2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Representation of Manipulation-Relevant Object Properties and Actions for Surprise-Driven Exploration

Susanne Petsch and Darius Burschka

***Abstract*— We propose a framework for the sensor-based estimation of manipulation-relevant object properties and the abstraction of known actions in a learning setup from the observation of humans. The descriptors consists of an object-centric representation of manipulation constraints and a scene-specific action graph. The graph spans between the typical places, where objects are placed. This framework allows to abstract the strongly varying actions of a human operator and to monitor unexpected new actions, that require a modification of the knowledge stored in the system. The usage of an abstract, object-centric structure enables not only the application of knowledge in the same situation, but also the transfer to similar environments. Furthermore, the information can be derived from different sensing modalities.**

The proposed system builds up the representation of manipulation-relevant properties and actions. The properties, which are directly related to the object, are stored in the *Object Container*. The *Functionality Map* links the actions with the typical action areas in the environment. We present experimental results on real human actions, showing the quality of the results, that can be obtained with our system.

I. Motivation

A robot should be able to learn unsupervised through the observation of human actions in its environment. Unfortunately, humans do not follow exact trajectories, while performing repetitive manipulation tasks. The system needs to be able to abstract the manipulation actions, in order to focus only on information, which is necessary to accomplish a manipulation or to cooperate with a human in a given environment. A mismatch between the expectation of the robot as an observer system and a current human action should occur only in situations, in which the change appears to be a result of a changed function or physical property of the object. We will call such a mismatch a *surprise* event in the following. A surprise event triggers the refinement or the modification of the stored information. Important examples are the following: Motion constraints are suddenly changed in the object transport phase (e.g., a cup carried always upright is now tilted arbitrarily); an object is suddenly placed on an unexpected place, e.g., a cup on the floor. These observations are usually an indication, that the physical properties of the object (e.g., the level of the liquid in the object) or their function (not a drinking cup, but a dirty dish) changed. This needs to be considered in the internal representation of the manipulation system.

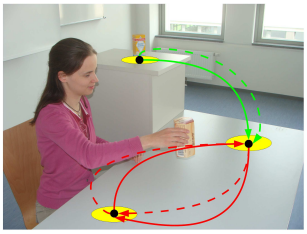

Fig. 1. The system creates an abstract map of possible manipulation actions and goals in the environment.

Our aim is to define a model, that allows to map different physical properties of the object to modifications in the handling properties. The model should efficiently abstract known actions applied to a given object, in order to be able to correctly predict the often strongly varying transport trajectories and goals. This second property of the system allows to reason about changes in the function of a specific object in a given environment. For example, a tool is not used for its specific purpose anymore, but just put away. The representation of the object specific properties and actions needs to be independent of exact Cartesian motion.

A-priori knowledge about an object class is stored in an *Atlas*, introduced in [1]. It contains already known properties of the objects as well as a-priori knowledge about the handling alternatives. Different handling properties might occur for the same object depending on the context. The correct alternative has to be selected based on the observation. This handling property may change over time. Such a modification is triggered by a mismatch between the expectation and the observation of the human action. The abstract knowledge from the Atlas can be mapped into the current context and stored in the *Working Memory*. The Working Memory is the explored representation of the current scene, where the a-priori (Atlas) knowledge gets registered to provide additional information about the structures observed by the system. The Atlas information gets mapped onto representations in the working memory, that describe the handling properties of objects and the observed functional relations between the typical places, where these objects can be found. In this paper, we focus on these representations of the manipulation-relevant properties in a given environment. The requirements on this representation have to be specified and the relevant knowledge has to be extracted from the observation in an appropriate manner. A very important aspect is, that not

This work was supported by the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215821 (GRASP project)

Susanne Petsch and Darius Burschka are with the Machine Vision and Perception Group, Department of Informatics, Technische Universität München. 85748 Garching, Germany
{petsch|burschka}@in.tum.de

only the object itself (e.g., its properties or physical states) is defining the way, how it is manipulated, but also the location at which the manipulation is performed. Certain actions takes usually place at specific locations, which have certain properties. For example, washing the dishes is usually done in the sink and not on a flat table without any water source around. The conclusion is, that we need not only a collection of object properties, but also a map, which links locations in the environment to the specific way how objects are handled at these locations (see Fig. 1). It is important to notice, that we are not interested in the exact registration of the actions to the environment in the sense of navigation, but in an abstract representation of the functionalities in the environment. We are not using any semantic information about the environment. Furthermore, the system does not rely on any linguistic information.

The representation of the knowledge about the human actions is split into an object-centric representation, reflecting the physical properties of an object stored in an *Object Container*, and a *Functionality Map*, representing possible actions related to the environment. While the *Object Container* is linked only to the object, the *Functionality Map* is anchored to the geometric model of the environment. This framework allows us to limit unexpected events (surprise events), that cannot be explained with the current knowledge, to situations, where the physical state or the function of an object changed. The system is insensitive to variations in the execution of the same action. Predictions about the current situations are based on the information stored in Object Container or the Functionality Map. Therefore, mismatches between these predictions and observations occur just at an abstract level. They signal the *right moment* to update the stored information.

It is important to consider, that the robot might face different types of input like, e.g., vision data. This can be useful in a household environment. Further examples include procedural descriptions, which provide knowledge in topological form (e.g., the steps of a surgical procedure). Here, the trajectories of the human motions are neither used for workflow applications nor for planning of actions. We focus on the object-centric constraints and the object's function in the environment. Our system does not rely on a trajectory in a certain representation, like x,y,z-coordinates, or on a certain colored texture of an object. The properties acquired in our system have to be sufficient for a manipulation task. At the same time, they have to be generic and extractable from different sources.

The goal of this paper is the presentation of a system, which provides the manipulation-relevant knowledge in a way, such that the described requirements can be met.

The paper is structured as follows: our approach is presented in the next section. First, the manipulation-relevant object properties and the Functionality Map of the environment are described, followed by the presentation of the knowledge extraction. The results of the experiments with real human actions are described in Section III. We end with conclusions and future work.

A. Related Work

An extensive work exists in the field of imitation learning. In [2], HMMs are used for imitation learning of arm movements in manipulation tasks for humanoid robots, in order to achieve a human-like reproduction of the motions. It is important to point out the difference between our approach and non-object-centric approaches. Such approaches are, for example, imitation learning of motor skills or imitation of movements with Dynamic Movement Primitives (DMPs), which encode the trajectories themselves directly [3], [4]. This paper does not aim to encode the trajectories with, e.g., DMPs or using the model in [5]. Calinon *et al.* use imitation learning, in order to learn control strategies [6]. Approaches related to Reinforcement Learning [7] use the observations of humans as reward [8], [9] in the context of imitation learning. In contrast to theses approaches, our aim goes beyond simple imitation. We want to generalize the observation to cope with variations in repetitive human actions.

The intention in imitation tasks is addressed by Jansen and Belpaeme [10]. They train their agent in a grid with blocks in a computer simulation. In contrast, we deal with more complex, real-world environments and our system needs much less training instances than the one presented in [10]. A real-world example of capturing the user's intention about sequential task constraints is presented in [11]. Their system reasons about the existence of sequential dependencies of operations.

The object's motion has to be analyzed, in order to achieve a further understanding of the object's functionality. The effects on a manipulated object (position, orientation) are taken into account in [12]. The difference to our approach is, that we obtain information about the manipulation properties of the object and, furthermore, to the objects functionality in the environment. In [13], function from motion is analyzed for "primitive motions", which are translations or rotations relative to the main axes of primitive objects. Our approach goes further to more and more general manipulation-relevant object properties. In [14], functional roles of objects, like "pour out", have been explicitly introduced. These roles do not refer to the object's properties, which are directly observable during manipulation.

It should be to pointed out, that we are not interested in the reconstruction or the analysis of the environment, like [15]. We split the information in pure grasp related object-centric information and the information for trajectory planning represented by the Functionality Map. The relative/absolute position of objects to each other have been used for the consideration of the environment in manipulation properties in [15]. In [16], a perceptual space (for the color and shape object properties) and a situation space (for the displacement of the objects in the scene) are introduced. In contrast, the object properties in our paper are beyond the pure visual appearance of the object, since we are interested in the manipulation-relevant object properties (like motion constraints, and known linkages between geometric static occurrences of the object).

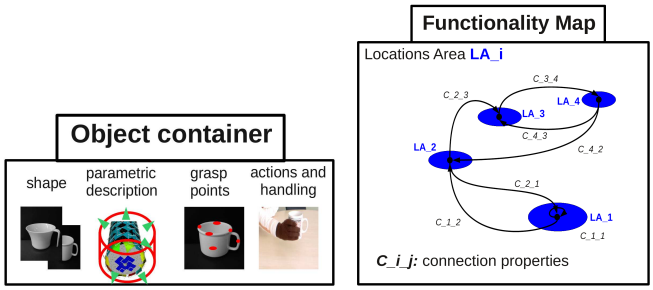

Fig. 2. Object Container and Functionality Map. The Object Container stores the object properties. The Functionality Map is an abstract representation of the manipulation-relevant operating areas in the environment.

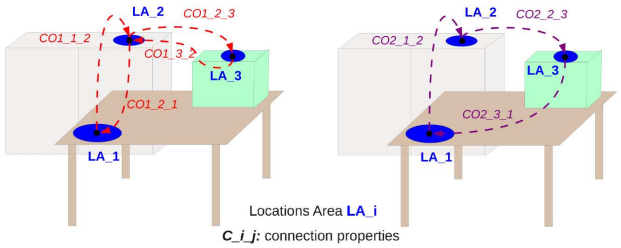

Fig. 3. Functionality Map of the environment for two exemplary objects.

II. Approach

An overview of the system is given in Fig. 2. It illustrates the *Object Container* with the object properties, and the representation of the actions in the *Functionality Map*. The map is represented by a graph, which encodes *Location Areas* and their connections between them.

A. Manipulation-Relevant Object Properties

Since we are interested in a general knowledge of the object properties, we do not want to list the x,y,z-coordinates of the recorded trajectory points, but the abstract handling properties important for grasp planning. The properties, we consider as important, are the variation of orientation, the maximal allowed acceleration, the grasp type allowing a successful grasp with a given manipulator, the mass and the center of mass. Some of these properties are not observable with a pure vision or tracking system. Therefore, the already described information database Atlas [1], which contains the "experience" (*a-priori information*), is used to provide initial information. The other properties need to be extracted, using, for example, a vision system.

The handling properties themselves are constraints, which limit the handling possibilities of the object in a certain situation. Our object-centric representation has the advantage, that the representation of a constraint does not rely on a specific Cartesian position in space. For a specific object, the internal properties, like fragile or liquid content, constraint merely the velocity and acceleration parameters independent of the position in the environment.

B. Functionality Map of the Environment

The Functionality Map provides information about possible trajectories in the environment. The first component of the Functionality Map are the Location Areas. These areas are the locations in the 3D space, where a manipulation sequence can start or end. We define explicitly location *areas* and not single points, since an object is usually placed in a certain area and not on one certain point in space. A Location Area does not necessarily imply, that the manipulated object is standing on a surface. A hand-over step (e.g., changing hands) can also establish a Location Area, which is, therefore, not connected to a surface, but to an area in space.

The connections between different Location Areas are the next component of the Functionality Map. A connection exists, if an action has been performed directly between both areas without visiting another one in between. It is important to consider, that a connection is directed. A connection itself stores the different manipulation properties of the actions, which are performed on this connection. The properties depend on different factors. The first factor are the objects themselves. The other factor are the different grasp alternatives, that can occur for each object. Two exemplary instantiations of a Functionality Map can be seen in Fig. 3.

The properties, which are stored in the Functionality Map, are the following:

- **pushed object vs. lifted object** - An object can be manipulated by lifting or by pushing it. A pushed object needs just to be pushed in the desired direction, whereas lifting an object requires an entire grasp planning (including knowledge about the object's weight).
- **arbitrary movement vs. constrained trajectory** - The trajectory between two Location Areas has either an arbitrary shape or it is the result of a constrained motion. A constrained motion connects the Location Areas in a direct manner, avoiding deviations. In contrast, an arbitrary movement is unconstrained.
- **connection relevance** - The connection relevance shows the probability of a connection property, based on the observed actions.
- **velocity constraints during pick-up** - The three phases defining an action introduced in [1] are used: the pick-up, the transportation and the placement phase. The maximal speed during the pick-up phase is stored as velocity constraint in the Functionality Map. It is an indicator for the difficulty to pick up the object.
- **grasp taxonomy** - The grasp type is mainly important for the pick-up and placement phase of the manipulation and not part of this paper. The grasp taxonomy we consider for the system is summarized in [17].
- **grasp approach vector** - The grasp approach vector is, similarly to the grasp type, mainly important for the pick-up and placement phase of the manipulation and is not part of this paper. The grasp direction is the direction, from which the object is grasped in the object-centric point of view.

The assignment of the properties to the Object Container or the Functionality Map depends on the type of the property. A property, which is related to the function in the environment, is assigned to the Functionality Map. For example, the velocity constraint during the pick-up is part of the Functionality Map, since the possible velocity constraint depends on the environment of the object (e.g., obstacles). In contrast, a property, which is directly related to the object and its state, is a part of the Object Container. An example for such a property is the maximal allowed acceleration for an object in a certain state (e.g., no high acceleration for a filled cup).

C. Acquisition of Knowledge

The presented Object Container and Functionality Map need to be filled with information. For example, a scene can be observed with a system, which provides a 6-DoF trajectory of the manipulated objects.

1) Object Container: The properties for the Object Container, which we want to consider in this paper, are the maximal acceleration value and the variation of observed orientation of the object during the manipulation.

a) Maximal Acceleration: The maximal acceleration value is approximated by the difference of two consecutive velocity samples, which, in turn, are computed as the difference of two consecutive samples.

b) Orientation: The observed orientation is determined from the given 6-DoF trajectory. Just the orientation change relative to the gravitational vector is of interest for the constraints in the manipulation task. The aim is to distinguish a motion with rotation from a motion without tilting. We use Hidden Markov Models (HMMs) [18] for the classification because of their ability of generalization. They are statistical classifiers, which use an observation sequence for the estimation of the underlying state-sequence. Moreover, they take into account knowledge of the past (previous state) in the sequential input. Discrete HMMs with $\lambda = (A, B, \Pi)$ are chosen. They comprise a transition probability matrix A, an observation symbol probability distribution matrix B and an initial state distribution $\mathit{\Pi}$.

First, the preprocessing takes place, until a codebook of the rotation information is built. An overlapping window of 400 ms with a 200 ms overlap (according to [19]) is applied on the sequential input. For each window, the angles between the axes of the current coordinate frame and the coordinate frame at the beginning of the manipulation are measured. Depending on the object and the way of recording its trajectory, different angular variations can occur for different objects. A relative amount of change is needed for each object. Therefore, the angles are normalized for each object with its maximum angle, occurring in all movements of the object. After this preprocessing, the collected data of the rotation information is clustered with the K-means algorithm [20] independent of its time of occurrence. The result is a 64 symbol rotation information codebook.

Then two HMMs (each with 10 states) are built, in order to classify the motions as ones which contain a rotation (λ_r) or as rotation-free ones (λ_{noR}). The transition and emission probabilities for each model are calculated with the maximum likelihood estimation, using the labeled training sequences.

For evaluation, the system receives test sequences, which are preprocessed as described above. The k-nearest-neighbors-method (knn) is used for the assignment to the corresponding symbols in each codebook. To evaluate the classification performance of the trained HMMs, the maximum log likelihood $log\ P(o_{test}|\lambda_i)$ of a given model λ_i is computed for each test sequence with observations o_{test} similar to [21]:

$$\lambda_r^* = \arg\max[\log\ P(o_{test}|\lambda_{noR}), \log\ P(o_{test}|\lambda_r)]\ . \quad (1)$$

2) Functionality Map:

a) Location Areas: The possible Location Areas have to be determined first. Therefore, the available trajectories are split up in single sequences. A sequence starts as soon as the human grasps the object. The end of the sequence is reached, when the hand and the object are not in contact any more. The collected 3D-points of the start and end positions are clustered. The resulting cluster-centers are the centers of the Location Areas. It is possible, that the system detects two Location Areas, which coincide in fact, but appear randomly as two. If these Location Areas are close to each other and have the same connection properties, they can be fused.

b) Connection Properties: The next step is the determination of the connections between the detected Location Areas and the corresponding properties of the connections for each object. The properties, we are using in this paper, are the distinction of a pushed vs. a lifted object, an arbitrary movement vs. a movement with a constrained trajectory, the velocity constraints during the pick-up phase and the connection relevance of a movement property on a certain connection. If possible, the grasp type of the manipulation is determined.

Pushed Object vs. Lifted Object: An object is pushed, if it is in contact with its background during the whole manipulation. In order to check this contact, the distance between the object and the supporting surface is measured along the normal vector of the surface (see [1] for the computation).

Arbitrary Movement vs. Movement with Constrained Trajectory: A Principle Component Analysis PCA (with rescaling) is performed for the distinction of an arbitrary movement and a movement with a constrained trajectory. The PCA is done on a 4.8 s window with a 2.4 s overlap, the resulting principal components are normalized. We check for arbitrary motion, where the motion has no major direction component, but the movement is relatively large in all directions. Therefore, we are especially interested in the third (and smallest) component of PCA, since it shows the variation of motion. If this component has a high amplitude, then all three principal components have relatively high amplitudes. In this case, the movement is large in all directions and it is an arbitrary movement. We define the amplitude of the third component as "high", if it meets one of two requirements. The first one

is a comparison with the main direction of motion (= the first principal component): If the magnitude of the first and the third component are relatively "close" to each other, there is hardly any main direction of the movement and the movement is arbitrary. "Close" means the following: The component of the smallest movement is multiplied with a factor (multiplication factor arbitrary-movement). This factor is the maximal ratio of the first and the third principal component among all arbitrary movements. It determines, how many times the largest movement is maximally allowed to be larger than the smallest movement to classify it still as an arbitrary motion. The second requirement for a "high" amplitude of the third component is occurring, when the this component is higher than a threshold (arbitrary-movement-threshold). The arbitrary-movement-threshold has to be chosen in the magnitude of the third principal component of the arbitrary movements. If all the described criteria are not met, the direction of the smallest motion is not high and the movement is a movement with a constrained trajectory.

Velocity constraints during the pick-up: The pick-up phase is defined manually with 50 samples from the starting position. The speed is computed for two consecutive samples.

Connection relevance: The connection relevance can easily be determined by dividing the number of occurrences of a certain movement property on a connection by the number of all movements on this connection.

Grasp Type: In the current implementation, grasp type is determined by manual labeling.

III. Results

The proposed system is tested on sequences (seq.) of real human actions. First, we test our system on external tracking data (subsection III-A). This data provides directly the 6-DoF trajectory of the tracked markers, which are placed on top of the manipulated objects (obj.). The system is also evaluated on stereo data directly from the manipulation system (subsection III-B).

A. Basic Results on Tracking Data

The tracking data is recorded with a marker-based IR tracking system[1] at 50 Hz. The data is preprocessed first: Each sample, which does not differ from the consecutive one, is deleted. The remaining motion corresponds to an at least minimal movement. Furthermore, the trajectories are smoothed with a 1.4 s moving-average-window, in order to eliminate high-frequency noise, which can especially occur at the beginning of the movement. After this basic preprocessing, the sequences vary between 4.3 s and 17.72 s, the average is 7.45 s (example in Fig. 4, *left*).

The sequences are recorded with four different objects: a milk carton, a spoon, a cup and a vase. The cup is grasped twice: at the handle and at the cylindrical part from the side. This leads to five "different" objects for the test. For each object, there are 18 different actions of a person, shown in Table I. The implementation is done in Matlab (Statistics Toolbox: PCA, HMM, K-mean algorithm. Bioinformatics Toolbox: knn-classification.).

[1]Advanced Realtime Tracking system. Advanced Realtime Tracking GmbH, *url:* http://www.ar-tracking.de/ .

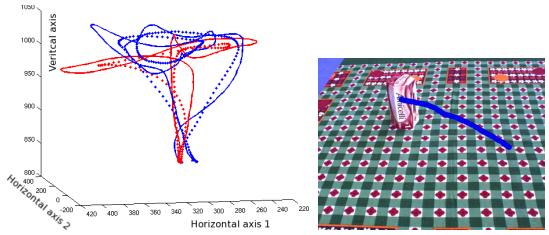

Fig. 4. *Left*: Arbitrary movements of the tracking data (in mm) for the cup in red (without rotation) and in blue (with rotation) (line = original movements, dotted line = result of the basic preprocessing). *Right*: Trajectory of a pushed object (seq. 1, vision data).

TABLE I

Left: Description of the seq. (Tracking data). *Right*: Further description of the four constrained trajectory (seq. 1-4, 5-8, 9-12, 13-16). Positions: *table-start* = pos. on the table, *box* = pos. on a box on the table, *corner* = corner of the table.

Seq	Movement	Rotation	Seq	Start Pos.	End Pos.
1-2	constr.: line		1	table-start	box
3-4	constr.: pushed		2	box	table-start
5-8	constr.: curve		3	table-start	corner
9-12	constr.: line	x	4	corner	table-start
13-16	constr.: curve	x			
17	arbitrary				
18	arbitrary	x			

The parameters and the initial values for the knowledge extraction (see Section II-C) are set as follows. The knn-assignment of a new value to a cluster in the rotation information codebook is done with k=3. The multiplication factor arbitrary-movement for the third component is 15, and the arbitrary-movement-threshold is 0.06. The height difference to the table is measured along its vertical axis for the distinction pushed vs. lifted object. The object is pushed, if its height difference to the table is not changing ($\pm 5\ mm$). The maximal acceleration is computed for a window of 8 ms. Furthermore, the initialization of the cluster for the build-up of the rotation information codebook is set. Otherwise, the results of the clustering are not always deterministic, even though they look mostly very similar. The initialization values are chosen between 0 and 1, since the input values are the normalized changes of the angles.

1) Object Container: A leave-one-out cross validation is made for the rotation classification. The results show, that 42 of 45 of the motions without tilting are correctly labeled, and 30 of 45 motions with titling are correctly classified (see Table II).

The final result of the Object Container can be seen in Table III. Acceleration classes are introduced, in order to make the Object Container more generic. The number of acceleration classes is set to three for illustration. Each class represents an approximately equal sized part of the achieved acceleration values. Table III shows the number of observations per acceleration class and the rotation-classification.

TABLE II

STATISTICAL RESULTS OF THE CLASSIFICATIONS.

Property: (T = Tracking data, V = Vision data)	Accuracy	True positive rate	True negative rate
Rotation (T)	80.0%	66.7%	93.3%
Pushed Object (T)	98.9%	100.0%	87.5%
Arbitrary Traj. (T)	94.4%	80.0%	96.3%
Rotation (V)	77.5%	93.8%	66.7%
Pushed Object (V)	95.0%	87.5%	96.9%
Arbitrary Traj. (V)	80.6%	100.0%	78.6%

TABLE III

RESULT: OBJECT CONTAINER. (R = MOTION WITH ROTATION).

Acceleration class	1		2		3	
Objects Tracking	no R	R	no R	R	no R	R
Milk	3	3	6	3	3	0
Spoon	3	1	3	4	4	3
Cup-handle	2	4	3	8	1	0
Cup	5	2	8	1	2	0
Vase	10	4	4	0	0	0
Objects Vision	no R	R	no R	R	no R	R
Object 1	1	1	0	1	2	5
Object 2	3	0	0	0	4	3
Object 3	2	1	1	1	1	4
Object 4	0	2	2	2	1	3

2) Functionality Map: All three used location areas have been correctly identified. A leave-one-out cross validation is done for the classification of the (non-)arbitrary movements (at first without the distinction of a pushed or lifted object). 8 of 10 arbitrary movements are correctly labeled, and 77 of 80 movements with a constrained trajectory are correctly classified. This shows, that the system performs definitely better than guessing. One has to consider for the true positive rate (see Table II), that there are just 10 arbitrary movements among all 90 sequences, leading to a significant influence of every mislabeled arbitrary movement.

The classification of the pushed vs. the lifted object is successful for all sequences except one spoon-sequence.

The results of the Functionality Maps show, that the system is able to handle some misclassifications, since the correct high connection relevance is gained for all objects except for the cup-handle. The best (= completely correct) results are achieved for the cup and the milk (Fig. 5, *left*). The worst result is the Functionality Map of the cup-handle, since it contains the highest number of misclassifications (two misclassifications) among all Functionality Maps. The misclassified arbitrary movements (red self-loop LA_2) and the misclassified movements with a constrained trajectory (magenta connection from LA_1 to LA_2) are drawn in Fig. 5 (*top, right*). The Functionality Maps of the other two objects have just one misclassification.

B. Results from a Vision System

The vision data is recorded with a Firewire Marlin FO46C camera at 30 Hz and an image size of 640x480 pixel (width x height). The trajectories are acquired as described

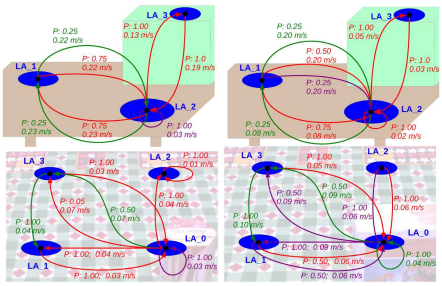

Fig. 5. *Top*: Functionality Maps of the tracking data (*left*: milk, *right*: cup-handle). *Bottom*: Functionality Maps of the vision data (*left*: obj. 1, *right*: obj. 2). Red arrow = constrained trajectory, green arrow = pushed obj., magenta arrow = arbitrary movement. P = probability.

TABLE IV

SEQUENCE PROPERTIES - VISION SYSTEM. THE START AND END POSITIONS ARE THE BOTTOM RIGHT (BR), THE BOTTOM LEFT (BL), THE TOP RIGHT (TR) AND THE TOP LEFT (TL) OF A TABLE.

Seq.:	Movement	Rotation	Start Pos.	End Pos.
1, 11, 21, 31	constr.: push		br	tl
2, 12, 22, 32	constr.: push		tl	bl
3, 13, 23, 33	constr.: curve		bl	br
4, 14, 24, 34	arbitrary		br	br
5, 15, 25, 35	constr.: curve		br	tl
6, 16, 26, 36	constr.: curve		tl	br
7, 17, 27, 37	constr.: curve	x	br	tr
8, 18, 28, 38	constr.: curve	x	tr	br
9, 19, 29, 39	constr.: curve	x	br	bl
10, 20, 30, 40	constr.: curve	x	bl	br

in [1]. The C++ Implementation of Hidden Markov Model by Dekang Lin [2] is (slightly modified) used for the implementation of HMMs. The PCA, the K-means algorithm and the knn-classification are done with OpenCV. Ten sequences are recorded with each of the used four objects. The properties of the sequences are listed in Table IV. An example is shown in Fig. 4 (*right*).

A basic preprocessing is performed (minimal movement $> 0.01/sample$, 140 sample moving-average-window) similarly to the tracking data. Furthermore, the first and last 20 samples are cut of, in order to deal with the arbitrary motions at the beginning and at the end of the sequences. The initialization and threshold values are set as for the experiment with the tracking data, except for the arbitrary-movement-threshold (0.06 for the vision data) and the window for the acceleration-computation (16 ms).

1) Object Container: The occurrence of (non-) rotation is correctly identified for 31 of 40 sequences (Table II). Eight sequences are mislabeled as sequences with rotations. All of them show, that one or both horizontal angles vary during the manipulation. The variations are not as strong as for most of the sequences with rotation, but it is still visible. Table III shows the final result of the Object Container.

2) Functionality Map: The Location Areas themselves are successfully determined. The assignment is successful

[2]Copyright (C) 2003 Dekang Lin, lindek@cs.ualberta.ca, *url:* http://webdocs.cs.ualberta.ca/ lindek/hmm.htm .

for 77 of 80 positions (96.3%). The misclassifications occur for the end positions of sequence 8, 21 and 33. These misclassifications are mainly caused by the z-components (the depth) of the end positions, which are closer to other Location Areas.

As the statistical measures in Table II show, the result of the distinction between a pushed object and an object, which is lifted for the movement, is remarkable. There is just one sequence mislabeled as pushed object, and one sequence mislabeled as raised object. The performance of the classification as arbitrary movement or as movement with a constrained trajectory achieves a true positive rate of 100.0%. Consequently, no arbitrary movement is mislabeled as non-arbitrary movement. Six sequences are misclassified as arbitrary movements instead of movements with constrained trajectory. These movements contain small parts with an arbitrary shape. The kind of grasp is analyzed according to [17]. All used grasps are power grasps with an abducted position of the thumb.

The Functionality Maps of object 1 and 4 have just one wrong assignment of an end location each, everything else is correct (see object 1 in Fig. 5, *left*). The Functionality Map of object 2 suffers mainly from misclassifications as arbitrary movements (see Fig. 5, *right*). One movement of object 3 can be seen a outlier, since its connection property, as well as the assignment of its end location, are wrong. Besides one further misclassified connection property, the Functionality Map of object 3 is correct.

IV. Conclusions and Future Work

The proposed system is developed for abstract representation of manipulation-relevant knowledge about objects. This system aims to monitor object properties and function in a given environment. The experiments on external tracking and vision data show, that the system can derive the knowledge from different sources. The presented system allows an efficient monitoring scheme for the detection of unexpected (surprising) event, that require an update of the information in the internal representation. The proposed framework allows to deal with the strong variations in actions performed by a human operator, reducing the number of false positive surprise events to a minimum. The proposed descriptors, consisting of an Object Container and a Functionality Map spanning typical object locations in a graph, allow a close monitoring of changes in a physical state of the object and its function in the environment.

The results from the vision system show, that a single trajectory is not enough to avoid a misclassification. However, an observation of multiple actions along a given edge of the Functionality Map allows a robust estimation. Our next goal is to focus more on unknown situations and environments. They provide new information to the system.

References

[1] S. Petsch and D. Burschka, "Estimation of Spatio-Temporal Object Properties for Manipulation Tasks from Observation of Humans," in *IEEE International Conference on Robotics and Automation*, Anchorage, USA, 2010, pp. 192–198.

[2] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 183–202, 2008.

[3] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," in *IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 2002, pp. 1398–1403.

[4] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and Generalization of Motor Skills by Learning from Demonstration," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 763–768.

[5] K.Ogawara, J.Takamatsu, K.Kimura, and K.Ikeuchi, "Generation of a task model by intergrating multiple observations of human demonstrations," in *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA '02)*, May 2002, pp. 1545–1550.

[6] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. Billard, "Learning and Reproduction Gestures by Imitation," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 44 – 54, 2010.

[7] R. S. S. und A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[8] D. Verma and R. P. N. Rao, "Imitation Learning Using Graphical Models," in *ECML 2007*, J. N. K. et al., Ed., vol. 4701. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2007, pp. 757–764.

[9] G. Bombini, N. D. Mauro, T. M. A. Basile, S. Ferilli, and F. Esposito, "Relational Learning by Imitation," in *KES-AMSTA 2009*, A. H. et al., Ed., vol. 5559. Lecture Notes in Artificial Intelligence, Springer-Verlag Berlin Heidelberg, 2009, pp. 273–282.

[10] B. Jansen and T. Belpaeme, "A Model for Inferring the Intention in Imitation Tasks," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN'06*, 2006, pp. 238–243.

[11] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zöllner, "Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments ," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 37, no. 2, pp. 322–332, April 2007.

[12] V. Krüger, D. L. Herzog, S. Baby, A. Ude, and D. Kragic, "Learning actions from observations," *IEEE Robotics and Automation Magazine*, pp. 30–43, June 2010.

[13] Z. Duric, J. A. Fayman, and E. Rivlin, "Function from Motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 579–591, June 1996.

[14] R. D. Zöllner and R. Dillmann, "Using multiple probabilistic hypothesis for programming one and two hand manipulation by demonstration," in *IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, USA, 2003, pp. 2926–2931.

[15] M. Mitani, M. Takaya, A. Kojima, and K. Fukunaga, "Environment Recognition Based on Analysis of Human Actions for Mobile Robot ," in *The 18th International Conference on Pattern Recognition (IEEE)*, 2006, pp. 782–786.

[16] A. Chella, H. Dindo, and I. Infantino, "Learning high-level tasks through imitation," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 3648–3654.

[17] T. Feix, R. Pawlik, H.-B. Schmiedmayer, J. Romero, and D. Kragic, "The generation of a comprehensive grasp taxonomy," in *Robotics, Science and Systems Conference: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation, Poster Presentation*, June 2009.

[18] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[19] M. Kawato, "Trajectory formation in arm movements: Minimization principles and procedures," in *Advances in Motor Learning and Control, ser. Human Kinetics, H. N. Zelaznik, Ed. Human Kinetics Publishers, Chanpaign Illinois*, 1996, pp. 225–259.

[20] J. Hartigan and M. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 8, no. 1, pp. 100–108, 1979.

[21] C. Reiley and G. Hager, "Task versus subtask surgical skill evaluation of robotic minimally invasive surgery," in *Medical Image Computing and Computer-Assisted Intervention -MICCAI 2009*, 2009, pp. 435–442.