames after the appearance of a new 3D-primitive. After n frames, the confidence is fixed for good. If the confidence is deemed too low at this stage, the primitive is forgotten. During our experiments $n = 5$ proved to be a suitable value.

The end-effector of the robot follows the same motion as the object. Therefore, this end-effector becomes extracted as well. Since we know the geometry of this

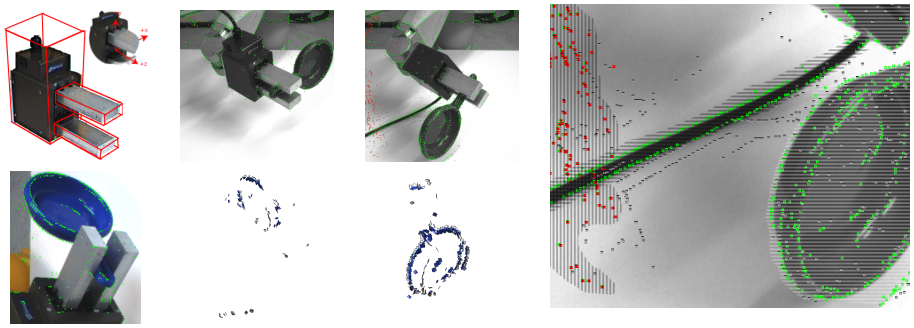


Fig. 6. Birth of an object. (a) top: bounding boxes of grasper body and its fingers used to eliminate grasper features and grasper coordinate system, bottom: image with eliminated grasper features. (b)–(c) two steps in the accumulation process. Top: 2D projection of the accumulated 3D representation and newly introduced primitives, bottom: accumulated 3D representation. (d) newly introduced and accumulated primitives in detail. Note that, the primitives that are not updated are red (dominant in the area with the vertical stripes (left side of the image)), the ones that have low confidence are grey and the high confidence ones are green (dominant in the areas with the horizontal stripes (the cable, the gripper and the object)).

Fig. 7. Objects and their related accumulated representation.

end-effector (Fig. 6(a)top), we can however easily subtract it by eliminating the 3D primitives that are inside the bounding boxes that bounds the body of the gripper and its fingers. For this operation, three bounding boxes are defined in the grasper coordinate system. Fig. 6(a)bottom shows the 2D projection of the remaining primitives after the ones produced by the gripper have been eliminated.

4.2. *Experiments*

We applied the accumulation scheme to a variety of scenes where the robot arm manipulated several objects. The motion was a rotation of 5 degrees per frame. The accumulation process on one such object is illustrated in Fig. 6. The top images of Fig. 6(b,c) show the predictions at two frames. The bottom images show the 3D-primitives that were accumulated. The object representation becomes fuller over time, whereas the primitives reconstructed from other parts of the scene are discarded. Fig. 7 shows the accumulated representation for various objects. The hole in the model corresponds to the part of the object occluded by the gripper. Accumulating the representation over several distinct grasps of the objects would yield a complete representation.

5. Conclusion

We introduced a scheme in which two modules in terms of Object Action Complexes (OACs) become combined to extract world knowledge in terms of the objectness of a set of local features as well as the object shape. Although this exploration scheme is completely autonomous, we argued that there is a significant amount of prior knowledge in terms of generic properties of the world built into the system. Starting with a rather sophisticated feature extraction process covering common visual modalities, functional relations defined on those features such as co-planarity, co-linearity, basic laws about Euclidean geometry and the motion of rigid object has been exploited. Furthermore and at least of equal importance was the capability to act on the world that made this process possible. Here the embodiment of the agent is of high importance. The option to grasp and move the objects in a controlled way is rather unique to few species and with high likelihood linked to develop higher cognitive capabilities.

The work described in this paper is part of the EU project PACO-PLUS[21] which aims at a system covering different levels of cognitive processing from low-level processes as described here up to a planning AI level (see Geib et al.[19]). This work introduced describes an important module of such a cognitive system which gives information that higher levels require to start operating. First, it segments the world in objects which are the basic entities that higher level reasoning is based on. Moreover, it generates 3D object representations in a procedural way which then can be used for object identification and pose estimation (see, e.g., Lowe[2] for the use of 3D models for object recognition and Detry and Piater[22] for first steps in directly making use of the extracted representations described in this paper). By the described exploratory procedure, a natural mechanism is given that enlarges the internal world model that then can be used by higher levels for reasoning and planning.

Acknowledgements

We would like to thank Tamim Asfour, Mark Steedman, Christopher Geib and Ron Petrick for fruitful discussions. This work was conducted within the EU Cognitive Systems project PACO-PLUS[21] funded by the European Commission.

References

1. J. Gibson, *The Ecological Approach to Visual Perception* (Boston, MA: Houghton Mifflin, 1979).
2. D. Lowe, Fitting parametrized 3D–models to images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(5), 441–450 (1991).
3. N. Krüger, N. Pugeault and F. Wörgötter, Multi-modal primitives: Local, condensed, and semantically rich visual descriptors and the formalization of contextual information, *IEEE Transactions on Pattern Analysis and Machine Intelligence, (also available as Technical report (2007-4) of the Robotics Group, Maersk Institute, University of Southern Denmark)* (under review).
4. N. Krüger, M. Lappe and F. Wörgötter, Biologically motivated multi-modal processing of visual primitives, *Interdisciplinary Journal of Artificial Intelligence & the Simulation of Behaviour, AISB Journal* **1**(5), 417–427 (2004).
5. Y. Aloimonos, I. Weiss and A. Bandopadhay, Active vision, *International Journal of Computer Vision* **2**, 333–356 (1987).
6. R. Rao and D. Ballard, An active vision architecture based on iconic representations, *Artificial Intelligence Journal* **78**, 461–505 (1995).
7. P. Fitzpatrick and G. Metta, Grounding vision through experimental manipulation, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **361**, 2165 – 2185 (2003).
8. L. Natale, F. Orabona, G. Metta and G. Sandini, Exploring the world through grasping: A developmental approach, in *IEEE International Symposium on Computational Intelligence in Robotics and Automation* 2005, pp. 559–565.
9. J. Modayil and B. Kuipers, Bootstrap learning for object discovery, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* **1**, 742–747 (2004).
10. P. J. Kellman and M. E. Arterberry, *The Cradle of Knowledge* (MIT-Press, 1998).
11. N. Krüger and F. Wörgötter, Statistical and deterministic regularities: Utilisation of motion and grouping in biological and artificial visual systems, *Advances in Imaging and Electron Physics* **131**, 82–147 (2004).
12. D. Aarno, J. Sommerfeld, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft and N. Krüger, Early reactive grasping with second order 3D feature relations, in *Recent Progress in Robotics; ViableRobotic Service to Human, selected papers from ICAR'07*, eds. S. Lee, I. H. Suh and M. S. Kim (Springer-Verlag Lecture Notes in Control and Information Sciences (LNCIS), 2007).
13. N. Pugeault, E. Baseski, D. Kraft, F. Wörgötter and N. Krüger, Extraction of multi–modal object representations in a robot vision system, in *International Conference on Computer Vision Theory and Applications (VISAPP)* 2007.
14. J. H. Elder, Are edges incomplete?, *International Journal of Computer Vision* **34**, 97–122 (1999).
15. N. Pugeault, Early cognitive vision: Feedback mechanisms for the disambiguation of early visual representation, PhD thesis, University of Göttingen, 2008.
16. S. Kalkan, N. Pugeault and N. Krüger, Perceptual operations and relations between 2D or 3D visual entities, Tech. Rep. 2007–3, Robotics Group, Maersk Institute, University

of Southern Denmark (2007).
17. O. Faugeras, *Three–Dimensional Computer Vision* (MIT Press, 1993).
18. A. Miller and P. Allen, GraspIt!: A versatile simulator for robotic grasping, *IEEE Robotics and Automation Magazine* **11**(4), 110–122 (2004).
19. C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krüger and F. Wörgötter, Object action complexes as an interface for planning and robot control, in *Workshop 'Toward Cognitive Humanoid Robots' at IEEE-RAS International Conference on Humanoid Robots (Humanoids 2006)* 2006.
20. N. Krüger, M. Ackermann and G. Sommer, Accumulation of object representations utilizing interaction of robot action and perception, *Knowlede Based Systems* **15**, 111–118 (2002).
21. PACO-PLUS: Perception, Action and Cognition through learning of Object-Action Complexes. EU Cognitive Systems project (IST-FP6-IP-027657), http://www.paco-plus.org/, (2006–2010).
22. R. Detry and J. Piater, Hierarchical integration of local 3D features for probabilistic pose recovery, in *Robot Manipulation: Sensing and Adapting to the Real World, (Workshop at Robotics, Science and Systems)* 2007.

A Scenario for Integrating Low-Level Robot/Vision, Mid-Level Memory, and High-Level Planning with Sensing

Ronald Petrick, Christopher Geib, Mark Steedman

30 November 2007

In this document we provide an overview of a proposed scenario for integrating SDU's robot/vision system, Leiden's mid-level memory/reasoning component, and Edinburgh's high-level planner and plan execution monitor.[1] This document is primarily based on discussions between SDU, Leiden, Liège, and Edinburgh at the September 2007 meeting in Leiden, and captures some of our thinking from the point of view of the planning task and required high-level representation. Since our integration discussions are ongoing this document should be viewed as a snapshot of our current thinking and subject to change.

1 Object stacking with sensing

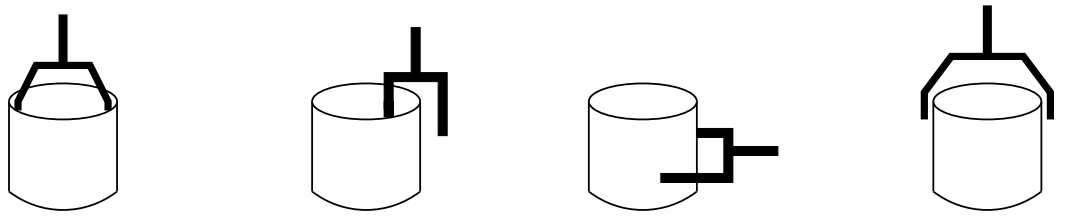
The domain we have chosen for our integration task is a simple object manipulation scenario that builds on the integration scenario involving SDU and Edinburgh (see the Edinburgh document *A Scenario for Integrating Low-Level Robot/Vision and High-Level Planning* for details). We assume a *table* with a number of *objects* that are graspable by the robot. We consider situations with no more than 5 objects and, initially, only 1-2 objects. For simplicity we assume that objects are generally cylindrical in shape but not necessarily identical. In particular, each object can have a different *radius* which determines its size. Objects may or may not be *open* containers, which determines whether or not we can *stack* objects inside other objects, provided the object sizes permit such stacking.

The goal of the scenario is to clear all open objects from the table, by removing them to some designated location (e.g., a box, a shelf, a hole, a corner of the table, etc.). The location may furthermore be restricted in such a way as to force object stacking in order to successfully complete the task. For instance, there might only be room for 2 objects to sit side by side on a shelf, meaning all other objects would have to be appropriately stacked. The high-level planning system will typically have only incomplete information concerning the openness of objects and must therefore plan explicit *sensing* actions to determine whether a particular object is open or not. Object openness plays two important roles in this scenario: (i) as a goal condition that determines which objects should be removed from the table, and (ii) as a prerequisite for stacking operations.

This scenario is meant to provide a basis for integrating the robot/vision, mid-level memory, and high-level planning components of the system. The planner is responsible for constructing a plan that achieves the goal of clearing open objects from the table, by working with a high-level representation of the scenario. The job of the mid-level component in this case is to *refine* such plans with regard to the sensing actions contained in these plans. In particular, the robot/vision system will be able to ascertain whether an object is open or not by one of two means: (i) it can *poke* an object in order to verify its concavity, or (ii) it can *focus* the vision system on the object at a higher level of resolution. The mid-level memory system must make an informed choice between poking and focusing operations, update the plan as appropriate, and pass the augmented plan on to the robot/vision system. Ultimately, the robot/vision system must be able to interpret, understand, and execute the plans generated and refined by the upper levels.

For the remainder of this document we will mainly focus on the top-down task, in particular, describing the high-level planning representation that is passed to the mid-level memory system, and the message passing protocol that supports the exchange of messages between the levels.

[1] For the purposes of distinguishing between the three levels in this document we will use the tags "robot", "memory", and "planner" to denote the SDU, Leiden, and Edinburgh components, respectively.



(a) Grasp Type A (b) Grasp Type B (c) Grasp Type C (d) Grasp Type D

Figure 1: Robot grasp types available to the planner

2 High-level planning representation

Given the above scenario description, we define a set of high-level actions and properties that allows the planner to operate in this domain, and provide some insights as to how these actions/properties relate to the memory and robot/vision levels.

2.1 Physical actions

In discussions with SDU we have agreed to model four types of grasping actions at the planning level, as illustrated in Figure 1. These actions correspond to a subset of the possible grasping options the robot is capable of performing. In general, these actions exhibit the following behaviour:

Grasp Type A - This action can only be used to grasp objects at the top of a stack, or an empty object on the table. Objects must also satisfy a minimum and maximum radius restriction.

Grasp Type B - This action can only be used to grasp objects on the table that are not part of a stack. Objects must also satisfy a minimum radius restriction.

Grasp Type C - This action can only be used to grasp objects that aren't contained in other objects, i.e., the "outermost" object which must be on the table. Objects must also satisfy a maximum radius restriction.

Grasp Type D - This action can only be used to grasp objects that aren't contained in other objects, i.e., objects that are on the table. Objects must also satisfy a maximum radius restriction. For simplicity, we will assume that objects stacked within the object being grasped will not affect the grasp.

For the planner's domain encoding it is necessary to subdivide Grasp Type A into two separate actions, to avoid reasoning about conditional effects. The planner therefore has five grasp actions available to it, corresponding to the four types of grasps available to the robot. (For the purposes of the sample plans in this document we only require Grasp Types A and D.) Each grasping action takes a single argument, `?x`, denoting the label of an object. We have agreed that each object in the world will be designated by a string of the form `objN`, where `N` is a non-negative integer, e.g., `obj42`.

`graspA-fromTable(?x)` - Grasp object `?x` from the table using Grasp Type A.

`graspA-fromTopOfStack(?x)` - Grasp object `?x` from the top of a stack using Grasp Type A.

`graspB-fromTable(?x)` - Grasp object `?x` from the table using Grasp Type B.

`graspC-fromTable(?x)` - Grasp object `?x` from the table using Grasp Type C.

graspD-fromTable(?x) - Grasp object ?x from the table using Grasp Type D.

We have also encoded four actions for moving and manipulating objects when successfully grasped:

putInto-objectOnTable(?x,?y) - Put object ?x into object ?y, which is on the table.

putInto-stack(?x,?y) - Put object ?x into object ?y, which is at the top of a stack on the table.

putOnTable(?x) - Put object ?x onto the table.

putAway(?x) - Put object ?x away.

Each manipulation action is *object centric* and modelled with a high degree of abstraction. For instance, we do not provide plan-level actions that specify 3D spatial coordinates, joint angles, or similar real-valued parameters. The `putAway` action is particularly generic and should be considered a placeholder for a more complex (possibly, predefined) operation that clears an object from the table to its final destination location. For the purpose of this document we will assume that objects are being put away onto a shelf. We also note that both `putInto-objectOnTable` and `putInto-stack` actions denote stacking operations which will have as a prerequisite the property that objects can only be stacked into open objects.

2.2 Sensing actions

The high-level planning representation also consists of a single sensing action:

sense-open(?x) - Determine whether object ?x is open or not.

At the planning level, this action is modelled an information gathering or *knowledge-producing action* that provides the planner with additional information about an object's state. At the robot/vision level, `sense-open` will ultimately be executed as either a *poke* operation which tests the object's concavity, or a *focus* operation which directs the vision system to study the object at a higher resolution. The mid-level memory system is responsible for refining high-level `sense-open` actions into robot/vision operations that are appropriate, given the context, and that can be understood by the lower level.

2.3 Properties

Our planning-level domain encoding makes use of a set of predicates and functions which we have agreed could reasonably be provided to the planner as a result of sensor information from the robot/vision level. These properties are subject to change as the domain model is refined through further discussions.

open(?x) - A predicate indicating that object ?x is open.

gripperempty - A predicate describing whether the robot's gripper is empty or not.

ingripper(?x) - A predicate indicating that the robot is holding object ?x in its gripper.

ontable(?x) - A predicate indicating that object ?x is on the table.

onshelf(?x) - A predicate indicating that object ?x is on the shelf.

isin(?x,?y) - A predicate indicating that object ?x is stacked in object ?y.

clear(?x) - A predicate indicating that no object is stacked in ?x.

instack(?x,?y) - A predicate indicating that object ?x is in a stack with object ?y at its base.

radius(?x) = ?y - A function indicating that the radius of object ?x is ?y.

shelfspace = ?x - A function indicating that there are ?x empty shelf spaces.

reachableA(?x)
reachableB(?x)
reachableC(?x)
reachableD(?x) - Predicates indicating that object ?x is reachable by the gripper using a particular grasp.

graspAMinRadius = ?x
graspAMaxRadius = ?x
graspBMinRadius = ?x
graspCMaxRadius = ?x
graspDMaxRadius = ?x - Functions indicating the min./max. radius restrictions for each grasp type.

2.4 Domain encoding

Using the above properties we can write PKS operators for the actions we require. For simplicity, we have made the following restrictions in our domain encodings: (i) all objects are initially assumed to be on the table, (ii) grasp type C will initially be omitted (grasp type B is not required for our initial examples), and (iii) the `putOnTable` action will initially be omitted (since there are no initial stacks).

Our current domain encoding is given in Table 1. These actions are formalized for use with the PKS planner, however, we have simplified the syntax here. Although most of the details of the actual action encodings can be ignored, we mention two important points. First, each action operator is parametrized with a set of arguments that can denote any object in the world. Thus, all of our actions are object centric. Second, our encoding takes advantage of PKS's ability to work with functions and simple numerical expressions, which we include as part of the action preconditions and effects. For instance, the radius of an object plays a role in determining whether or not it can be stacked inside another object, and the minimum/maximum grasp values help determine whether or not a particular grasp action can be applied. Our domain encoding can be extended as needed to accommodate new actions or properties that may arise from future discussions.

PKS action description notation: the domain encoding in Table 1 is very much like a standard STRIPS encoding except that PKS, unlike STRIPS, uses multiple databases as the basis for its representation. Thus, references to K_f and K_w in the "effects" section of an action denote two of PKS's databases. (K_f is very much like a standard STRIPS databases that stores the planner's knowledge of facts, and K_w is a specialized database for storing the effects of sensing actions.) As well, $\neg K_w$`open(?x)` in the description of `sense-open` is a knowledge precondition that ensures the planner does not include a sensing action in a plan if it already knows the outcome of the sensing (i.e., if the planner already knows whether an object is open or not then it shouldn't sense the object).

Action	Preconditions	Effects
sense-open(?x)	$\neg K_w$(open(?x)) ontable(?x)	add(K_w,open(?x))
graspA-fromTable(?x)	reachableA(?x) clear(?x) gripperempty ontable(?x) radius(?x) ≥ graspAMinRadius graspAMaxRadius ≥ radius(?x)	add(K_f,ingripper(?x)) add(K_f,¬gripperempty) add(K_f,¬ontable(?x))

graspA-fromTopOfStack(?x)	reachableA(?x)
	clear(?x)
	gripperempty
	radius(?x) ≥ graspAMinRadius
	graspAMaxRadius ≥ radius(?x)
	(∃ ?z).
	  instack(?x,?z)
	  ontable(?z)
		add(K_f,ingripper(?x))
		add(K_f,¬gripperempty)
		(∀ ?y). isin(?x,?y) ⇒
		  del(K_f,isin(?x,?y))
		  add(K_f,clear(?y))
		(∀ ?z). instack(?x,?y) ⇒
		  del(K_f,instack(?x,?z))

graspB-fromTable(?x)	reachableB(?x) clear(?x) gripperempty ontable(?x) radius(?x) ≥ graspBMinRadius	add(K_f,ingripper(?x)) add(K_f,¬gripperempty) add(K_f,¬ontable(?x))
graspD-fromTable(?x)	reachableD(?x) gripperempty ontable(?x) graspDMaxRadius ≥ radius(?x)	add(K_f,ingripper(?x)) add(K_f,¬gripperempty) add(K_f,¬ontable(?x))

putInto-objectOnTable(?x,?y)	?x ≠ ?y
	ingripper(?x)
	open(?y)
	clear(?y)
	ontable(?y)
	radius(?y) > radius(?x)
		add(K_f,gripperempty)
		add(K_f,isin(?x,?y))
		add(K_f,instack(?x,?y))
		del(K_f,clear(?y))
		del(K_f,ingripper(?x))
		(∀ ?w). instack(?w,?x) ⇒
		  del(K_f,instack(?w,?x))
		  add(K_f,instack(?w,?y))

putInto-stack(?x,?y)	?x ≠ ?y
	ingripper(?x)
	open(?y)
	clear(?y)
	radius(?y) > radius(?x)
	(∃ ?z).
	  instack(?y,?z)
	  ontable(?z)
		add(K_f,gripperempty)
		add(K_f,isin(?x,?y))
		del(K_f,clear(?y))
		del(K_f,ingripper(?x))
		(∀ ?z). instack(?y,?z) ⇒
		  add(K_f, instack(?x,?z))
		  (∀ ?w). instack(?w,?x) ⇒
		  del(K_f,instack(?w,?x))
		  add(K_f,instack(?w,?z))

putAway(?x)	ingripper(?x) shelfspace > 0	add(K_f,onshelf(?x)) add(K_f,gripperempty) del(K_f,ingripper(?x)) shelfspace = shelfspace - 1

Table 1: PKS-style action descriptions

3 Example plans

In this section we give three examples of planning problems we can solve using PKS and the above action descriptions. In each example we consider a scenario with 2 objects initially on the table. Each object also has a size as indicated by its radius. We also assume certain minimum/maximum values for the grasps but these values don't play a large role in these examples. (For simplicity we use integer values in our examples however we also permit real-valued quantities.) In each example we assume the following initial conditions:

- Objects: `obj1`, `obj2`
- `radius(obj1) = 1`, `radius(obj2) = 4`
- `shelfspace = 1`
- All objects are on the table (no initial stacks)

The goal in each example is to clear the open objects from the table by placing the objects on a shelf which has limited space. In Example 1, the planner initially knows that both objects are open and, thus, does not need to include sensing actions in the plan. In Examples 2 and 3, sensing actions are required: in the second example, the planner knows that one object is not open but does not know whether the second object is open or not; in the third example, the planner does not know whether either object is open or not.

When PKS constructs a plan that includes sensing actions, it can build into the plan a set of *conditional branches* for reasoning about the possible outcomes of a sensing operation. In particular, one branch is constructed for each possible value the sensed property might have. The resulting plans in this case are structured as trees rather than simple linear sequence of actions. In our examples, branch points are denoted by expressions like "`branch(open(objX))`," meaning "branch on the truth value of `open(objX)`." In this scenario, we will only consider branches on binary properties, i.e., properties that can be either true or false. A branch point is followed by two plan sections, labelled as "`K+`" and "`K-`," denoting two disjoint plan branches. The `K+` branch indicates the "knowledge positive" branch where `open(objX)` is assumed to be true. The `K-` branch indicates the "knowledge negative" branch where `open(objX)` is assumed to be false (i.e., ¬`open(objX)` is assumed to be true). Each branch can contain a sequence of actions and possibly other branch points. A `nil` tag along a branch indicates that no further operation takes place along that branch. At execution time, the information returned from a sensing action will let the plan execution monitor decide which branch of the plan it should follow at a branch point. The planner ensures that when conditional plans are constructed, the goals are achieved along every branch of the plan.

3.1 Example 1

The planner initially knows that `open(obj1)` and `open(obj2)` are true.

Plan:

```
graspA-fromTable(obj1)
putInto-objectOnTable(obj1,obj2)
graspD-fromTable(obj2)
putAway(obj2)
```

Since `obj1` and `obj2` are both initially known to be open the planner does not need to include any sensing actions in the plan. The two objects can simply be stacked and removed from the table.

3.2 Example 2

The planner initially knows that ¬open(obj1) is true but does not know the state of open(obj2).

Plan:

```
sense-open(obj2)
branch(open(obj2))
K+:
    graspA-fromTable(obj2)
    putAway(obj2)
K-:
    nil
```

Since the planner does not initially know whether obj2 is open or not it includes a sense-open action in the plan. The plan then branches on the two possible outcomes of open(obj2). If open(obj2) is true (the K+ branch) then obj2 is grasped and removed from the table; if open(obj2) is false (the K- branch) then no further action is taken. Since the planner initially knows that obj1 is not open, this object does not need to be removed from the table.

3.3 Example 3

The planner does not initially know the state of open(obj1) and open(obj2).

Plan:

```
sense-open(obj1)
sense-open(obj2)
branch(open(obj2))
K+:
    branch(open(obj1))
    K+:
        graspA-fromTable(obj1)
        putInto-objectOnTable(obj1,obj2)
        graspD-fromTable(obj2)
        putAway(obj2)
    K-:
        graspA-fromTable(obj2)
        putAway(obj2)
K-:
    branch(open(obj1))
    K+:
        graspA-fromTable(obj1)
        putAway(obj1)
    K-:
        nil
```

Since the planner does not initially know whether obj1 or obj2 is open, it includes two sense-open actions in the plan. It then considers each possible outcome of these actions by constructing a plan with four branches (an initial branch point, followed by a second branch point along each of the top-level branches):

(i) Along the K+/K+ branch where open(obj2) and open(obj1) are true, both objects are grasped and put away as in Example 1.

(ii) Along the K+/K- branch where open(obj2) and ¬open(obj1) are true, object obj2 is grasped and put away.

(iii) Along the K-/K+ branch where ¬open(obj2) and open(obj1) are true, object obj1 is grasped and put away.

(iv) Along the K-/K- branch where ¬open(obj2) and ¬open(obj1) are true, no further action is taken.

4 Message passing protocol

In this section we describe a simple message passing protocol for exchanging information between the robot/vision, memory, and planning levels. We begin by defining the kinds of messages that can be passed between the system levels. We then describe a simple control architecture that is sufficient for our proposed integration task, and provide some details of a communication library (supplied by Edinburgh) that implements this protocol.

4.1 Message definitions

We define a set of 10 messages that capture the interactions between the three levels of the system. Each message is defined by its *type* and its *content*. A message's type is simply its name or label. Depending on the message type, a message may also contain specific content or data to be sent. The message passing protocol we have defined is currently based on a *point-to-point* model, where each message is sent by a particular system component to another component. Moreover, the message set is designed in such a way that messages are (generally) defined in send/receive pairs so that only certain messages can be initiated by a "sending" level, with an appropriate response being sent by the "receiving" level. The complete set of messages is given in Table 2 and the send/receive message pairs are given in Table 3.[2]

4.2 Message passing control loop

The message passing protocol is initially driven by the robot/vision level of the system. Because of the paired send/receive nature of our message set, the upper system levels are forced to coordinate their operations in order to respond appropriately to lower-level messages. Currently, communication only takes place between two "adjacent" levels of the system, i.e., the robot and memory, or the memory and planner (see Figure 4). This means that all communication between the robot and planner must flow through the memory level, which typically acts as a forwarding service, but may also observe or refine the flow of messages (see below). Because the message passing protocol is mainly driven by the robot level, the memory and planning levels operate as message servers that respond to message queries. This protocol also permits certain message exchanges between the planner and memory levels, however, that can interrupt the standard robot-driven process. It is also worth noting that nothing in the implementation of the communication architecture prevents us from expanding this protocol in the future to permit direct point-to-point communication between any two components of the system.

4.2.1 Robot-level control loop

At the robot level, the message-processing control loop follows a relatively simple repeating pattern where the robot essentially drives the message-passing process and the upper levels of the system respond to

[2]The message set is still subject to change and may be expanded or streamlined in the future.

MSG_STATE_UPDATE – Provide updated state information

Sender/Destination: Robot to Memory, or Memory to Planner

Content: World state specification

ACK_STATE_UPDATE – Acknowledge state update message

Sender/Destination: Planner to Memory, or Memory to Robot

Content: NONE

MSG_ACTION_REQUEST – Request a new action

Sender/Destination: Robot to Memory, or Memory to Planner

Content: NONE

ACK_ACTION_REQUEST – Acknowledge new action request for execution

Sender/Destination: Planner to Memory, or Memory to Robot

Content: NONE

MSG_ACTION_SUBMIT – Submit a new action for execution

Sender/Destination: Planner to Memory, or Memory to Robot

Content: Action specification

ACK_ACTION_SUBMIT – Acknowledge receipt of new action and start of action execution

Sender/Destination: Robot to Memory, or Memory to Planner

Content: NONE

MSG_ACTION_STOPPED – Provide alert that execution of last submitted action has stopped

Sender/Destination: Robot to Memory, or Memory to Planner

Content: Action execution return value (1 = success or 0 = failure).

ACK_ACTION_STOPPED – Acknowledge termination of last submitted action

Sender/Destination: Planner to Memory, or Memory to Robot

Content: NONE

MSG_PLAN_REQUEST – Request entire plan from planner

Sender/Destination: Memory to Planner

Content: NONE

MSG_PLAN_SUBMIT – Submit a complete plan

Sender/Destination: Planner to Memory

Content: Plan specification

Table 2: Available message types in the message passing protocol

Message type sent	Expected response
MSG_STATE_UPDATE	ACK_STATE_UPDATE
MSG_ACTION_REQUEST	ACK_ACTION_REQUEST
MSG_ACTION_SUBMIT	ACK_ACTION_SUBMIT
MSG_ACTION_STOPPED	ACK_ACTION_STOPPED
MSG_PLAN_REQUEST	MSG_PLAN_SUBMIT

Table 3: Send/receive message pairs

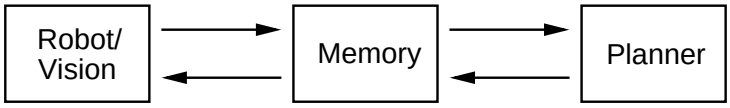


Table 4: Flow of messages between the three system levels

queries. The robot-level control loop defines a very simple synchronous cycle wherein a message is sent and its acknowledgement is received before the next message can be sent. As a result, the robot only executes one action at a time and provides updates on the state of the world before the next action begins.

At an abstract level, we see the interaction between the robot and the higher levels follow the *RobotLevel-ControlLoop* pseudo code given in Figure 2(a). After an initial report on the world state, the main communication cycle consists of an action request by the robot, which is fulfilled by the upper levels (ultimately the planner), an indication from the robot when the action has finished executing, followed by an update on the new state of the world. Messages to and from the robot level all pass through the memory level. Thus, a request made by the robot for a planning-level service (e.g., requesting a new action) will ultimately reach the planner after being forwarded through the memory.

4.2.2 Memory-level control loop

Unlike the more tightly-regulated control loop of the robot level, communication at the memory level is more loosely structured using a client-server architecture. In particular, the memory is able to respond to requests from both the robot and the planner, as well as initiate certain messages of its own.

In most cases, the memory will initially act as a forwarding service that delivers messages from the robot to the planner, and messages from the planner to the robot. One notable exception is the receipt of `MSG_ACTION_SUBMIT` messages from the planner. Before forwarding such messages, the memory must inspect the message contents to check for sensing actions, which may need to be refined. In the context of the simple integration scenario described in this document, the memory must transform all `sense-open` actions into *poke* or *focus* operations before passing them on to the robot. (In the future, the memory may also be responsible for refining grasp operations specified by the planner. This protocol also supports a future bottom-up role for the memory, where the middle level "abstracts" subsymbolic robot-level information into a symbolic form understandable by the planner.)

The memory is also able to directly request information about the structure of a plan from the planner. The planner will respond with a complete description of the current plan, which may be a conditional plan with branches. The memory can then use this information as needed, for instance to help in its decision making concerning refinement activities.

The pseudo code for the memory-level control algorithm is given in Figure 2(b).

4.2.3 Planning-level control loop

The planning level control loop also operates in a client-server fashion, responding to messages sent from the memory level (but typically originating from the robot level). The planning level is responsible for constructing high-level plans and feeding the actions, one at a time, to the robot level through the memory level. The planner also receives world state updates from the robot (again, through the memory) as well as status reports as to the success or failure of performed actions.

The memory level is also able to interact with the planner to request a complete description of the current plan. This part of the protocol provides the memory level with greater information about a plan's structure, which could be analysed in order to help direct future operations of the memory level, or refine future actions sent to the robot. Future versions of the communication protocol may also allow the planner to directly "push" such plan information to the memory, for instance as a result of replanning operations. The general planning-level control algorithm is given in Figure 2(c).

The message passing architecture we have outlined has a number of advantages. First, the protocol clearly separates the operations of the three system levels and the interactions between the levels, with the memory level acting as a form of mediator or interpreter. For instance, this protocol allows for the possibility of different content formats for messages flowing between the lower and upper levels of the system (e.g., messages with subsymbolic information between the robot and memory, and messages with symbolic information between the memory and planner). Also, future changes to the communication protocol involving one pair of levels need not force changes to the interaction of another pair of levels. Finally, we have designed our message set to support much more complex and flexible control architectures, which might arise in the future. For our initial integration tasks, however, the existing process we have outlined is more than sufficient.

Direct link between SDU and Edinburgh: In terms of the initial integration efforts between SDU and Edinburgh, the above protocol does not specify any major changes to existing work. Instead, the memory level can be viewed as a message-forwarding module that holds the place of the full mid-level memory component, in order to bring the existing SDU/Edinburgh architecture in line with the protocol described here. Although this module will simply pass messages to the other system levels, its addition should facilitate the inclusion of a more fully-featured module into current integration efforts at a later date when a memory system is made available. The necessary code for the message-forwarding module is provided as part of Edinburgh's supplied communication library.

4.3 Socket communication library and sample code

For ease of implementation we have defined a set of C++ classes for manipulating message types and message contents. These classes work in conjunction with a lightweight socket library (also written in C++) that we have developed for Linux, to facilitate communication between system components.

At the code level, message types are chosen from a list of predefined `enum` types, and message contents are simple C++ strings.[3] Currently, the content of the `MSG_STATE_UPDATE` message must be a list of instantiated properties from Section 2.3 that form the world state. Similarly, the action specification content of the `MSG_ACTION_SUBMIT` message is a single instantiated action from Sections 2.1 or 2.2. The content of the `MSG_PLAN_SUBMIT` message will be a plan similar to those in Section 3, but encoded as a Prolog-style list

[3]The current version of the communication library also defines messages for introducing new objects, new properties, and new actions into the planning-level domain description. We are still in the process of extending the message passing protocol to include these new message types and, thus, we have not included a discussion of these messages here. Such additions will appear in a future version of this document.

```
Proc RobotLevelControlLoop
    Send: MSG_STATE_UPDATE; Receive: ACK_STATE_UPDATE;
    while !termination loop
        Send: MSG_ACTION_REQUEST; Receive: ACK_ACTION_REQUEST;
        Receive: MSG_ACTION_SUBMIT; Send: ACK_ACTION_SUBMIT;
        Send: MSG_ACTION_STOPPED; Receive: ACK_ACTION_STOPPED;
        Send: MSG_STATE_UPDATE; Receive: ACK_STATE_UPDATE;
    endLoop
endProc
```

(a)

```
Proc MemoryLevelControlLoop
    while !termination loop
        choose
            Send: MSG_PLAN_REQUEST;
        or
            Wait for message receive;
            case MSG_ACTION_SUBMIT:
                if action is sense-open then
                    Replace sense-open with poke or focus operation;
                endIf
                Forward message;
            case MSG_PLAN_SUBMIT:
                Update memory with received plan;
            case all other message types:
                Forward message;
        endChoose
    endLoop
endProc
```

(b)

```
Proc PlannerLevelControlLoop
    while !termination loop
        Wait for message receive;
        case MSG_STATE_UPDATE:
            Update world model;
            Send: ACK_STATE_UPDATE;
        case MSG_ACTION_UPDATE:
            Send: ACK_ACTION_REQUEST
            Construct plan/get next action in plan;
            Send: MSG_ACTION_SUBMIT; Receive: ACK_ACTION_SUBMIT;
        case MSG_ACTION_STOPPED:
            Process action success/failure;
            Send: MSG_ACTION_SUBMIT;
        case MSG_PLAN_REQUEST:
            Construct plan/get entire plan;
            Send: MSG_PLAN_SUBMIT;
    endLoop
endProc
```

(c)

Figure 2: Message passing control algorithms

(see Section 5 for an example). A plan iterator class is provided for inspecting the structure of conditional plans in this format. (For more details, refer to the sample code provided with the socket library.)

For initial testing purposes we terminate a plan by having the planner send a `MSG_ACTION_SUBMIT` message to the memory level in response to an action request, with the string `"EOP"` as its content. The memory level will then pass this message on to the robot. Both the memory and robot levels must then send a final `ACK_ACTION_SUBMIT` message to the level above, at which point all system levels are free to terminate communication. In the future, plan termination will force the suspension of the main control loop (i.e., the planner will not send an action) until a new goal is given to the planner and a new plan is constructed.

The communication library is distributed with a set of sample programs that implement the basic message passing protocol described in this document for the robot, memory, and planner components. These programs focus solely on the communication interface, with little additional functionality. (For instance, the memory level program simply forwards messages without "refining" sensing actions and always requests a complete plan after the first robot level request for an action.) It is hoped that these programs can serve as the basis for the development of more sophisticated modules that can simply be "plugged" into the communication architecture. A series of pregenerated plans are included with this software, however, to test the message exchanges between the three levels.

5 Message passing example

To better understand the flow of messages between the three system levels, we consider the scenario in Example 2, where the robot is tasked with the goal of clearing the open objects from a table. Figure 3 shows the messages sent by the three levels during the execution of the first action, `sense-open(obj2)`, in the conditional plan constructed for Example 2 (i.e., one complete cycle of the robot-level control loop).

We note that the first message sent by the robot, `MSG_STATE_UPDATE`, provides the planner with its initial description of the world. We assume that on initialization the robot/vision system will send such an "unusually complete" world description, as a bootstrapping action. From the perspective of the planning system such a message is no more than a particularly large state update, and requires no extra machinery.

Given an initial state description, the planner can construct a plan to achieve a given high-level goal. The planner sends the actions in this plan to the robot/vision system one step at a time, through the memory, in response to action requests. After the execution of each action the robot/vision system reports an update of the world state back to the planner, again, through the memory. In Figure 3 these updates are described in terms of state changes, however, we have agreed that state updates will initially include a complete (or as near as possible to complete) description of the new world state.

For many of the messages sent in the system, the memory level acts as a forwarding service between the robot and the planner. (In the future the memory may take on a more active role as a mediator or translator between the robot and planner.) One notable exception is the occurrence of the `MSG_ACTION_SUBMIT` message. Since the action specified in this message is a sensing action, `sense-open(obj2)`, the memory refines this action by choosing between a *poke* and a *focus* operation. In this case, `focus(obj2)` is chosen as the refined action and the modified message is forwarded to the robot.

Figure 3 also illustrates the results of a `MSG_PLAN_REQUEST` message from the memory to the planner. In this case, the planner responds with a plan of the form:

```
[sense-open(obj2),branch(open(obj2),[graspA-fromTable(obj2),putAway(obj2)],[])].
```

This plan corresponds to the complete conditional plan given in Example 2, encoded in a Prolog-style list format for transmission using the communication library.[4]

[4]The communication library provides a helper class for processing plans in the compact list format.

We note that according to the message passing protocol, `MSG_PLAN_REQUEST` messages could be sent by the memory at other times during its control loop, or not at all, producing slightly different message orderings than those shown in Figure 3. (In the sample code the memory sends a `MSG_PLAN_REQUEST` after receipt of the first `MSG_ACTION_SUBMIT` message from the planner.) Similarly, alternate message orderings—including messages sent in parallel from different levels—could also arise since the robot, memory, and planner all run as independent processes (e.g., message 13 could be sent at the same time as message 11, or even before it). The implementation of our message passing protocol ensures that such ordering differences do not lead to problems like deadlock, however.

6 Plan execution and monitoring

Although we are able to construct plans for the proposed object manipulation scenario, and communicate those plans to the other system levels using the message passing protocol, we must still be concerned with how plan failure information should be exchanged between the planner and the other system levels.

In discussions with SDU we have identified the need for a high-level mechanism that would operate closely with the planner in order to monitor plan execution and control replanning/resensing activities. A *plan execution monitor*, currently being built by Edinburgh, will be responsible for assessing both action failure and unexpected state information that result from feedback provided to the planner from the execution of planned actions at the robot level. The difference between predicted and actual state information will be used to decide between (i) continuing the execution of an existing plan, (ii) resensing activities that target a portion of a scene at a higher resolution to produce a more detailed state report, and (iii) replanning from new/unexpected states.

In support of the resensing activities described in (ii), we have agreed in discussion with SDU that the plan monitor could initially provide the vision system (possibly through the memory level) with a list of the objects that were "relevant" to the execution of the action that is reported to have failed, as based on the high-level action description. Using this information, the vision system could then target particular parts of the scene with greater resolution in order to reevaluate the sensors that provide information about these objects. This operation may lead to new information about the world state and, possibly (as future work), an updated high-level action model.

In terms of the integration scenario described in this document, the plan execution monitor will have the added task of managing the execution of plans with conditional branches. Plan branches result from the inclusion of explicit sensing operations (like `sense-open`) into a plan. When a sensing action is ultimately executed at the robot level, the result of the sensing will be returned to the planner through the memory level, as part of the standard state update cycle. When faced with a conditional branch point in a plan, the plan execution monitor will then make a decision as to the correct plan branch it should execute, based on the current state information. If such information is unavailable, for instance due to a failure at the robot/vision level, resensing or replanning activities will be triggered as above. It is important to note that the robot/vision system will never be aware of the conditional nature of a plan, and will never receive a "branch" operation like those shown in the example plans above. From the point of view of the robot, it will only receive a sequential stream of actions. This will also be the case for the memory level, except when a complete plan is requested. In such situations a fully-specified conditional plan will be transmitted to the memory level.

Initially, we expect that most plans will fail early, and often, and that most monitoring operations will trigger replanning activities. Our goal is to implement the basic framework for the plan monitor in the short term, in order to evaluate its effectiveness on plans being executed in the actual robot environment.

	ROBOT	MEMORY	PLANNER
1.	**MSG_STATE_UPDATE**: "ontable(obj1),...,!clear(obj1)"		
2.		(Forward to planner) **MSG_STATE_UPDATE**: "ontable(obj1),...,!clear(obj1)"	
3.			**ACK_STATE_UPDATE**
4.		(Forward to robot) **ACK_STATE_UPDATE**	
5.	**MSG_ACTION_REQUEST**		
6.		(Forward to planner) **MSG_ACTION_REQUEST**	
7.			**ACK_ACTION_REQUEST**
8.		(Forward to robot) **ACK_ACTION_REQUEST**	
9.			**MSG_ACTION_SUBMIT**: "sense-open(obj2)"
10.		*Refine* sense-open(obj2) *to* focus(obj2) (Forward to robot) **MSG_ACTION_SUBMIT**: "focus(obj2)"	
11.		(Send to planner) **MSG_PLAN_REQUEST**	
12.			**MSG_PLAN_SUBMIT**: "[sense-open(obj2), branch(open(obj2), [graspA-fromTable(obj2), putAway(obj2)], [])]"
13.	**ACK_ACTION_SUBMIT**		
14.		(Forward to planner) **ACK_ACTION_SUBMIT**	
15.	**MSG_ACTION_STOPPED**: "1"		
16.		(Forward to planner) **MSG_ACTION_STOPPED**: "1"	
17.			**ACK_ACTION_STOPPED**
18.		(Forward to robot) **ACK_ACTION_STOPPED**	
19.	**MSG_STATE_UPDATE**: "open(obj2)"		
20.		(Forward to planner) **MSG_STATE_UPDATE**: "open(obj2)"	
21.			**ACK_STATE_UPDATE**
22.		(Forward to robot) **ACK_STATE_UPDATE**	
23.	...	...	...

Figure 3: Example of messages sent during the execution of the sense-open(obj2) action in Example 2

7 Discussion

- All high-level grasp operators abstract the task of grasping into single action steps. We may extend the planner's representation to provide "finer-grained" actions that split the act of grasping into a sequence of steps like `positionForGraspA(obj1)`, `graspA-fromTable(obj1)`, `lift(obj1)`. Such actions would provide more detailed execution instructions to the robot system and, on failure, the robot system could more accurately indicate to the planner the specific components of the grasp that failed.

- In this document we only consider the refinement of `sense-open` actions at the memory level. In the future, we could also accommodate the refinement of grasp actions. For instance, the planner could generate plans with abstract actions like `grasp(obj1)`. It would then be the task of the memory level to make a decision as to the choice of grasping option and transform such actions into more specific robot-level actions like `graspA(obj1)` or `graspD(obj1)`.

- As future work, we believe that a more complex interaction between the robot, memory, and planning levels might be desirable. For instance, we may want the planning level to have the ability to terminate an action during its execution if it is having an undesirable outcome, or alert the memory about a replanning operation. This would require a more asynchronous ("push") architecture, including state update messages from the robot during action execution, as well as the ability to issue halt commands from the planning level. Moreover, we also see the possibility of a more comprehensive "bottom-up" role for the memory level, as an abstraction component that mediates between the robot/vision level and the high-level planner. We believe that such extensions will not require a significant reworking of the message passing protocol, but only slight extensions to the message set and control algorithm.

- We also envision a more significant extension to the message passing protocol to support the addition of new objects, new properties, and new actions (i.e., "the birth of an object/property/action") into the high-level planning representation as a result of memory-level reasoning. Partial support for such messages already exists at the code level of the socket library, however, future versions of the message passing protocol will more fully specify the operation of these message types.

- There are a number of places where incompleteness of information in the world state update can come into the system. Some of these are endemic to the interaction of a resource bounded agent working in a real world setting. As a result, we believe that we must seriously examine the limitations of the system's capability for providing complete state updates, as well as the traditional AI assumption that we have a complete model of the state changes that result from executed actions. This is a very interesting area for future work and something we are committed to looking at in detail. Initially, however, we will simply ensure that our action models and state updates are complete and correct.

- More issues to come as discussions progress...

Robotics Group
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark

Technical Report no. 2008 – 3

Object Model Learning using Bayesian Filtering

Nicolas Pugeault, Florentin Wörgötter, Norbert Krüger

January 29, 2008

Title	Object Model Learning using Bayesian Filtering
Author(s)	Nicolas Pugeault, Florentin Wörgötter, Norbert Krüger
Publication History	

Object Model Learning using Bayesian Filtering

Nicolas Pugeault
University of Edinburgh
npugeaul@inf.ed.ac.uk

Florentin Wörgötter
University of Göttingen
worgott@bccn-goettingen.de

Norbert Krger
University of Southern Denmark
norbert@mmmi.sdu.dk

Abstract

In this paper we present an active system that learns objects models by manipulating them. Information about the object's 3D shape is provided by reconstructing local contour descriptors. This shape information is accumulated over time in three ways: 1) disambiguation: erroneous stereo correspondences that are unsuccessfully tracked are discarded. We make use of aspect cues to increase the data association selectivity. 2) correction: the full pose of the reconstructed features is corrected over time using an Extended Kalman Filter approach; 3) completeness: multiple $2\frac{1}{2}D$ representations become merged, constructing a full 3D representation of the object. Combined with a grasping reflex, the system presented herein is intended to endow an active system with the capability to discover autonomously its environment. The described system is evaluated quantitatively on artificial scenario and qualitatively on a real scenario.

1. Introduction

In order to be really autonomous, a system need to be able to adapt to new, unplanned situations. It has been argued that a cognitive robot system need to

World knowledge, and more specifically the knowledge of the objects that inhabit it, their shape and their affordances, is of critical importance for an autonomous system. Given an appropriate model, robust methods exist for object recognition and pose estimation (see, e.g., [8]), and for grasping and manipulation (e.g. [3], [10]). This information is generally provided by the system's designer, e.g., as CAD model or collections of 2D views of the object spanning all perspectives,. However, extracting such representations from unknown objects is a difficult task. A quality of a cognitive system is the ability enrich its world knowledge by exploration, thus increasing its cognitive abilities (see, e.g., [5]).

This paper introduce a system that generates an internal representation of unknown objects by manipulation. First, a set of local contour descriptors are extracted from the image, providing a first representation of the 2D shapes. Two such representations are matched across two views using stereopsis, and the matches are used to reconstruct a first representation of the 3D shapes in the scene. At this stage, the representation is merely a collection of 3D primitives, objects and background are not segmented in any way. Assuming that nothing is known about the objects in the scene, the system will learn its representation by attempting to grasp and manipulate parts of the scene. Such a representation has been shown to be sufficient to generate simple grasping reflexes [1]. This will create a simple behavior where the system will attempt to randomly grasp parts of the scene, until it succeed. A successful grasp allows the system to take control over the object and to manipulate it to learn its shape. By using the motion knowledge provided by the robot, we segment the object from the scene (by selecting primitives that move according to the robot's arm motion) and accumulate the representation. Such an exploratory mechanism enable the system to explore its environment and learn the shape of the objects that inhabit it.

This paper presents the mechanism used for accumulating the visual representation over time. Having control over the object provides a very accurate knowledge of its motion that can be used to track individual 3D primitives. At each frame, new observations are used to correct the 3D primitives' full pose and to enrich the representation with new aspects of the object (e.g., parts that were previously occluded). The mechanism presented herein improves the representation in three respects:

- *Accuracy*: the representation is corrected over time using new observations.
- *Reliability*: tracking primitives allows to discard erroneous ones. Because we track 3D primitives, the likelihood for erroneous primitives to be tracked successfully is vanishingly small.
- *Completeness*: by manipulating the object, the system can witness it under a wide range of viewpoints, and accumulate $2\frac{1}{2}D$ representations into a full $3D$ representation.

Solutions to this problem belongs to two groups:

The first groups consists of the geometric analytic solutions, including multifocal tensors [6] and bundle adjustment [12]. These techniques are ideal solutions to the problem ("Gold Standard", cf. [6]) and are prominent in the strict batch–SFM scenario. They can be designed to be robust to erroneous data association (see [12] for a discussion). One major flaw with these solutions stems from the fact that they are *batch* processes: all views need to be available. This can make the problem intractable for large sequences, and implies a delay in any active system. It has been proposed to split the problem into groups of, e.g., 3 frames, reducing both delay and computational cost. Nonetheless, these approaches face the dead–reckoning problem and need an additional global integration stage.

The second group uses various flavors of the Bayesian filtering theory. This provides an on–line solution by formalizing the problem as a Markov process where a state vector combining the current pose and the landmarks' bearing can be formalized as the general Bayesian Tracking problem — see [2] for a review. This theoretical formulation allows for an optimal solution called Kalman filter if the state vector as a multivariate normal distribution and if the prediction and observation processes are linear.

Because of the on–line constraint, the approach presented in this paper belongs to the second category. Kalman filters and its non–linear brothers (e.g., Extended or Unscented Kalman filters) have been used extensively to solve the simultaneous localization and map–building problem (see, e.g., [4, 13, 11]) The present scenario is quite different, because the motion prediction is very accurate, the framerate is high, and a large proportion of the primitives is visible at any time. On the other hand, because they only encode local contour information, primitives are not very distinctive. SLAM, on the other hand, focuses on scenario where the motion is very inaccurate, successive frames are generally far apart and landmarks are very distinctive (e.g., SIFT [9]) The differences of the approach presented herein are:

- tracking in 3D space: the tracking of the primitives is done in 3D space, allowing to represent the rigid motion of the robot's arm as a linear operation using homogeneous coordinates.

- data association in stereo 2D space: 3D primitives are re–projected in both image planes to be compared with extracted primitives. This allows to reduce the uncertainty generated by stereopsis.

- full pose tracking: because the primitives are local contour descriptors, they encode local position and orientation. Therefore we use a pair of Kalman filters to filter the full pose.

The system is presented in section 2 and results on artificial and real scenarios are shown in section 3.

2. Framework presentation

This paper presents a framework that extracts 3D representations of object's shape from stereopsis, then accumulates and refine them using robot's arm motion knowledge. In a first stage, local contour descriptors are extracted from both images (section 2.1), then matched and reconstructed (section 2.1.1). The resulting 3D–primitives are tracked over time using the robot's arm motion knowledge and the object's shape representation is corrected, disambiguated and completed (section 2.2).

2.1. Primitive–based shape representation

In this work, we use local contour descriptors called primitives (see [7]). The primitives are sparsely extracted from local signal operators, and reconstructed in 3D using stereopsis. A primitive encodes local orientation, phase and color. They, therefore describe local tangents to image contours, augmented with these contours' color and phase information.

Therefore, a primitive π is encoded by the vector:

$$\pi = (\mathbf{x}, \theta, \omega, \mathbf{c}) \tag{1}$$

where $\mathbf{x}$ contains the primitive position, θ its orientation, ω its phase, and c its color. An appearance metric d_m between two primitives is defined as the mean of the distance between those two primitives in all three components θ, ω, c — see [7] for a discussion of the individual metrics.

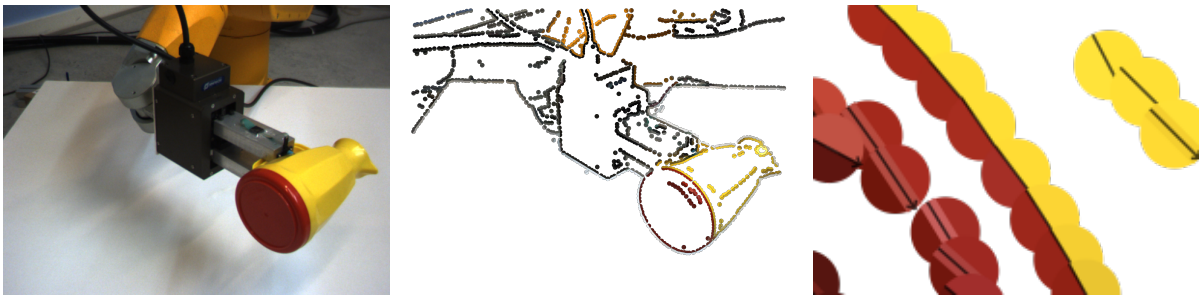
2.1.1 Stereopsis and 3D–reconstruction

Given one 2D–primitive in the first image, we compute the corresponding epipolar line in the second image. All 2D–primitives in the second image that lie close to the epipolar line are considered as *putative correspondences*. Each putative correspondence is associated a confidence depending on how similar it is to the primitive in the first image.

At this stage, a simple heuristic would be to select the correspondence with the highest confidence. We will see in the following that it is worthwhile preserving competing correspondences until more information is available.

Once we have a pair of corresponding 2D–primitives, we can reconstruct a 3D–primitive in space, that describes the 3D contour they describe. Because 2D–primitives are local tangents to the image contours, the reconstructed 3D–primitives are also defined by a position and an orientation in space, alongside with phase and color information. Therefore, a 3D–primitive $\mathbf{\Pi}$ is encoded by

$$\mathbf{\Pi} = (\mathbf{X}, \mathbf{\Theta}, \Omega, \mathbf{C}) \tag{2}$$



(a) image (b) 2D–primitives (c) detail

Figure 1. Illustration of the 2D–primitive extraction. (a) original image; (b) extracted 2D–primitives; and (c) detail of (b).

where $\mathbf{X}$ is the position, $\mathbf{\Theta}$ the orientation, Ω the phase, and C the color.

2.2. 3D–Primitive tracking

In order to improve the reconstructed shape model of the object, the 3D–primitives' pose is tracked using known motion. The predicted representation is then assessed, merged and corrected using the 3D–primitives reconstructed from the next stereo pair of images.

2.2.1 Prediction

In this work it is assumed that the object's motion is known from the robot's arm (Note that it could alternatively be computed using visual odometry). Considering a primitive with a pose $(\mathbf{X}_t, \mathbf{\Theta}_t)$, its pose $(\mathbf{X}_{t+1}, \mathbf{\Theta}_{t+1})$ after a motion $\mathbf{M}$ can be predicted by

$$\left\{ \begin{array}{rcl} \mathbf{X}_{t+1} & = & \mathbf{M} \cdot \mathbf{X}_t \\ \mathbf{\Theta}_{t+1} & = & \mathbf{M} \cdot \mathbf{\Theta}_t \end{array} \right. \tag{3}$$

2.2.2 Data association

The data association is performed in both 2D image planes, in order to reduce uncertainty. The predicted 3D–primitive is re–projected onto both image planes, and compared with the extracted stereo pairs of 2D–primitives that lead to a valid reconstruction in 3D. Two 2D–primitives $\mathbf{\Pi}_p$ and $\mathbf{\Pi}_e$ are considered as matched if they are *proximate* and *similar*, i.e.,

$$\left\{ \begin{array}{rcl} ||\mathbf{x}_p - \mathbf{x}_e|| & < & \tau_E \\ d_m(\mathbf{\Pi}_p, \mathbf{\Pi}_e) & < & \tau_A \end{array} \right. \tag{4}$$

Here d_m is an appearance constraint that is 0 if the two primitives look perfectly similar and 1 if they are totally different — see section 2.1. τ_M is the threshold on this appearance constraint. τ_E is the maximal distance between two primitives for a match to be valid. This is set to the size of the 2D–primitives, 3 pixels.

By extension, two 3D–primitives are matched if their re–projections in both image planes are matched.

2.3. Confidence re–evaluation

Stereopsis suffers from a certain level of ambiguity. Accumulating visual representations over time allows to resolve some of this ambiguity, and to discard erroneous assumptions. This section describes how the accumulated primitives' confidence is re–evaluated as a function of success of the primitive tracking.

2.3.1 Confidence update

The confidence in a putative 3D–primitive was originally estimated as the similarity between the stereo–pair of 2D–primitives that reconstructed it. In the general case where the scene contains repetitive structures, this assumption does not hold. Tracking the 3D–primitives over time allows to re–evaluate this confidence, and to discard erroneous primitives. Therefore, it is worthwhile conserving competing stereo–correspondences as discussed in section 2.1.1, to avoid losing correct hypotheses early on.

We will write the fact that a primitive $\mathbf{\Pi}_i$ that predicts a primitive $\hat{\mathbf{\Pi}}_i^t$ at time t is matched (as described above) as $\mu_t(\hat{\mathbf{\Pi}}_i)$. We define the matching history of a primitive $\mathbf{\Pi}_i$ from its apparition at time 0 until time t as:

$$\boldsymbol{\mu}(\mathbf{\Pi}_i) = \left(\mu_t(\hat{\mathbf{\Pi}}_i), \mu_{t-1}(\hat{\mathbf{\Pi}}_i), \cdots, \mu_0(\hat{\mathbf{\Pi}}_i)\right)^T \tag{5}$$

thus, applying Bayes formula:

$$p\left(\mathbf{\Pi}_i | \boldsymbol{\mu}(\hat{\mathbf{\Pi}}_i)\right) = \frac{p\left(\boldsymbol{\mu}(\hat{\mathbf{\Pi}}_i)|\mathbf{\Pi}\right) p\left(\mathbf{\Pi}\right)}{p\left(\boldsymbol{\mu}(\hat{\mathbf{\Pi}}_i)|\mathbf{\Pi}\right) p\left(\mathbf{\Pi}\right) + p\left(\bar{\boldsymbol{\mu}}(\hat{\mathbf{\Pi}}_i)|\bar{\mathbf{\Pi}}\right) p\left(\bar{\mathbf{\Pi}}\right)} \tag{6}$$

where $\mathbf{\Pi}$ and $\bar{\mathbf{\Pi}}$ are correct and erroneous primitives, respectively.

Assuming independence between the different observations, that $\mathbf{\Pi}$ has been tracked since n iterations, and has been matched successfully m times, the confidence can be

estimated to:

$$\begin{aligned} p\left(\boldsymbol{\mu}(\hat{\mathbf{\Pi}}_i)|\mathbf{\Pi}\right) &= \prod_t p\left(\mu_t(\hat{\mathbf{\Pi}}_i)|\mathbf{\Pi}\right) \\ &= p\left(\mu_t(\hat{\mathbf{\Pi}}_i)=1|\mathbf{\Pi}\right)^m p\left(\mu_t(\hat{\mathbf{\Pi}}_i)=0|\mathbf{\Pi}\right)^{n-m} \end{aligned} \tag{7}$$

In this case the μ_t are equiprobable and therefore, defining the quantities $\alpha = p(\mathbf{\Pi})$, $\beta = p\left(\mu_t(\hat{\mathbf{\Pi}})=1|\mathbf{\Pi}\right)$ and $\gamma = p\left(\mu_t(\hat{\mathbf{\Pi}})=1|\bar{\mathbf{\Pi}}\right)$, Eq. (6) can be rewritten as:

$$p\left(\mathbf{\Pi}_i|\bar{\boldsymbol{\mu}}(\hat{\mathbf{\Pi}}_i)\right) = \frac{\beta^m(1-\beta)^{n-m}\alpha}{\beta^m(1-\beta)^{n-m}\alpha + \gamma^m(1-\gamma)^{n-m}(1-\alpha)} \tag{8}$$

The prior and conditional probabilities were estimated to the values of $\alpha = 0.46$, $\beta = 0.83$, and $\gamma = 0.41$ using a variety of video sequences with known motion where depth ground truth was provided by a range scanner. This means that, in these examples, the prior likelihood for a stereo hypothesis to be correct is 46%, the likelihood for a correct hypothesis to be confirmed is 83% whereas for an erroneous hypothesis it is of 41%. These probabilities show that Bayesian inference can be used to identify correct correspondences from erroneous ones.

2.3.2 Confidence freeze

While the robot manipulate the object, it is expected that some parts of the object become occluded and therefore fail to be updated. For example, if the object is fully rotated, 3D–primitives on the occluded side will repeatedly fail to be matched during half of the rotation. This would normally lead to a large decrease in the primitive's confidence. In order to preserve primitives that become occluded, we state that if the a primitive's confidence become high enough (> 0.9) it is not updated anymore. Conversely, if a primitive's confidence drops below 0.1, it is discarded. This allows to reduce the system's computational load by only tracking promising hypotheses.

2.4. Correction of the representation

Even when the motion knowledge is very accurate, tracking a primitive by successive predictions is bound to become inaccurate after a large number of iterations. To prevent this problem, we correct the accumulated 3D–primitives at each time step with the observed 3D–primitives they were matched with. Moreover, correcting the primitives' full pose at each iteration with newly observed data allows to reduce uncertainty.

In order to correct the primitives' full pose, we use a pair of Kalman filters, one for the position and one for the orientation. Both states are handled as homogeneous vectors, as a rigid body motion in space is linear in homogeneous coordinates (the orientation vector is encoded as an homogeneous point at infinity, with an homogeneous coordinate of zero).

When considering rigid motion of 3D poses, the Kalman equations simplify to

$$K_{t+1} = \Lambda[x]_{t+1|t} \cdot \left(\Lambda[x]_{t+1|t} + \Lambda[z]_{t+1}\right)^{-} \tag{9}$$

$$x_{t+1|t+1} = x_{t+1|t} + K \cdot (z_{t+1} - x_{t+1|t}) \tag{10}$$

$$\Lambda[x]_{t+1|t+1} = \Lambda[x]_{t+1|t} - K_{t+1} \cdot \Lambda[x]_{t+1|t} \tag{11}$$

where $x_{t+1|t}$ is the predicted state, z_{t+1} is the observed state, $x_{t+1|t+1}$ the corrected state, $\Lambda[x]$ stands for the covariance matrix of x, and K_{t+1} is the Kalman gain.

2.4.1 Integration of new 3D–primitives

Finally, all reconstructed 3D–primitives that are not associated with a predicted one are added to the representation. This allows to complement the representation, but also to recover primitives that may have been mistakenly discarded.

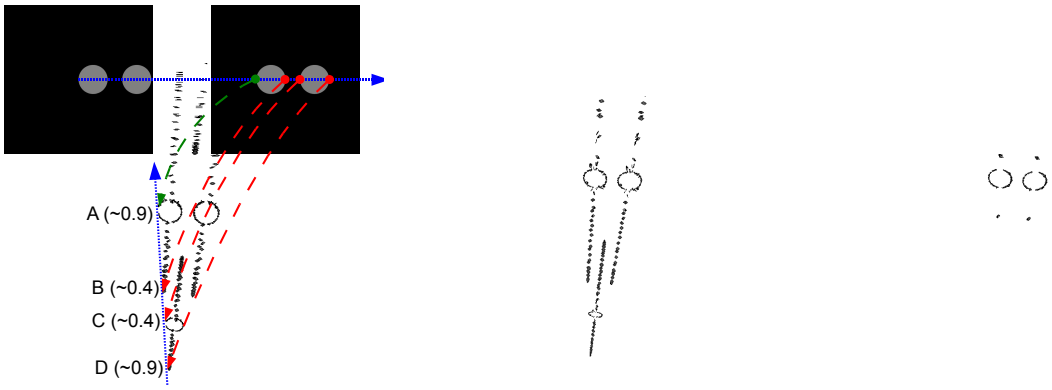
3. Results and discussion

The scheme presented in the above section has been evaluated on artificial and real video sequences, to evaluate different the aspects of the accumulation. As discussed in the introduction, a good accumulation framework should improve visual representation in three ways:

- *Accuracy*: the representation is corrected over time using new observations. This is evaluated in section 3.1 on artificial data.
- *Reliability*: tracking primitives allows to discard erroneous ones. This is evaluated on artificial data in section 3.2
- *Completeness*: by manipulating the object, the system can witness it under a wide range of viewpoints, and accumulate $2\frac{1}{2}D$ representations into a full $3D$ representation. This is evaluated in section 3.3 on a real scenario.

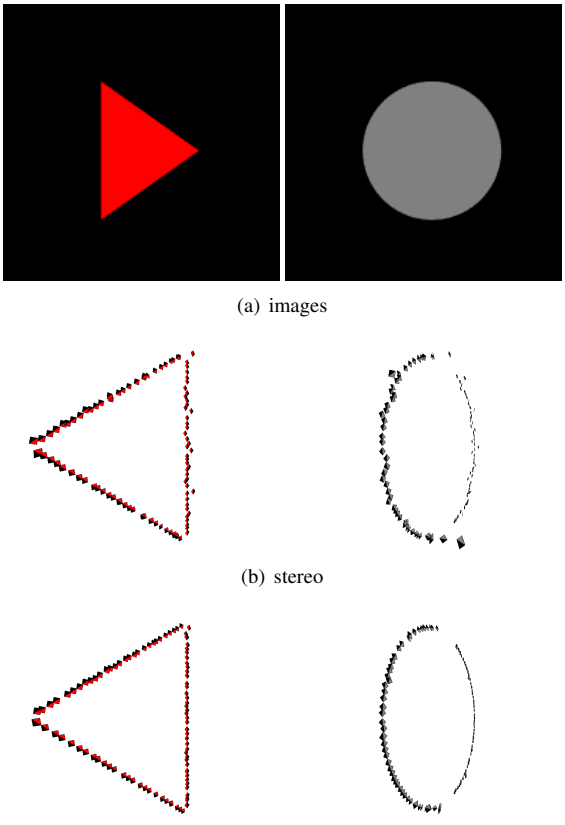
3.1. Correction

Figs. 3 and 2 shows the 3D pose error of the accumulated primitives during a pure translation. The same experiment was done with a triangle and a circle. The mean position and orientation errors both fall quickly to half their initial values, and the variance becomes neglectable. The same experiments was done with a rotation of 180 degrees (18 degrees per frame), with similar results. This shows that using a linear correction of rigid body motion is suitable approach for our scenario.



(a) correspondence problem (b) all candidates (c) after accumulation

Figure 4. Illustration of the correspondence problem and its disambiguation by tracking. In (a) the correspondence problem is illustrated: The primitive P in the left image has four putative correspondences on that satisfy the epipolar constraint, that lead to the reconstruction of four mutually exclusive 3D–primitives A, B, C and D. Because B and C have opposite phase, their similarity (0.4) is lower than the one of A and D (0.9). On the other hand, A and D are locally indistinguishable. Fig. (b) shows all putative correspondences reconstructed in 3D. Fig. (c) shows the result of the accumulation.



(a) images

(b) stereo

(c) accumulation 10 frames

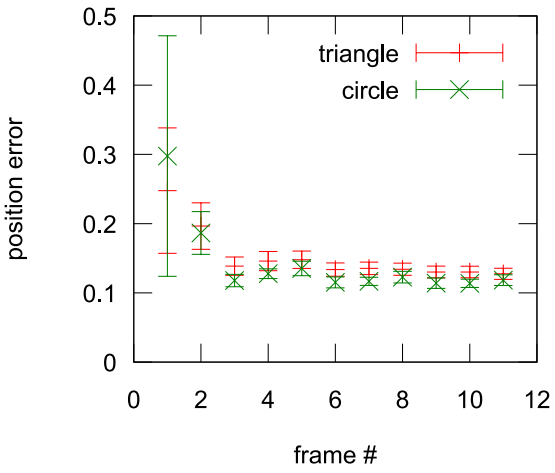
Figure 2. Pose correction over time, for a pure lateral translation.

3.2. Selection

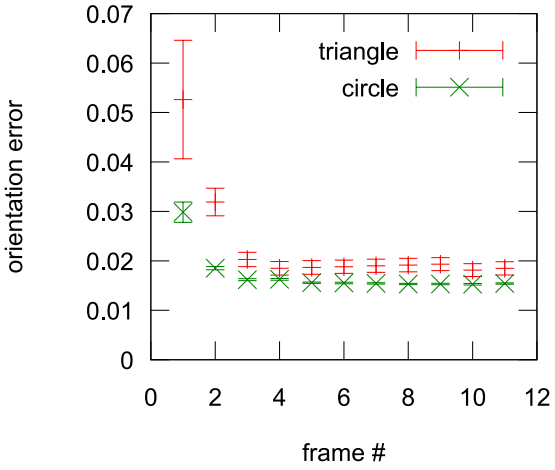
Fig. 4 illustrates how the accumulation process resolves the ambiguity generated by the stereo correspondence problem. Consider one primitive in the left image, its potential correspondences all lie on the epipolar line χ. In the example of Fig. 4(a), the epipolar line cross four image contours (two per circle) and therefore stereopsis allow for four mutually exclusive reconstructions, noted A, B, C and D. Out of those four, two (B and C) imply that the left side of the left circle be matched with the right side of one circle in the left image. Although the primitives' orientation is similar, the phase information contradicts such association, the similarity of these matches is low (0.4). Therefore these two hypotheses could be discarded by a local appearance constraint. On the other hand, the two other hypotheses (A and D) are locally indistinguishable. Fig. 4(b) show the reconstruction of all hypotheses: the two far circles represent the correct reconstruction. The nearby circle is generated by the left side of the left–hand circle matched with the left side of the right hand circle (e.g., case D). The lines are generated by the unlikely match of the left and right sides of the circles (e.g., cases B and C). Fig. 4(b) shows the hypotheses left after accumulation: all erroneous hypotheses have been removed.

3.3. Completeness

We applied the scheme presented above to some real video sequences. We use an industrial robot arm (Staubli RX60) that allows for a very precise motion control. The images were captured using a pre–calibrated Bumblebee camera system. The hand–eye calibration was also calculated off–line. Fig. 5 shows the result of the applications



(a) position



(b) orientation

Figure 3. Pose correction over time, for a pure lateral translation, see Fig. 2.

the scheme presented herein to this scenario. The object is rotated by the robot's arm so that all perspectives are shown to the cameras. The bottom part of the figure shows the reconstructed (left) and accumulated (right) representations, from two different views. The accumulated representation is free of most outliers, while offering a representation of the whole 3D shape. On the other hand simple stereo reconstruction lead to an incomplete and less reliable representation. In order to provide the reader with a better perception of the reconstructed and accumulated representations, three videos are attached to this paper. The first one, named `accumulation-process.avi` shows the accumulated representation, every 5 frames until frame #45. The second one, named `stereo.avi` shows the visual representation reconstructed by stereopsis, under different

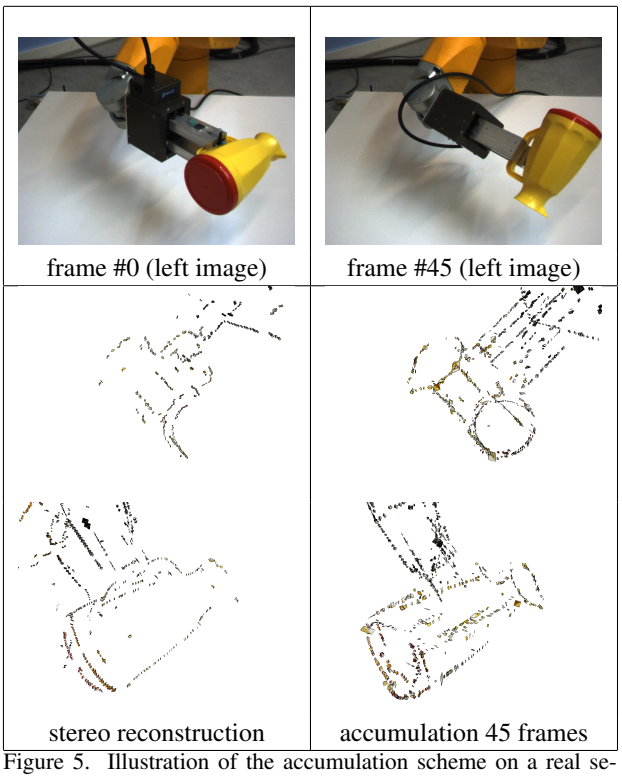


Figure 5. Illustration of the accumulation scheme on a real sequence. The plastic jug held by the robot's arm is rotated, showing all parts of the object to the camera. Two views on the reconstructed and accumulated representation are shown.

viewpoints (at frame #45). Finally, the last one, named `accumulated.avi` shows the accumulated representation (also at frame #45) from different perspectives.

4. Conclusion

This paper presented a Bayesian filtering framework for accumulating instantaneous $2\frac{1}{2}D$ visual representations, reconstructed using stereopsis, into full 3D representations. The proposed framework is shown to improve the quality of the representation, comparatively to pure stereopsis, in three respects: accuracy, reliability, and completeness. Implemented into an active system with a grasping reflex and a basic exploratory behavior, this mechanism enables a system to discover objects in its environment.

Future work includes using visual odometry in cases where the knowledge of the robot's arm is inaccurate of absent. Inaccurate motion requires to correct the motion estimate along with the primitives. Also, the confidence re–evaluation scheme could be refined by considering the quality of each match instead of the binary matched / not–matched that is presently used.

References

[1] D. Aarno, J. Sommerfeld, D. Kragic, N. Pugeault, S. Kalkan, F. Wörgötter, D. Kraft, and N. Krüger. Early Reactive Grasping with Second Order 3D Feature Relations. In *IEEE International Conference on Advanced Robotics (ICAR'07)*, 2007. 1

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non–Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 2002. 2

[3] C. Borst, M. Fischer, and G. Hirzinger. A fast and robust grasp planner for arbitrary 3D objects. In *Ieee international conference on robotics and automation*, pages 1890–1896, Detroit, Michigan, May 1999. 1

[4] P. Dissanayake, P. Newman, H. Durrant-Whyte, S. Clark, and M. Csorba. A solution to the simultaneous localisation and mapping (SLAM) problem. *IEEE Transactions in Robotics and Automation*, 17(3):229–241, 2001. 2

[5] P. Fitzpatrick and G. Metta. Grounding vision through experimental manipulation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 361:2165 – 2185, 2003. 1

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000. 2

[7] N. Krüger, M. Lappe, and F. Wörgötter. Biologically motivated multi-modal processing of visual primitives. *Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour, AISB Journal*, 1(5):417–427, 2004. An full technical description is also available as the technical report 2007-4 of the Maersk Institute. (www.mip.sdu.dk/covig/publications/primitives.pdf). 2

[8] D. Lowe. Three–dimensional object recognition from single two images. *Artificial Intelligence*, 31(3):355–395, 1987. 1

[9] D. Lowe. Distinctive Image Features from Scale–Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. 2

[10] A. Miller, S. Knoop, H. Christensen, and P. Allen. Automatic grasp planning using shape primitives. In *Ieee international conference on robotics and automation*, volume 2, pages 1824–1829, 2003. 1

[11] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *International Journal of Robotics Research*, 23(7–8):693–716, 2004. 2

[12] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. Technical report, 1999. 2

[13] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. The Unscented Particle Filter. Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, 2000. 2

Object Separation using Active Methods and Multi-View Representations

Kai Welke, Tamim Asfour and Rüdiger Dillmann
University of Karlsruhe (TH), IAIM, Institute of Computer Science and Engineering (CSE)
P.O. Box 6980, 76128 Karlsruhe, Germany
Email: {welke, asfour, dillmann}@ira.uka.de

Abstract— **In this paper an approach for object separation using active methods and multi-view object representations is presented. Daily life objects reveal natural similarities, which cannot be resolved with the perception of a single view only. The presented framework allows for the separation of different object hypotheses which have similar views to the current percept. By actively rotating the object, the coherence between controlled path, inner models and percept is observed and allows to reject implausible hypotheses. Using the same framework, pose and object correspondence can be determined. With the benefit of active methods the perceptual task can be solved using very coarse features, which facilitates a compact multi-view object representation. The proposed approach is independent from a specific visual feature descriptor and thus suitable for multi-modal object recognition.**

I. Introduction

In this work an approach is presented which solves a task that is natural to humans. A humanoid robot which acts in a natural environment has to cope with a large variety of different objects. In order to perceive the surrounding world in a robust manner the robot has to be able to learn, classify and act on objects accordingly. The amount of different objects imposes a challenging problem for the research in machine vision. Many approaches for object recognition are restricted to a small amount of objects and are also restricted to objects that are separable with the feature extraction method deployed. Since visual perception is one of the building blocks of cognition, this restriction hinders the application of cognitive systems in real environments. Inference and reasoning based upon usually require a large set of examples which cover the variety of percepts required to solve a cognitive task.

One typical problem when dealing with a large amount of objects consists in the natural similarities of daily life objects. Many objects can not be discriminated when observed from one unique view point using a single feature descriptor. There are two different approaches to cope with this fact. One possibility consists in the integration of different modalities and senses which allows for a reduction of the uncertainty that is imposed by the similarities. Another possibility consists in the active exploration of objects in order to determine the correct correspondence between inner models and perceived world. In this paper we present an approach which uses active vision to reduce the uncertainty deriving from similarities in the world surrounding. Within a coupled action-perception framework, the robot generates views of the object until the perception system is capable of narrowing the possibilities of matches between inner models and perceived object.

Since the active vision paradigm was introduced ([1], [2], [3]) the availability of humanoid robot systems with distinct manipulation capabilities has opened the possibility to study and implement active methods in real environments. Recent research focuses on solving some ill-posed problems in machine vision with active methods. Fitzpatrick et. al use the manipulator of a robot to gain an initial idea of the presence and the shape of an object [4]. Omrcen et. al propose a control scheme and an active vision approach which addresses the problem of figure ground segmentation [5].

In the work introduced in this paper, an active approach for object separation is presented which is designed for the implementation on a humanoid robot platform.

The next section will introduce some principles from cognitive science and neuroscience which were taken into consideration during the development of the approach. The subsequent section gives a brief overview of the different modules of the proposed system. The algorithm used for interpretation of the current percept and adaption of the movement will be explained in Section IV. Finally, experimental results are presented and discussed.

II. Guiding Principles

In the following section some of the guiding principles in the development of the system are presented. All principles stand in line with the conviction that perception systems aimed for the application in cognition should agree with the basic findings of cognitive science and neuroscience in the past years. Here we present principles which directly influence the approach, explain how and give references to work which focuses on similar aspects.

1) Object representation is based on two dimensional views: In our work we use view based representations of objects. Psychophysical experiments on humans [6] and on monkeys [7] have lead to a view-based model of how the visual system achieves consistent identification of objects. View-based methods model the object by selected views rather than by constructing a 3D-model to match the object in the scene. Logothetis et al. found cells in the macaque Inferotemporal cortex (IT) that are tuned to specific views of an object [8]. Psychophysical studies carried on with human subjects indicate that object recognition is performed

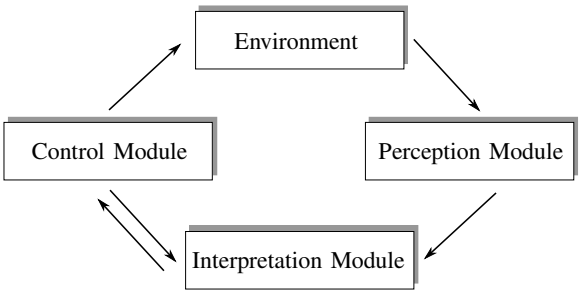


Fig. 1. Overview of the system structure. The system couples control and perception modules via the environment. The interpretation module receives input from perception and control modules and guides the control module.

around views presented while training [9]. Logothetis et al. identified view-tuned cells which respond with a significant spike rate for specific views of objects. In the presented approach each object model consists of a set of object views which are represented by nodes in a spherical graph. This representation of objects is usually referred to as *aspect graph* or *viewing sphere*. In recent research the aspect graph showed to be a feasible representation for multi-view object representations. Aspect graphs usually contain prototypical views of objects. These views are used during recognition tasks (see e.g. [10]). A main interest also has been to extract outstanding views of objects as prototypes which allow for a more compact and descriptive representation ([11], [12]). Also the problem of pose estimation was addressed using a combination of local features and aspect graphs [13].

2) Representations should allow for multi modal integration: One possibility to reduce uncertainty in perception consists in the integration of different cues for the task of object separation. There have been many efforts to integrate different visual modalities [14] and modalities from different senses ([15], [16]) into a unified percept. In this paper an approach is proposed that allows for the integration of different visual modalities. As described later, only very coarse global features are used for the experiments. Features can be used with the approach if they are global and rotationally invariant in the viewing plane.

3) Sensory memory is transient and of limited capacity: The model of human sensory memory has been introduced with the Atkinson-Shiffrin memory model [17]. Sensory memory contains rich sensory information and is transient. The proposed system accounts for this model in the sense that rich sensory information is only stored in order to be processed immediately. The approach is designed in a way that, despite the inner models, only the features from the current percept are required for processing.

III. System description

Figure 1 shows the structure of the proposed system. The control module guides the rotation of the object using the manipulator of the robot. In simulation the control module sets the rotation of the simulated object model. In execution we plan to use a control scheme based on the null space of the Jacobian as presented in [5]. The perception module provides the feature extraction methods. In this work global features are extracted from the current percept. For the application on the robot, background subtraction techniques are required. In the interpretation module the system checks the coherence between object hypotheses, percepts and the currently controlled movement. For this purpose path hypotheses on the surface of the viewing sphere are generated and compared with the controlled path and the sequence of processed percepts. From the path hypotheses the pose between controlled path and object hypotheses is calculated. With the pose, the best separating view of object hypotheses is determined and the controlled movement is adjusted accordingly. Object separation is performed using quality ratings for path hypotheses. Since the movement approaches a view which is ideally only valid for one object, only path hypotheses belonging to that object will be plausible and rated accordingly.

All three parts of the system run at different speeds. The interpretation module runs at about 3Hz. Each time an iteration has been completed, the feature of the current percept is requested from the perception module. The timing of the control module is independent. In the experiments a new movement is initiated every second.

As feature descriptor color cooccurrence histograms (CCHs) are used throughout the experiments. CCHs offer some properties which allow the application in real world recognition tasks. The resulting description of the objects' appearance is invariant to rotation in the viewing plane and robust to scaling. CCHs combine texture information in terms of the distribution of pixel pair colors as well as color information. For more detailed information the reader is referred to [18]. In our work CCHs based on red and green color channels as well as on the gradient image of red and green color channels are used. This results in a feature vector of 320 dimensions.

A. Building the model

Prior to object separation using the proposed approach, object models in the desired form have to be generated. Object models consist of aspect graphs with features corresponding to the views associated to each node. Object views are generated in simulation by rotating a 3D model of the object and sampling views at predefined positions. For the experiments in this paper, only block shaped models are deployed. The nodes in the aspect graph are selected to be distributed equidistant on the viewing sphere surface. Considering the Euclidian distance, the points which are closest to one node describe a Voronoi region. To encode the neighbour relationship between Voronoi regions in the aspect graph, Delauney triangulation is performed. Nodes are connected with an edge if their corresponding Voronoi regions have a common edge. All extracted views are processed with the feature extraction method. As the inner model for the objects, the extracted features for each view, the node positions and connections are stored.

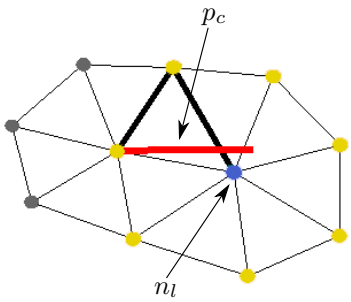


Fig. 2. The initial algorithm connects nodes to the path hypothesis which are neighbours of the last node n_l and similar to the current view from the controlled path p_c

B. Perception Module

The perception module extracts descriptors from the appearance of the currently perceived object. Since background subtraction is not necessary in the simulation, CCHs are directly extracted from the current view of the object.

C. Control Module

The control module specifies which view of the object will be visible to the perception module. In all our experiments we start with a random movement. Once the most separating view of the object hypotheses is determined within the interpretation module, control is guided towards this view.

D. Interpretation Module

The interpretation module checks the coherence between object hypotheses, current percepts and the controlled movement. To accomplish that, path hypotheses are rated considering their plausibility in comparison to the controlled path and the current percept. From the interpretation, the transformation from controlled path to hypothesis paths can be determined. The controlled path is adjusted by the interpretation module once the transformation of all object hypotheses has been calculated. The controlled movement is directed towards the most separating views. With the rating of the path hypotheses, the correct correspondence for the current percept can be calculated within the object hypotheses.

IV. Path Hypothesis Generation

Since the system has to cope with very coarse and rotational invariant feature descriptors the perception of one object view alone is not sufficient to discriminate between objects. In order to separate between multiple object hypotheses the ability of a humanoid robot to actively rotate objects in front of its cameras is exploited. While rotating, the current percept of the object changes and reveals new object views at different aspects. The direction of the object rotation can be specified in the control module and is made availabe to the interpetation module.

A. Initial Algorithm

Each path hypothesis p_i can be expressed with a sequence of nodes in the following way:

$$p_i = (((\alpha_0, \beta_0), c_0), \ldots, ((\alpha_{N-1}, \beta_{N-1}), c_{N-1})) \qquad (1)$$

where α describes the rotation of the node about the vertical axis and β describes the rotation about the horizontal axis. The value c_m describes the number of percepts which have been valid for the path element m and is referred to as hit counter.

From the sequence of rotations initiated by the control system the controlled path p_c can be determined in a similar way. The angles α and β describe the currently controlled rotation. The hit counter c_m for each path element of the controlled path p_c is set to one.

In the course of rotating, path hypotheses on the viewing sphere of the object hypotheses considering the feature of the current view are determined. Algorithm 1 describes the initial idea on how to generate path hypothesis:

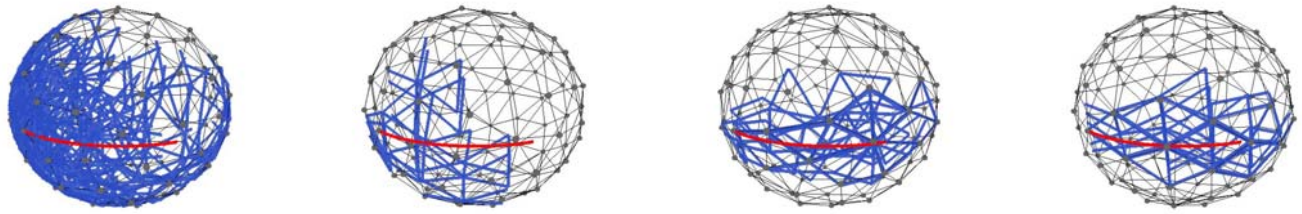
Input: Current Views and Object Models
Output: Path Hypothesis $P = \{p_i\}$

```
1  {p_i} = searchExhaustive(current_view, models);
2  foreach Path p_i do
3      n_l = lastNodeOfPath(p_i);
4      {n_j} = neighbours(n_l);
5      {n_j} = {n_j} ∪ n_l;
6      n_b = mostSimilar_{w_0}(current_view, {n_j});
7      if n_b == n_l then
8          increaseHitCounter(n_l);
9      end
10     else
11         appendNodeToPath(p_i,n_b);
12     end
13 end
```

Algorithm 1: Initial algorithm for path hypothesis generation

The feature of the current view and all object models are made available as input to the algorithm. Once, on start of the algorithm, all features of the object models have to be traversed and compared with the extracted feature of the current view. Only the N_{hypo} best matches are stored in the set of path hypotheses P. In each iteration, the last node n_l of each path p_i is considered. With the edges stored during the model building step, the direct neighbours of n_l can be identified. The stored features of all neighbours n_j including the last node n_l are compared with the input feature. The most similar node n_b is determined using the similarity measure of the feature extraction method used denoted with w_0. If the most similar node corresponds to the last node of the path n_l the hit counter of that node is increased. Otherwise, the most similar node n_b is appended to the path p_i.

Figure 2 illustrates how the algorithm searches for new nodes. The object was already rotated according to the controlled path p_c. During rotation, the path hypothesis



(a) Initial Algorithm (b) Particles Algorithm (c) Loop Elimination (d) Similarity Accumulation

Fig. 3. All 50 path hypothesis for the four different variations of the algorithm. The object was rotated 70 degree about the vertical axis. The controlled path p_c is displayed in red. All path hypothesis have been transformed according to the estimated pose.

extended from an initial node to two connected nodes. In each iteration, the algorithm searches in the set of nodes connected to the path's last node n_l for the most similar view. If the most similar view is not the last node, the path is extended accordingly.

Figure 3(a) shows 50 path hypotheses after rotating an object 70 degrees around the vertical axis. The generated hypotheses are distributed over the aspect graph and no good hypothesis could be identified. The bad performance of the algorithm is due to the fact that using CCHs, neighbouring nodes usually exhibit very similar descriptors. If the controlled path moves away from one starting node the current view will be similar to a large amount of neighbouring nodes. It can not be guaranteed that the most similar neighbour will be in the direction of the controlled path. Once a node is chosen which leads to a different direction than the controlled path, the algorithm cannot backup to the correct path, since only neighboured nodes are considered.

In the following we will explain how the initial algorithm was adapted in order to gain the ability to deal with coarse features like CCHs. All adpations were made with the principles presented in section II in mind.

B. Hypothesis generation using Particles

In the initial algorithm the knowledge of the controlled path p_c was not considered. Path hypotheses were only generated on base of the similarity of features from the current percept and nodes of the object models. Since the course of the controlled path is known, path hypotheses can be rated according to their similarity not only to the current view, but also to the controlled paths course. To judge the quality of the path course the similarity between controlled path and path hypothesis has to be determined. In order to obtain comparable paths, the rotations required to transform the hypothesis path to the controlled path are calculated. Figure 4 illustrates how a path hypothesis with starting point s_1 and end point e_1 are transformed into the controlled path with starting point s_2 and end point e_2. Three rotations with different rotation axes are performed. The first two rotations with the angles α and β ensure that the starting points of both paths match. The third rotation with the angle ψ assures that the transformed starting point s''_1, the starting point s_2, the end point e_1 and the end point e_2 lie on the same arc of the viewing sphere.

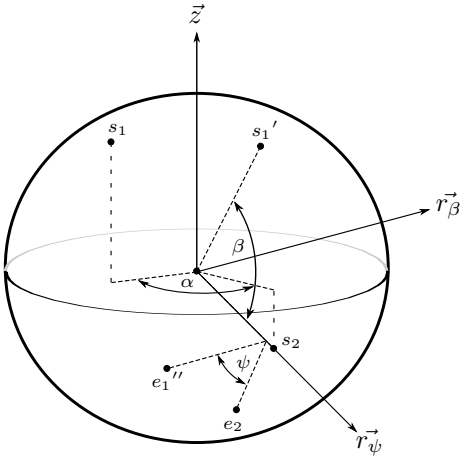


Fig. 4. Transformation of the path (s_1, e_1) into the path (s_2, e_2). First a rotation around the z-axis with angle α is performed until the projection into the xy-plane of the startingpoints s_1, s_2 lie on the same line. Then a rotation with the angle β is performed in order to tranform s_1 to the position of s_2. Finally the rotation ψ is performed to ensure that the transformed endpoint e_1'' lies on the same arc as s_2 and e_2.

Overall the complete transformation t_i can be described with the following parameters:

$$t_i = (\alpha, \beta, \psi, r_{\vec{\beta}}, r_{\vec{\psi}}) \tag{2}$$

where $r_{\vec{\beta}}, r_{\vec{\psi}}$ are the rotation axis for the rotation with β and ψ respectively. The initial rotation α is performed around the z-axis.

Once the transform is determined, the similarity of the path's courses can be calculated. In order to compare paths according to their elements it is ensured during the composition of the controlled path that the overall number of hits of hypothesis path and controlled path is identical. This is achieved by adding one element to the controlled path each time a new iteration of the hypothesis generation is started and setting the hit counter to one. Let $h(p, e)$ be the function that returns the path element which has been valid at the time of the e-th hit. The similarity of the path's course can then be calculated in the following way:

$$w_1(p_c, p_i) = \frac{\sum_{e=0}^{H-1} d(h(p_c, e), h(p_i, e))}{H} \tag{3}$$

where H is the sum of hits over the complete path and d is returns the distance of both points on the sphere.

In order to integrate the path course rating, an approach similar to particle filtering is deployed. In a hypothesis generation step, different hypotheses referred to as particles are generated. In a verification step, the best particles are determined and kept for the next iteration of the algorithm. The initial algorithm is altered in the following way. Instead of appending the most similar node to a hypothesis path (Alg. 1 line 11) particles are generated for each neighbour of the last node n_l and the hit counter of the node n_l in the currently considered particle is increased. After all path hypotheses have been generated, the similarity of the path courses with the controlled path $w_1(p_c, p_i)$ is calculated. Furthermore, the similarity of the last new node n_l of each path hypothesis with the current view is determined using the similarity measure for the feature descriptor w_0. In order to combine both measures independent from their actual values all path hypotheses are inserted into one priority queue using their similarity for each measure. The overall rating of the path is calculated by the mean position of the hypothesis in both queues. Since the number of generated hypotheses is increased with this approach, only the N_{hypo} hypotheses with the best overall rating are stored for the next iteration.

Figure 3(b) illustrates all path hypotheses after applying the adapted algorithm to a rotation around the vertical axis of 70 degrees. Still the course of the generated hypothesis does not resemble the controlled path. Again the similarity between neighboured nodes leads to the generation of hypotheses that do not approximate the controlled path in a satisfying way. While iterating, the similarity measure w_0 causes the paths to loop between similar nodes which results in a bad rating using the measure w_1.

C. Loop Elimination

In order to take account for this behaviour, a loop elimination step is integrated into the algorithm. With restricting the allowed controlled movements to loop free movements, all elements of the path that have the same start and end node are deleted and the hit counters of the remaining elements are adjusted accordingly. Afterwards all path hypothesis are compared and paths are deleted if there is another path in the set of hypothesis with the same node sequence. This is necessary to keep the number of required hypothesis small. Figure 3(c) illustrates the resulting hypothesis after applying loop elimination and the removal of identical paths. The generated hypotheses form a good approximation of the controlled path.

D. Similarity Accumulation

Until now, only the similarity of the last node of each path hypothesis and the current view were considered for the rating of path hypotheses. Now, in order to improve the approximation of the controlled path, the best similarity that was encountered while rotating the object is stored for each node. These best similarities can be accummulated in another measure:

$$w_2(p_i) = \frac{\sum_{j=0}^{N-1} b(n_j)}{N} \tag{4}$$

where N is the number of nodes of path p_i and $b(n_j)$ returns the best similarity of an input view to node n_j as encountered while rotating. The measure w_2 is integrated with the other measures in the same way as described above to generate an overall path hypothesis rating. The resulting paths after integrating similarity accumulation into the algorithm is illustrated in Fig. 3(d). The visible difference between the outcome with and without similarity accumulation is marginal. Nevertheless in the results section we will show that the convergence of the algorithm can be improved by introducing the measure w_2.

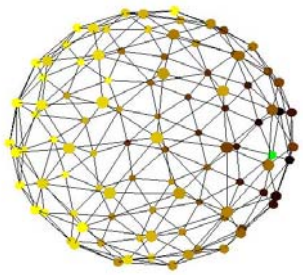
E. Object Separation

In order to calculate the best separating view between objects the relative pose between the object hypotheses has to be known. The relative pose can be derived from the transformations of the path hypotheses to the controlled path. The calculation of the best separating view is initiated, once the running variance of the mean transformation of all path hypotheses is below a threshold. For all object hypotheses the running variance is calculated using the mean rotation angles α, β and ψ over all path hypotheses. A window size of 5 is used for the running variance and the sum of the variances of all three angles is used to establish a threshold for convergence of the transformation. The relative pose is approximated by the mean of all rotation angles for each object hypothesis.

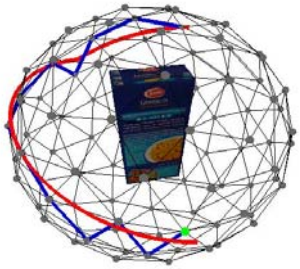
With the relative pose, the aspect graphs of all object hypotheses are transformed into one common base coordinate system. Once having all views in the base coordinate system, the similarity graph between the object hypotheses can be calculated. Therefore, the best object hypothesis is determined using the path ratings w_0, w_1 and w_2 and its structure is copied to the similarity graph. For each node in the similarity graph the closest node in the common coordinate system of each remaining object hypothesis is determined. In order to derive a measure for the similarity of corresponding nodes, the variance of all features associated to a set of closest nodes is calculated. The result is stored in the nodes of the similarity graph. Figure 5(a) illustrates the similarity graph for two object hypotheses with different textures on their backsides.

In order to discriminate between the object hypotheses, the robots ability to rotate objects is exploited to reveal the most separating view. Since also small inaccuracies of the pose estimation and control side have to be coped with, the most separating view is determined, also considering neighboured nodes in the similarity graph. For all nodes in the similarity graph the mean variance of the corresponding features of the node itself and all its neighbours is calculated. The node with the highest mean variance is used as the most separating view. In figure 5(a) the most separating view is denoted with green color.

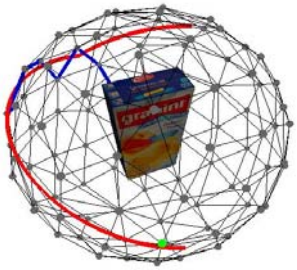
Since also the transformation between controlled path and object hypothesis is known, the position of the best separating view is available in the control coordinate system. The closest path between the current view and the best



(a) Similarity graph as calculated from two object hypothesis and related estimated poses.



(b) The path hypothesis for the correct object hypothesis approximates the controlled path p_c.



(c) The path hypothesis for the incorrect object hypothesis does not converge to the controlled path p_c.

Fig. 5. Object Separation

separating view is calculated and the movement is adapted accordingly.

In order to separate between object hypotheses, the measure w_1 of the best path hypothesis is considered. Once the current percept is not coherent with one of the object hypotheses, the path will not be able to proceed in the desired direction and will vary from the controlled path. This state can be determined by monitoring the path rating w_1. There are two possible stages of the system, where one object hypothesis becomes invalid. Either the system approaches an object view which is not coherent with one hypothesis before the relative pose could be estimated or the system already moves towards the most separating view, which will force incoherent object hypotheses to be eliminated.

Figure 5(b) illustrates the estimated path for a valid object hypothesis. The controlled path could be approximated well, which results in a good rating with the path measure w_1. In Fig. 5(c) the backside of the object was exchanged compared to the inner model. Once views of the object that are not similar to the inner model are revealed, the algorithm does not find its way in the vicinity of the controlled path. This will cause a reduction in the path rating measure w_1.

V. Experimental Results

All results in this paper were achieved in simulation. As mentioned in section III-A, block shaped models of objects were used to generate views at the desired viewing angles. The application of block shapes does not imply a simplification for the approach. Since the algorithm has to cope with planar surfaces, many neighboured views of one object look similar and thus do not reveal information which can be exploited to generate path hypotheses.

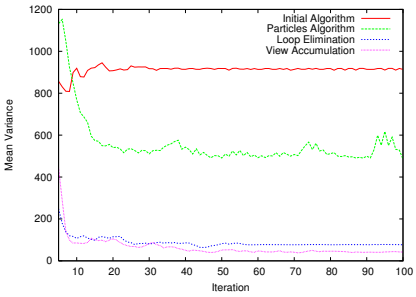
All inner models were represented with aspect graphs consisting of 100 views. The number of maximum hypotheses per iteration was set to $N_{hypo} = 50$. Both parameters were chosen empirically prior to the experiments. The approach was evaluated using randomly selected starting points in spherical polar coordinates in all experiments. All inner models were transformed using random rotation axes and angles. The initial direction of the movement was also selected randomly with constant increments in the azimuth and zenith in spherical polar coordinates. By using spherical polar coordinates, it was ensured that not only straight paths were generated.

A. Path hypothesis accurracy

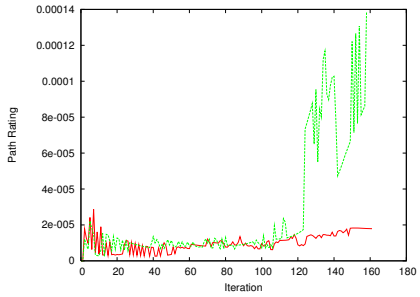
The parameters for path hypothesis convergence were chosen in a way that the generated pose hypotheses are accurate enough to find the transformation between multiple object hypotheses and between object hypothesis and the control space in order to identify and reach the most separating view. During model building, the viewing sphere was discretised to 100 views. The mean angle between neighboured views amounts to 22.6 degrees. This gives a benchmark to select the thresholds for convergence accordingly.

Figure 6(a) illustrates how the different variations of the initial algorithm perform in comparison. The results were achieved by using views of the same model for the representation as well as for the view generation in order to allow convergence of the path hypotheses. To capture the correct trend for the different approaches, 100 rounds with random starting points and transformations were executed for each variation. The graph shows the development of the sum of mean running variances over the three rotations measured during 100 rounds. As can be seen, the initial algorithm did not converge at all to a robust object pose estimation. With the introduction of particles the pose approximation performed better but settled down at a very high variance. The restriction to loop free paths allowed to produce much more realistic paths on the surface of the viewing sphere. Combined with the elimination of similar paths this approach converged very fast to a stable amount. The accumulation of views again improved the convergence of the hypotheses.

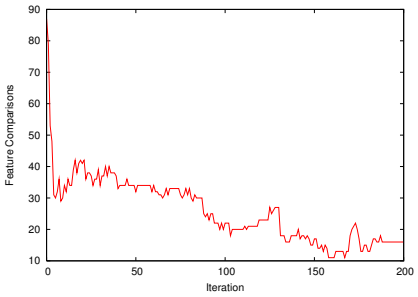
To measure the accuracy of the pose approximation process the mean pose error was calculated for 100 rounds. Each round was limited to 400 iterations. If the path hypothesis did not converge after 400 iterations, the round was marked as failure and the pose estimation was not considered. Eight of the 100 rounds were marked as a failure. All eight errors occurred in cases where the randomly chosen controlled path was close to the singularities of the spherical polar coordinate system. In these cases, only a few very similar views were



(a) Convergence of the mean variance of path hypothesis for all four different variations of the algorithm. The best results could be achieved using the particles approach with similarity accumulation.



(b) The path rating measure w_1 is shown for two object hypothesis with different backsides. Once the backside is reveald through the control movement, the rating of the invalid hypothesis increases.



(c) Number of feature comparisons required in relation to the iterations of the algorithm. With convergence of path hypothesis the number of required comparisons decreases.

Fig. 6. Experimantal results using the proposed approach.

presented to the system which was not sufficient to allow the path hypothesis to converge to a valid solution. The resulting mean pose error using the remaining 92 rounds amounted to 14.89 degrees which lies within in the desired accuracy of 22.6 degrees.

B. Object Separation

To evaluate object separation, two pairs of object models were deployed. In each pair, the backside of the objects differed. One of the objects from each pair was used as input and both models of one pair were stored as inner models. The task was the identification of the correct correspondence for the input model.

In Fig. 6(b) the path rating measure w_1 for the best path hypothesis is shown for both object hypotheses over the iterations until the movement reached the most separating view. Once a view is revealed which is not coherent with one of the object hypotheses, the distance between controlled path and best path hypothesis increases immediately. This circumstance is deployed to discriminate between the object hypotheses. We assign a perceived object to an object hypothesis if its path rating is 5 times better than the path rating of the remaining object hypothesis.

Using this threshold, 100 separation tasks were performed. For each task the convergence of the pose estimation and the calculated correspondence were determined. In the 100 test cases, the object be assigned to the correct object models with only the path rating w_1 in 88% of the cases. In the remaining 12% of the cases, the threshold used for the path rating was not sufficient to separate between the object hypotheses. With the help of the similarity rating for the current percept w_0, 95% of presented objects could be assigned to the correct object hypothesis. The remaining 5% of tries failed because of controlled movements close to singularities which can be avoided by restricting movements to be far from singularities thus revealing enough object views.

C. Performance

All experiments were accomplished using an Intel Centrino Duo 2.0GHz notebook. The interpretation module achieved a cycle time of about 3-5Hz. As in most vision applications, feature extraction and comparison is the most time consuming task. Compared to a brute-force algorithm, where in each iteration the 100 nodes of the complete inner model is compared with the current view, the hypothesis generation reduces the necessary feature comparison. Figure 6(c) illustrates the number of comparisons between different features required in relation to the number of iterations of the algorithm. Initially the complete neighbourhood of all candidate views has to be examined. Once path hypotheses develop, only the neighbouring nodes of the last path element have to be examined. Since the pose converges towards the same controlled path many hypothesis paths have the same last nodes and examine the same neighbours. This helps reducing the required feature comparisons to about 14 when the hypothesis have converged.

VI. Conclusion

In this work an active vision approach for object separation and pose approximation is presented. The experimental results show that the proposed approach allows to reliably separate between multiple object hypotheses. Also it has been shown that the transformation from the path hypotheses to the controlled path gives a good approximation of the object pose. Both results, pose and object correspondence, are achieved within the same active vision framework only on basis of the coherence between controlled movement and percept.

It has been shown that coarse features are sufficient to achieve a good performance. With the use of active vision methods, the inner model of objects could be kept compact. Each object can be described by only 32000 feature values. This reveals the benefit that can be obtained by using active methods in solving visual tasks.

Acknowledgment

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

REFERENCES

[1] J. Aloimonos, I. Weiss, and A. Bandopadhay, "Active vision," *International Journal on Computer Vision*, pp. 333–356, 1987.

[2] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 996–1006, 1988.

[3] D. H. Ballard, "Animate vision," *Artif. Intell.*, vol. 48, no. 1, pp. 57–86, 1991.

[4] P. Fitzpatrick and G. Metta, "Grounding vision through experimental manipulation," *Royal Society of London Philosophical Transactions Series A*, vol. 361, pp. 2165–2185, Oct. 2003.

[5] D. Omrcen, A. Ude, K. Welke, T. Asfour, and R. Dillmann, "Sensorimotor processes for learning object representations," in *Proceedings of the IEEE/RAS Int. Conf. on Humanoid Robots (Humanoids 07)*, 2007, (submitted to).

[6] M. Tarr, P. Williams, W. Hayward, and I. Gauthier, "Three-dimensional object recognition is viewpoint dependent," *Nature Neuroscience*, pp. 275–277, 1998.

[7] N. Logothetis, J. Pauls, H. Bülthoff, and T. Poggio, "View-dependent object recognition by monkeys," *Current Biology*, vol. 4, pp. 401–414, 1994.

[8] H. Bülthoff and S. Edelmann, "Psychophysical support for a two-dimensional view interpolation theory of object recognition," in *Proceedings of the National Academy of Sciences*, vol. 89, 1992, pp. 60–64.

[9] N. Logothetis, J. Pauls, and T. Poggio, "Shape representation in the inferior temporal cortex of monkeys," Current Biology 5, pp. 552–563, 1995.

[10] C. M. Cyr and B. B. Kimia, "A similarity-based aspect-graph approach to 3d object recognition," *Int. J. Comput. Vision*, vol. 57, no. 1, pp. 5–22, 2004.

[11] H. Yamauchi, W. Saleem, S. Yoshizawa, Z. Karni, A. Belyaev, and H.-P. Seidel, "Towards stable and salient multi-view representation of 3d shapes," in *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*. Washington, DC, USA: IEEE Computer Society, 2006, p. 40.

[12] K. Welke, E. Oztop, G. Cheng, and R. Dillmann, "Exploiting similarities for robot perception," in *Proceedings of the IEEE Int. Conf. on Intelligent Robot Systems (IROS 2007)*, 2007, (accepted to).

[13] G. Peters, "Efficient pose estimation using view-based object representations," *Mach. Vision Appl.*, vol. 16, no. 1, pp. 59–63, 2004.

[14] N. Krüger and F. Wörgötter, "Multi-modal primitives as functional models of hyper-columns and their use for contextual integration," in *BVAI*, 2005, pp. 157–166.

[15] T. Cooke, S. Kannengiesser, C. Wallraven, and H. H. Bülthoff, "Object feature validation using visual and haptic similarity ratings," *ACM Transactions on Applied Perception*, vol. 5, pp. 1–23, 2006.

[16] C. Beltrán-González and G. Sandini, "Visual attention priming based on crossmodal expectations," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS 2005)*, 2005.

[17] R. Atkinson and R. Shiffrin, "Human memory: A proposed system and its control processes," *The psychology of learning and motivation*, vol. 8, 1968.

[18] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *Proceedings of the IEEE/RSJ International Conference Intelligent Robots and Systems*, 2005.

Exploiting Similarities for Robot Perception

Kai Welke* , Erhan Oztop†‡ , Gordon Cheng†‡ and Rüdiger Dillmann*
*University of Karlsruhe (TH), IAIM, Institute of Computer Science and Engineering (CSE)
P.O. Box 6980, 76128 Karlsruhe, Germany
†JST, ICORP, Computational Brain Project
4-1-8 Honcho, Kawaguchi, Saitama, Japan
‡ATR Computational Neuroscience Lab., Dept. of Humanoid Robotics and Computational Neuroscience
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan

***Abstract*— A cognitive robot system has to acquire and efficiently store vast knowledge about the world it operates in. To cope with every day tasks, a robot needs to learn, classify and recognize a manifold of different objects. Our work focuses on an object representation scheme that allows storing perceived objects in a compact way. This will enable the system to store extensive information about the world and will ease complex recognition tasks. The human visual system deploys several mechanisms to reduce the amount of information. Our goal is to develop an artificial system that mimics these mechanisms to create representations that can be used in cognitive tasks. In particular, in this paper we will present an approach that exploits similarities among different views of objects. The proposed representation scheme allows for reduction of storage required for the representation of objects and preserves the information about the similarity among objects. This is achieved by selecting 'important views' of objects, depending on their stability. Furthermore, by extending the same approach to multiple objects, we are able to exploit similarities between objects to find a common representation and to further reduce the storage requirements.**

I. Introduction

The main focus of our work is to develop an object representation and learning scheme, that is suitable for learning in humanoid robots, e.g. via action-perception coupling, much the same way as humans learn about objects in their environment. To achieve the ability of recognizing objects from all viewing directions, we introduced a learning and representation scheme that allows generalizing to specific views of objects [1]. We have shown that, given locations of important views, the objects can be represented in a depth rotational invariant manner, with a reduced amount of views stored as representations. However, in the former work the important views were selected manually.

In this paper, we introduce a solution on how to select specific important views of objects automatically. Our approach is driven by the observation that there are specific views of an object, that allow recognizing a wide range of rotational variations of the object. Such views are often referred to as *stable* views [2]. With these stable views of an object, its appearance can be described using a minimal set of views.

The robot perceives the world in different modalities, depending on the sensors and feature extraction methods used. In real world scenarios, the robot will face objects, which are similar in at least one modality and are only separable by combining different modalities. Furthermore, learning of objects from all possible viewing directions will reveal even more similarities between views of different objects. In our approach, we identify views that are shared between objects. Similar views can be subsumed and stored only once. In such cases we do not want to recognize objects in the modality, in which the similarities exist, but rather aim at a representation that preserves the information, which objects are candidates for the specific view and modality. The ability to discriminate such objects has to be achieved by combining multiple modalities. The shared view of objects in one modality can then be used to restrict the possibilities in other modalities to only a few objects.

Our approach follows a global appearance-based representation scheme of objects. In appearance-based vision systems, objects are represented with multiple retinal projections of object views. In contrast, model-based representations need more structural information, like full 3D models, which are hard to aquire during online learning [3]. Furthermore, we use global object descriptors to identify important views of the object. The majority of recent work on object recognition uses local features, which describe important locations in the object's appearance, considering measures for texturedness or cornerness. These systems perform well in real environments, are able to handle occlusion, and usually offer invariances to at least shift and rotation in the camera plane [4] [5]. Recognition of all rotations of an object with local features is possible, but impractical in terms of efficiency due to the amount of stored local feature representations.

The selection of important views of an object has strong correspondence to the notion of canonical views used in psychology [6]. In the past, different criterias that define canonical views have been introduced. Blanz et al. give a good overview of different criteria [7]. As our aim is to implement an object recognition system based on our representation and learning scheme, we are mainly interested in the *goodness for recognition* criteria. More precisely, we identify views of the objects, that are stable for at least small transformations.

While the work on canonical views copes largely with one outstanding view of the object, a representation scheme which is used for recognition has to rely on multiple important views of the object, which together should cover the

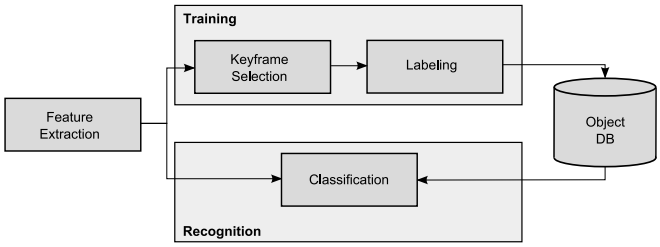


Fig. 1. System structure of the proposed learning and recognition system.

complete appearance.

Hall et al. presented an approach to extract multiple important views of an object by identifying the most unique views of the object [8]. The identification of unique views is suitable, if the objects are to be visualized or if the resulting views are used only to discriminate objects. The approach did not take into account the similarity of views. Moreover, the resulting views did not capture similarities among different objects.

Yamauchi et al. introduced an approach for the identification of important views which combines the saliency of views and the stability criterion [2]. They proposed an approach based on spherical graphs which reflect all available viewing directions of the object. Important views were identified using Zernike Moments [9] to measure the similarity between neighbored views in the spherical graph. In their work, Yamauchi et al. did not take into account similarities among different objects. Each object had its own set of keyframes regardless of the appearance of other objects. Furthermore the number of extracted views per object had to be predefined.

In the following, we present an approach that can be applied to the output of different feature extraction methods. Our approach will identify stable views for the objects considering the Euclidian distance of the output from the used extraction method. In the following we will referr to these stable views as *keyframes*. Furthermore, our method exploits similarities among different objects. The number of keyframes per object does not have to be predefined. Rather, the accuracy can be defined with an overall maximum error, thus the system generates a different number of keyframes per object, depending on the object's appearance.

II. System description

A. Overview

Figure 1 gives a schematic overview of the system structure used throughout this paper. The system can be divided into a training part and a recognition part. In the application on a robot system, both parts have to be executed simultanously to allow the acquisition of new objects during interaction with the environment. The following sections will primarily focus on the training part, since the recognition part needs to rely on more than one modality as explained later (subsection II-E).

As mentioned earlier, the presented approach does not depend on a certain feature extraction method. The extraction method used should fulfill the following requirements:

- The extraction method has to capture the global appearance of an object.
- The extraction method should represent each view invariant to rotations in the viewing plane.
- The extracted feature vectors should be of reasonable size to allow fast extraction of keyframes.

For the experiments in this paper, we use color cooccurrence histograms (CCHs) to extract descriptors of the global appearance of views. The extraction of CCHs will be explained in subsection II-B.

During keyframe selection, significant views of the objects are identified by clustering the feature space into classes containing similar views. Each class is identified with its centroid, which is referred to as a keyframe.

Objects are assigned to keyframes in the labeling step. Each keyframe will be associated with all objects that have views in the corresponding class. Furthermore, we define the activation of a keyframe as the number of views an object participates with in the corresponding class. The keyframes are stored in the object database, together with the labels and activations determined in the labeling phase.

During recognition, the extracted features are classified using the stored keyframes from the object database. The classification will output all objects that have views similar to the current percept and the corrsponding activations.

The following subsections will explain the different elements of the system structure in detail.

B. Feature Extraction

Throughout this paper, we will use color cooccurrence histograms (CCHs) for the description of object appearances. CCHs were chosen because they offer some properties which allow the application in real world recognition tasks. For instance CCHs offer a description of the object, which is invariant to the rotation in the viewing plane, when the parameters are chosen accordingly. Furthermore, CCHs offer some robustness towards scaling. Finally, CCHs combine texture information (in terms of information about pairs of neighbored pixels) as well as color information.

Based on work performed by Haralick et al. [10], CCHs were first introduced by Chang et al. [11]. In their work they define an entry in the color coocurrence histogram by the cooccurring colors and their distances in an observed image:

$$CH(c_1, c_2, \Delta x, \Delta y), \tag{1}$$

where c_1 and c_2 describe two colors in RGB space and Δx and Δy describe their distances in terms of pixels in the observed image. To achieve a rotation invariant description, only the absolute distance of the two colors is used in their approach:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{2}$$

The cooccurrence histogram is derived by counting all occurrences of entries CH in the observed images.

In our implementation only cooccurences with a distance $d < 1.5$ are observed. This restricts the cooccurences to

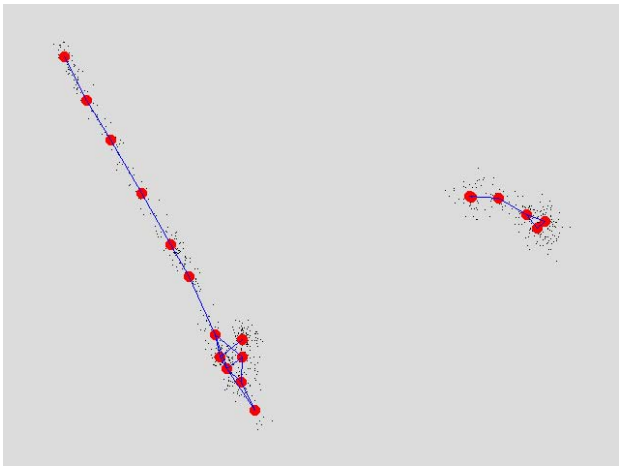


Fig. 2. Resulting growing neural gas network after 2000 iterations. 10 objects with 72 views each were used as input. The features and node positions were projected into 2D eigenspace.

neighboring pixels. Furthermore, the red and green color channels (I_r, I_g) and the gradient magnitude of both color channels $(\nabla I_r, \nabla I_g)$ are used as histogram dimensions. The choice of these image descriptors is motivated by the previous works of Ekvall et al. [12]. They showed that with the combination of intensity and gradient descriptor calculated on the basis of the red and green channels good recognition results could be achieved. The cooccurrences in each channel are considered separately. Each channel is quantized to 80 clusters in a preprocessing step. This results in a feature vector of 320 dimensions, which is still of reasonable size.

C. Keyframe Selection

The identification of similar views in the set of CCH features can be achieved by clustering the feature space into similar classes. Since the keyframe selection process will run autonomously, an unsupervised clustering method is required for our approach. Furthermore, the applied method should select the number of generated clusters dependent on the distribution of the input data rather than on a prespecified number of keyframes. One algorithm that fulfills these requirements is the Growing Neural Gas algorithm (GNG) which was first introduced by Fritzke [13]. The GNG is a self organizing map, which grows in the process of training according to the distribution of the input data. Thus the GNG algorithm creates a topological map which represents the distribution of the training data.

The GNG algorithm combines the Competitive Hebbian Learning and the Neural Gas method proposed by Martinez et al. [14] with an incremental learning approach. GNG thus overcomes the problem of prespecifying the number of nodes that is required to reach a certain goal. Heinke et al. [15] provided a comparison of different incremental neural network algorithms. Their comparison comprises Growing Cell Structures (GCS), Fuzzy Artmap (FAM), and Growing Neural Gas (GNG). As benchmark the performance of the multi-layered perceptron (MLP) was used. The GNG algorithm outperforms FAM on nearly all datasets and generates less nodes then GCS with similar performance for most datasets.

In the following, a brief introduction to the GNG algorithm is given to ease the understanding of the choice of certain parameters and the termination criterion. For a more detailed description of the algorithm the reader is referred to [13]. The network consists of the following components:

- A set of nodes N, each node $n \in N$ has an associated position vector w_n.
- A set of edges E, each edge $c \in E$ connects pairs of nodes and has an associated age.

The algorithm can be described with the following steps:

1) Create two nodes n_1 and n_2 with random positions w_{n_1} and w_{n_2}.
2) Select one feature f from the training set randomly.
3) Identify the nearest and second nearest nodes n_a and n_b to the feature f.
4) Increment the age of all edges starting from n_a.
5) Accumulate the error of node n_a by the squared distance of node position w_{n_a} and input signal f:
$$\Delta e_{n_a} = ||w_{n_a} - f||^2$$
6) Move the nearest node n_a and the second nearest node n_b towards the input signal f using the learning rates sp_a and sp_b:
$$\Delta w_{n_a} = sp_a(f - w_{n_a})$$
$$\Delta w_{n_b} = sp_b(f - w_{n_b})$$
7) Reset the age of the edge from the nearest to second nearest node c_{n_a,n_b} to zero. If no edge exists, create a new edge.
8) Remove edges with an age larger than a_{max}.
9) If the accumulated error e_{n_e} of one node n_e exceeds the maximum error e_{max} insert a new node in the following way:
   - Identify the node connected to n_e with the maximum error n_m.
   - Insert a new node n_c halfway between the two nodes n_e and n_m:
   $$w_{n_c} = \frac{(w_{n_e} + w_{n_m})}{2}$$
   - Insert edges c_{n_c,n_e} and c_{n_c,n_m} and remove the edge c_{n_e,n_m}.
   - Set the accumulated error e_{n_c} of the new node to the mean error of the nodes n_e and n_m.
   - Decrease the accumulated error e_{n_e} and e_{n_m} by multiplication with a constant factor $\alpha < 1$.
10) Decrease all error variables by multiplication with a constant $\gamma < 1$:
$$e'_n = \gamma e_n$$
11) Check termination criterion. If not matched restart with step 2.

Depending on the termination criterion and the parameters used for training, the GNG algorithm will produce a topological map of the input data with respect to the distribution

of the input data. Figure 2 shows an example outcome of the GNG clustering for 720 CCH features, which describe the rotations of 10 objects.

The parameters used for training the GNG were determined empirically. Aim of the parameter choice was a balance between stability of the network and fast convergence. Throughout the experiments a maximum edge age $a_{max} = 20$ was used. The learning rates were set to $sp_a = 0.16$ and $sp_b = 0.01$. The factors for the adjustment of the accumulated error in the case of a new node (α) and for each iteration (γ) were set to $\alpha = 0.001$ and $\gamma = 0.995$.

The parameters for the maximum accumulated error per node e_{max} and the termination criterion directly influence the number of nodes created for the input data. The choice of these parameters will be discussed in section III.

Each node from the network represents one cluster in the space of input features and is considered a keyframe.

D. Labeling

The clustering results in a set of nodes $N = (n_1, \ldots, n_r)$. In order to use these nodes for object representation and recognition we have to restore the association of object views with the clusters formed by the nodes. In the following, s objects $W = (F_1, \ldots, F_s)$ each described with t features $F_x = (f_{x,1}, \ldots, f_{x,t})$ are considered.

To associate object labels with nodes all objects x and their features $f_{x,v}$ are traversed. For each node n_i the number of features of the object where the node is the nearest neighbor to the corresponding feature is determined as:

$$b_{i,x} = |\{f_{x,v} : i = argmin_{u \in \{1,\ldots,r\}} ||w_{n_u} - f_{x,v}||^2\}| \quad (3)$$

If $b_{i,x}$ is not zero, the object label x is appended to the list of object labels L_i for node n_i, if not already present:

$$L_i' = (L_i, x) \quad (4)$$

Additionally, the activation $a_{i,x}$ of the node n_i for the object x is calculated by the following equation:

$$a_{i,x} = \frac{b_{i,x}}{\sum_x b_{i,x}} \quad (5)$$

The activation describes how likely a feature which is associated to the node n_i will belong to the object x. If $b_{i,x}$ is non-zero, the activation $a_{i,x}$ is appended to the list of activation A_i of the node:

$$A_i' = (A_i, a_{i,x}) \quad (6)$$

It is guaranteed that for all labels of one object the sum of the corresponding activations is equal one, i.e.:

$$\sum_{x=1}^{s} a_{i,x} = 1 \quad (7)$$

This shows that if the activation for an object for the node equals 1, then the corresponding keyframe describes one object uniquely. The node will only contain one object label in this case.

In the object database, the node positions $w_{n_1}, \cdots, w_{n_r}$ are stored along with the associated labels $L_{n_1}, \cdots, L_{n_r}$ and activations $A_{n_1}, \cdots, A_{n_r}$.

E. Classification

In the classification step, a perceived view of an object in terms of its CCH f is matched with the keyframes stored in the object database. This can be accomplished by identification of the nearest neighbor n_i in the set of keyframes:

$$i = argmin_{u \in \{1,\ldots,r\}} ||w_{n_u} - f||^2 \quad (8)$$

If the label list L_i contains only one label, the corresponding object is found. Otherwise the classification can not be performed in a unique way. The list of labels L_i contains objects that have views similar to the currently perceived view. The corresponding activations A_i describe the probabilities for the individual objects.

In the case of multiple potential candidates for the current view, the feature extraction method used is not sufficient to separate between the objects in this class. In this case other modalities are required to uniquely detect the object corresponding to the perceived view. For this purpose our approach reduces the number of possibilities to similar objects in the modality observed and allows the restriction to only a few objects for the search in other modalities.

III. Parameter Evaluation

For all experiments in this paper, object views from the Amsterdam Library of Object Images (ALOI) [16] are used. The ALOI contains images of objects on black background from 72 distinct viewing angles, which are generated by rotating the object around the vertical axis. We use 10 objects for the evaluation of our approach, which results in 720 CCHs.

As mentioned earlier, the maximum error e_{max} and the termination criterion are crucial for the number of nodes that are generated by the GNG algorithm. In the following, our choice of these parameters is explained.

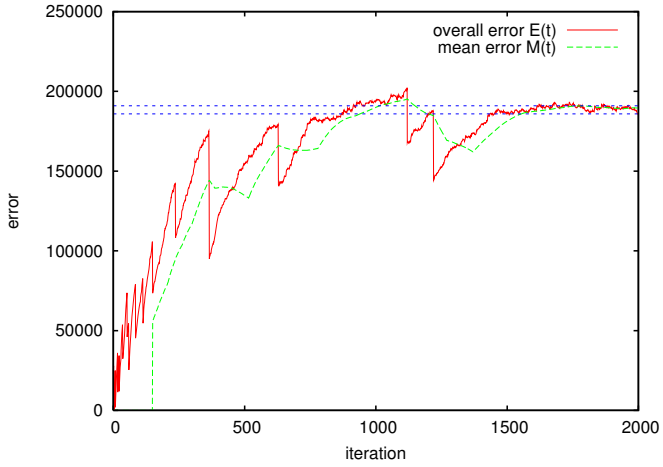
To verify if the network has converged, the overall error $E(t)$ of the network is monitored for each iteration t. The overall error can be determined by summing up the accumulated errors of all nodes:

$$E(t) = \sum_{x=1}^{r} e_{n_x}(t) \quad (9)$$

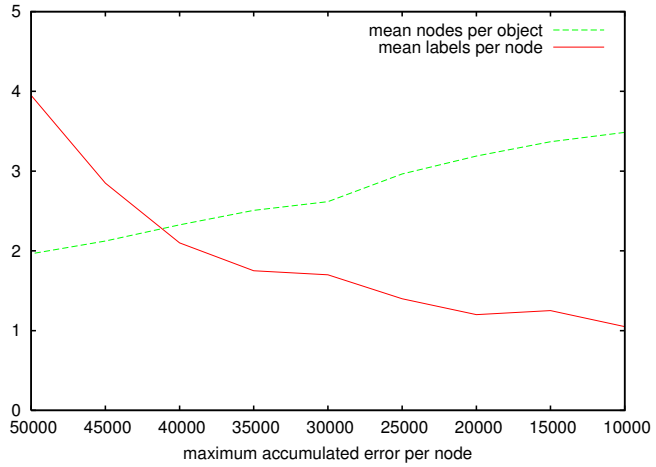
The overall error is smoothed by calculating the mean overall error $M(t)$ over the last 200 iterations. This helps in coping with local peaks in the course of the error over the iterations. To detect the convergence of the network, we check if $M(t)$ is in a defined range r for a minimum number of iterations Δt. The termination criterion $c(t)$ is defined in the following way:

$$c(t) = \begin{cases} 0 & a \leq M(t - t_0) < b; 0 \leq t_0 < \Delta t; b - a < r \\ 1 & \text{otherwise} \end{cases}$$

We choose a range of $r = 5000$ and set the minimum time the mean overall error has to stay in this range to $\Delta t = 500$ iterations. Figure 3(a) shows the development of the overall network error $E(t)$ and the mean error $M(t)$ during one training phase. Every time the accumulated error



(a) Overall network error during one training phase. The upper and lower bounding from the termination criterion are denoted with horizontal lines.



(b) Development of the mean number of labels per node and the mean number of nodes per object dependent on the maximum accummulated error e_{max}.

Fig. 3. Results from the parameter evaluation

of one node exceeds e_{max} the accumulated error is adjusted and a new node is inserted. This results in a diminution of the overall error. On new input data, the accumulated error of both nodes increases again. The overall error of a network containing more nodes can exceed the overall error of a network with less nodes because each single node can accumulate an error of up to e_{max}. The termination criterion terminates the training, if the error stays inside the range r denoted by the two horizontal lines.

In order to determine the maximum node error e_{max} for our experiments, two measures were observed using a range of $e_{max} \in [10000 : 50000]$. First the mean number of labels per node $\overline{L}$ was observed. Furthermore, the mean number of labels per object $\overline{I}$ was observed. In figure 3(b) both measures $\overline{L}$ and $\overline{I}$ are recorded. The graphs show that with a large e_{max}, the mean number of labels per node decreases fast. With decreasing e_{max}, the gradient of $\overline{L}$ reduces. The number of nodes per object $\overline{I}$ grows about linear with decreasing e_{max}. A suitable choice of e_{max} should reduce the number of labels produced per node, since this decreases the uncertainty during recognition. Furthermore, not too many nodes per object should be generated, since the resulting representation has to be compact. For this reason, a maximum accumulated error of $e_{max} = 25000$ was chosen for our experiments. The choice of a maximum accumulated error less than 25000 would result in the generation of more nodes without significantly decreasing the number of labels per node.

IV. Experimental Results

In our experiments, the GNG algorithm proved to be very stable. For the 10 objects with 720 views the number of nodes usually converged to 19. Depending on the sequence of the random selection of input data that was exposed to the network, occasionally 18 or 20 nodes were created.

Fig. 4. Important views of objects as extracted by our approach. Only views corresponding to an activation $a_{i,x}$ above 20% are shown.

A. Keyframe Selection

The GNG network produces clusters of similar views and corresponding nodes with object labels L_i and activations A_i. The node positions do not exactly correspond to object views. In order to visualize important views for the objects, the nearest neighbor from the set of samples for each object x which is in the list of labels L_i is identified. Views are reported only if the activation $a_{i,x}$ from the list of activations A_i is above a given threshold. Thus, only those views are reported that are produced by clusters where the object participates with a significant amount of views.

Figure 4 shows the important views produced by our approach with a threshold of $a_{i,x} > 0.2$. The selected views depend on the used feature extraction method. Using a color descriptor like CCH results in the selection of views which are stable in terms of color.

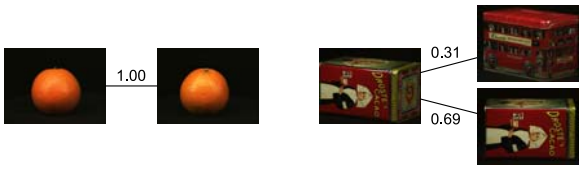


Fig. 5. During recognition, the orange on the left side can be identified uniquely. The can on the right side is associated to a keyframe with two labels. The connections are tagged with the corresponding activations $a_{i,x}$.

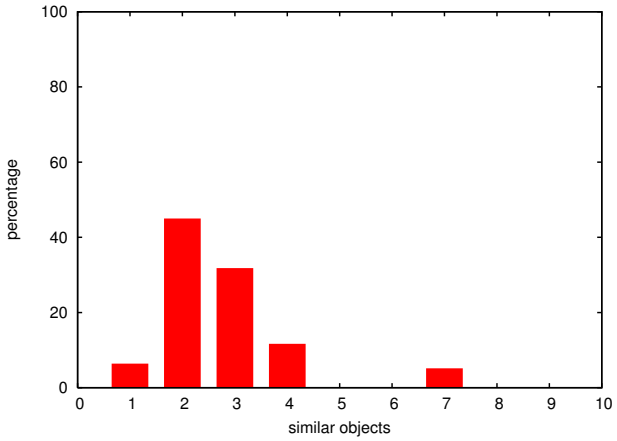


Fig. 6. Percentage of views of all 10 objects in relation to the number of similar objects associated.

B. Recognition

In the recognition phase all object views are associated to the corresponding keyframes. Figure 5 shows two examples of associated views. In the first case, the view was associated to a keyframe which containes only one label. In the second case, the keyframe contained two labels. The keyframes are visualized with the corresponding closest views for each label contained in the label list. The connections are tagged with the activations $a_{i,x}$.

In order to provide a measure on how our approach reduces the uncertainty about the perceived object, we associate all object views to their keyframes. For each view the uncertainty can be expressed with the number of similar objects obtained from the label list. Figure 6 shows the percentage of views in relation to the number of similar objects. 6% of the object views are associated to keyframes which contain only one view and thus can be uniquely identified. 80% of the views are associated to keyframes which contain two or three labels. The remaining views are associated to keyframes with four and more views. The mean number of similar objects per view is about 2.7.

V. Conclusion

The proposed approach allows for the extraction of keyframes on the basis of similarities among objects. For 10 objects with overall 720 views we were able to reduce the number of stored features for one modality to only 19. The experiments show, that with these 19 features, the potential candidates for a perceived object can be reduced to 2.7 on average.

An artificial perception system for a cognitive robot has to rely on more then one modality to identify and classify the manifold of different object types it encounters in real world tasks. The proposed approach will be used in conjunction with a combination of different descriptors for the object appearances. Despite the mentioned CCHs we plan to apply the same approach to other feature extraction methods eg. Zernike Moments. If chosen accordingly, the combination of different modalities will allow to identify the perceived objects uniquely.

Finally, the system will be implemented on our humanoid robot to ease the acquisition of objects during exploration of the environment.

Acknowledgment

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

[1] K. Welke, E. Oztop, A. Ude, R. Dillmann, and G. Cheng, "Learning features for an object recognition system," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Genova, Italy, 2006, pp. 290–295.

[2] H. Yamauchi, W. Saleem, S. Yoshizawa, Z. Karni, and H.-P. Seidel, "Towards stable and salient multi-view representation of 3d shapes," in *Proceedings of the International Conference on Shape Modeling and Applications*, 2006.

[3] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua, "Fully automated and stable registration for augmented reality applications," in *Proceedings of the Second IEEE and ACM Symposium on Mixed and Augmented Reality*, 2003, pp. 93–102.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 1150–1157.

[5] K. Welke, P. Azad, and R. Dillmann, "Fast and robust feature-based recognition of multiple objects," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Genova, Italy, 2006, pp. 264–269.

[6] S. Palmer, E. Rosch, and P. Chase, "Canonical perspective and the perception of objects," in *Attention and Performance IX*, J. Long and A. Baddeley, Eds. Hillsdale, NJ: Erlbaum, 1981.

[7] V. Blanz, M. J. Tarr, H. H. Bülthoff, and T. Vetter, "What object attributes determine canonical views?" Max Planck Institute for Biological Cybernetics, Tech. Rep. Technical Report No. 42, 1996.

[8] P. Hall and M. Owen, "Simple canonical views," in *Proceedings of the british machine vision conference*, 2005, pp. 7–16.

[9] S.-H. Kim, I.-C. Kim, and I.-S. Kweon, "Probabilistic model-based object recognition using local zernike moments," in *IAPR workshop on Machine Vision Applications*, Nara , Japan, Dec. 2002.

[10] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," in *IEEE Transactions on Systems, Man and Cybernetics*, 1973, pp. 610–621.

[11] P. Chang and J. Krumm, "Object recognition with color cooccurrence histogram," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[12] S. Ekvall and D. Kragic, "Receptive field cooccurrence histograms for object detection," in *Proceedings of the IEEE/RSJ International Conference Intelligent Robots and Systems*, 2005.

[13] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems*, vol. 7, 1995.

[14] T. Martinez and K. Schulten, "Topology representing networks," in *Neural Networks*, vol. 7, 1994, pp. 507–522.

[15] D. Heinke and F. H. Hamker, "Comparing neural networks: A benchmark on growing neural gas, growing cell structures, and fuzzy artmap," in *IEEE Transactions on Neural Networks*, vol. 9, 1998.

[16] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.