# Learning the semantics of object-action relations by observation

Eren Erdal Aksoy[1], Alexey Abramov[1], Johannes Dörr[1], Kejun Ning[1], Babette Dellen[1,2] and Florentin Wörgötter[1]

[1]Bernstein Center for Computational Neuroscience, University of Göttingen, III. Physikalisches Institut, Friedrich-Hund Platz 1, 37077 Göttingen, Germany

Emails: eaksoye,abramov,jdoerr,worgott@physik3.gwdg.de

[2]Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028 Barcelona, Spain

Email: bdellen@iri.upc.edu

## Abstract

Recognizing manipulations performed by a human and the transfer and execution of this by a robot is a difficult problem. We address this in the current study by introducing a novel representation of the relations between objects at decisive time points during a manipulation. Thereby, we encode the essential changes in a visual scenery in a condensed way such that a robot can recognize and learn a manipulation without prior object knowledge. To achieve this we continuously track image segments in the video and construct a dynamic graph sequence. Topological transitions of those graphs occur whenever a spatial relation between some segments has changed in a discontinuous way and these moments are stored in a transition matrix called the *semantic event chain (SEC)*. We demonstrate that these time points are highly descriptive for distinguishing different manipulations. Employing simple sub-string search algorithms, semantic event chains can be compared and type-similar manipulations can be recognized with high confidence. As the approach is generic, statistical learning can be used to find the archetypal SEC of a given manipulation class. The performance of the algorithm is demonstrated on a set of real videos showing hands manipulating various objects and performing different actions. In experiments with a robotic arm, we show that the SEC can be learned by observing human manipulations, transferred to a new scenario, and then reproduced by the machine.

**Keywords:** Semantic scene graphs, unsupervised learning, action recognition, object categorization, affordances, Object-Action Complexes (OACs)

## 1 Introduction

It is long known that raw observation and naive copying are insufficient to execute an action by a robot. Execution requires capturing the action's essence and often this problem is discussed in conjunction with imitation learning (Breazeal and Scassellati, 2002).

Humans are – without problems – able to capture "the essence" and recognize the consequences of their own actions as well as those performed by others. While, the mirror-neuron system is suspected to be involved in this feat (Rizzolatti and Craighero, 2004), it is until now completely unknown how newborns gradually learn to recognize and imitate and thereby develop advanced motor skills aiding their cognitive development.

Part of the problem lies in the fact that usually there are many ways to perform a certain manipulation. Movement trajectories may be different and even the order of how to perform it may change to some degree. On the other hand, certain moments during a manipulation will be similar or even identical. For example, during a manual assembly process certain object combinations must occur without which mounting would render nonsense.

This points to the fact that at certain pivotal time-points one needs to comprehend specifically the momentarily existing *relation* between manipulator (hand) and manipulated object as well as the resulting relations (and their changes) between objects and object parts.

This defines the requirements for a potentially useful manipulation representation for artificial agents: It needs to be (1) based on sensory signals and (2) learn-able by observation. At the same time it should encode the (3) relations between objects in an invariant way, which it should do only on certain decisive (4) moments during the manipulation. Furthermore, preferably it should also be (5) human-comprehensible and (6) compatible with model-based knowledge. Aspects (1) and (2) will assure grounding as the process is bootstrapped in a generic way. Aspect (3) would lead to the required categorization property (invariance against irrelevant object-specifics) and aspect (4) to a dramatic data compression as only a few moments need to be stored. The last two aspects (5,6) would allow human access – very practically – for debugging and improving the algorithm(s) but also for being able to better understand and possibly interact with the artificial system and for entering model based knowledge.

To arrive at such a representation is a very difficult problem and commonly one uses models of objects (and hands) and trajectories to encode a manipulation (see next section for literature discussion). These approaches, however, can easily lack grounding because models are almost always given by the designer and not learned by the agent itself. Furthermore, it is so far unknown how to solve the variability problem of manipulations in a model-based way. Or, more plainly: What is the correct model (model-class) for bringing objects and actions together in all those vastly differing manipulations?

In this study it is our goal to introduce the so-called "Semantic Event Chain" (SEC) as a novel, generic encoding scheme for manipulations, which, to a large degree, fulfills the above introduced requirements (grounded, learnable, invariant, compressed, and human-comprehensible). We will show that these SECs can be used to allow an agent by observation to learn distinguishing between different manipulations and to classify parts of the observed scene. Furthermore, we will demonstrate that an agent can decide by self-observation whether or not a manipulation *sequence* was correct. Thus, our algorithms give the machine a basic tool by which it can assess the consequence of a manipulation step directly linking the symbolic planning domain to the signal domain (image) addressing the difficult cause-effect problem of how to observe and verify self-induced changes.

To show this we will use at the sensor front-end computer vision methods for image segmentation and tracking. On the motor side we will employ methods for dynamic trajectory generation and control. Both aspects are not in the core of this paper, which focuses – as discussed above – on the semantics of manipulations, their representation, and learning.

Parts of this study have been published at a conference (Aksoy et al., 2010).

## 2 Related Work

To date, there exists no common framework for manipulation recognition. Different approaches have been presented for vision-based recognition of manipulations (which is the focus of the work presented in this paper), vision-based recognition of human-motion patterns, and non-visual recognition of other types of activities (Modayil et al., 2008; Liao et al., 2005; Hongeng, 2004). The latter will not be discussed any further, because our work focuses on vision. In the following we give short summaries of previous achievements obtained in these areas. We also review previous work on vision-based object recognition, since there exist relations to our work.

## 2.1 Recognition of manipulations

The visual analysis of manipulations, e.g., a hand manipulating an object, represents an important subproblem in vision-based manipulation recognition and is relevant for many vision-based applications such as learning from demonstration, work-flow optimization, and automatic surveillance. However, manipulations are far less understood than for example human motion patterns and only a few solutions have been proposed so far (Vicente et al., 2007; Sridhar et al., 2008; Kjellstrom et al., 2008).

Sridhar et al. (2008) analyzed manipulations in the context of a breakfast scenario, where a hand is manipulating several objects (cups, knifes, bread) in a certain order. The whole image sequence is represented by an activity graph which holds spatiotemporal object interactions. By using statistical generalization, event classes are extracted from the activity graphs. Here, each event class encodes a similar pattern of spatiotemporal relations between corresponding objects, and object categories can be learned by calculating the similarity between object roles at each event class. They demonstrated that objects can be categorized by considering their common roles in manipulations. However, large activity graphs and the difficulty of finding exact graph isomorphisms make this framework expensive and sensitive to noise. Furthermore, object knowledge was provided beforehand. This way the vision problem was (artificially) separated from the manipulation-recognition problem.

Kjellstrom et al. (2008) segmented hand and objects from the video and then defined hand/object features (shape based) and manipulation features, providing a sequence of interrelated manipulations and object features. Semantic manipulation-object dependencies, e.g. drink/glass, are then extracted using conditional random fields (CRFs) and connected hierarchical CRFs. Hand/manipulator and the manipulated object together define the manipulation, and for this reason the recognition process simultaneously involves both hand/manipulator and objects (Vicente et al., 2007; Kjellstrom et al., 2008). In Vicente et al. (2007), manipulations are represented as sequences of motion primitives. Here, five different manipulations of different levels of complexity were investigated. The process is modeled using a combination of discriminative support vector machines and generative Hidden Markov Models (HMMs). HMMs have also been used by Ogawara et al. (2002) to extract primitive of manipulations by learning several HMMs and then to cluster these HMMs such that each cluster represents one primitive. Raamana et al. (2007) recognized simple object manipulations such as pointing, rotating and grasping in a table-top scenario using HMMs and selected the best features for recognition automatically. These works demonstrate that HMMs are a useful tool if the manipulation primitives are hidden in the sensory feature set provided to solve the recognition tasks. Usually this the case if low-level features are used instead of higher-level "object" like entities. However, in our case, manipulations are represented by chained relations between image segments (see Section 3), which directly represent manipulation primitives, and as such they can be compared, grouped, and superimposed without having to assume a hidden model. This holds at least for the manipulation examples considered in this paper.

## 2.2 Recognition of human motion patterns

Recognition of human motion has received much attention in recent years and many contributions exist, but are often unrelated to manipulation recognition (Laptev and Perez, 2007; Niebles et al., 2008; Dee et al., 2009; Hakeem and Shah, 2005; Calinon and Billard, 2004, 2005, 2007; Maurer et al., 2005; Gilbert et al., 2009). Much work has been done by the group of Aude Billard (Calinon and Billard, 2004, 2005, 2007; Maurer et al., 2005) addressing the aspect of gesture recognition. Naturally a strong focus lies here on finding a way to describe complete trajectories and different methods (including PCA, ICA, HMM and Hopfield nets) have been used in different combinations to address this problem and also to deal with the question of sequence learning (Maurer et al., 2005). In Laptev and Perez (2007) spatiotemporal volumes of optical flow are used to classify human motion patterns. In Niebles et al. (2008) human actions are learned in an unsupervised way by using spatiotemporal words that represent space-time interest points. Dee et al. (2009) segment images into regions of similar motion structure and learn pair wise spatial relations between motion regions, roughly corresponding to semantic relations such as "above", "below", and "overlapping". By combining these learned spatial relations with the segmentations learned from data, a compact representation can be provided for each video, representing a motion-based model of the scene, which allows classifying videos containing different kinds of motion patterns, e.g. indoor scenarios with moving people, roads, squares or plazas. In Hakeem and Shah (2005) events involving multiple agents are detected and learned considering temporally correlated sub-events. In Gilbert et al. (2009) simple 2D corners are grouped in both spatial and temporal domains using a hierarchical process at each stage and the most descriptive features are then learned by using data mining. This way, fast and accurate action recognition in video sequences is achieved in real time.

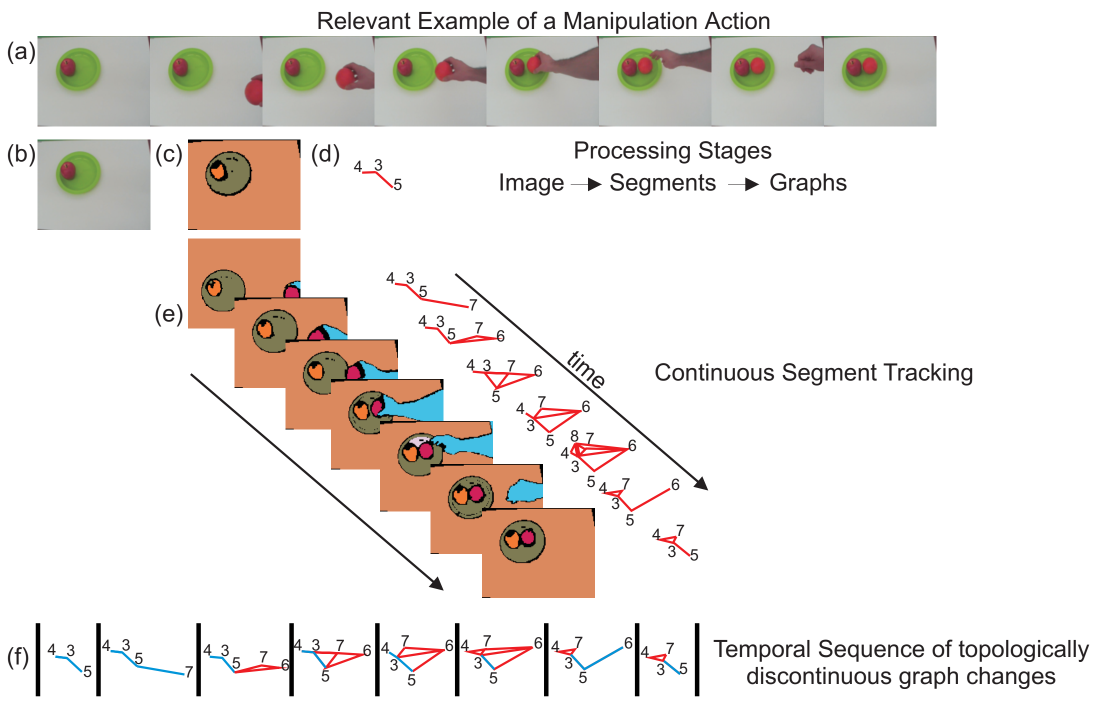
## 2.3 Object recognition and the role of context

Despite progress that has been made in the past decades, the recognition of objects using visual cues remains a highly challenging task and still there exists no vision system reaching human object-recognition capabilities. This is mainly due to the fact that objects take vastly different appearances in images because of the following factors: (i) relative pose of an object to a camera, (ii) lighting variations, and (iii) variance in appearance of objects (size, color, shape) belonging to the same class. Object recognition systems extract certain object-relevant characteristics in images and match them against stored object representation or models, which can be either 2D or 3D. We roughly distinguish between geometry-based, appearance-based, and feature-based approaches. Geometry-based approaches use a geometric description of a 3D object and match its projected shape against the image of the object (Mundy, 2006; Mundy and Zisserman, 1992). This approach however requires that the object can be initially segmented from the image. Appearance-based algorithms use global image patterns to perform recognition (Turk and Pentland, 1991; Murase and Nayar, 1995; Belhumeur and Kriegmant, 1996). For example, Turk and Pentland (1991) projected face images onto a face-specific feature space and used the distance of a projected image to the eigenvectors of the face space for classification.

These methods show invariance to changes in viewpoint and lighting conditions, but are sensitive to occlusions. Feature-based algorithms find local interest points in the image, e.g., SIFT (Lowe, 2004), that have invariant properties with respect to pose, lighting, and affine transformations (Fergus et al., 2003; Nister and Stewenius, 2006; Sivic and Zisserman, 2003). Local feature histograms are then matched against model representations for object recognition. Feature-based methods depend on the quality and number of features that can be extracted from the image, and thus perform best for images containing rich texture.

In the above described "classical" approaches to object recognition, the context in which the object is embedded is usually considered to be of minor importance or even harmful to the recognition procedure, and the problem is sometimes eased by segmenting the object from the background prior to recognition. On the other hand, evidence from visual cognition as well as computer vision suggests that objects appearing in a consistent or familiar background can be more accurately detected and recognized than objects appearing in an inconsistent scenario (Torralba, 2003; Helbig et al., 2010; Hoiem et al., 2008; Oliva and Torralba, 2009). Recently it has been shown in psychophysical experiments that also action context can facilitate human object recognition (Helbig et al., 2010).

This observation is to some extent in agreement with our study, where objects, which can be associated with certain manipulations, are obtained indirectly by classifying and recognizing actions and without using *prior* object knowledge.


Relevant Example of a Manipulation Action
(a)
(b)
(c)
(d)
Processing Stages
Image → Segments → Graphs
(e)
time
Continuous Segment Tracking
(f)
Temporal Sequence of topologically discontinuous graph changes

(g) *Event Chain*

$$
\begin{array}{c}
\rho_{7,5} \\ \rho_{6,5} \\ \rho_{4,3} \\ \rho_{7,3} \\ \rho_{6,3} \\ \rho_{7,4} \\ \rho_{6,7}
\end{array}
\begin{pmatrix}
9 & 1 & 2 & 2 & 0 & 0 & 0 & 0 \\
9 & 9 & 2 & 2 & 2 & 2 & 1 & 9 \\
1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 \\
9 & 0 & 0 & 2 & 2 & 2 & 2 & 2 \\
9 & 9 & 0 & 2 & 2 & 2 & 0 & 9 \\
9 & 0 & 0 & 0 & 0 & 2 & 2 & 2 \\
9 & 9 & 2 & 2 & 2 & 2 & 0 & 9
\end{pmatrix}_{\mathbf{7X8}}
\xrightarrow[\text{time}]{}
$$

Figure 1: *Processing example and semantic event chain representation.* **(a)** *Frames from a movie recorded during a manipulation. All frames* **(b)** *are segmented* **(c)** *by superparamagnetic clustering in a spin-lattice model (Dellen et al., 2009), which also allows for consistent marker-less tracking* **(e)** *of the individual segments. From the image segments, graphs are constructed* **(d)** *where graph nodes represent the segments' centers and graph edges encode whether or not two segments touch each other. Then we encode a manipulation by storing only main graphs between which a topological change has taken place* **(f)***. Such a change happens whenever an edge or a node has been newly formed or has been deleted. This type of representation is then given by the semantic event chain* **(g)***, which is a sequence-table, where each entry encodes the spatial relations between each segment pair $\rho_{i,j}$ counting graph edges (2 means that segments touch (denoted by red edges), 1 means that they overlap (denoted by blue edges), 0 means that there is no edge between two segments, and absence of a previously existing segment yields 9).*

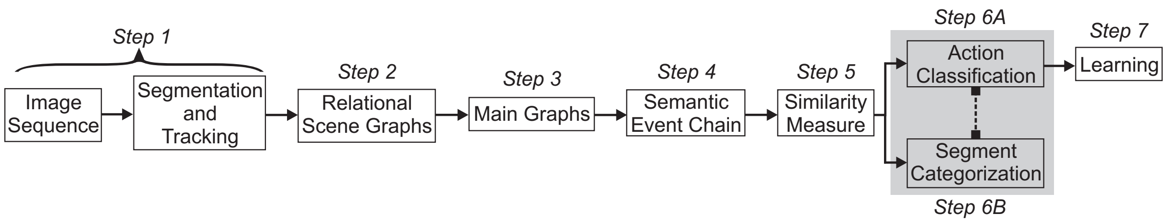


Figure 2: *Block diagram of the algorithm*

## 3 METHODS

### 3.1 Overview of the Algorithm

Before discussing details we provide an overview of the different algorithmic steps (see Fig. 1 and 2).

Fig. 1 shows a processing example of a manipulation resulting in its semantic event chain representation. We first extract all frames from the manipulation movie (Fig. 1 (a)). Frames (Fig. 1 (b)) are then segmented (Fig. 1 (c)) by superparamagnetic clustering in a spin-lattice model (Dellen et al., 2009; Abramov et al., 2010), which allows for consistent marker-less tracking (Fig. 1 (e)) of the individual segments due to spin-linking across images using optic-flow information. The scene is then represented by undirected and un-weighted graphs (Fig. 1 (d)), the nodes and edges of which represent segments and their neighborhood relations, respectively. Graphs can change by continuous distortions (lengthening or shortening of edges) or, more importantly, through discontinuous changes (nodes or edges can appear or disappear). Such a discontinuous change represents a natural breaking point: All graphs before are topologically identical and so are those after the breaking point. Hence, we can apply an exact graph-matching method at each breaking point and extract the corresponding topological main graph. The sequence of these main graphs (Fig. 1 (f)) thus represents all structural changes in the scene. This type of representation is then encoded by the semantic event chain (Fig. 1 (g)), which is a sequence-table, where 0 means that there is no edge between two segments, corresponding to two spatially separated segments, 1 means that one segment overlaps with the other completely, and 2 represents segments that touch each other. A special case exists when segment has disappeared, which will be denoted by 9. Note that the complete image sequence, which has here roughly 100 frames, is represented by an event chain with a size of only $7 \times 8$. The above described steps (1-4) are also presented in Fig. 2, showing the block diagram of the complete algorithm. The following steps (5-7) utilize the SEC to compute similarity values between videos showing manipulations (step 5), to perform action classification (step 6A) and conjointly performed segment categorization (step 6B), and, finally, the learning of archetypal SECs (step 7). In the following, we describe the different algorithmic steps in detail.

### 3.2 Recording, Preprocessing, Segmentation & Tracking (Step 1)

Manipulation movies are recorded in indoor environments with limited context. All movies used in this study can be found at `http://www.nld.ds.mpg.de/~eren/Movies.html` (see Extension 1-2). Typical examples are shown in Fig. 4. We use a stereoscopic camera setup using AVT Marlin F080C CCD firewire cameras and lenses with variable focal length of 2.7-13.5mm (see Fig. 3(a)). Distance to the manipulation scene is about $1.0 - 1.5$ $m$. Images are rectified (see Fig. 3(b-c)), stereo and optic-flow information is extracted by different standard algorithms (Pauwels and Van Hulle, 2008; Sabatini et al., 2007). An example of a resulting sparse phase-based disparity map is shown in Fig. 3(d). For step 1 (Fig. 2), we use an image-segmentation method, developed by us earlier, in which segments are obtained and tracked by a 3D

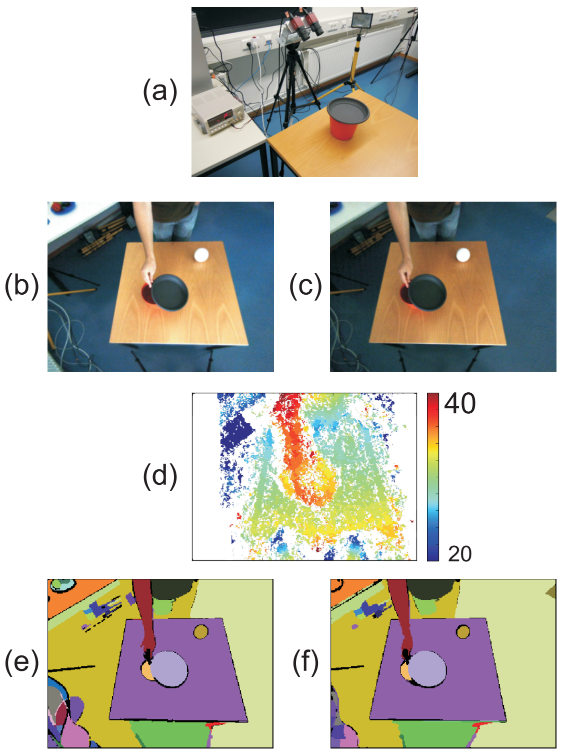


Figure 3: *Schematic of recording and visual preprocessing. (a) Stereo camera setup. (b,c) Original example frames from the left and right image sequences. (d) Sparse phase-based disparity map. (e,f) Extracted segments for the left and right image.*

linking process (see Fig. 3(e-f)). The method has been been described in detail elsewhere (Shylo et al., 2009; Dellen et al., 2009; Dellen and Wörgötter, 2009; Abramov et al., 2010). It is mainly implemented on GPUs and operates close-to real-time at about 23 fps at a resolution of $256 \times 320$ pixels. For reasons of brevity details will be omitted here. The main result from these steps is that we receive consistently tracked image segments, the fate of which can be used to encode a manipulation as described next.

## 3.3 Relational Scene Graphs (Step 2)

Following the extraction of segments (Step 1), we analyze the spatial relations between each segment pair. We denote spatial relations by $\rho_{i,j}$ in which $i$ and $j$ are the segment numbers. Note that spatial relations are symmetric, i.e. $\rho_{i,j} = \rho_{j,i}$.

As mentioned in the algorithmic overview above, we define four relations between segments: *Touching=2*, *Overlapping=1*, *Non-touching=0*, and *Absent=9*, which refers to an image segment that is not observed in the scene. We redefined standard concepts used in the field of topology (e.g. hole, neighbor, etc.) on purpose to make the terminology more appropriate in the context of manipulation recognition. Terms such as overlapping and touching are directly referring to primitive manipulations. Whenever necessary, we use 3D-information from our stereo setup to disambiguate perspective effects, which would lead to false relations when using only 2D. Note 3D information could also be used to define additional relations (like “inside”, “above”, etc.). This is currently not done as classification results are already highly satisfactory even without.

Furthermore we note that ultimately additional information must be stored if one wants to use such graphs (or event chains) also for execution of a manipulation. At least one needs to additionally define the required movement trajectories as well as the desired poses of the objects which are brought into contact with each other as the relation of just ”touching” will not be sufficient for - for example - a mounting procedure. The current paper does not address these issues, though, as we are here focusing on manipulation recognition but not on execution.

Given that image segments often have strangely-shaped as well as noisy borders, the correct assignment of these relations is non-trivial and we had to design a fast and efficient special algorithm for this. As this is not in the core of this study, we will provide details only in Appendix 1. This algorithm gives us the required relations.

Once the image sequence has been segmented and spatial relations have been extracted, we represent the scene by undirected and unweighted labeled graphs. The graph nodes are the segment labels and

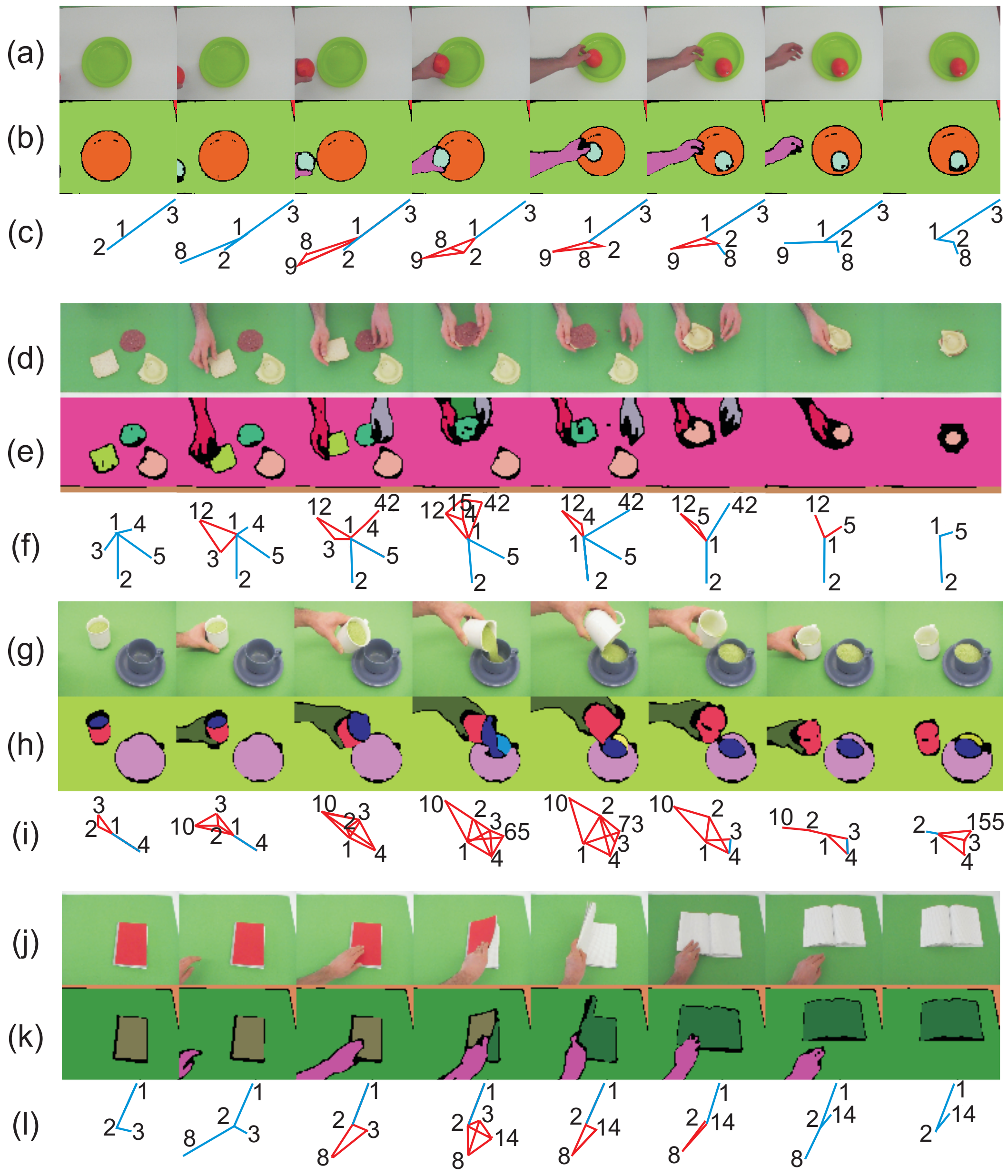


Figure 4: *Four different real action types. (a),(d),(g),(j) Original images, (b),(e),(h),(k) corresponding image segments, and (c),(f),(i),(l) scene graphs from the following manipulations: Moving Object, Making Sandwich, Filling Liquid, and Opening Book. In blue and red are indicated Overlapping and Touching relations.*

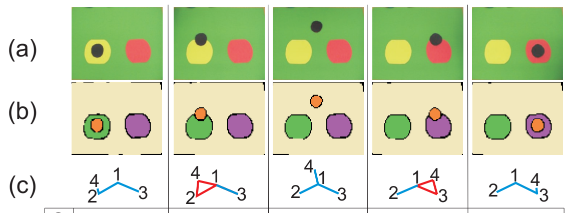
plotted at the center of each segment. Nodes are then connected by an edge if segment relations are either *Touching* or *Overlapping*.

Fig. 4 shows original frames and corresponding segments and their scene graphs from four different real action types: *Moving Object*, *Making Sandwich*, *Filling Liquid*, and *Opening Book*. In the *Moving Object* action a hand is putting an orange on a plate while moving the plate together with the orange (Fig. 4 (a-c)). The *Making Sandwich* action represents a scenario in which two hands are putting pieces of bread, salami, and cheese on top of each other (Fig. 4 (d-f)). In the *Filling Liquid* action a cup is being filled with liquid from another cup (Fig. 4 (g-i)). The *Opening Book* action describes a scenario in which a hand is opening a book (Fig. 4 (j-l)).

## 3.4 Main Graphs (Step 3)

To understand the remainder of the algorithm better, we will now use simple, artificial scenes to describe steps 3 to 6 of Fig. 2. Fig. 5 (a-b) depicts original frames and their corresponding segments of an artificial *Moving Object* action (sample action 1) in which a black round object is moving from a yellow vessel into a red vessel. We will come back to real scenes later.

Scene graphs, such as those depicted in Fig. 4, represent spatial relations between nodes. Unless spatial relations change, the scene graphs remain topologically the same. The only changes in the graph structures are the node positions or the edge lengths depending on the object trajectory and speed. Consequently, any change in the spatial relation between nodes corresponds to a change in the main structure of the scene graphs. Therefore, those changes in the graphs can be employed to define manipulation primitives. Considering this fact, we apply an exact graph-matching method in order to extract the main graphs by computing the eigenvalues and eigenvectors of the adjacency matrices of the graphs (Sumsi, 2008). A change in the eigenvalues or eigenvectors then corresponds to a structural change of the graph. The whole image sequence of the artificial *Moving Object* action has 92 frames, however, after extracting the main graphs, only 5 frames are left, each defining



(d)

| | | | | | |
|---|---|---|---|---|---|
| $\rho_{2,1}$ | 1 | 2 | 1 | 1 | 1 |
| $\rho_{3,1}$ | 1 | 1 | 1 | 2 | 1 |
| $\rho_{4,1}$ | 0 | 2 | 1 | 2 | 0 |
| $\rho_{3,2}$ | 0 | 0 | 0 | 0 | 0 |
| $\rho_{4,2}$ | 1 | 2 | 0 | 0 | 0 |
| $\rho_{4,3}$ | 0 | 0 | 0 | 2 | 1 |

$\xi_o^1$

(e)
$$\begin{array}{c} \rho_{2,1} \\ \rho_{3,1} \\ \rho_{4,1} \\ \rho_{4,2} \\ \rho_{4,3} \end{array} \begin{bmatrix} 12 & 21 & 11 & 11 \\ 11 & 11 & 12 & 21 \\ 02 & 21 & 12 & 20 \\ 12 & 20 & 00 & 00 \\ 00 & 00 & 02 & 21 \end{bmatrix}$$
$\xi_d^1$

(f)
$$\begin{bmatrix} 12 & 21 & & \\ 12 & 21 & & \\ 02 & 21 & 12 & 20 \\ 12 & 20 & & \\ 02 & 21 & & \end{bmatrix}$$
$\xi_c^1$

Figure 5: *Simple example of the Moving Object action (sample action 1). (a) Original images. (b) Corresponding image segments. (c) Semantic scene graphs. In blue and red are indicated Overlapping and Touching relations. (d) Original semantic event chain ($\xi_o^1$). (e) Derivative of the semantic event chain ($\xi_d^1$). (f) Compressed semantic event chain ($\xi_c^1$).*

a single manipulation primitive (Fig. 5 (c)).

## 3.5 Event Chains (Step 4)

All existing spatial relations in the main graphs are saved in the form of a table where the rows represent spatial relations between each pair of nodes. The maximum total number of spatial relations, hence the maximum total number of rows, is defined as

$$\rho_{\text{total}} = n(n-1)/2 \quad , \tag{1}$$

where $n$ is the total number of segments. For the sample *Moving Object* action we have $n = 4$ (yellow and red vessels, a black moving object, and a green background) and therefore $\rho_{\text{total}} = 6$. Those relations are $\rho_{2,1}$, $\rho_{3,1}$, $\rho_{4,1}$, $\rho_{3,2}$, $\rho_{4,2}$, and $\rho_{4,3}$.

Since any change in the spatial relations represents an event that defines an action, we refer to this table as *original semantic event chain* ($\xi_o$). Fig. 5 (d) shows it for the artificial action explained above.

It is now important to understand that these tables contain spatial-relational information (rows) as well as temporal information in the form of a sequence of time-points (sequence of columns) when a certain change has happened. To compare two manipulations with each other, spatial and temporal aspects are being analyzed in two steps by different sub-string search algorithms.

To achieve this, we first perform two data-compression steps. In general, it suffices to only encode the transitions from one state (one column) in the original chain ($\xi_o$) to another (next column). Therefore, we can perform a derivative-like operation on $\xi_o$ and represent the result by $\xi_d$ to simplify the chains.

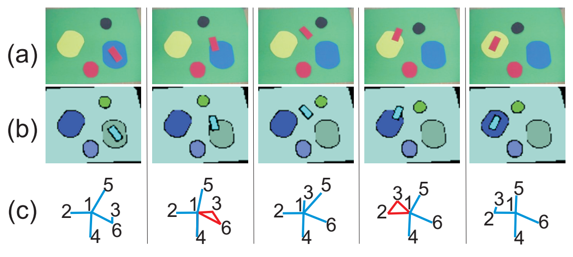
For this we scan each row of $\xi_o$ from left to right and substitute "changes" by combining their numerical values into a two-digit number. For example a change from Overlapping to Touching, hence from 1 to 2, will be now encoded by 12. When nothing has changed a double digit, like 11, occurs. Rows where nothing ever happens (e.g. row $\rho_{3,2}$ in Fig. 5 (d)) are immediately removed. The resulting representation ($\xi_d$) is, thus, a mild, loss-less compression of the original one. It is required for temporal analysis. Fig. 5 (e) shows $\xi_d^1$ for the sample *Moving Object* action.

Then, in a second compression step all double-digits (00, 11, 22, and 99) are removed leading to $\xi_c$. This representation has lost all temporal information and will be used for the spatial-relational analysis. $\xi_c^1$ of the artificial action is given in Fig. 5 (f).

## 3.6 Similarity Measure (Step 5)

Next we will discuss how to calculate the similarity of two actions. Essential this comes down to sub-string search algorithms in the spatial as well as the temporal domain. In the spatial domain we are searching for the correspondences between rows of two compressed event chains to reduce the combinatorics (see Section 3.6.1). Then in the temporal domain the order of columns is examined to get the final recognition result (see Section 3.6.2).

To explain this we created one more sample for the artificial *Moving Object* action. Fig. 6 depicts the main graphs with respective image segments of sample action 2 in which a red rectangular object is moving from a blue vessel into a yellow vessel following a different trajectory with different speed as compared to the first sample. Moreover, the scene contains two more objects which are either stationary (red round object) or moving randomly (black round object). Following the same procedure, the *event chain* $\xi_o^2$ and their compressed versions ($\xi_d^2$ and $\xi_c^2$) for the second sample are calculated and given



$$\begin{array}{c} \rho_{2,1} \\ \rho_{3,1} \\ \rho_{6,1} \\ \rho_{3,2} \\ \rho_{3,6} \end{array}
\begin{bmatrix} 1 & 1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 2 & 0 \\ 1 & 2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 1 \\ 1 & 2 & 0 & 0 & 0 \end{bmatrix}
\begin{bmatrix} 11 & 11 & 12 & 21 \\ 02 & 21 & 12 & 20 \\ 12 & 21 & 11 & 11 \\ 00 & 00 & 02 & 21 \\ 12 & 20 & 00 & 00 \end{bmatrix}
\begin{bmatrix} 12 & 21 & & \\ 02 & 21 & 12 & 20 \\ 12 & 21 & & \\ 02 & 21 & & \\ 12 & 20 & & \end{bmatrix}$$

$\xi_o^2$ (d) $\xi_d^2$ (e) $\xi_c^2$ (f)

Figure 6: *Different version of the simple Moving Object action (sample action 2). (a) Original images. (b) Respective image segments. (c) Semantic scene graphs. In blue and red are indicated Overlapping and Touching relations. (d) Original semantic event chain ($\xi_o^2$). (e) Derivative of the semantic event chain ($\xi_d^2$). (f) Compressed semantic event chain ($\xi_c^2$).*

in Fig. 6 (d-f). Note that even though the second sample contains more objects, the dimensions of the different chains are accidentally the same. This is of no importance as the sub-string search described next does not rely on dimensions, allowing to compare arbitrarily long action sequences.

### 3.6.1 Spatial Similarity Analysis

The goal of this subsection is to provide the first of two subsequent analysis steps, required to obtain a final measure of similarity between two event chains. The first step is based on a spatial analysis comparing the rows of compressed event chains ($\xi_c^1$ and $\xi_c^2$) accounting for a possibly shuffling of rows in different versions of the same manipulations. This way the number of possible relations is reduced before we can finally, in the second step, find the true similarity measures.

Let $\xi_c^1$ and $\xi_c^2$ be the sets of rows for the two manipulations, written as a matrix (e.g. Fig. 5 (f) and 6 (f)):

$$\xi_c^1 = \begin{bmatrix} r^1_{1,1} & r^1_{1,2} & \cdots & \cdots & r^1_{1,\gamma_1^1} & & \\ r^1_{2,1} & r^1_{2,2} & \cdots & r^1_{2,\gamma_2^1} & & & \\ \vdots & \vdots & \ddots & \vdots & & & \\ r^1_{m,1} & r^1_{m,2} & \cdots & \cdots & \cdots & r^1_{m,\gamma_m^1} \end{bmatrix},$$

and

$$\xi_c^2 = \begin{bmatrix} r^2_{1,1} & r^2_{1,2} & \cdots & \cdots & \cdots & \cdots & r^2_{1,\gamma_1^2} \\ r^2_{2,1} & r^2_{2,2} & \cdots & \cdots & r^2_{2,\gamma_2^2} & & \\ \vdots & \vdots & \ddots & \vdots & & & \\ r^2_{k,1} & r^2_{k,2} & \cdots & \cdots & \cdots & r^2_{k,\gamma_k^2} & \end{bmatrix},$$

where $r_{i,j}$ represents a relational *change* between a segment pair

$$r_{i,j} \in \{01, 02, 09, 10, 12, 19, 20, 21, 29, 90, 91, 92\} \ .$$

The lengths of the rows are usually different and given by indices $\gamma$.

Now each row of $\xi_c^1$ is compared with each row of $\xi_c^2$ in order to find the highest similarity. The comparison process searches for equal entries of one row against the other using a standard sub-string search, briefly described next. Assume that we compare the $a^{th}$ row of $\xi_c^1$ with the $b^{th}$ row of $\xi_c^2$. If row $a$ is shorter or of equal length than row $b$ ($\gamma_a^1 \le \gamma_b^2$), the $a^{th}$ row of $\xi_c^1$ is shifted $\gamma_b^2 - \gamma_a^1 + 1$ times to the right. At each shift its entries are compared with the one of the $b^{th}$ row of $\xi_c^2$ and we get as a result set $F_{a,b}$ defined as:

$$F_{a,b} = \{f_t : \ t \in [1, \gamma_b^2 - \gamma_a^1 + 1]\} \ ,$$

$$f_t = \frac{100}{\gamma_b^2} \sum_{i=1}^{\gamma_a^1} \delta_i \quad , \tag{2}$$

where $\gamma_b^2$ is the normalization factor and $i$ is the row index and with

$$\delta_i = \begin{cases} 1 & \text{if } r^1_{a,i} = r^2_{b,i+t-1} \\ 0 & \text{else} \end{cases} \quad , \tag{3}$$

where the set $F_{a,b}$ represents all possible similarities for every shift $t$, given by $f_t$, which holds the normalized percentage of the similarity calculated between the shifted rows.

As usual for sub-string searches, we are only interested in the maximum similarity of every comparison hence we define:

$$M_{a,b} = \max(F_{a,b}),$$

For the case $\gamma_a^1 > \gamma_b^2$, a symmetrical procedure is performed by interchanging all indices of Eqs. (2), (3) above.

Spatial similarity values between all rows of $\xi_c^1$ and $\xi_c^2$ are stored in a matrix $\zeta_{spatial}$ with size $m \times k$ as

$$\zeta_{spatial} = \begin{bmatrix} M_{1,1} & M_{1,2} & \cdots & M_{1,k} \\ M_{2,1} & M_{2,2} & \cdots & M_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m,1} & M_{m,2} & \cdots & M_{m,k} \end{bmatrix},$$

The final similarity value ($\psi_{spatial}$) between the rows of two compressed event chains is calculated by taking the mean value of the highest similarities across rows of $\zeta_{spatial}$ as

$$\psi_{spatial} = \frac{1}{m}\sum_{i=1}^{m} \max_{j}(M_{i,j}), \quad j \in [1, \cdots, k] \quad . \quad (4)$$

The complete similarity matrix ($\zeta_{spatial}$) between the artificial *Moving Object* samples ($\xi_c^1$ and $\xi_c^2$) is given in Table 1. Visual inspection of $\xi_c^1$ (Fig. 5 (f)) and $\xi_c^2$ (Fig. 6 (f)) immediately confirms these similarity values. We find 100% similarity ($\psi_{spatial}$) between both artificial "manipulations". One can see that Eq. 4 can still lead to multiple assignments of permutations with the same maximal similarity. This will be resolved by the temporal similarity measurement stage following below. In more realistic scenes 100% is, of course, often not reached (see Fig. 7 below) and one needs to define a threshold above which one would consider two similarity values as equal.

However, we also observe that there are several 100% matches between rows in these examples. As a consequence, for the second example two permutations $\xi_c^{2,1}$ and $\xi_c^{2,2}$ exist with equal row-matching probability as given in Table 2. Note, for real scenes after thresholding, even more permutations might exist. Hence, the analysis in not yet complete.

### 3.6.2 Temporal Similarity Analysis

In the now following second step we can use the time sequence, encoded in the order of events in the event chains to find the best matching permutation and thereby arrive at the final result. To this end will now use temporal information, hence a derivative like term $\xi_d$, to find the truly matching permutation.

| $\xi_c^1$ \ $\xi_c^2$ | $\rho_{2,1}$ | $\rho_{3,1}$ | $\rho_{6,1}$ | $\rho_{3,2}$ | $\rho_{3,6}$ |
|---|---|---|---|---|---|
| $\rho_{2,1}$ | **100%** | 25% | **100%** | 50% | 50% |
| $\rho_{3,1}$ | **100%** | 25% | **100%** | 50% | 50% |
| $\rho_{4,1}$ | 25% | **100%** | 25% | 50% | 50% |
| $\rho_{4,2}$ | 50% | 50% | 50% | 0% | **100%** |
| $\rho_{4,3}$ | 50% | 50% | 50% | **100%** | 0% |

Table 1: *Similarity table ($\zeta_{spatial}$). Similarity values between the rows of $\xi_c^1$ and $\xi_c^2$ the artificial Moving Object samples.*

$$\begin{array}{lcl|l}
\xi_c^1 & & \xi_c^{2_1} & \qquad\quad \xi_d^{2_1} \\
\rho_{2,1} & \Leftarrow 100\% \Rightarrow & \rho_{2,1} & \\
\rho_{3,1} & \Leftarrow 100\% \Rightarrow & \rho_{6,1} & \\
\rho_{4,1} & \Leftarrow 100\% \Rightarrow & \rho_{3,1} & \\
\rho_{4,2} & \Leftarrow 100\% \Rightarrow & \rho_{3,6} & \\
\rho_{4,3} & \Leftarrow 100\% \Rightarrow & \rho_{3,2} &
\end{array}
\qquad
\begin{bmatrix} 11 & 11 & 12 & 21 \\ 12 & 21 & 11 & 11 \\ 02 & 21 & 12 & 20 \\ 12 & 20 & 00 & 00 \\ 00 & 00 & 02 & 21 \end{bmatrix}$$

(a) Permutation $\xi_d^{2_1}$

$$\begin{array}{lcl|l}
\xi_c^1 & & \xi_c^{2_2} & \qquad\quad \xi_d^{2_2} \\
\rho_{2,1} & \Leftarrow 100\% \Rightarrow & \rho_{6,1} & \\
\rho_{3,1} & \Leftarrow 100\% \Rightarrow & \rho_{2,1} & \\
\rho_{4,1} & \Leftarrow 100\% \Rightarrow & \rho_{3,1} & \\
\rho_{4,2} & \Leftarrow 100\% \Rightarrow & \rho_{3,6} & \\
\rho_{4,3} & \Leftarrow 100\% \Rightarrow & \rho_{3,2} &
\end{array}
\qquad
\begin{bmatrix} 12 & 21 & 11 & 11 \\ 11 & 11 & 12 & 21 \\ 02 & 21 & 12 & 20 \\ 12 & 20 & 00 & 00 \\ 00 & 00 & 02 & 21 \end{bmatrix}$$

(b) Permutation $\xi_d^{2_2}$

Table 2: *Permutations $\xi_d^{2,p}$.*

Thus, we will compare both permutations $\xi_d^{2_p}$, $p = 1, 2$ of the second event chain, shown in Tab. 2 (a-b), with the first one $\xi_d^1$. In such cases where $\xi_d^1$ has more rows than $\xi_d^{2_p}$, hence rows which have no correspondences, $\xi_d^{2_p}$ will be filled with dummy rows that have no possible similarities.

Let $\xi_d^1$ and $\xi_d^{2_p}$ be matrices with the sizes of $q \times u$ and $q \times v$ and assume that $u \leq v$ as

$$\xi_d^1 = \begin{bmatrix} e_{1,1}^1 & e_{1,2}^1 & \cdots & e_{1,u}^1 \\ e_{2,1}^1 & e_{2,2}^1 & \cdots & e_{2,u}^1 \\ \vdots & \vdots & \ddots & \vdots \\ e_{q,1}^1 & e_{q,2}^1 & \cdots & e_{q,u}^1 \end{bmatrix},$$

and

$$\xi_d^{2_p} = \begin{bmatrix} e_{1,1}^{2_p} & e_{1,2}^{2_p} & \cdots & e_{1,v}^{2_p} \\ e_{2,1}^{2_p} & e_{2,2}^{2_p} & \cdots & e_{2,v}^{2_p} \\ \vdots & \vdots & \ddots & \vdots \\ e_{q,1}^{2_p} & e_{q,2}^{2_p} & \cdots & e_{q,v}^{2_p} \end{bmatrix},$$

where $\xi_d^{2_p}$ is a permutation of $\xi_d^2$ and $e_{i,j}^1$ and $e_{i,j}^{2_p}$ represent the possible relational changes between any segment pair

$$e_{i,j} \in \{00, 01, 02, 09, 10, 11, 12, 19, 20, 21, 22, 29, 90, 91, 92, 99\} . \quad (5)$$

Following this, the columns of $\xi_d^1$ and $\xi_d^{2_p}$ are compared. Note that by contrast to rows, columns of event chains will never be shuffled unless they represent different types of actions. Therefore, the column orders of type-similar event chains have to be the same. Assume that we compare the $a^{th}$ and $b^{th}$ columns of $\xi_d^1$ and $\xi_d^{2_p}$, respectively. The procedure is very similar to the one for the spatial analysis.

Since the lengths of the columns $q$ are the same, no shift-operator is required and columns are directly compared index-wise as

$$\theta_{a,b} = \frac{100}{q} \sum_{j=1}^{q} \delta_j \quad , \quad (6)$$

where $j$ is the column index and with

$$\delta_j = \begin{cases} 1 & \text{if } e_{j,a}^1 = e_{j,b}^{2_p} \\ 0 & \text{else} \end{cases} \quad , \quad (7)$$

where $\theta_{a,b}$ holds the normalized percentage of the similarity value calculated between columns.

Similarity values between all columns of $\xi_d^1$ and $\xi_d^{2_p}$ are stored in a matrix $\zeta_{temporal}^p$ with the size of $u \times v$ as

$$\zeta_{temporal}^p = \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,v} \\ \theta_{2,1} & \theta_{2,2} & \cdots & \theta_{2,v} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{u,1} & \theta_{u,2} & \cdots & \theta_{u,v} \end{bmatrix},$$

The final similarity value $\psi_{temporal}^p$ between the columns of two event chains is then calculated by taking the mean value of the highest similarities across rows as

$$\psi_{temporal}^p = \frac{1}{u} \sum_{i=1}^{u} \max_j(\theta_{i,j}), \quad j \in [1, \cdots, v] \quad (8)$$

For each permutation $\xi_d^{2_p}$ given in Tab. 2 a $\psi_{temporal}^p$ value is calculated by using Eqs. (6), (7), and (8), yielding $\psi_{temporal}^1 = 60\%$ and $\psi_{temporal}^2 = 100\%$. Note that we use Longest Common Subsequence (LCS) in order to guarantee that the order of columns is the same. LCS is generally used to find the longest sequence observed in both input sample sequences. Columns of event chains are used as sequences for this task. Since the number of sequences is constant, the problem is solvable in polynomial time by dynamic programming. Consequently, our two sample actions have 100% similarity and permutation $\xi_d^{2_2}$ represents the final row correspondences to the first action. The best matching permutation is further used for categorizing objects as described in sub-section 3.8.

As mentioned above, in real scenes often 100% are not reached and we will call two actions "type-similar" as soon as their final similarity value exceeds a certain threshold. We would also like to point out that the final spatial and temporal similarity values are not necessarily identical. Action classification should use the final temporal similarity values as this measure is more restrictive and therefore provides the final means for classification.

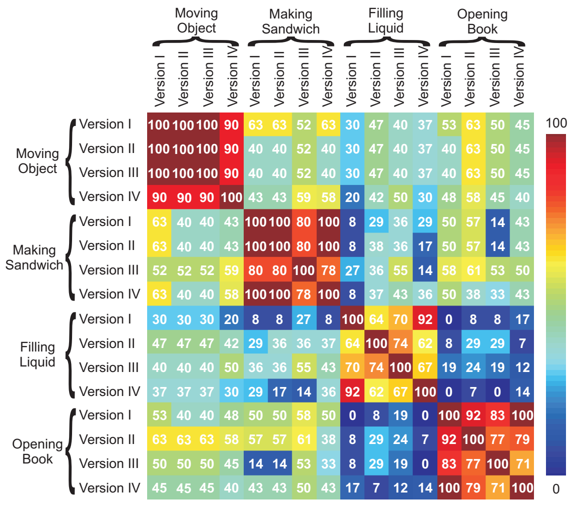
The question arises, why we use a 2-step process as the second step might suffice on its own. In this case however *all* possible permutations would have to be analyzed, which can be very costly (up to $O(n!)$) for big event chains. Particularly, decisive "no-match decisions", which occur for all non-type similar manipulations – hence, quite often – could only be obtained at the end of the complete permutation analysis. Whereas, when performing spatial analysis which has the time complexity of $O(n^3)$, this result is obtained faster. This leads to a substantial algorithmic speed-up and makes the choice of a two-step algorithm useful.

## 3.7 Action Classification (Step 6A)

We applied our framework to four different real action types: *Moving Object*, *Making Sandwich*, *Filling Liquid*, and *Opening Book* (see Fig. 4). For each of these actions, we recorded four movies with highly different trajectories, speeds, hand positions, and object shapes. These examples were introduced to show that really different instantiations of a manipulation will still be recognized as belonging to the same type.

Event chains of each real test data are compared with each other by using the similarity measurement algorithm explained in Step 5. The resulting final maximal similarity values are given in Fig. 7. Each test data has high similarity with the other versions of its type-similar action (see close to diagonal) and almost always, the similarity between type-similar actions is much bigger than the similarity between non-type-similar actions. The minimum similarity value for type-similar actions is measured as 62% between the forth and second versions of *Filling Liquid*, but the similarity between *Filling Liquid* and non-type-similar actions is far less. Setting a threshold at 60% would, across all examples, lead to zero false negatives and to two false positives (opening book version II and moving object version I), which would be confused with other manipulations. This is interesting as opening a book can indeed look very similar to the moving of an object (picking up a book-cover and moving it - say - up and towards you will indeed look very similar from picking up an object and moving it up and towards you). The same is true for the comparison of moving-object with making-sandwich. In the making-sandwich action many subcomponents exist where objects are being moved. Thus, in general the manipulation analysis shown in Fig. 7 corresponds very well to our human understanding of action (sub-)components!

Note, as soon as the complete table has been measured, it is also accessible to unsupervised classification (X-means, (Dan Pelleg, 2000)). We have done this by using correlation values between columns as features and we receive four classes with no outliers. Hence with clustering one gets completely correct classification of all individual manipulations.



| | | Moving Object | | | | Making Sandwich | | | | Filling Liquid | | | | Opening Book | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Version I | Version II | Version III | Version IV | Version I | Version II | Version III | Version IV | Version I | Version II | Version III | Version IV | Version I | Version II | Version III | Version IV |
| Moving Object | Version I | 100 | 100 | 100 | 90 | 63 | 63 | 52 | 63 | 30 | 47 | 40 | 37 | 53 | 63 | 50 | 45 |
| | Version II | 100 | 100 | 100 | 90 | 40 | 40 | 52 | 40 | 30 | 47 | 40 | 37 | 40 | 63 | 50 | 45 |
| | Version III | 100 | 100 | 100 | 90 | 40 | 40 | 52 | 40 | 30 | 47 | 40 | 37 | 40 | 63 | 50 | 45 |
| | Version IV | 90 | 90 | 90 | 100 | 43 | 43 | 59 | 58 | 20 | 42 | 50 | 30 | 48 | 58 | 45 | 40 |
| Making Sandwich | Version I | 63 | 40 | 40 | 43 | 100 | 100 | 80 | 100 | 8 | 29 | 36 | 29 | 50 | 57 | 14 | 43 |
| | Version II | 63 | 40 | 40 | 43 | 100 | 100 | 80 | 100 | 8 | 38 | 36 | 17 | 50 | 57 | 14 | 43 |
| | Version III | 52 | 52 | 52 | 59 | 80 | 80 | 100 | 78 | 27 | 36 | 55 | 14 | 58 | 61 | 53 | 50 |
| | Version IV | 63 | 40 | 40 | 58 | 100 | 100 | 78 | 100 | 8 | 37 | 43 | 36 | 50 | 38 | 33 | 43 |
| Filling Liquid | Version I | 30 | 30 | 30 | 20 | 8 | 8 | 27 | 8 | 100 | 64 | 70 | 92 | 0 | 8 | 8 | 17 |
| | Version II | 47 | 47 | 47 | 42 | 29 | 36 | 36 | 37 | 64 | 100 | 74 | 62 | 8 | 29 | 29 | 7 |
| | Version III | 40 | 40 | 40 | 50 | 36 | 36 | 55 | 43 | 70 | 74 | 100 | 67 | 19 | 24 | 19 | 12 |
| | Version IV | 37 | 37 | 37 | 30 | 29 | 17 | 14 | 36 | 92 | 62 | 67 | 100 | 0 | 7 | 0 | 14 |
| Opening Book | Version I | 53 | 40 | 40 | 48 | 50 | 50 | 58 | 50 | 0 | 8 | 19 | 0 | 100 | 92 | 83 | 100 |
| | Version II | 63 | 63 | 63 | 58 | 57 | 57 | 61 | 38 | 8 | 29 | 24 | 7 | 92 | 100 | 77 | 79 |
| | Version III | 50 | 50 | 50 | 45 | 14 | 14 | 53 | 33 | 8 | 29 | 19 | 0 | 83 | 77 | 100 | 71 |
| | Version IV | 45 | 45 | 45 | 40 | 43 | 43 | 50 | 43 | 17 | 7 | 12 | 14 | 100 | 79 | 71 | 100 |

Figure 7: *Similarity values between event chains of the real test data set.*

## 3.8 Segment Categorization (Step 6B)

The row correspondence, determined by finding the best matching permutation, also implicitly encodes the similarity of the graph *nodes* between the two different examples.

We will explain this using again the two artificial examples (Fig. 5 and Fig. 6). The row similarity values of the second permutation given in Table 2 (b) represent the correspondences between manipulated nodes in $\xi^1$ and $\xi^2$ (now we can drop all other indices as descriptions will not suffer).

The question, which we would like to answer is: *which nodes represent segments that play the same role in type-similar actions.*

This can be achieved in a fully unsupervised way by a simple counting procedure. We first analyze which node number in $\xi^1$ is repeating in conjunction with which node number in $\xi^2$ in Table 2. We start with node number 1 in $\xi^1$, which occurs in relations $\rho_{2,1}, \rho_{3,1}$ and $\rho_{4,1}$. Its corresponding best matching

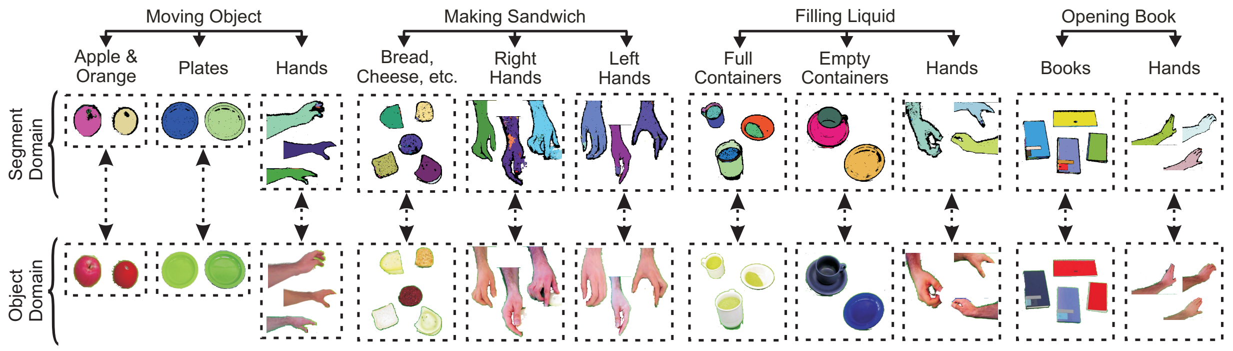


Figure 8: *Segment categorization results. In each action type, the manipulated segments can be classified and grouped based on their action roles. Note, classification happens at the level of segments or segment groups. A link to the object-domain (indicated by the dashed arrows) could be introduced by including explicit object models, but this is not part of this study.*

relations are given by:

$$\begin{array}{lcl} \rho_{2,1} & \Leftarrow 100\% \Rightarrow & \rho_{6,1} \\ \rho_{3,1} & \Leftarrow 100\% \Rightarrow & \rho_{2,1} \\ \rho_{4,1} & \Leftarrow 100\% \Rightarrow & \rho_{3,1} \end{array} \Rightarrow \quad 1 \approx 1 \quad .$$

While node 1 is repeating three times in $\xi^1$ (left side), the same node number 1 in $\xi^2$ (right side) is also repeating three times. However, node numbers 2, 3, and 6 in $\xi^2$ occur only once. Therefore, we conclude that graph nodes 1 in both examples, $\xi^1$ and $\xi^2$, had the same roles. In fact, both graph nodes represent the green background in both actions.

We continue the node relation analysis with node number 2 in $\xi^1$, and obtain

$$\begin{array}{lcl} \rho_{2,1} & \Leftarrow 100\% \Rightarrow & \rho_{6,1} \\ \rho_{4,2} & \Leftarrow 100\% \Rightarrow & \rho_{3,6} \end{array} \Rightarrow \quad 2 \approx 6 \quad .$$

Node number 2 in $\xi^1$ is repeating twice with node number 6 in $\xi^2$. Those graph nodes represent the yellow and blue vessels within which the moving objects are initially placed and from which they then move away.

For the case of node number 3 in $\xi^1$ we obtain

$$\begin{array}{lcl} \rho_{3,1} & \Leftarrow 100\% \Rightarrow & \rho_{2,1} \\ \rho_{4,3} & \Leftarrow 100\% \Rightarrow & \rho_{3,2} \end{array} \Rightarrow \quad 3 \approx 2 \quad .$$

Node number 3 in $\xi^1$ corresponds to node number 2 in $\xi^2$ because both of them are repeating twice. Those graph nodes define the destination vessels for the moving objects.

The last node number 4 in $\xi^1$ is obtained as

$$\begin{array}{lcl} \rho_{4,1} & \Leftarrow 100\% \Rightarrow & \rho_{3,1} \\ \rho_{4,2} & \Leftarrow 100\% \Rightarrow & \rho_{3,6} \\ \rho_{4,3} & \Leftarrow 100\% \Rightarrow & \rho_{3,2} \end{array} \Rightarrow \quad 4 \approx 3 \quad .$$

Node number 4 in $\xi^1$ and node number 3 in $\xi^2$ are both repeating three times. In fact, both graph nodes represent the moving objects, which are the round black object in $\xi^1$ and the rectangular red object in $\xi^2$.

In the case of having the same highest value more than once, e.g. having two times 100% similarity values in the same row of the similarity matrix, segment categorization might lead to ambiguous results, i.e. one segment would correspond to two different segment in the other manipulation. This sort of conflicts can be solved by taking the second highest values in the similarity matrix and calculating the node-similarity again. This way we always achieve unique segment categorization results.

We applied this categorization algorithm to our four different real manipulation scenes (Fig. 4). The results showed that the manipulated segments in each

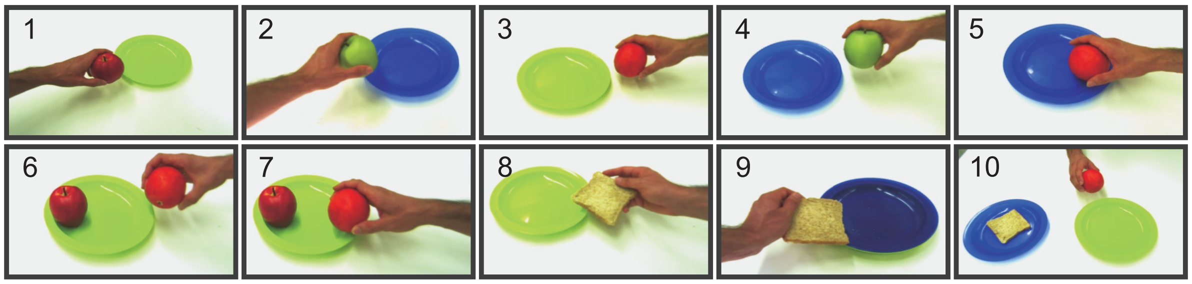


Figure 9: *Sample frames from* 10 *different versions of a "Putting an object on a plate" action. In this action type a hand is appearing in the scene, putting different kinds of objects (e.g. apple, orange, a piece of bread, etc.) on a plate following different trajectories with different speeds, and then leaving the scene.*

action type can be categorized according to their roles in the actions. Fig. 8 illustrates the categorization results, e.g. the *Moving Object* actions include three different segment groups here named by their object-names for simplicity (apples or oranges, plates, and hands) each of which performed different roles. In the *Filling Liquid* action the hands are grouped correctly despite having different poses. Note that for the sake of simplicity the backgrounds are ignored in Fig. 8 although they are also detected and grouped correctly.

While we are here strictly at the level of segments it is evident (albeit non-trivial!) that this unsupervised categorization process could be coupled to object models, thus, providing access to object categorization, too. It is interesting to remark that in this case any object-like entity will be classified strictly in the context of the observed manipulation. Thus, steps 6A and 6B are tightly linked as depicted by the gray box in Fig.2. For example a "cup-being-filled" would be grouped with other objects-being-filled. The same cup, when occurring in an action of "cup-used-as-pedestal" (where the cup is first turned upside down and then something is put on top), would be classified together with other objects-used-as-pedestals. This relates to the cognitive concept of affordances (Gibson, 1977) and will be discussed later in more detail.

## 3.9 Learning (Step 7)

In the next step we will show that the SECs of different instantiations of type-similar manipulation can be combined by statistical learning to render a model SEC for this manipulation type. Also this is done in an unsupervised way.

We know that rows of the event chain encode the main relational changes between segments. To arrive at a model, the learning procedure just needs to search for all common relational changes observed across repeated type-similar manipulations. A simple averaging algorithm suffices for this.

We describe an on-line version of the learning, but the same procedure could also be employed in batch-mode. Learning is initiated by assigning small weights $\omega_i^r$ to all rows and $\omega_i^c$ to all columns of the first observed chain. When observing the next manipulation, we use Step 6A (action classification) to find out if it is type-similar. If this is the case the weights of each row and column are incremented by a small amount $\Delta\omega_i$ if the row and column have a correspondence in the new event chain. If the new chain has additional, so far unobserved rows, the model is extended by these rows, which start with the initial small weight value. This is repeated as long as desired but usually 10 instantiations suffice for a stable model. After this, weights are thresholded, deleting all rows and columns which are subthreshold and returning the resulting model event chain for this ma-

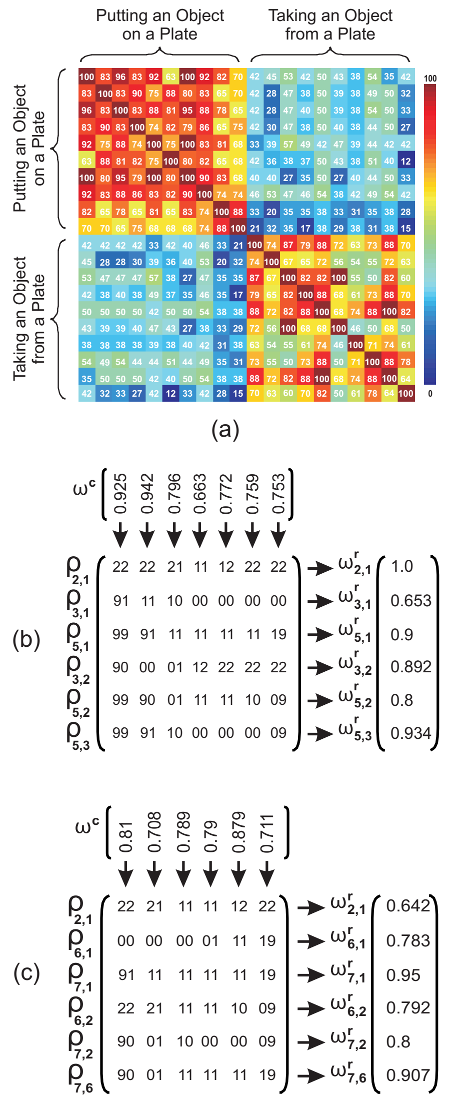


Figure 10: *(a) Similarity values between event chains of "Putting an object on a plate" and "Taking an object from a plate" actions. (b) The learned SEC model for the action type "Putting an object on a plate" with corresponding row ($\omega_i^r$) and column ($\omega_i^c$) weight values. These weight vectors are just for illustration since different weight values might be observed for different action types due to degree of noise in the event chains. (c) Same for "Taking an object from a plate".*

nipulation type.

In addition to this, for each manipulation instance, action-relevant segments (segment groups) are extracted and labeled according to their roles within the observed action as explained in Step 6B (segment categorization).

Note, online learning could suffer from bad first examples with which all next following manipulations would be classified. There are obvious work-arounds, for example cross-comparing the manipulations with each other. Ultimately, batch-mode learning is more useful. For this one would first record scenes from many manipulations, then perform clustering of the similarity matrix (e.g. Fig. 10) after which learning can be done for each cluster in the same way as described above.

We applied the learning framework in batch-mode to two different manipulation types: *"Putting an object on a plate"* and *"Taking an object from a plate"* each of which has 10 different versions with strongly different trajectories, speeds, hand positions, and objects (see Fig. 9 to get an impression of the level of difference).

Unsupervised classification of the similarity matrix (see Fig. 10) is used to classify those 20 versions. Note, many times a high similarity values (around 50%) is observed between non-type-similar actions. The reason is that except for the sequencing, which is inverted for "putting" versus "taking", primitives of both action types necessarily look similar. Differences are big enough, though, such that unsupervised classification will still lead to completely correct classification.

Next, a SEC model is learned for each manipulation class by searching for the similar common rows and columns observed in all 10 different versions as explained above. Fig. 10 (b-c) shows the learned SEC models for both action types with corresponding row ($\omega_i^r$) and column ($\omega_i^c$) weight values. To prove the accuracy of the learned SEC models we prepared 5 test movies which all contain *both* action types – putting and taking –, but performed in different temporal order (or sometimes with two hands at the same time!). Fig. 11 shows some sample frames from each of the test movies.

Fig. 12 depicts the similarity results between two

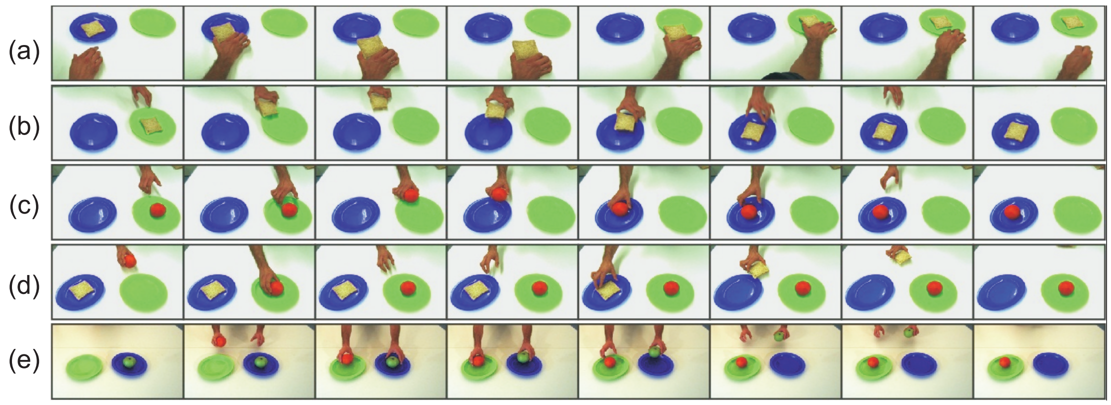


Figure 11: *Sample frames from* 5 *different mixed actions in which both manipulation types "Putting an object on a plate" and "Taking an object from a plate" are performed in different orders. (a) A hand is first taking a piece of bread from a plate and then putting it on a different plate. (b) Another piece of bread is moved from one plate to another with a different trajectory. (c) A hand is replacing an orange. (d) A hand is first putting an orange on a plate and then taking a piece of bread from another plate. (e) A hand is putting an orange on a plate and in the mean time the other hand is simultaneously taking an apple from the second plate.*

learned models and all 25 movies, 20 of which are the training data and the remaining 5 are unknown test data. Similarity is measured as described in Step 5. In red and blue are indicated the similarities for a given movie with the *"Putting an object on a plate"* and *"Taking an object from a plate"* models by leave-one-out method, respectively. For the first 10 training data the learned model of *"Putting an object on a plate"* has higher a similarity, whereas the model of *"Taking an object from a plate"* has a lower one (Fig. 12, green area). It is the other way around for the next 10 training data (Fig. 12 yellow area). However, for the last 5 test data, in which both manipulation types are performed in different orders both learned models have high similarity. (Fig. 12 blue area). When doing time-slicing (data not shown) one sees that the similarity in the last 5 data for either manipulation increases together with the completion of the respective manipulation. Thus, one after the other in the first 4 movies and simultaneously in the last one, where both actions are performed simultaneously.

## 4 Case Study: Learning and replaying an action sequence

Artificial intelligence (AI) systems almost always follow logic rules structured as: pre-condition, action, post-condition. Assessment of success of rule-execution requires measuring the post-condition. Hence, such systems rely on Thorndike's law of cause and effect (Thorndike, 1911) and, traditionally, they were defined by their programmers. Thus, it is difficult to find ways for an agent to learn cause-effect rules by itself (without explicit interference of a supervisor, see "grounding problem", (Harnad, 1990)). Furthermore, especially in complex situations, agents are faced with the problem of how to assess "effect" as many aspects of a situation might change following an action (see "frame problem", (McCarthy and Hayes, 1969)).

In the following we show our first results of a system that allows learning the rules of an action sequence without explicit supervision and then executing actions in a scenario self-assessing "action-

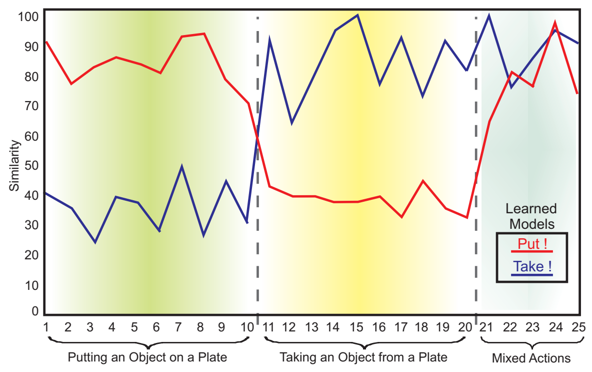


Figure 12: *Similarity results between the two learned modes and all* 25 *movies. In red and blue are indicated the similarities for a given movie with the "Putting an object on a plate" and "Taking an object from a plate" models, respectively. First* 20 *data are the training data and represent different versions of the "Putting an object on a plate" and "Taking an object from a plate" actions, respectively. The last* 5 *data represent the mixed actions used for testing the learned models.*

effects". Both processes rely on the event chains and the agent can without any pre-defined rule set learn the sequence and then assess the (in-)correctness of its actions just by comparing the resulting chains. Condensation into event chains thus helps solving the grounding- as well as the frame problem.

Our robot system is quite simple, consisting of a 3 DOF arm an with magnetic gripper (Neurorobotics, Sussex). Thus, we used "pushing" as well as "pick-and-place" as action repertoire. To generate trajectories we used predefined dynamic movement primitives (Ijspeert et al., 2002; Ning et al., 2010) and trajectory start- and end-points (for touching) were visually pre-defined and transferred onto the robot via a standard inverse kinematics procedure (no servoeing). Motion generation and control are not in the focus of this study, therefore we kept this simple here (for an advanced treatment of these aspects see Ning et al. (2010)). Objects for pick-and-place were magnetic.

The desired action sequence was first demonstrated by a human. Fig. 13 (a-b) (blue frame) shows sample frames of the action sequence in which a hand is "pushing" a lid off a container and then "picking-and-placing" a ball inside. The event chains of this action sequence is learned by our system. It can be broken into two sub-chains and the final result is shown in Fig 14 (a,b).

In the next step we confront the robot with a scene, provide it with a possible set of motion-trajectory start points, and let the robot randomly try out pushing and pick-and-place actions. Fig. 13 (c-f)(red frame) shows a subset of the different types of actions the robot has tried out (many more were performed but cannot be shown here). The blue tip of the robot arm is visible in the images. Note, objects are usually different from the ones used by the human. In Fig. 13 (c) the robot is only pushing a lid but does not continue with pick&place. In (d) a black ball is pushed. Fig. 13 (e) shows how the robot picks up a ball and then drops it on the table. Panel (f) represents an action where the robot is taking the ball from a container and places it on the table. All these examples do not (or only incompletely in (c)) reproduce the observed action sequence. Fig 13 (g-h) (green frame) shows the correct action sequence which at some point was also executed by the robot.

Corresponding event chains of all those action sequences are given in Fig 14. Due to different noise sources (in tracking, segmentation or depth information) the sizes of individual event chains can vary considerably. Still, as discussed in the section on Learning, individual chains contain the relevant information, which is not harmed by noise-induced rows and columns. As a consequence, even very different looking event chains can be robustly compared to the learned models (a,b) using the above described similarity algorithm. Figure labels (a-h) in Fig. 13 correspond to those in Fig. 14. Colored boxes in Fig. 14 show rows with high similarities. This occurs for panel (c) and (g), which are similar to (a), as well as for (h), which is similar to (b). A similarity table is shown in Fig. 14 (i). It shows that manipulation (c) is similar to the learned pushing model (a). The same is true for manipulation (g), which both are above 60% similarity. Only manipulation (h) is similar to the pick and place-inside model (b) with 87% similarity.

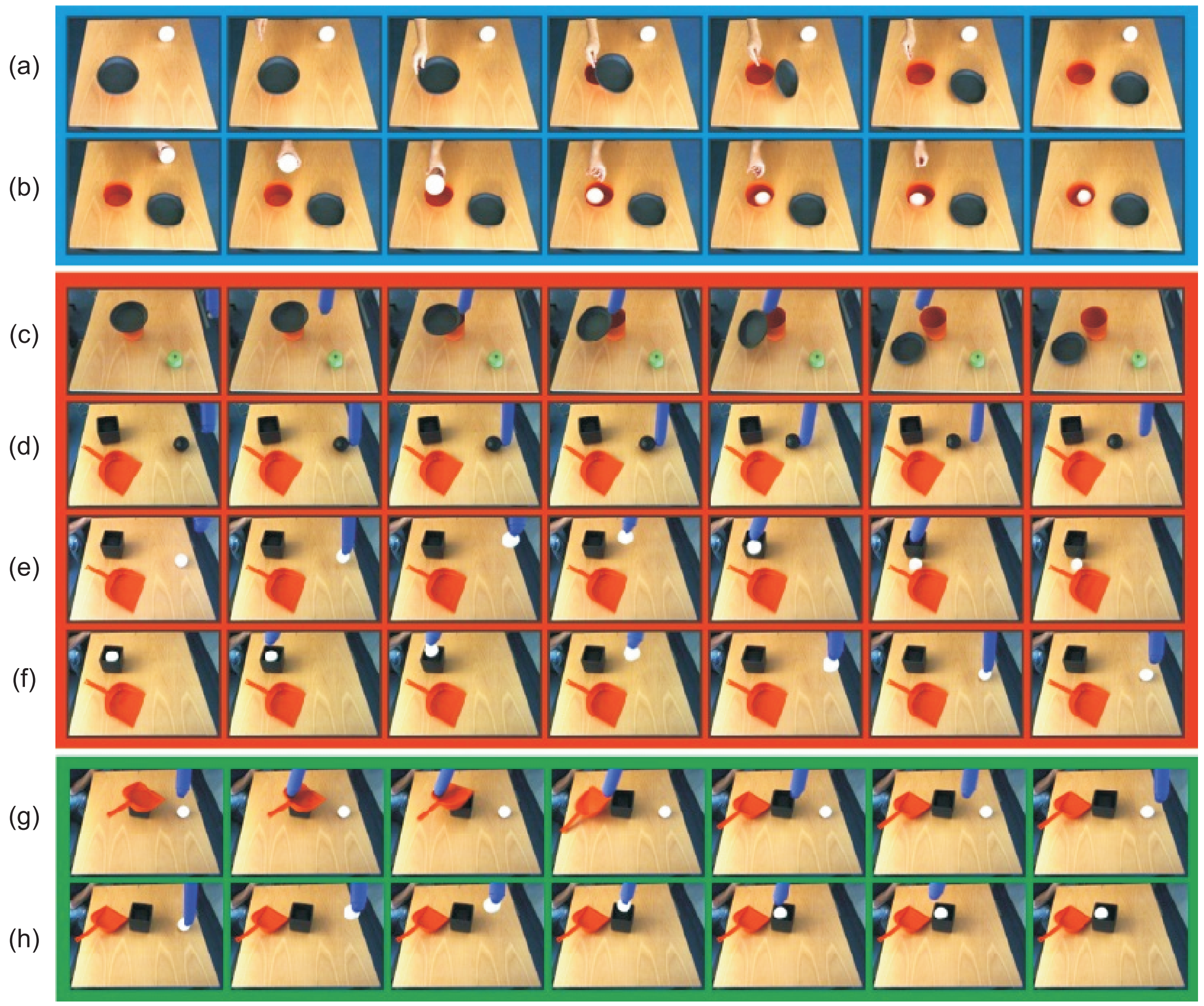


Figure 13: *Action sequence of (a) pushing a lid off a container and then (b) putting a ball inside demonstrated by a human (blue frame). Different types of robot actions (red frame). (c) pushing a lid, (d) pushing a ball, (e) lifting a ball and dropping it on the table, (f) taking the ball from a container and putting it on the table. The green frame shows a robot action sequence similar to the one performed by the human, in which (g) a lid is first pushed off and then (h) a ball is placed inside a container.*

$$\text{(a)}\quad \begin{matrix} \rho_{2,1} \\ \rho_{4,1} \\ \rho_{5,1} \end{matrix} \begin{pmatrix} 00 & 00 & 00 & 02 & 22 \\ 90 & 01 & 10 & 00 & 00 \\ 99 & 99 & 91 & 11 & 10 \end{pmatrix}$$

$$\text{(b)}\quad \begin{matrix} \rho_{2,3} \\ \rho_{3,5} \\ \rho_{3,6} \end{matrix} \begin{pmatrix} 22 & 21 & 11 & 10 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 01 & 11 \\ 90 & 00 & 01 & 11 & 10 & 00 & 09 \end{pmatrix}$$

$$\text{(c)}\quad \begin{matrix} \rho_{2,1} \\ \rho_{5,1} \\ \rho_{6,1} \\ \rho_{6,5} \\ \rho_{4,1} \\ \rho_{4,2} \\ \rho_{6,4} \end{matrix} \begin{pmatrix} 00 & 00 & 00 & 00 & 00 & 02 & 22 & 22 & 22 & 22 \\ 90 & 01 & 11 & 11 & 11 & 10 & 00 & 00 & 00 & 09 \\ 99 & 99 & 99 & 91 & 11 & 11 & 10 & 00 & 00 & 00 \\ 99 & 99 & 99 & 91 & 11 & 11 & 11 & 10 & 00 & 09 \\ 00 & 00 & 01 & 11 & 11 & 11 & 10 & 00 & 00 & 00 \\ 22 & 22 & 22 & 22 & 21 & 11 & 11 & 11 & 12 & 22 \\ 99 & 99 & 99 & 91 & 11 & 11 & 11 & 11 & 11 & 11 \end{pmatrix}$$

$$\text{(d)}\quad \begin{matrix} \rho_{2,1} \\ \rho_{4,1} \\ \rho_{3,5} \end{matrix} \begin{pmatrix} 01 & 11 & 11 & 10 \\ 00 & 01 & 10 & 00 \\ 10 & 00 & 00 & 00 \end{pmatrix}$$

$$\text{(e)}\quad \begin{matrix} \rho_{1,2} \\ \rho_{1,3} \\ \rho_{2,3} \end{matrix} \begin{pmatrix} 01 & 11 & 11 & 11 & 10 & 09 \\ 00 & 01 & 11 & 11 & 11 & 19 \\ 22 & 22 & 21 & 10 & 00 & 09 \end{pmatrix}$$

$$\text{(f)}\quad \begin{matrix} \rho_{1,3} \\ \rho_{4,3} \\ \rho_{2,3} \\ \rho_{5,4} \end{matrix} \begin{pmatrix} 00 & 00 & 02 & 22 \\ 22 & 20 & 00 & 00 \\ 01 & 11 & 10 & 00 \\ 22 & 22 & 22 & 20 \end{pmatrix}$$

$$\text{(g)}\quad \begin{matrix} \rho_{5,6} \\ \rho_{3,6} \\ \rho_{6,7} \\ \rho_{3,5} \\ \rho_{5,7} \end{matrix} \begin{pmatrix} 00 & 00 & 00 & 02 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 \\ 01 & 11 & 11 & 11 & 11 & 10 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 09 \\ 00 & 00 & 01 & 11 & 10 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 11 & 11 & 11 & 10 & 00 & 00 & 00 & 01 & 10 & 01 & 11 & 10 & 00 & 00 & 09 \\ 22 & 21 & 12 & 22 & 21 & 11 & 12 & 22 & 22 & 22 & 21 & 12 & 21 & 12 & 22 \end{pmatrix}$$

$$\text{(h)}\quad \begin{matrix} \rho_{5,8} \\ \rho_{8,7} \\ \rho_{3,8} \\ \rho_{3,6} \\ \rho_{3,5} \\ \rho_{5,7} \end{matrix} \begin{pmatrix} 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 21 & 10 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01 & 11 & 11 & 11 & 11 & 11 & 11 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 01 & 11 & 11 & 11 & 10 & 00 & 01 & 11 & 11 & 11 & 11 & 11 & 10 & 09 \\ 01 & 11 & 11 & 11 & 11 & 11 & 10 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 09 \\ 11 & 11 & 11 & 10 & 00 & 00 & 00 & 01 & 11 & 11 & 11 & 10 & 00 & 01 & 11 & 11 & 11 & 10 & 00 & 00 & 00 & 09 \\ 22 & 21 & 12 & 22 & 21 & 11 & 12 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 21 & 12 & 21 & 12 & 22 & 22 \end{pmatrix}$$

| (i) | Pushing Model (a) | Picking & Placing Model (b) |
|---|---|---|
| (c) | $\boxed{92\ \%}$ | 25 % |
| (d) | 22 % | 33 % |
| (e) | 11 % | 28 % |
| (f) | 50 % | 17 % |
| (g) | $\boxed{67\ \%}$ | 42 % |
| (h) | 22 % | $\boxed{87\ \%}$ |

Figure 14: *Corresponding event chains of human demonstrated actions and different types of robot actions. Labels (a-h) correspond to the manipulations shown in Fig 13 (a-h). (a,b) Event chain model extracted from human demonstration of (a) "pushing" as well as (b) "pick-and-place-inside". (c-f) Event chains corresponding to the wrong or incomplete actions in Fig 13 (c-f, red frame). (g,h) Event chains corresponding to the correct sequence in Fig 13 (g,h, green frame). (i) Similarity table between all actions the robot has tried (c-h) and the learned models (a,b) demonstrated by the human.*

Sequence (g-h) of both manipulations following each other is, thus, correctly recognized as being the one that reproduces the complete learned model (a-b).

Thus, this (still rather simple) set of examples demonstrates that by using SECs learning and recognition of manipulations is possible for a robot. The main achievement, we believe, lies here in the very high level of abstraction, which allows the machine to recognize (in-)correctness of its actions even when objects and their arrangements are very different in the different scenes.

## 5 Discussion

In this paper we have introduced a novel representation for manipulations, called the semantic event chain, which focuses on the relations between objects (including hands) in a scene. The representation generates column vectors in a matrix where every transition between neighboring vectors can be interpreted as an action rule, which defines which object relations have changed in the scene. Hence event chains reach a rather high level of abstraction but on the other hand they remain tightly linked to the images from which they originate, because they rely on continuously tracked segments. We have devised simple algorithms based on sub-string comparisons and counting procedures by which event chains can be compared and actions as well as segment (-groups) can be classified in an unsupervised way. No prior object models are required in this approach and the learning of archetypal event chains ("models") relies only on weight upgrade of repeating columns (repeating "rules"). Thus, learning also operates in an unsupervised way. The method was shown to be successful in classifying and recognizing video showing different manipulations, and also in learning the archetypal SEC for a given action class. We further demonstrated the feasibility of the approach through experiments with an robotic arm. By observation, the machine extracted the SEC of a human manipulation, and then reproduced the associated manipulation type in a new scenario via repeated experimentation.

To our knowledge this is the first approach towards manipulation recognition, which reaches an almost symbolic level of abstraction[1] while being fully grounded in the signal domain. In the following we will discuss related approaches and also problems and possible extensions of our algorithm.

### 5.1 Related approaches

Our framework introduces the semantic event chain which is a novel representation that seems to hold some promise for extracting action semantics. It directly encodes the observed manipulations without hidden states. Event chains remain tightly linked to perception but are of semantic character. Thus, they are more invariant with respect to viewpoint changes and object features than most other approaches. This however only holds if the visual entities used, here image segments, carry sufficient meaning, i.e. they represent parts of objects. As a consequence, the semantic event chain is composed of action primitives, and no hidden model needs to be assumed (e.g. Hidden Markov Model) as required in other works to bridge the gap between signal and symbol (Ogawara et al., 2002; Raamana et al., 2007).

Similarities exist between our approach and the work by Sridhar et al. (2008), who analyzed manipulations in the context of a breakfast scenario and represented the whole image sequence by an activity graph which holds spatiotemporal object interactions. By using statistical generalization, event classes are extracted from the activity graphs. Object categories are learned by calculating the similarity between object roles at each event class. However, large activity graphs and the difficulty of finding exact graph isomorphisms are a major drawback of this method. Furthermore, unlike in our work, object knowledge was provided beforehand, thus lacking grounding in the signal domain.

Our approach is different from the one by Kjellstrom et al. (2008). In our method, event chains are already highly invariant with respect to viewpoint

[1]Note, the symbolic domain can be directly coupled to the event chains as these chains allow the parsing of meaningful rule-like commands such as: "In the next transition perform an action that assures the segment 17 touches segment 12 and that segment 3 overlaps with segment 5."

changes and object features because the relations between segments are of qualitative character - e.g., touching is already a semantic description -, while in the work of Kjellstrom et al. (2008) semantics only emerge at the last stage of the modeling process.

In the present paper, we do not perform any object recognition in the classical sense (see Section 2.3). The image segments used for finding the SECs are below the object level, which means that a "true" object may be composed of several segments. Nevertheless, during the manipulation recognition procedure, image segments emerge naturally in conjunction with their associated action, providing a means to extract "action-relevant" objects from the scene by recognizing the respective actions in the SECs. This result is congruent with psychophysical evidence that humans recognize objects more easily if they are embedded in a consistent action context (Helbig et al., 2010). The approach has the advantage that it is highly invariant to the object's appearance and only takes into account the functionality of the object with respect to a given set of actions (see Fig. 8). However, the rich information provided by the object's appearance in the image is ignored and thus the algorithm does not allow recognizing objects without providing any action context.

## 5.2 Features and Problems of the Algorithm

The realm of object and action recognition is exceedingly rich carrying many facets and so far there is no algorithm existing which would reach the level of human proficiency. Adults can robustly classify objects and actions using a very high degree of invariance and generalization. To reach classification robustness in artificial systems usually the application scenarios are restricted and models of objects and/or actions are introduced, which however limits the generalization properties of such systems. To improve on this, preferentially life-long model-learning would be required leading to the extension of existing models and to the acquisition of new ones across all scenarios. Little is known how to do this.

In the introduction we had discussed that at least six properties define the requirements for a useful manipulation representation for artificial agents: (1) sensory signal based, (2) learnable, (3) relational, (4) time-sliced (5) human-comprehensible and (6) compatible with models.

We believe that the here introduced representation carries these aspects at least to some degree and that the approach also reaches across scenarios, but we are also aware of the fact that we are still far away from the final goal, which would be to reach human proficiency.

What are the drawbacks we are still facing and which extensions need to be made in future work to improve on this?

We heavily rely on advanced computer vision methods and we are aware that failures in the computer vision can harm our approach. Measures are undertaken to reduce such failures (Dellen et al., 2009), but this was not in the core of this study.

It is evident that image segments do not correspond to objects and our approach remains "beneath" the object level. While this way being strongly grounded in the image domain, we do not address complex aspects such as "permanence" and "feature binding", which are both required to assure the continuous existence of an object in the memory of a system.

In the case of heavily textured objects, feature binding based on color alone as employed in the segmentation framework will lead to a large number of segments, i.e. objects will be highly fragmented. This may cause problems to the tracking procedures (matching complexity) and thus will affect the quality of the SECs in an undesirable way. For textured images, additional mechanisms should be used at the segmentation stage to avoid over-segmentation and/or fragmented segments could be merged using segment-based grouping criteria. We are currently investigating potential solutions to this problem.

However, probably the most important feature of the here presented algorithms is that they do not rely on image segments as their input. Any (!) continuously trackable entity, as long as it is sufficiently close to the semantic level, hence also object-model-based tracking, can be used to design an event chain. Thus, while we still think that image segments are in many ways useful, the event chain representation and its core algorithms are transferable to other inputs too.

Furthermore, as mentioned above, our approach and the resulting event chains cannot directly be used to execute an action. For this, additional information about movement trajectories of the hands as well as information on the required poses of the objects needs to be stored and used. It will be a topic of future research to introduce these aspects into the framework.

### 5.3 Affordances and Object-Action Complexes

The rule-character of the event chains was used in this study to let an agent assess the success of its own actions. The experiments shown in Figs. 13 and 14 have demonstrated that an agent can first learn event chains and then connect them to (predefined) motion patterns trying out how to reproduce an action sequence. In doing so correlations between actions and outcomes (the resulting states) exist and the agent can produce triplets of *[starting state, action, resulting state]*, where starting and resulting state are given by adjacent columns of the event chain. If the resulting state matches the one remembered from the learning phase then the agent can store the complete triplet. This way actions can be attached to starting states taking the process forward from recognition to purposeful execution. Also this process works in an unsupervised way, where repetition would allow the agent to consolidate a triplet, but is also accessible by supervision, where a teacher would tell the agent which action to perform when.

For easy of talking we will use the term "thing" and "object" now in this discussion avoiding the clumsy terms segment-group, entity, etc.

As mentioned above, in our representation objects are being classified always in the context of the performed manipulation. A thing, when being filled with liquid, could be grouped with fillable objects into a category, which a human might call "containers". Equally well the same thing (when turned around) would be grouped together with other platforms into a category "pedestals" as soon as someone puts something on top of it. This aspect strongly relates to Gibson's concept of affordances (Gibson, 1977), where objects suggest actions. This concept had been extended by a group of European researchers from the PACO-PLUS project (http://www.paco-plus.org/) stating the objects and action are combined by humans into the cognitive concept of Object-Action Complexes (OACs) (Wörgötter et al., 2009; Krüger et al., 2010). A physical "thing" gets its meaning through the actions performed with it and will consequentially be interpreted – like the container/pedestal above – always in an action context.

Krüger et al. (2010) had given a formal definition of the Object-Action Complex. In an abbreviated way it states that an OAC is a function that compares obtained against excepted outcome when an action is being executed at an object. The OAC thereby also measures success of the action by assessing the sensor states of the agent. An OAC, thus, represents a model of the transformations that takes place at an object following an action (Wörgötter et al., 2009). OACs are learnable by the agent.

Adjacent columns of an event chain together with the processes of action learning described above are closely related to this concept. Thus, a complete event chain (together with its actions and objects) represents a chain of OACs and can be understood as a category which groups objects and actions into the cognitive concept of a manipulation. Thus, the here suggested set of algorithms provides – as far as we know – the first entry point to a grounded, agent-learnable cognitive categorization process of rather high complexity. In addition, it provides a link to the symbolic, language domain because of its rule-like character.

As discussed above, many problems still exist, but we would hope that this research might open some new avenues into the difficult cognitive field of action understanding.

## Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No 247947 - GARNICS. We thank Tomas Kulvicius for valuable discussion. B.D. acknowledges support from the Spanish Ministry for Science and Innovation via a Ramon y Cajal Fellowship.

## 6 Appendix

### 6.1 Appendix 1 - Defining segment relations in a fast and efficient way

As defined in the main text, possible spatial relations of each segment pair are *Touching=2*, and *Overlapping=1*, *No Connection=0*, and *Absence=9*. The process of calculating those relations has two main steps. In the very first step the segmented image is scanned horizontally (from left to right) and vertically (from top to down) to calculate the existing segment sequences. Following the scanning process, all lines (vertical+horizontal) are counted where a certain segment sequence has been observed and are stored in a list

$$L: \quad (i_1, i_2, i_3, ...) \mapsto n_S$$

where $n_S$ is the number of all vertical and horizontal lines with the segment sequence $(i_0, i_1, i_2, ...)$.

Fig. 15 illustrates how the sequences between 5 different segments can be calculated, e.g. (1) and $(1, 2, 1, 3, 1)$ are observed as 225 and 40 times, respectively.

The second main step analyzes the existing sequences to calculate the spatial relations between segment pairs. For this purpose, each sequence is iterated by considering the following rules:

- **"Touching"**: Segments follow one right after the other in any sequence are touching, e.g. segments 5 and 3 are touching each other in such sequences $(..., 5, 3, ...)$ or $(..., 3, 5, ...)$.
- **"Overlapping"**: (i) If a segment is observed twice in a sequence, all segments in between are overlapping with it, e.g. in $(..., 1, 5, 3, 1, ...)$ 5 and 3 are both overlapped (surrounded) by 1. (ii) However, the inner segments are not overlapping with each other, e.g. in $(..., 1, 5, 3, 1, ...)$ 5 cannot overlap with 3 because it is not observed twice.

To each rule corresponds a counter of hints (either $\mathbf{C}^{\mathrm{t}}_{i,j}$ or $\mathbf{C}^{\mathrm{o}}_{i,j}$). For each segment pair, counters store number of hints that show the rules are fulfilled for each segment pair as

- $\mathbf{C}^{\mathrm{t}}_{i,j} \mapsto n_t$: Number of hints that $i$ and $j$ are touching.
- $\mathbf{C}^{\mathrm{o}}_{i,j} \mapsto n_o$: Number of hints that $i$ is overlapping with $j$.

Note that $\mathbf{C}^{\mathrm{t}}_{i,j} \equiv \mathbf{C}^{\mathrm{t}}_{j,i}$ since the *Touching* relation is undirected, whereas $\mathbf{C}^{\mathrm{o}}_{i,j}$ is not symmetric.

Each sequence $S$ is processed separately. Its elements are stored in a stack one after another. When the next element $i_n$ is stored, the first rule indicates that $i_n$ and the previous element $i_{n-1}$ have the *Touching* relation. Since the current sequence has been found multiple times in the image (given by $L(S)$), the touching entry $(i_{n-1}, i_n)$ is incremented by $L(S)$:

$$\mathbf{C}^{\mathrm{t}}_{i_{n-1},i_n} \mathrel{+}= L(S) \ .$$

*Example:* The sequence $S := (1, 5, 3, 1)$ is analyzed by storing the first element $i_1 = 1$ in the stack. Since there are always more than one element required for the stack, the algorithm immediately skips to adding the next element $i_2 = 5$. The first rule indicates that the pair $(1, 5)$ has the *Touching* relation. As a result, $\mathbf{C}^{\mathrm{t}}_{1,5}$ is increased by $L(S) = 40$. The same operations are applied to the pair $(5, 3)$ in the next step.

To fulfill the second rule the stored element needs to be checked whether it is already in the stack. In this case, the elements of the first occurrence $i_s$ and $i_n$ are recognized as having the *Overlapping* relation with $i_n$. Therefore, the corresponding counter will be updated as follows:

$$\mathbf{C}^{\mathrm{o}}_{i_n,j} \mathrel{+}= L(S), \quad \forall j \in \{i_{s+1}, ..., i_{n-1}\} \ .$$

*Example:* In the same sequence given in the previous example the next element $i_4 = 1$ is added to the stack and $\mathbf{C}^{\mathrm{t}}_{1,3}$ is incremented by 40. Since $i_4$ occurred earlier ($i_s = i_1$), all elements in between, hence $i_2 = 5$ and $i_3 = 3$, $\mathbf{C}^{\mathrm{o}}_{1,5}$ and $\mathbf{C}^{\mathrm{o}}_{1,3}$ are increased by $L(S) = 40$.

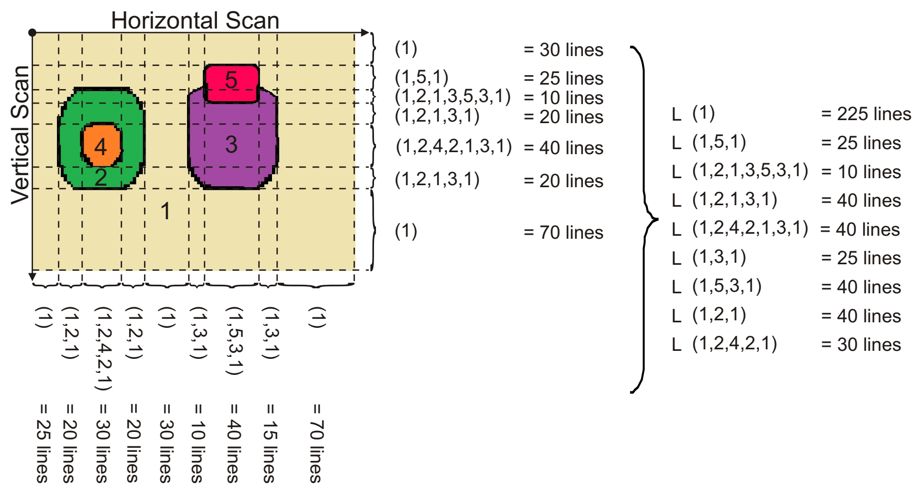


Figure 15: *Calculating the spatial segment relations between background, two vessels, and two contents which are represented by segment number 1, 2, 3, 4, and 5, respectively.*

The second rule also indicates that those inner elements $j$ do not overlap with each other, thus:

$$\mathbf{C}^{\mathrm{o}}_{j_n,j_m} -= L(S), \forall j_n, j_m \in \{i_{s+1}, ..., i_{n-1}\},\ n \neq m.$$

*Example:* Due to this rule, $\mathbf{C}^{\mathrm{o}}_{3,5}$ and $\mathbf{C}^{\mathrm{o}}_{5,3}$ are decreased by 40.

Next, the inner elements are removed from the sequence. This is important in cases of having recursive overlapping situations to get *Overlapping* relations only between neighbor segments. In Fig. 15 segment pairs $(1,2)$ and $(2,4)$ have the *Overlapping* relations, whereas $(1,4)$ has *No Connection.*

*Example:* For the sequence $S := (1,2,4,2,1)$, $i_4$ is added to the stack in the fourth step. By considering the description given above, we compute $\mathbf{C}^{\mathrm{t}}_{2,4} += L(S)$ and $\mathbf{C}^{\mathrm{o}}_{2,4} += L(S)$. The elements $i_3$ and $i_4$ are then removed from the stack, which leads to $(1,2)$. The algorithm is continuing by adding $i_5 = 1$ to the stack and by computing $\mathbf{C}^{\mathrm{t}}_{1,2} += L(S)$ and $\mathbf{C}^{\mathrm{o}}_{1,2} += L(S)$ as described above. In the end it is observed that segment pairs $(2,4)$ and $(1,2)$ have the *Overlapping* relation, however, $(1,4)$ has *No Connection.*

Once all sequences are iterated, the values in $\mathbf{C}^{\mathrm{t}}_{i,j}$ and $\mathbf{C}^{\mathrm{o}}_{i,j}$ are used to compute the final spatial relations of the segments. Note that some counter values might be wrong due to noisy segments. Instead of defining a minimum value as a static threshold, each entry is normalized first using the size of the corresponding segments:

$$\bar{\mathbf{C}}^{\mathrm{t}}_{i,j} := \frac{\mathbf{C}^{\mathrm{t}}_{i,j}}{\min(N_i, N_j)}$$

where $N_i$ is a list that stores the pixel size of segment $i$. Normalization considers only the smaller segment that makes the algorithm robust against noise and accurate for small segments. Note that $\mathbf{C}^{\mathrm{o}}_{i,j}$ is also normalized in the same way. Each normalized entry $\bar{\mathbf{C}}^{\mathrm{t}}_{i,j}$ and $\bar{\mathbf{C}}^{\mathrm{o}}_{i,j}$ is then thresholded. Unless $\bar{\mathbf{C}}^{\mathrm{t}}_{i,j}$ and $\bar{\mathbf{C}}^{\mathrm{o}}_{i,j}$ exceed the thresholds, relations are set to *No Connection.*

The main advantage of the proposed algorithm is that each step explained above can be calculated separately and hence can be parallelized.

Note, more complex 3D spatial segment relations (e.g. inside above, under, etc.) directly relate to the overlapping and touching relations as only a third dimension needs to be added. The following example makes this clear. Consider two 2D-*Overlapping* cases: "lying on top" (e.g. two flat objects) or "being inside" (of one smaller object inside a container). Both are 2D-identical in the sense of being an overlapping-

| Extension | Type | Description |
|---|---|---|
| 1 | Video | Four Different Action Types |
| 2 | Video | Case Study |

Table 3: Multimedia Extensions

relation, but with adding 3D one could define new relations ("on-top" and "inside").

## 6.2 Appendix 2 - Index of Multimedia Extensions

The multimedia extensions (see Table 3) to this article are at: `http://www.ijrr.org`.

## References

Abramov, A., Aksoy, E. E., Dörr, J., Pauwels, K., Wörgötter, F., and Dellen, B. (2010). 3d semantic representation of actions from efficient stereo-image-sequence segmentation on gpus. In *3DPVT*.

Aksoy, E. E., Abramov, A., Wörgötter, F., and Dellen, B. (2010). Categorizing object-action relations from semantic scene graphs. In *IEEE International Conference on Robotics and Automation, ICRA2010 Alaska, USA*.

Belhumeur, P. N. and Kriegmant, D. J. (1996). What is the set of images of an object under all possible lighting conditions. *IEEE CVPR*, pages 270–277.

Breazeal, C. and Scassellati, B. (2002). Robots that imitate humans. *Trends Cogn. Sci. (Regul. Ed.)*, 6:481–487.

Calinon, S. and Billard, A. (2004). Stochastic Gesture Production and Recognition Model for a Humanoid Robot. In *Proceedings of the IEEE/RSJ international Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2769–2774.

Calinon, S. and Billard, A. (2005). Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 105–112.

Calinon, S. and Billard, A. (2007). Incremental learning of gestures by imitation in a humanoid robot. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262, New York, NY, USA. ACM.

Dan Pelleg, A. M. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco. Morgan Kaufmann.

Dee, H., Hogg, D., and Cohn, A. (2009). Scene modelling and classification using learned spatial relations. In *Proc. Spatial Information Theory*, volume 5756, pages 295–311. Springer N.Y.

Dellen, B., Aksoy, E. E., and Wörgötter, F. (2009). Segment tracking via a spatiotemporal linking process in an n-d lattice model. *Sensors*, 9(11):9355–9379.

Dellen, B. and Wörgötter, F. (2009). Disparity from stereo-segment silhouettes of weakly textured images. In *Proceedings of the British Machine Vision Conference.*

Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *In CVPR*, pages 264–271.

Gibson, J. (1977). The theory of affordances. In perceiving, acting, and knowing. Eds. Robert Shaw and John Bransford.

Gilbert, A., Illingworth, J., and Bowden, R. (2009). Action recognition using mined hierarchical compound features. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Hakeem, A. and Shah, M. (2005). Multiple agent event detection and representation in videos. In *AAAI.*

Harnad, S. (1990). The symbol grounding problem. *Physica D 42*, pages 335–346.

Helbig, H. B., Steinwender, J., Graf, M., and Kiefer, M. (2010). Action observation can prime visual object recognition. *Experimental Brain Research*, 200(3-4):251–258.

Hoiem, D., Efros, A. A., and Hebert, M. (2008). Putting objects in perspective. *Int. J. Comput. Vision*, 80(1):3–15.

Hongeng, S. (2004). Unsupervised learning of multi-object event classes. In *Proc. 15th British Machine Vision Conference*, pages 487–496.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *in Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398–1403.

Kjellstrom, H., Romero, J., and Kragic, D. (2008). Simultaneous visual recognition of manipulation actions and manipulated objects. In *European Conference on Computer Vision.*

Krüger, N., Piater, J., Geib, C., Petrick, R., Steedman, M., Wörgötter, F., Ude, A., Asfour, T., Kraft, D., Omrcen, D., Agostini, A., and Dillmann, R. (2010). Object-action complexes: Grounded abstractions of sensorimotor processes (in revision). *Robotics and Autonomous Systems.*

Laptev, I. and Perez, P. (2007). Retrieving actions in movies. In *ICCV.*

Liao, L., Fox, D., and Kautz, H. (2005). Location-based activity recognition using relational markov networks. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 773–778.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

Maurer, A., Hersch, M., and Billard, A. (2005). Extended Hopfield Network for Sequence Learning: Application to Gesture Recognition. In *Proceedings of ICANN'05.*

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, pages 195–204.

Modayil, J., Bai, T., and Kautz, H. (2008). Improving the recognition of interleaved activities. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 40–43.

Mundy, J. and Zisserman, A. (1992). *Geometric Invariance in Computer Vision.* MIT Press.

Mundy, J. L. (2006). Object recognition in the geometric era: A retrospective. In *Toward CategoryLevel Object Recognition, volume 4170 of Lecture Notes in Computer Science*, pages 3–29. Springer.

Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-d objects from appearance. *Int. J. Comput. Vision*, 14(1):5–24.

Niebles, J., Wang, H., and Fei-Fei, L. (2008). Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318.

Ning, K., Kulvicius, T., Tamosiunaite, M., and Wörgötter, F. (2010). A novel trajectory generator based on dynamic motor primitives. *IEEE Transactions on Robotics (Submitted).*

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168.

Ogawara, K., Takamatsu, J., Kimura, H., and Katsushi, I. (2002). Modeling manipulation interactions by hidden markov models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Oliva, A. and Torralba, A. (2009). The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520–526.

Pauwels, K. and Van Hulle, M. (2008). Real-time phase-based optical flow on the GPU. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Computer Vision on the GPU*, Anchorage, Alaska.

Raamana, P. R., Grest, D., and Krueger, V. (2007). Human action recognition in table-top scenarios: an HMM-based analysis to optimize the performance. In *CAIP'07: Proceedings of the 12th international conference on Computer analysis of images and patterns*, pages 101–108, Berlin, Heidelberg. Springer-Verlag.

Rizzolatti, G. and Craighero, L. (2004). The mirror-neuron system. *Annual Review of Neuroscience*, 27:169–192.

Sabatini, S., Gastaldi, G., Solari, F., Diaz, J., Ros, E., Pauwels, K., Van Hulle, M., Pugeault, N., and Krüger, N. (2007). Compact and accurate early vision processing in the harmonic space. In *International Conference on Computer Vision Theory and Applications*, pages 213–220, Barcelona.

Shylo, N., Wörgötter, F., and Dellen, B. (2009). Ascertaining relevant changes in visual data by interfacing AI reasoning and low-level visual information via temporally stable image segments. In *Proceedings of the International Conference on Cognitive Systems (Cogsys 2008).*

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1470, Washington, DC, USA. IEEE Computer Society.

Sridhar, M., Cohn, G. A., and Hogg, D. (2008). Learning functional object-categories from a relational spatio-temporal representation. In *Proc. 18th European Conference on Artificial Intelligence*, pages 606–610.

Sumsi, M. F. (2008). *Theory and Algorithms on the Median Graph. Application to Graph-based Classification and Clustering.* PhD thesis, Universitat Autonoma de Barcelona.

Thorndike, E. (1911). Animal intelligence. New York. Macmillan.

Torralba, A. (2003). Modeling global scene factors in attention. *JOSA - A*, 20:1407–1418.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86.

Vicente, I., Kyrki, V., and Kragic, D. (2007). Action recognition and understanding through motor primitives. *Advanced Robotics*, 21(15):1687–1707.

Wörgötter, F., Agostini, A., Krüger, N., Shylo, N., and Porr, B. (2009). Cognitive agents - a procedural perspective relying on predictability of object-action complexes (oacs). *Robotics and Autonomous Systems*, 57(4):420–432.