Project no.: **027657**

Project full title: **Perception, Action & Cognition through Learning of Object-Action Complexes**

Project Acronym: **PACO-PLUS**

Deliverable no.: D2.1.5

Title of the deliverable: Report or scientific publ. of real time segmentation and tracking of image parts

Contractual Date of Delivery to the CEC:	**Jan 31, 2010**
Actual Date of Delivery to the CEC:	**Jan 31, 2010**
Organisation name of lead contractor for this deliverable:	**BCCN**
Author(s):	**F. Wörgötter,**
Participants(s):	**BCCN**
Work package contributing to the deliverable:	**WP2**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**5 + Appendix**
Start date of project:	**1st Feb. 2006** **Duration: 52 months**

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

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

Abstract:

This deliverable describes two papers on real time segmentation and model free tracking of image parts in the context of manipulation recognition. Images are being segmented using super-paramagnetic clustering with a Metropolis relaxation algorithm. Segments are tracked by binding them together via the optic flow information extracted in parallel from the image sequences. Left and right images provide stereo information (disparity) and left and right segments are bound by the disparity. Segment centers define graph nodes. Neighboring segments are being represented by graph-edges connecting the nodes. Topological changes (forming of breaking of nodes) are encoded numerically in a table: The semantic Event Chain. This table, thus, encodes relations of objects and their changes, which allows recognizing a manipulations type.

Keyword list: Real time segmentation, model free tracking, manipulation recognition, OACs

Contents

EXECUTIVE SUMMARY **2**
CONTRIBUTIONS TO PACO_PLUS **4**
CONTRIBUTIONS BEYOND THE STATE OF THE ART **4**

Executive Summary

This deliverable consists of two publications (in press), which address the important issue of how to represent and recognize manipulations.

Different from recognizing general movements and actions, like ”walking”, ”dancing”, ”standing up”, ”drawing a gun” (security issue!), etc., the recognition of a manipulations needs to achieve two things:

1) It must be able to deal with the *relations* between hand(s) and object(s). (Think of a pick and place manipulation, where hands and objects touch and un-touch.)
2) It must handle sometimes rather *long and convoluted sequences* of such relations and their changes. (Think of screwing a screw into a wood panel.)

Fundamentally, this is what is being captured by the OAC concept of PACO-PLUS: which states that objects and actions are only meaningful in conjunction with each other.

The goal of this deliverable is to introduce a certain representation, called “**semantic Event Chain**”, by which such relations can be captured, as well as to describe the procedures which are being used to arrive at this representation. Thus, the semantic Event Chain can be seen as a chain of OACs.

We are currently at the stage that we can automatically…

A) …extract semantic Event Chains from medium complex movie sequences, like the making of a breakfast in a clean- background situation, which contains about ten different manipulations.
B) We can learn by repetition the relevant parts of the individual (still noisy) Event Chains and…
C) …we can then in the end also recognize the different manipulations found in a longer sequence (of a different movie).
D) Furthermore, we can categorize objects into the same group, which belong to the same manipulation. For example, we can assign the label “pickable” to all objects, regardless of their other physical properties, that are being picked up in a “pick-up-manipulation”-chunk.

We are in this deliverable not concerned with the execution (by a robot) of an Event Chain. This is planned for a follow-up project.

Most of the procedures are working in video real-time, because they are implemented on a multi-core GPU/CPU architecture.

The procedure of extracting a semantic Event Chain consists of two complex steps:

1) Extraction of 3D-relational graphs
2) Extraction of the semantic Event Chain proper.

Fig. 1 taken from the first publication [1] summarized the different components of the first step. It relies on 3D-image segmentation and the real-time tracking of the segments. Graphs are being constructed by defining the segment centers as graph-nodes. Edges between nodes are drawn for all neighboring, hence, "touching" segments. Thus, an edge indicates that two objects touch each other in the real world.

The **central idea** behind all this is now to analyze how these "touching relations" change during a manipulation.

To this end segments are being tracked across image frames. This relies on the binding of segments from frame t to t+1 by optic flow. The optic flow determines where a segment found in frame t has moved to in frame t+1. Similarly, the stereo disparity information determines where a segment from the left image frame is found in the right image frame. Thus, using flow and disparity we can consistently track segments in 3D across long movie sequences. The difficulties in actually achieving stable tracking are being discussed in both papers, but they are not unsurmountable and good results for complex scenes – like the breakfast-making – exist already.

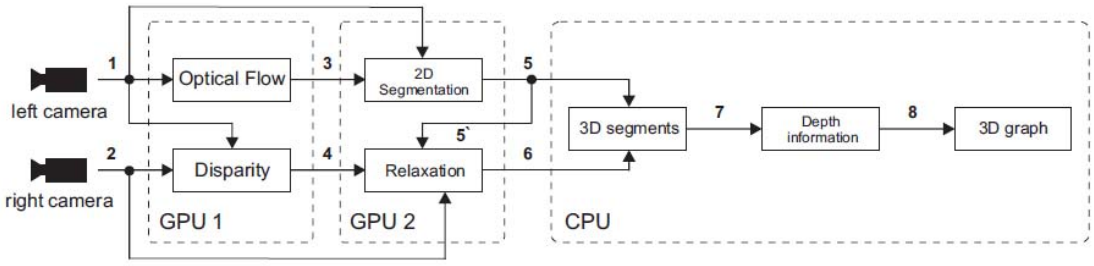


Fig. 1) Flow diagram of step 1: Extraction of 3D-graphs

In the second step, the semantic Event Chain is being generated. Fig.2, top from [2], right to the dashed line, shows the performed algorithmic steps, which lead to action (viz. manipulation!) recognition but also to object categorization. Note in this paper [2], we still had not arrived at the final nomenclature and there we call the Event Chain an "event table". Fig.2, bottom, shows for a very primitive example of three disks that move relative to each other, how a semantic Event Chain is being encoded. We assign alpha-numerical values to the different relations between segments, for example O=Overlapping, T=touching , N=non-existing. These labels are being assigned to all nodes in the 3D-graphs creating a column in the Event Chain. Only when at least one such relation changes a new column is being introduced and the Event Chain grows. Thus, the Event Chain encodes only the topological changes that happen to the 3D-graphs during a manipulation. Time becomes sequenced and the complexity of the original images gets extremely reduced[1]. For example, pose information is not being considered. (This may be a drawback for certain sub-manipulation types and we are working on an algorithmic extension to also capture pose, whenever two objects touch.)

[1] Data reduction also much lives from the sitation the we are dealing only with true 3D. Thus, all apparent touching relations which one would see in a 2D projective image (any single camera image) are being eliminated and only real touching (within a 3D-distance threshold) is being taken into account.

Originally one huge Event Chain would this way be constructed for a long movie, like the breakfast making example. However, one finds that there are natural breaking points in such a chain, due to the fact that hands can only perform one action at a time. Hence, the huge Event Chain diagonalizes into a diagonal block-matrix, where the individual blocks represent manipulation-primitives. These are manipulations that can be performed individually (i.e., independent from any other manipulation!).

In general we find that the Event Chain does this way capture in a very unique and very compressed way the different manipulations.

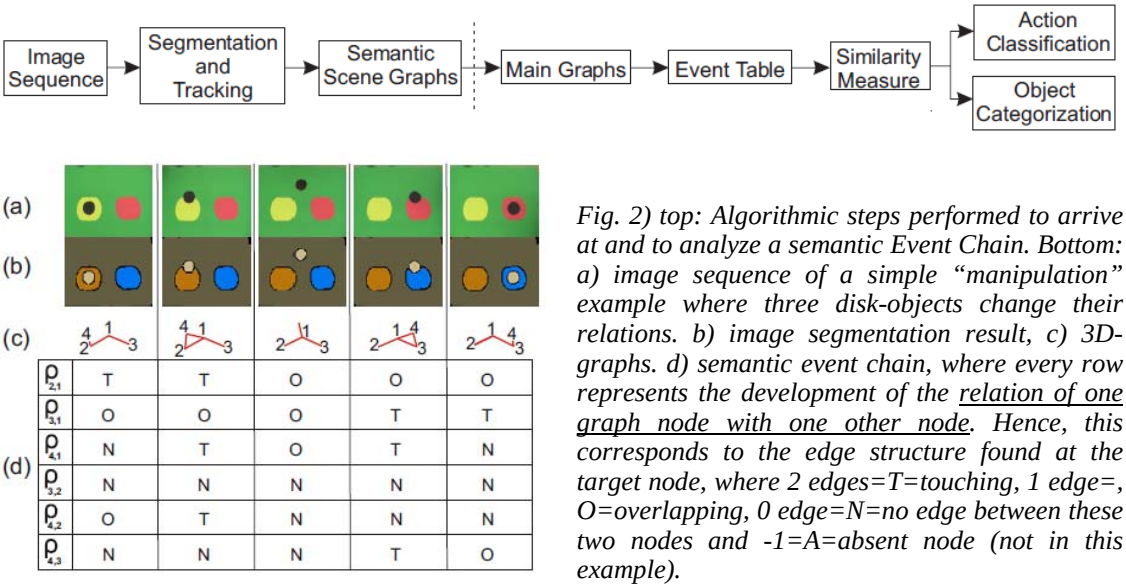


Fig. 2) top: Algorithmic steps performed to arrive at and to analyze a semantic Event Chain. Bottom: a) image sequence of a simple "manipulation" example where three disk-objects change their relations. b) image segmentation result, c) 3D-graphs. d) semantic event chain, where every row represents the development of the relation of one graph node with one other node. Hence, this corresponds to the edge structure found at the target node, where 2 edges=T=touching, 1 edge=, O=overlapping, 0 edge=N=no edge between these two nodes and -1=A=absent node (not in this example).

Contributions to PACO_PLUS

1. First instantiation of the chaining of OACs.

2. A framework for the learning and recognition of complex manipulations.

3. A framework for the categorization of objects by their role within a manipulation.

Contributions beyond the state of the art

Manipulation recognition has so far not been much addressed in the literature. There seems to be indeed only one other group, who have addressed this in its own right [3]. On the other hand there is quite an extended literature on "action recognition", which is only very mildly related to this approach. Actions are being recognized in these works without (or largely without) relation to any object (see literature discussion in [1,2]). Furthermore, essentially all approaches that we know of are model-based. Our Ansatz is model free. Even in the work of [3] the authors have to resort to pre-labeling the objects that exist in the scene (they use differently colored, uniform objects), thereby candidly instilling an object model. Furthermore in [3] time is not being discarded. This leads to

additional (unwanted) complexity as the fact that any manipulation can be executed with different speed is not being eliminated.

Thus, we believe that the semantic Event Chain approach is indeed a genuine and very novel approach with clear impact and relation to the OAC concept developed by PACO-PLUS. A detailed discussion of the less related works is given in [1,2].

References

[1] A. Abramov, E. E. Aksoy, J. Dörr, F. Wörgötter, K.Pauwels, B. Dellen (2010). 3D Semantic Representation of Actions from efficient stereo-image-sequence segmentation on GPUs, 3DPVT 2010- the Fifth International Symposium on 3D Data Processing, Visualization and Transmission (in press).

[2] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen (2010) Categorizing Object-Action Relations from Semantic Scene Graphs, ICRA (in press).

[3] M. Sridhar, G. A. Cohn, and D. Hogg, (2008) Learning functional object categories from a relational spatio-temporal representation," in Proc. 18th European Conference on Artificial Intelligence.

APPENDIX: Two accompanying publications

3D Semantic Representation of Actions from efficient stereo-image-sequence segmentation on GPUs

and

Categorizing Object-Action Relations from Semantic Scene Graphs

3D Semantic Representation of Actions from efficient stereo-image-sequence segmentation on GPUs

Alexey Abramov Eren Erdal Aksoy Johannes Dörr
Florentin Wörgötter
Georg-August University, BCCN Göttingen, III Physikalisches Institut
Göttingen, Germany
{abramov,eaksoye,jdoerr,worgott}@bccn-goettingen.de

Karl Pauwels
Laboratorium voor Neuro- en Psychofysiologie
K.U.Leuven, Belgium
karl.pauwels@med.kuleuven.de

Babette Dellen
BCCN Göttingen, Max-Planck-Institute for Dynamics and Self-Organization
Göttingen, Germany
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Barcelona, Spain
bkdellen@bccn-goettingen.de

Abstract

A novel real-time framework for model-free stereo-video segmentation and stereo-segment tracking is presented, combining real-time optical flow and stereo with image segmentation running separately on two GPUs. The stereo-segment tracking algorithm achieves a frame rate of 23 Hz for regular videos with a frame size of 256×320 *pixels and nearly real time for stereo videos. The computed stereo segments are used to construct 3D segment graphs, from which main graphs, representing a relevant change in the scene, are extracted, which allow us to represent a movie of e.g. 396 original frames by only 12 graphs, each containing only a small number of nodes, providing a condensed description of the scene while preserving data-intrinsic semantics. Using this method, human activities, e.g., handling of objects, can be encoded in an efficient way. The method has potential applications for human-activity recognition and learning, and provides a vision-front end for applications in cognitive robotics.*

1. Introduction

Movies contain abundant information about the visual scene, which, if choosing the raw signals for representations, render the application of any logic or learning scheme intractable. This problem occurs for example if we want to understand and learn both courses and consequences of manipulations from visual data. A reduced representation of the scene is urgently required to make such problems tractable by algorithms operating on a small number of abstract descriptors (symbols). Finding this reduced representation without prior knowledge on the data (model free) thus represents a major challenge in cognitive-vision applications – this problem is also known as the signal-symbol gap [12].

In this paper, we present a novel framework for bridging this gap. We aim at creating a condensed description of stereo movies by computing the 3D relations between tracked image segments for generating 3D semantic graphs, which, in the future, will allow us to encode human actions in an efficient way. To achieve this goal, three main problems need to be solved: (i) Stereo images need to be segmented in real time in a consistent way and image segments need to be tracked along the movie, i.e., segments representing the same part of an object should carry the same label

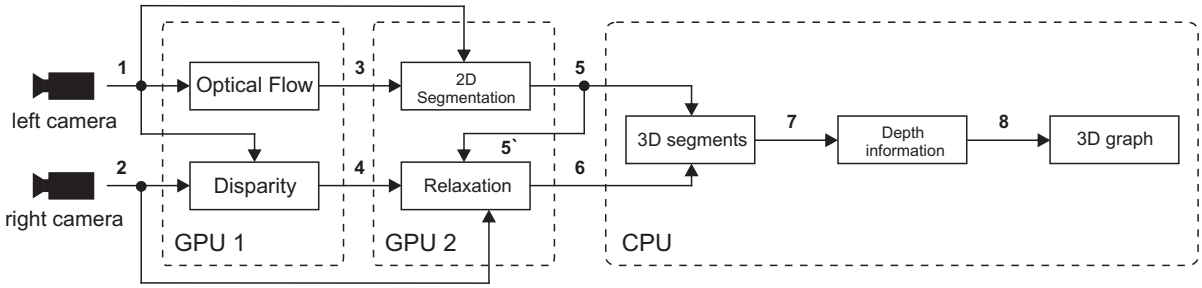


Figure 1. The architecture of 3D segment tracking framework.

all the time. Note that image segmentation represents a logical step towards our goal because redundant information is thereby grouped and condensed into higher level representational entities (segments). The method should be entirely data driven. (ii) The 3D relations of the segments need to be derived with sufficient accuracy, and graphs need to be constructed. Then, only the graphs representing a relevant cha nge in the scene should be extracted, further removing redundant information. (iii) The algorithms should run in real time or close to real-time[1] to allow the framework to be used for robotic applications.

Several approaches for video segmentation have been proposed in the past, where some methods rely on segmenting each frame independently, followed by a segment matching step based on their low-level features [5, 20, 23, 8], while other methods use motion projection to link segments [16, 28, 26, 14]. It was shown recently that image segments can be tracked along the frames of a movie in a model-free way, i.e. without assuming a data model of some kind, using the method of superparamagnetic clustering of data [6].

Real-time requirements render the video segmentation algorithms currently inadequate for most robotic applications. Furthermore, stereo movies have not been treated by any of these works. To overcome these limitations, we developed real-time model-free image segmentation on GPUs based on a novel parallel method, combined with real-time phase-based optical flow [17] and stereo [19], executed on GPU as well, for segment tracking.

The framework will potentially be applicable to a wide range of problems in the field of action and manipulation recognition and learning, and may serve as vision-front end in cognitive robots.

[1]By real-time we understand processing of a full frame at 25Hz or faster.

2. 3D segment tracking framework

2.1. Overview

The architecture of the 3D segment tracking framework is shown in Fig. 1. It consists of a stereo camera, two GPUs, one CPU, and various processing components that are connected by channels in the framework. Output data of all components can be accessed from any component in the framework. The left and right images from a stereo camera enter into the framework via channels 1 and 2 respectively. Optical flow and disparity are computed on GPU 1 using a real-time algorithm [17], and the results are accessible from channels 3 and 4 respectively (see Section 2.3). For the images from the left camera, the labels from a previous segmentation are warped to the current frame using optical flow (channel 3). The new label configuration is used as an initialization for the real-time segmentation algorithm running on GPU 2 (see Section 2.2-2.4). This way, the required time for label relaxation can be reduced, and, even more importantly, a consistent labeling of the frames can be achieved, i.e. segments describing the same object part are likely to carry the same label (segment tracking) (see Section 2.4). The results of the segmentation can be accessed from channel 5 and used to compute the label initialization for the segmentation of the right frame via channel $5'$ (see Section 2.5). This time, the labels are warped using phase-based disparity information obtained from channel 4. The segmentation result of the right image, which is now consistently labeled with respect to the images from the left camera, is stored in channel 6. Segments larger than a predefined threshold are extracted and stored in channel 7. The stereo-segment correspondences are used to find the 3D structure of the segments using a recent stereo method on the CPU [7] and stored in channel 8 (see Section 2.6). In a final step, relevant information about the 3D structure of the segments is extracted from the computed disparity map and used together with the segmentation results to construct an abstract 3D description of the scene, which is represented by undirected graphs in which nodes and edges rep-

resent 3D segments and their neighborhood relations (see Section 2.7).

2.2. Image segmentation algorithm

The method of superparamagnetic clustering solves the segmentation problem by finding the equilibrium states of the energy function of a ferromagnetic Potts model in the superparamagnetic phase [18, 11, 25, 9, 3, 15, 24]. The Potts model [18] describes a system of interacting granular ferromagnets or spins that can be in q different states, characterizing the pointing direction of the respective spin vectors. Three phases, depending on the system temperature, i.e. disorder introduced to the system, are observed: the paramagnetic, the superparamagnetic, and the ferromagnetic phase. In the ferromagnetic phase, all spins are aligned, while in the paramagnetic phase the system is in a state of complete disorder. In the superparamagnetic phase regions of aligned spins coexist. Blatt et al. (1998) applied the Potts model to the image segmentation problems in a way that in the superparamagnetic phase regions of aligned spins correspond to a natural partition of the image data [3]. Finding the image partition corresponds to the computation of the equilibrium states of the Potts model.

The equilibrium states of the Potts model have been approximated in the past using the Metropolis-Hastings algorithm with annealing [10] and methods based on cluster updating, which are known to accelerate the equilibration of the system by shortening the correlation times between distant spins, such as Swendsen-Wang [22], Wolff [27], and energy-based cluster updating (ECU) [15, 24]. All of these methods obey detailed balance, ensuring convergence of the system to the equilibrium state. Here we achieve efficient performance using the Metropolis algorithm with annealing [10], which can be easily parallelized and implemented on a GPU. The method further has the advantage that results from a previous segmentation can be utilized by the algorithm to find a consistent segmentation of the next frame within a small number of Metropolis updates only, drastically reducing computation time.

The real-time image segmentation algorithm proceeds as follows. In the Potts model, a spin variable σ_k, which can take on q discrete values $v_1, v_2, \ldots, v_q$, called spin states, is assigned to each pixel of the image. The energy of the system is described by

$$E = -\sum_{<ij>} J_{ij}\delta_{ij} \quad , \tag{1}$$

with the Kronecker sign

$$\delta_{ij} = \begin{cases} 1 & \text{if } \sigma_i = \sigma_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where σ_i and σ_j are the respective spin variables of two neighboring pixels i and j. The function

$$J_{ij} = 1 - |\mathbf{g_i} - \mathbf{g_j}|/\overline{\Delta} \tag{3}$$

is a coupling constant, determining the interaction strength, where $\mathbf{g_i}$ and $\mathbf{g_j}$ are the respective color vectors of the pixels, and

$$\overline{\Delta} = \alpha \cdot (\sum_{<i,j>} |\mathbf{g_i} - \mathbf{g_j}| / \sum_{<i,j>} 1) \tag{4}$$

computes the averaged color vector difference of all neighbors $< i, j >$. The factor $\alpha \in [0, 10]$ is a system parameter.

The Metropolis algorithm allows generating spin configurations S which obey the Boltzmann probability distribution [4]

$$P(S) \sim \exp\left[-\beta E(S)\right] \quad , \tag{5}$$

where $\beta = 1/kT$, T is the temperature parameter, and k is the Boltzmann constant.

Initially, values are assigned randomly to all spin variables. According to the Metropolis algorithm, each spin-update procedure consists of the following steps [13]:

1. The system energy E_A of the current spin configuration S_A is computed according to Eq. 1.

2. A pixel i with spin variable σ_i in spin state v_l is selected and for each possible move to a new spin state $\sigma_i \neq v_l$ the energy E_B of the resulting new spin configuration S_B is computed according to Eq. 1. The number of possible moves is $(q - 1)$.

3. Among all new possible configurations we find the configuration with the minimum energy

$$E_{new} = \min(E_1, E_2, \ldots, E_{q-1}) \quad , \tag{6}$$

and compute the respective change in energy

$$\Delta E = E_{new} - E_A \quad . \tag{7}$$

4. If the total energy of the configuration is decreased by this move, i.e. $\Delta E < 0$, the move is always accepted.

5. If the energy increased, i.e. $\Delta E > 0$, the probability that the proposed move will be accepted is given by

$$P_{A\to B} = \exp\left(-\frac{|\Delta E|}{kT_n}\right) \quad , \tag{8}$$

and

$$T_{n+1} = \gamma T_n \qquad \gamma < 1 \quad , \tag{9}$$

where γ is the annealing coefficient. We draw a number ξ randomly from a uniform distribution in the range of $[0, 1]$. If $\xi < P_{A\to B}$, the move is accepted.

Each spin update involves only the nearest neighbors of the considered pixel. Hence, spin variables of pixels that are not neighbors of each other can be updated simultaneously [2]. Therefore the Metropolis algorithm fits very well to the GPU architecture. For the first frame of the image sequence, a short-cut via a pre-segmentation step, is used to allow sufficiently fast segmentation. More details can be found in [1]. In the experiments we will use a large number of spins, which then serve as a segment label. Note that these wordings are thus equivalent here.

2.3. Phase-based optical flow and disparity

Since fast processing is a very important issue in our work, we use the GPU-based real-time optical flow algorithm proposed by Pauwels et al. (2008) [17]. This algorithm belongs to the class of phase-based techniques, which are highly robust to changes in contrast, orientation and speed. This particular algorithm integrates the temporal phase gradient (extracted from five subsequent frames) across orientation and gradually refines its estimates by traversing a Gabor pyramid from coarser to finer levels. In our framework optical flow is computed for the left video stream only. The algorithm provides a vector, at each pixel indicating its motion

$$\mathbf{u}(\mathbf{x},\mathbf{y}) = (u_x(x,y), u_y(x,y)) \quad , \tag{10}$$

This allows us to link pixels of two subsequent frames $\{t\}$ and $\{t+1\}$ (see Fig. 2).

In order to link pixels of correspondent left and right frames displacements for correspondent left and right pixels have to be estimated. For this purpose the disparity algorithm can be used. Our system relies on rectified images and therefore only horizontal disparities need to be determined. The disparity algorithm that is used in our framework is also phase-based and operates on phase differences between the left and right image as opposed to temporal phase gradients in optical flow algorithm. It is described in more detail in [19].

2.4. Image-sequence segmentation using optical flow based label warping

We obtain information about pixel correspondences between subsequent frames via a phase-based optical flow algorithm applied to the frame sequence, as described in 2.3. Since we are using a local algorithm, optical flow cannot be estimated everywhere, for example not in weakly-textured regions. For pixels in these regions, vertical and horizontal flows, i.e. u_y and u_x, do not exist. Thus we make an assumption that these pixels did not change position between frames $\{t\}$ and $\{t+1\}$, i.e. $u_y = 0$ and $u_x = 0$. We find the new label configuration by translating all labels according to the derived flow maps.

Suppose frame $\{t\}$ is segmented and S_t is its final label configuration (see Fig. 2(d)). The labels can be transferred from frame $\{t\}$ to frame $\{t+1\}$ according to

$$S_{t+1}(x_{t+1}, y_{t+1}) = S_t(x'_t, y'_t) \quad , \tag{11}$$

where

$$x'_t = x_{t+1} - u_x(x_t, y_t) \tag{12}$$

$$y't = y_{t+1} - u_y(x_t, y_t) \quad . \tag{13}$$

Labels which did not obtain an initialization via Eq. 11 are then given a randomly chosen label between 1 and q. Once frame $\{t+1\}$ is initialized (see Fig. 2(e)), a relaxation process is needed in order to fix erroneous bonds that can take place during the spin states transfer (see 2.2). We found that 20 additional Metropolis updating iterations are sufficient to obtain satisfactory segmentation results (see Fig. 2(f)).

2.5. Stereo segmentation using disparity-based label warping

Segmentation of every right frame is obtained in a similar way. Here, label initialization is obtained by translating the labels from the segmentation of the left image using the disparities from phase-based stereo. The stereo algorithm yields a sparse disparity map D providing information not for all pixels. For this reason, we make the assumption that pixels from frame $\{t_L\}$ having no correspondence in frame$\{t_R\}$ have zero displacement, i.e. $D(x^t) = 0$.

We suppose that the left frame $\{t_L\}$ is segmented and S^t_L is its final label configuration (see Fig. 3(d)). Labels are transferred from frame $\{t_L\}$ to frame $\{t_R\}$ according to

$$S^t_R(x^t_R, y^t_R) = S^t_L(x'^t_L, y'^t_L) \quad , \tag{14}$$

where

$$x'^t_L = x'^t_R - D(x^t_L, y^t_L) \tag{15}$$

$$y'^t_L = y'^t_R \quad . \tag{16}$$

Pixels that did not obtain a label initialization this way are given a randomly chosen label between 1 and q (see Fig. 3(e)). Once frame $\{t_R\}$ is initialized, again a relaxation process is needed in order to fix erroneous bonds (see 2.2). We found that about $50-100$ additional Metropolis updating iterations are sufficient to obtain satisfactory 3D segmentation results (see Fig. 3(f)).

2.6. Disparity from stereo-segment correspondences

In weakly-textured scenes, segment correspondences can be used to derive the disparity map of the scene in situations

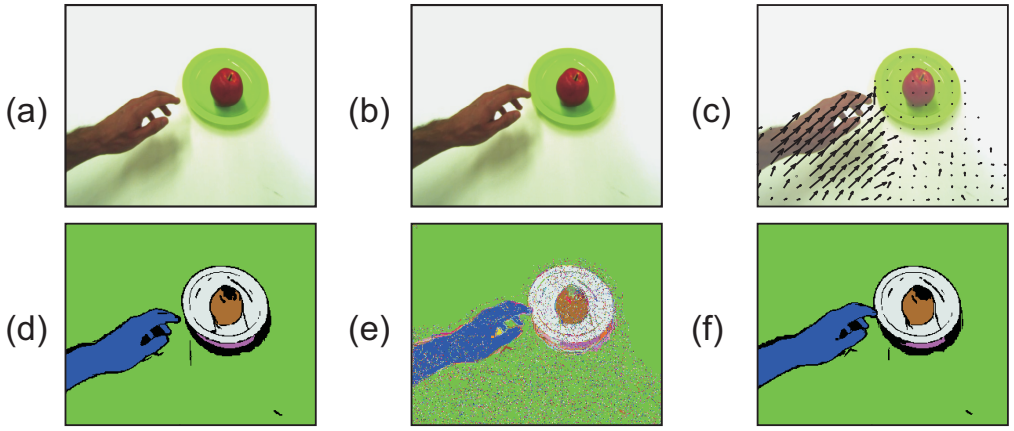


Figure 2. Segmentation of two adjacent frames in a sequence. (a) Original frame $\{t\}$. (b) Original frame $\{t+1\}$. (c) Estimated optical flow field from phase-based method (subsampled 16 times and scaled 5 times). (d) Extracted segments S_t for frame $\{t\}$. (e) Initialization of frame $\{t+1\}$ after the spin transfer. (f) Extracted segments S_{t+1} for frame $\{t+1\}$.

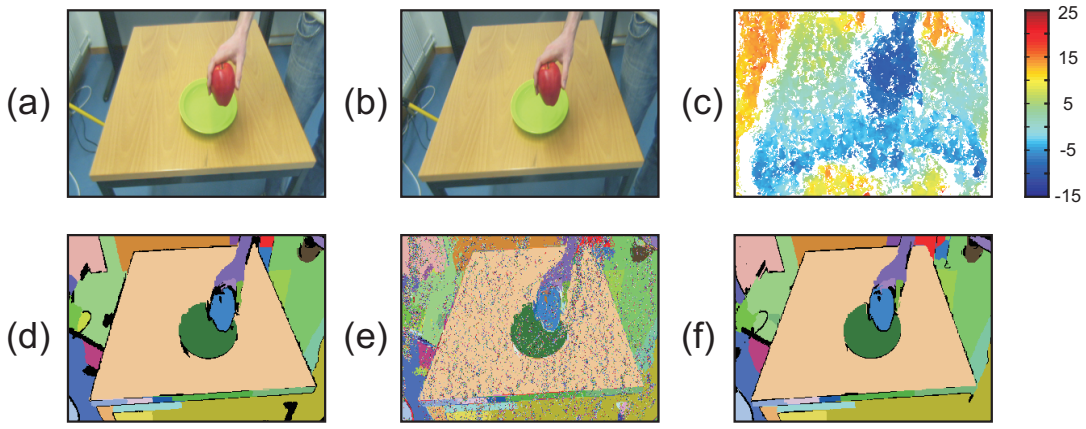


Figure 3. Segmentation of a stereo pair. (a) Original left frame $\{t_L\}$. (b) Original right frame $\{t_R\}$. (c) Estimated disparity map from phase-based stereo. (d) Extracted segments S_L^t for frame $\{t_L\}$. (e) Initialization of frame $\{t_R\}$ after the spin transfer. (f) Extracted segments S_R^t for frame $\{t_R\}$.

where texture-based methods are bound to fail [7]. Following the method proposed by [7], stereo segments are computed first using our framework, then, reliable disparity estimates from segment boundaries and weak inner-segment texture are extracted together with an occlusion map, which is derived from the approximate depth ordering of the stereo segments. Finally the information is fused and a segment-constrained interpolation algorithm is employed to obtain a dense disparity map (see Fig. 4(c) and Fig. 5(c)). The disparity map can then be used to establish 3D relations between segments in the graphical representation (see 2.7). We further defined a confidence value for each segment by summing the input confidence values (before interpolation - see [7]) for the respective segment and dividing it by the total size of the segment. Large segments for which little inner-segment disparities or edge disparities could be found will receive a lower confidence than smaller segments for which, e.g., the disparities are known along the whole segment boundary. Using this approach, we found that for the large white table of the sequences shown in Fig. 4(a) and Fig. 5(a), disparities could be interpolated only with very low confidence. This is because most of the boundaries of the table are cut off by the frame boundaries and because the table has no texture. As a consequence, stereo cannot provide sufficient depth information here.

2.7. 3D Semantic Scene Graphs

Once 3D segments are extracted, we represent the scene by undirected and unlabeled semantic graphs. The graph nodes are the segment labels and plotted at the center of each segment. The nodes are then connected by an edge if the segments are neighbors and their depth differences are less than a predefined threshold value (see Fig. 4(d) and Fig. 5(d)). We also define a 3D field of view boundary and ignore segments whose depth value does not exceed this boundary. With this method we create a focus of attention and ignore objects outside the 3D boundary.

In the temporal domain, 3D scene graphs 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 action 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 3D graphs [21]. A change in the eigenvalues or eigenvectors then corresponds to a structural change of the graph. As experimental data we use two real stereo-image sequences (see Section 3). The total frame number of the first image sequence "Making a sandwich" is 396, however, after extracting the main graphs, only 12 frames are left, each defining a single action primitive. Due to page limitations Fig. 4(d) shows only 7 of the main graphs. In the second image sequence we have interleaved chained actions, i.e., "Cutting a salami", "Making a sandwich", and "Putting on a plate", making a total of 2977 frames. In Fig. 5(d), 6 sample main graphs are shown, each representing an action primitive.

2.8. Experimental environment

As hardware platforms for our 3D segment tracking framework we use one NVIDIA card GeForce GTX 295 (with 896 MB device memory) consisting of two GPUs each of which has 30 multiprocessors and 240 processor cores in total and CPU 2.2GHz AMD Phenom Quad 9550 (using a single core) with 2 GB RAM.

We use the first GPU of the card to calculate optical flow and disparity. The second GPU is mainly used for frame-wise image segmentation. Using two GPUs allows us to run some parts of the framework physically in parallel and achieve better processing time as compared to having one GPU only.

3. Experimental Results

The developed framework is applied to two real stereo-image sequences: "Making a sandwich" (see Fig. 4(a)) and a sequence consisting of interleaved chained actions, i.e., "Cutting a salami", "Making a sandwich", and "Putting on a plate" (see Fig. 5(a)). In the first example (see Fig. 4(a)) two arms are appearing in the scene, putting the salami and cheese slices on a piece of bread, and then leaving the scene. The second sequence (see Fig. 5(a)) consists of two arms that are first taking a bread from a toaster, putting a piece of a cheese on it, and then cutting off a slice of salami with a knife. After putting the salami on top of the cheese, the sandwich is being placed on a plate and the arms are leaving the scene. Respective image segments of some sample frames from the left sequences are given in Fig. 4(b) and 5(b). Dense disparity maps obtained for extracted stereo segments are given in Fig. 4(c) and 5(c). The low-confidence-value area of the table segment is depicted with a black color in the dense disparity maps (see Section 2.6). Fig. 4(d) and 5(d) illustrate 3D semantic scene graphs of the selected frames. The graphs show that all relevant object parts in the scene can be represented by unique labels and tracked during the whole image sequence. Moreover, each graph defines a topological change in the scene. For instance, in frame number 112 of the first image sequence (see Fig. 4(d)) we observe that the arm represented by graph node number 113 has an edge only with the table (graph node number 3). In 2D, the arm would have an edge with the cheese since their respective segments are 2D neighbors, but this edge is ignored because the depth difference between the arm and the cheese is too large. In the next main graph (frame number 167) an edge connects the arm (node number 113) with the salami (node number 247) because the segments are touching in 3D.

For mono-image sequences with a frame size of 256×320 pixels a processing time of 23 Hz was achieved that demonstrates the applicability of the framework to mono-video processing tasks in real-time or close to real-time. However, for sequences containing weakly-textured regions more additional Metropolis updating iterations are needed in order to achieve a final stable configuration (see 2.4). For the stereo video segmentation with the same frame size the maximum processing time that can be achieved for the moment is 10 Hz. However this is not the case for all stereo-image sequences.

4. Discussion

In this work we presented a novel highly parallel framework for deriving a condensed 3D semantic representation of visual scenes based on a near real-time segment tracking procedure implemented in parallel on GPUs. We achieved (i) stereo image-sequence segmentation and segment track-

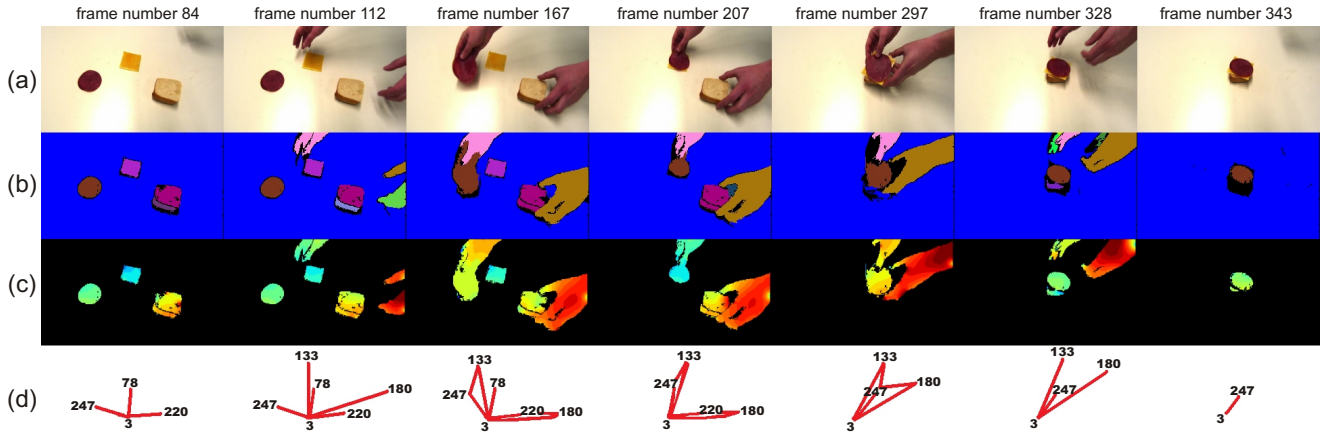


Figure 4. Results for the sample action "Making a sandwich". (a) Original frames from the left image sequence. (b) Extracted segments for frames of the left sequence. (c) The dense disparity maps obtained for extracted stereo segments. The disparity values are color-coded from blue (small) to red (large). Areas of low confidence are colored black, i.e., the uniform and untextured area of the table, for which only poor disparity results could be obtained. (d) Final 3D semantic scene graphs, representing action primitives.

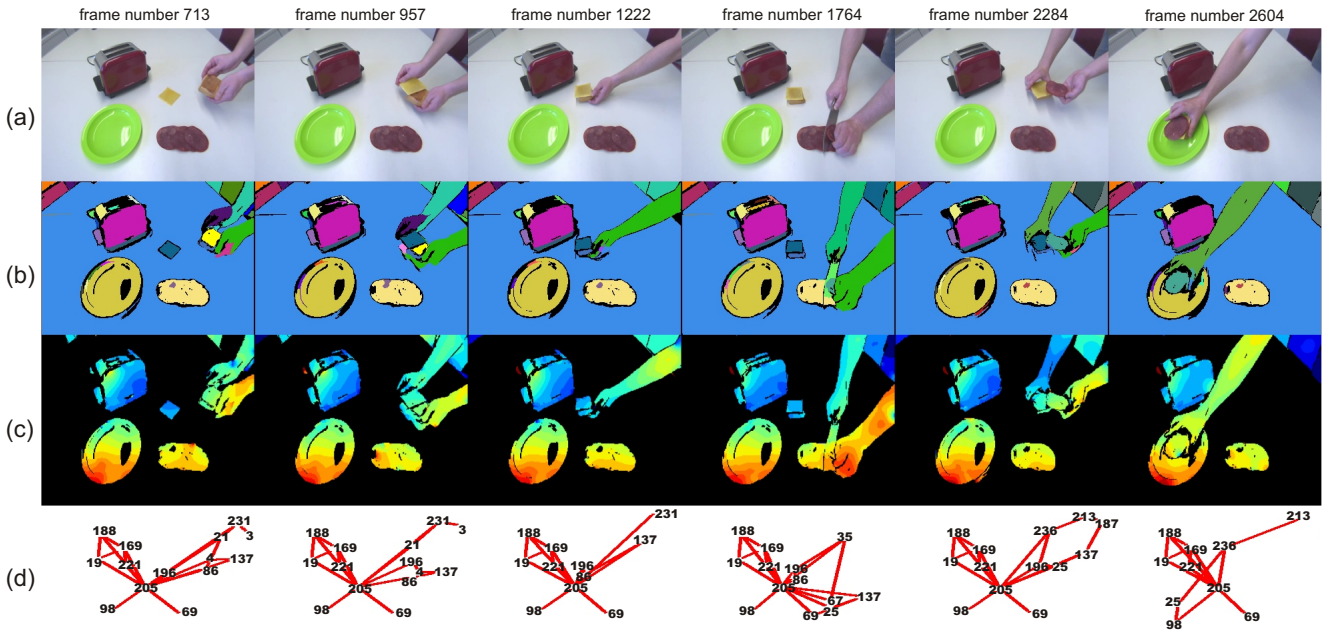


Figure 5. Results for the interleaved sample actions, i.e. "Cutting a salami", "Making a sandwich", and "Putting on a plate". (a) Original frames from the left image sequence. (b) Extracted segments for frames of the left sequence. (c) The dense disparity maps obtained for extracted stereo segments. The disparity values are color-coded from blue (small) to red (large). Areas of low confidence are colored black, i.e., the uniform and untextured area of the table, for which only poor disparity results could be obtained. (d) Final 3D semantic scene graphs, representing action primitives.

ing in near real time using a highly parallel architecture involving two GPUs, and, in a subsequent step, (ii) to build 3D semantic graphs using a recent stereo method for defining the 3D segment relations and exact graph matching for the extraction main graphs (action primitives). Algorithmic parts for (ii) are implemented on CPU and are not yet running in real-time, but potential parallel solutions are currently investigated. The main contribution of this work is the provision of a novel framework for representing image sequences in an efficient way, which may serve as vision-

front end for cognitive robots in the future. Most importantly, the segment tracking is entirely data driven (model-free) and thus does not require prior data models to be provided, making the method more robust and more widely applicable.

To our knowledge, the quite extensive framework presented in this paper is the first of its kind. Although video segmentation has been proposed before, the methods used therein are often model based, or not running real time [5, 20, 23, 8, 16, 28, 26, 14, 6]. Furthermore, our segment tracking procedure is developed for a specific goal, i.e. the construction of 3D semantic graphs, which has not been attempted in this way before.

The proposed framework has been applied to several real stereo-image sequences. Obtained results demonstrate stability of the segment tracking procedure both for mono-image sequences and for stereo-image sequences. For the segmentation of mono-image sequences with a frame size of 256×320 pixels we obtained processing time sufficient for real-time or close to real-time applications. Stereo-image sequence segmentation is not real time yet, but the developed framework is sufficiently fast for being applicable to robot-real-time 3D applications.

The algorithm has some weak points. At the moment, relaxation times increase if two previously disjoint regions ought to be joined. The processing time of stereo-video segmentation also depends on texture information and the size of occluded areas. For sequences with weakly textured areas and relatively large occlusions a longer relaxation process is required to label those areas. Currently, we are developing an improved parallel Metropolis procedure to cope with these challenges.

Furthermore, problems arise for scenarios with relatively high complexity. By high complexity we mean actions that entail splitting, e.g., cutting actions. In this case some parts become new independent segments, so new labels have to be assigned to them. Clearly, these problems cannot be resolved on the pixel level. A high level procedure is needed for the detection of object splittings. We are currently investigating this problem in more detail.

In the future, we aim to apply the developed framework to tasks in cognitive robotics. Since each main graph represents an action primitive and also the respective object relations, we are planning to use this information to recognize and classify the actions and categorize objects based on their role during the action.

5. Acknowledgements

The work has received support from the BMBF funded BCCN Göttingen and the EU Project PACO-PLUS under Contract No. 027657.

References

[1] Authors. Real-time image segmentation on a GPU. In *Facing the Multicore-Challenge, Submitted*. 4

[2] G. T. Barkema and T. MacFarland. Parallel simulation of the Ising model. *Physical Review E*, 50(2):1623–1628, 1994. 4

[3] M. Blatt, S. Wiseman, and E. Domany. Superparamagnetic clustering of data. *Physical Review Letters*, 76(18):3251–3254, 1996. 3

[4] P. Carnevali, L. Coletti, and S. Patarnello. Image processing by simulated annealing. *IBM Journal of Research and Development*, 29(6):569–579, 1985. 3

[5] J. G. Choi, S. W. Lee, and S. D. Kim. Spatio-temporal video segmentation using a joint similarity measure. *IEEE Trans. Circuits Syst. Video Technol.*, 7:279–286, 1997. 2, 8

[6] B. Dellen, E. E. Aksoy, and F. Wörgötter. Segment tracking via a spatiotemporal linking process including feedback stabilization in an n-d lattice model. *Sensors*, 9(11):9355–9379, 2009. 2, 8

[7] B. Dellen and F. Wörgötter. Disparity from stereo-segment silhouettes of weakly-textured images. In *British Machine Vision Conference*, London, UK, 2009. 2, 5

[8] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:800–810, 2001. 2, 8

[9] C. Eckes and J. C. Vorbrüggen. Combining data-driven and model-based cues for segmentation of video sequences. *World Congress on Neural Networks, San Diego*, 1996. 3

[10] D. Geman and S. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):721–741, 1984. 3

[11] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary detection by constrained optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):609–628, 1990. 3

[12] P. König and N. Krüger. Perspectives: Symbols as self-emergent entities in an optimization process of feature extraction and predictions. *Biological Cybernetics*, 94(4):325–334, 2006. 1

[13] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. of Chem. Phys.*, 21(11):1087–1091, 1953. 3

[14] V. Mezaris, I. Kompatsiaris, and M. G. Strintzis. Video object segmentation using bayes-based temporal tracking and trajectory-based region merging. *IEEE Trans. Circuits Syst. Video Technol.*, 14(6):782–795, 2004. 2, 8

[15] R. Opara and F. Wörgötter. A fast and robust cluster update algorithm for image segmentation in split-lattice models without annealing - visual latencies revisited. *Neural Computation*, 10(6):1547–1566, 1998. 3

[16] L. Patras, E. A. Hendriks, and R. L. Lagendijk. Video segmentation by map labeling of watershed segments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:326–332, 2001. 2, 8

[17] K. Pauwels and M. Van Hulle. Realtime 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, 2008. 2, 4

[18] R. B. Potts. Some generalized order-disorder transformations. *Proc. Cambridge Philos. Soc.*, 48:106–109, 1952. 3

[19] S. Sabatini, G. Gastaldi, F. Solari, J. Diaz, E. Ros, K. Pauwels, M. Van Hulle, N. Pugeault, and N. Krüger. Compact and accurate early vision processing in the harmonic space. In *International Conference on Computer Vision Theory and Applications*, pages 213–220, Barcelona, 2007. 2, 4

[20] P. Salembier and F. Marques. Region-based representations of image and video: Segmentation tools for multimedia services. *IEEE Trans. Circuits Syst. Video Technol.*, 9:1147–1169, 1999. 2, 8

[21] M. F. Sumsi. *Theory and Algorithms on the Median Graph. Application to Graph-based Classification and Clustering*. PhD thesis, Universitat Autònoma de Barcelona, 2008. 6

[22] R. Swendsen and S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Physical Review Letters*, 76(18):86–88, 1987. 3

[23] E. Tuncel and L. Onural. Utilization of the recursive shortest spanning tree algorithm for video-object segmentation by 2-d affine motion modeling. *IEEE Trans. Circuits Syst. Video Technol.*, 10:776–781, 2000. 2, 8

[24] C. von Ferber and F. Wörgötter. Cluster update algorithm and recognition. *Physical Review E*, 62(2):1461–1464, 2000. 3

[25] J. C. Vorbrüggen. Zwei modelle zur datengetriebenen segmentierung visueller daten. *Frankfurt am Main, Harri Deutsch, Thun*, 1976. 3

[26] D. Wang. Unsupervised video segmentation based on waterheds and temporal tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 8:539–546, 1998. 2, 8

[27] U. Wolff. Collective Monte Carlo updating for spin systems. *Physical Review Letters*, 62(4):361–364, 1989. 3

[28] Y. Yokoyama, Y. Miyamoto, and M. Ohta. Very low bit rate video coding using arbitrarily shaped region-based motion compensation. *IEEE Trans. Circuits Syst. Video Technol.*, 5:500–507, 1995. 2, 8

Categorizing Object-Action Relations from Semantic Scene Graphs

Eren Erdal Aksoy, Alexey Abramov, Florentin Wörgötter, and Babette Dellen

***Abstract*— In this work we introduce a novel approach for detecting spatiotemporal object-action relations, leading to both, action recognition and object categorization. Semantic scene graphs are extracted from image sequences and used to find the characteristic main graphs of the action sequence via an exact graph-matching technique, thus providing an event table of the action scene, which allows extracting object-action relations. The method is applied to several artificial and real action scenes containing limited context. The central novelty of this approach is that it is model free and needs *a priori* representation neither for objects nor actions. Essentially actions are recognized without requiring prior object knowledge and objects are categorized solely based on their exhibited role within an action sequence. Thus, this approach is grounded in the affordance principle, which has recently attracted much attention in robotics and provides a way forward for trial and error learning of object-action relations through repeated experimentation. It may therefore be useful for recognition and categorization tasks for example in imitation learning in developmental and cognitive robotics.**

I. INTRODUCTION

One central goal for humanoid robotics is to imitate, understand, and learn from human behavior. Part of this problem is to relate a manipulation to its manipulated object. The difficulty lies here in the fact that individual manipulations even when "doing the same thing" can take vastly different forms just due to changes in posture, action sequence, and/or differences in the general (visual) context surrounding the core manipulation. Nonetheless humans have no problem in classifying manipulation types, such as "moving an object", "closing a book", "making a sandwich" or "filling liquid", and to link objects with actions. The goal of this paper is to devise a method which can, at least to some degree, do the same and thereby classify manipulation types.

Recently, these questions have been approached in an abstract way by the concept of object-action complexes (OACs) [1], [2], claiming that objects and actions are inseparably intertwined. This is linked to the way humans perceive the world by relating objects with actions. The OAC concept proposes such a human-like description by which an object is identified considering both its (visual) properties *and* the actions that have been performed with it. The OAC concept attaches the performed actions to the objects as attributes. This approach is therefore related to the affordance principle [3], which especially in the recent years has had increasing influence in robotics [4]. Take, for example, a cup, which is an entity for filling and drinking. However, not only this single specific cup but any other cylindrical, hollow object could be used for the same actions. Thus, objects, which are supporting common actions, can be considered similar. Filling creates the object-type "container"! Consider now an inverted cup, which cannot be filled. Now the former container has become a pedestal on which you could put something. While physically the same thing, a "pedestal" is a different object type altogether. In cognitive vision, many new approaches for object recognition from 3D models have been introduced [5], [6], [7]. However, these model-based approaches cannot identify object-action relations. In this work, we introduce a novel approach for detecting spatiotemporal object-action relations using semantic scene graphs, leading to both action recognition and object categorization. Using this method, objects are connected to recognized actions considering their roles within a scenario.

The approach relies on a front-end algorithm which allows for the continuous tracking of scene segments using superparamagnetic clustering, with proven convergence properties [8], [9], [10], [11]. The presented core algorithm, used for recognition and classification, then relies on the sequence of neighborhood relations between those segments, which for a given action will always be "essentially" the same. Hence different from feature based (or model based) approaches our system operates on object-part relations without presupposing assumptions about the structure of object and action. Thus, it is model free. This leads to a high degree of invariance against position, orientation, etc. but we need to make sure that segment tracking is stable, which is currently achieved by several means described elsewhere [22]. Furthermore, at this stage we show examples from a 2D (projected) domain. True 3D tracking is currently being implemented for difficult action sequences like "making a complete breakfast".

Therefore, we would like to emphasize that the core contribution of this work is the novel categorization method. The computer vision front end is a required prerequisite, but other tracking methods could be used here as well and improvements are possible.

The structure of the paper is as follows. In Section II we discuss related works. In Section III we introduce the action classification and the object categorization algorithm. In Section IV experimental results with real images are presented.

The work has received support from the BMBF funded BCCN Göttingen, Grant No. 01GQ0430, Project 01GQ0432, and the EU Project PACO-PLUS. We thank Tomas Kulvicius for valuable discussion.

Eren Erdal Aksoy, Alexey Abramov, and Florentin Wörgötter are with Bernstein Center for Computational Neuroscience, University of Göttingen, Friedrich-Hund Platz 1, 37077 Göttingen, Germany eaksoye,abramov,worgott@bccn-goettingen.de

B. Dellen is with Bernstein Center for Computational Neuroscience, Max-Planck Institute for Dynamics and Self-Organization, Bunsenstr. 10, 37073 Göttingen, Germany and Institut de Robotica i Informatica Industrial (CSIC-UPC), Llorens i Artigas 4-6, 08028 Barcelona, Spain bkdellen@bccn-goettingen.de

In Section V we show directions for future research and how to extend the proposed framework for the process of more complex and longer scenes. Finally, in Section VI the results are discussed.

II. RELATED WORK

Action recognition and object categorization have received increasing interest in the Artificial Intelligence (AI) and cognitive-vision community during the last decade. The problem of action recognition has been addressed in previous works, but only rarely in conjunction with object categorization. Modayil *et al.* (2008) presented a framework focusing on the recognition of activities in daily living [12]. In order to detect the activities, the test subject (e.g. human) was equipped with a Radio Frequency Identification (RFID) reader and tags. The types of actions and used objects were recorded by the RFID reader to learn a model that recognizes the activities performed during observations, and an Interleaved Hidden Markov Model (HMM) was used to increase the accuracy of the learned model. Similar to this study, Liao *et al.* (2005) provided an approach to perform location-based activity recognition by using Relational Markov Networks [13]. This work also covered high-level activities, e.g. working, shopping, dining out, during long periods of time. The system used data from a wearable GPS location sensor and considered time, place of action, and sequence of action, which were extracted from the GPS sensor. Although those kinds of sensor-based multitasking-activity-recognition approaches provide promising results, they do not cover object-categorization issues and have handicaps like limited coverage area. Hongeng (2004) introduced a Markov network to encode the entire event space for scenes with limited context but without considering object classification [14]. Sridhar *et al.* (2008) showed that objects can also be categorized by considering their common roles in actions, resulting however in large and complex activity graphs, which have to be analyzed separately [15]. Li *and* Lee (2000) introduced sub-scene graph matching method just for object recognition, combining it with a Hopfield neural network to get local matches between graphs [16]. Our framework provides a novel approach that represents scenes by semantic graphs which hold spatiotemporal object-action relations. By analyzing semantic scene graphs, we not only recognize actions but also categorize objects based on their action roles.

III. METHODS

A. Overview of the Algorithm

In the current study, we analyze movies of scenes containing limited context. Fig. 1 shows the block diagram of the algorithm. As a first processing step, image segments are extracted and tracked throughout the image sequence, allowing the assignment of temporally-stable labels to the respective image parts [9], [11]. The scene is then described by semantic graphs, in which the nodes and edges represent segments and their neighborhood relations, respectively. For segmentation and graph examples of real images, see Fig. 2. Graphs can change by continuous distortions (lengthening or shortening of edges) or, more importantly, by 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 after a breaking point and extract the next following topological main graph. The sequence of these main graphs thus represents all structural changes in the scene. The temporal order by which those main graphs follow each other defines an "event table". An event signifies that something has happened in the scene which caused a true topological change in the graph. This method allows classifying object-action relations by calculating the similarity between event tables from different scenes. Furthermore, nodes playing the same role in an classified action sequence can be identified and then be used to categorize objects by returning to the signal level via image segments.

B. Segmentation and Tracking

We use an image-segmentation method in which segments are obtained trough a 3D linking process [9], [11], [10]. First, a spin variable σ_i is assigned to each pixel i of the stereo image. To incorporate constraints in form of local correspondence information, we distinguish between neighbors within a single frame (2D bonds) and neighbors across frames (3D bonds). We create a 2D bond $(i,k)_{2D}$ between two pixels within the same frame with coordinates (x_i, y_i, z_i) and (x_k, y_k, z_k) if $|(x_i - x_k)| \leq 1$, $|(y_i - y_k)| \leq 1$, and $z_i = z_k$. Across frames, we create a 3D bond $(i,j)_{3D}$ between two spins i and j if $|(x_i + d_{ij}^x - x_j)| \leq 0.5$, $|(y_i + d_{ij}^y - y_j)| \leq 0.5$, $z_i \neq z_j$, and $a_{ij} = 1$. The values d_{ij}^x and d_{ij}^y are the shifts of the pixels between frames z_i and z_j along the axis x and axis y, obtained from an initial optic flow map. The parameters a_{ij} are the respective amplitudes (or confidences). However, since the images in the examples given in this paper are changing only little from frame to frame, we will assume that the flow is zero everywhere. Hence the values d_{ij}^x and d_{ij}^y are zero, and $a_{ij} = 1$ everywhere.

The spin model is now implemented such that neighboring spins with similar color have the tendency to align. We use a q-state Potts model [17] with the Hamiltonian

$$H = -\sum_{\langle ik \rangle_{2D}} J_{ik} \delta_{\sigma_i, \sigma_k} - \sum_{\langle ij \rangle_{3D}} J_{ij} \delta_{\sigma_i, \sigma_j} \quad , \tag{1}$$

with $J_{ij} = 1 - \triangle / \bar{\triangle}$ and $\triangle_{ij} = |g_i - g_j|$, where g_i and g_j are the gray (color) values of the pixels i and j, respectively. The mean distance $\bar{\triangle}$ is obtained by averaging over all bonds.

Here, $\langle ik \rangle_{2D}$ and $\langle ij \rangle_{3D}$ denote that i,k and i,j are connected by bonds $(i,k)_{2D}$ and $(i,j)_{3D}$, respectively. The Kronecker δ function is defined as $\delta_{a,b} = 1$ if $a = b$ and zero otherwise. The segmentation problem is then solved by finding clusters of correlated spins in the low temperature equilibrium states of the Hamiltonian H. The total number M of segments is then determined by counting the computed segments. It is usually different from the total number q

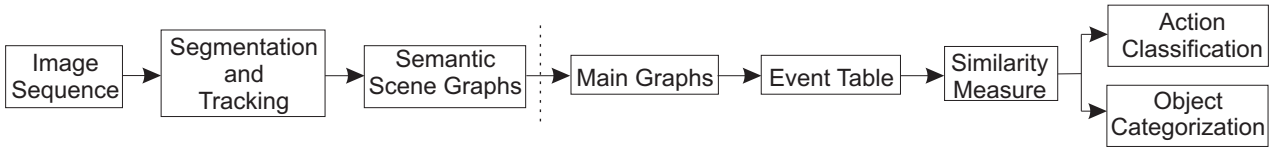


Fig. 1. Block diagram of the algorithm.

of spin states, which is a parameter of the algorithm (here $q = 10$).

We solve this task by implementing a clustering algorithm. In a first step, "satisfied" bonds, i.e. bonds connecting spins of identical spins $\sigma_i = \sigma_j$, are identified. Then, in a second step, the satisfied bonds are "frozen" with a some probability P_{ij}. Pixels connected by frozen bonds define a cluster, which are updated by assigning to all spins inside the same clusters the same new value [18]. In the method of superparamagnetic clustering proposed by [19] this is done independently for each cluster. In this paper, we will employ the method of energy-based cluster updating (ECU), where new values are assigned in consideration of the energy gain calculated for a neighborhood of the regarded cluster [8], [20]. The algorithm is controlled by a single "temperature" parameter, and has been shown to deliver robust results over a large temperature range. After a 100 iterations, clusters are used to define segments.

In this paper, we segment always two consecutive frames of the image sequence at the same time, i.e. frame i and $i+1$, then, we segment the next pair, i.e. $i+1$ and $i+2$, where the last image of the first pair is identical with the first image of the second pair. Then, consecutive pairs are connected by identifying the identical segments in the overlapping images. This strategy allows handling long motion image sequences [11].

C. Semantic Scene Graphs

Once the image sequence has been segmented and segments have been tracked, we represent the scene by undirected and unweighted labeled graphs. The graph nodes are the segment labels and plotted at the center of each segment. The nodes are then connected by an edge if segments touch each other.

Fig. 2 shows original frames with respective segments and semantic scene graphs from four different real action types: *Moving Object*, *Opening Book*, *Making Sandwich*, and *Filling Liquid*. In the *Moving Object* action a hand is putting an orange on a plate while moving the plate together with the orange (see Fig. 2(a-c)). The *Opening Book* action represents a scenario in which a hand is opening a book (see Fig. 2(d-f)). In the *Making Sandwich* action two hands are putting pieces of bread, salami, and cheese on top of each other (see Fig. 2(g-i)). The *Filling Liquid* action represents a scenario in which a cup is being filled with liquid from another cup (see Fig. 2(j-l)).

Larger sample images for each action type are shown in Fig. 3(a-d) to give an impression of the level of complexity, i.e. amount of texture, reflections, and shadows.

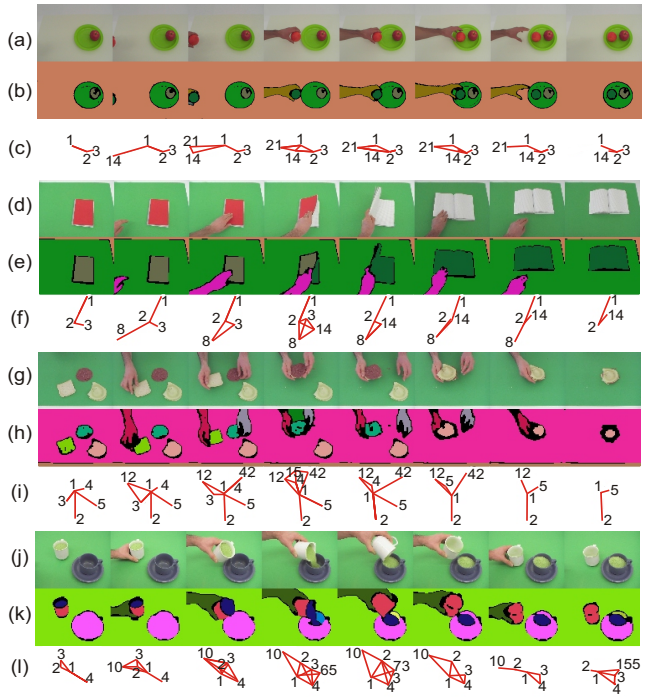


Fig. 2. Four different real action types. (a) Original images from the *Moving Object* action. (b) Respective image segments. (c) Semantic scene graphs. (d) Original images from the *Opening Book* action. (e) Respective image segments. (f) Semantic scene graphs. (g) Original images from the *Making Sandwich* action. (h) Respective image segments. (i) Semantic scene graphs. (j) Original images from the *Filling Liquid* action. (k) Respective image segments. (l) Semantic scene graphs.

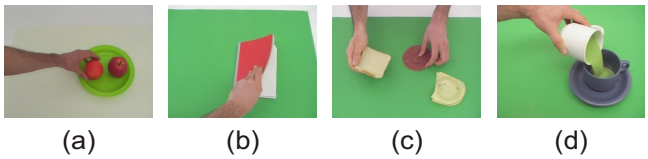
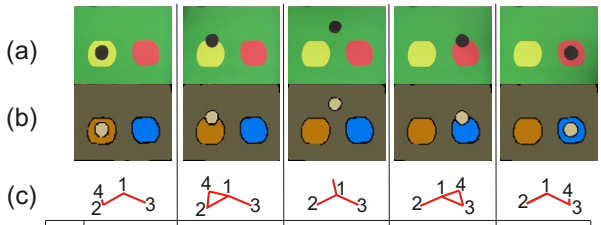


Fig. 3. Sample images taken from each real action type. (a) *Moving Object*. (b) *Opening Book*. (c) *Making Sandwich*. (d) *Filling Liquid*.

D. Main Graphs and Event Tables

In the following we will first use simpler scenes to describe the remaining parts of the algorithm (to the right of the dashed line in Fig. 1). Fig. 4(a-b) depicts original frames with respective 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.



$\rho_{2,1}$	T	T	O	O	O
$\rho_{3,1}$	O	O	O	T	T
$\rho_{4,1}$	N	T	O	T	N
$\rho_{3,2}$	N	N	N	N	N
$\rho_{4,2}$	O	T	N	N	N
$\rho_{4,3}$	N	N	N	T	O

Fig. 4. Simple example of the *Moving Object* action (sample action 1). (a) Original images. (b) Respective image segments. (c) Semantic scene graphs. (d) Event table.

In the temporal domain, scene graphs 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 action 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 [21]. A change in the eigenvalues or eigenvectors then corresponds to a structural change of the graph. The whole image sequence of the sample *Moving Object* action has 92 frames, however, after extracting the main graphs, only 5 frames are left, each defining a single action primitive (see Fig. 4(c)).

Following the extraction of the main graphs, we analyze the spatial relations between each pair of nodes in the main graphs. We denote the spatial relations by $\rho_{i,j}$ in which i and j are the nodes of interest. Note that the spatial relations are symmetric, i.e. $\rho_{i,j} = \rho_{j,i}$.

Possible spatial relations of each node pair are *absence* (A), *no connection* (N), *overlapping* (O), and *touching* (T). We define those relations by calculating the number of edges of both currently considered nodes i and j in each main graph. As an example, all possible spatial relations between the black object and yellow vessel are illustrated in Fig. 5. Since those objects are represented by graph nodes 4 and 2, we write the relation as $\rho_{4,2}$. The relation *absence* means that one of the considered nodes is not observed in the scene, i.e. the black object node 4 does not exist in the graph (see Fig. 5(a)). In the case of *no connection*, the considered nodes have no edge between them (see Fig. 5(b)). In the *overlapping* relation one of the considered nodes is completely surrounded by the other node. Therefore, the surrounded node has only one edge (see Fig. 5(c)). The *touching* relation represents the situation in which segments

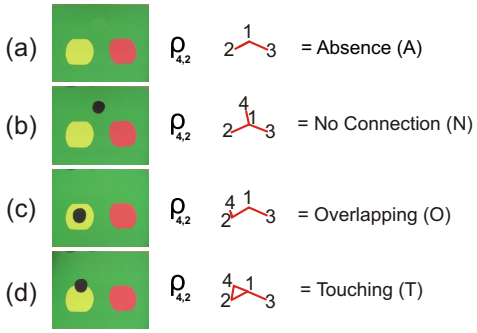


Fig. 5. Possible spatial relations between black object and yellow vessel which are represented by graph nodes 4 and 2, respectively. (a) The relation *absence*. (b) The relation *no connection*. (c) The *overlapping* relation. (d) The *touching* relation.

touch each other and both considered nodes have more than one edge (see Fig. 5(d)). More complex spatial relations between nodes are currently not considered but could be included in the future.

The total number of spatial relations is defined as

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

where n is the total number of objects. For the sample *Moving Object* action mentioned above 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 $\rho_{4,3}$.

All existing spatial node relations in the main graphs are saved in the form of a table where the rows represent spatial relations between each pair of nodes. Since any change in the spatial relations represents an event that defines an action, we refer to this table as an *event table* (ξ). Fig. 4(d) shows the *event table* of the action above. However, the fourth row of the *event table* does not hold any change in the sense of a changing spatial relation since the yellow and red vessels never move. For this reason, we ignore the fourth row. For the sake of simplicity, we substitute numbers -1, 0, 1, and 2 for possible spatial relations A, N, O, and T. The final *event table* of sample action 1 is given in Table 1.

E. Similarity Measure

So far we showed how to represent a long image sequence by an event table the dimensions of which are related to the spatial node relations in the main graphs. Next we will

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

TABLE I

EVENT TABLE (ξ_1) OF THE FIRST SAMPLE ACTION. SPATIAL RELATIONS BETWEEN THE NODES OF SAMPLE ACTION 1.

discuss how to calculate the similarity of two actions. To this end we created one more sample for the *Moving Object* action. Fig. 6 depicts the main graphs 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 table* for the second sample is calculated and given in Table 2. Note that even though the second sample contains more objects, the dimension of the event tables is accidentally the same. This makes explanations simpler, but, as we will see later, the dimensions of the event tables are not important and can even be different between two cases.

Similarity measurement of actions is based on the comparison of the event tables. Basically, each row of the first *event table* (ξ_1) is compared with each row of the second *event table* (ξ_2) in order to find the highest similarity. (For event tables with different dimensions, sub-matrices need to be used.) Considering this simple rule we start determining the similarity with the first rows of ξ_1 and ξ_2, giving [2 2 1 1 1] and [1 1 1 2 2], respectively. Those lines are written one below the other. Next, the amount of equal digits (equal relations!) are counted and divided by total number of digits. Since only one digit (third digit) out of five digits is the same, the similarity of those two rows is 20%. Once all rows have been compared with each other, the determined similarity values are saved in the form of a table where rows and columns give the similarity relations between ξ_1 and ξ_2. The resulting table is called *similarity table* (ζ) and shown in Table 3.

The final similarity measure is determined by calculating the arithmetic mean value of the highest values in each row of ζ. Consequently, our two sample actions have 100% similarity.

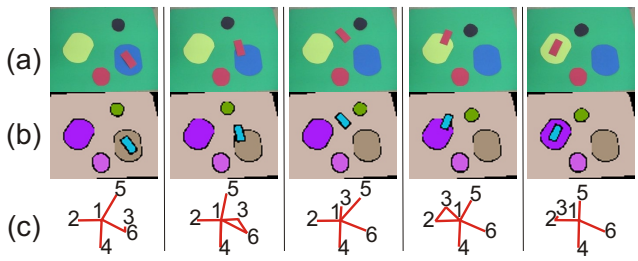


Fig. 6. Different version of the simple *Moving Object* action (sample action 2). (a) Original images. (b) Respective image segments. (c) Semantic scene graphs.

$\rho_{2,1}$	1	1	1	2	2
$\rho_{3,1}$	0	2	1	2	0
$\rho_{6,1}$	2	2	1	1	1
$\rho_{3,2}$	0	0	0	2	1
$\rho_{3,6}$	1	2	0	0	0

TABLE II

EVENT TABLE (ξ_2) OF THE SECOND SAMPLE ACTION. SPATIAL RELATIONS BETWEEN THE NODES OF SAMPLE ACTION 2.

ξ_1 \ ξ_2	$\rho_{2,1}$	$\rho_{3,1}$	$\rho_{6,1}$	$\rho_{3,2}$	$\rho_{3,6}$
$\rho_{2,1}$	20%	40%	**100%**	20%	20%
$\rho_{3,1}$	**100%**	40%	20%	20%	0%
$\rho_{4,1}$	40%	**100%**	40%	40%	40%
$\rho_{4,2}$	20%	40%	20%	20%	**100%**
$\rho_{4,3}$	20%	40%	20%	**100%**	20%

TABLE III

SIMILARITY TABLE (ζ). SIMILARITY VALUES BETWEEN ξ_1 AND ξ_2.

In case of having event tables with different dimensions, we apply a window-based search algorithm to the bigger table in order to find out a region that has the highest similarity with the smaller table. In this case, the number of total search is defined as

$$s_{\text{total}} = (|r_1 - r_2| + 1)(|c_1 - c_2| + 1) \quad , \tag{3}$$

where r_1, r_2, c_1, and c_2 are the row and column numbers of the first and second event tables. The final similarity measurement is the highest similarity observed during this total search. If the dimensions are inconsistent in size to decide which one is smaller (such as $r_1 < r_2$ and $c_1 > c_2$ or $r_1 > r_2$ and $c_1 < c_2$), the event table with less columns is extended by adding the last column until it has the same number of columns as the bigger table. This sort of operation does not affect the action content since we do not change spatial node relations in the temporal domain.

As a result we can now measure how similar the two actions are and we find 100%. Thus, these actions are of the same type ("type-similar").

F. Object Categorization

The *similarity table* also implicitly encodes the similarity of the nodes between the two different examples. Intriguingly, this can be used to extract *nodes with the same action roles* in type-similar actions. For this we first list all relations ρ of both actions with highest individual similarity. For instance, the relation between nodes 2 and 1 ($\rho_{2,1}$) in the first row has a 100% similarity with the relation between nodes 6 and 1 ($\rho_{6,1}$) in the third column. Doing this for all relations, we find the following maximal similarities in ζ:

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

Those similarity values represent the correspondences between manipulated nodes in ξ_1 and ξ_2. In order to determine these correspondences, we analyze which node number in ξ_1 is repeating in conjunction with which node number in ξ_2. We start with node number 1 in ξ_1, and obtain

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

While 1 is repeating three times in ξ_1, the same node number 1 in ξ_2 is also repeating three times. However, node numbers 2, 3, and 6 in ξ_2 occur only once. Therefore, we conclude that graph nodes 1 in both ξ_1 and ξ_2 had the same roles. In fact, both graph nodes represent the green background which plays same role in both actions.

We continue the spatial node relation analysis with node number 2 in ξ_1, and obtain

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

Node number 2 in ξ_1 is repeating twice with node number 6 in ξ_2. Those graph nodes represent the yellow and blue vessels within which the moving objects are initially located and from which they then move away.

For the case of node number 3 in ξ_1 we obtain

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

Node number 3 in ξ_1 corresponds to node number 2 in ξ_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 ξ_1 is obtained as

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

As node number 4 in ξ_1, node number 3 in ξ_2 is also repeating three times. In fact, both graph nodes represent the moving objects which are the round black object in ξ_1 and the rectangular red object in ξ_2.

In the case of having a *similarity table* which has the same highest value more than once in a column, e.g. having two times 100% similarity values in the same column, the object categorization section leads to ambiguous results, i.e. one object corresponds to two different objects. Since this sort of correspondence is not allowed in the framework, the final similarity value is calculated again by taking the second highest values into account. This way we can get rid of any kind of mismatching in the object categorization process.

IV. Results with Real Images

We applied our framework to four different real action types: *Moving Object*, *Opening Book*, *Making Sandwich*, and *Filling Liquid* (see Fig. 2). For each of these actions, we recorded four movies with different trajectories, speeds, hand positions, and object shapes. All those sixteen movies were recorded by a stable camera that was focused on the hands and the manipulated objects.

In Fig. 7, some sample frames of all four action types are shown which are different from those in Fig. 2. Here, for

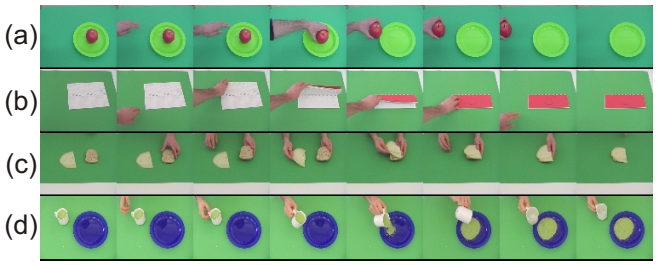


Fig. 7. Different versions of the real action types. (a) *Moving Object*. (b) *Opening Book*. (c) *Making Sandwich*. (d) *Filling Liquid*.

the *Moving Object* action a hand is *appearing* in the scene, taking an apple from a plate, and leaving the scene (see Fig. 7(a)). The *Opening Book* action type represents here a scenario in which a hand is *closing* a book (see Fig. 7(b)). In the *Making Sandwich* action two hands are putting pieces of bread and cheese on top of each other in *different* order (see Fig. 7(c)). The *Filling Liquid* action type includes a scenario in which a cup is filling a *plate* with liquid (see Fig. 7(d)). These examples were introduced to show that really different instantiations of a manipulation will still be recognized as belonging to the same type.

Event tables of each real test data are compared with each other. The resulting similarity values are given in Fig. 8. Each test data has at least 69% similarity with the other versions of its type-similar action (see close to diagonal). In general the similarity between type-similar actions is for all scenes much bigger than the similarity between non-type-similar actions, except in one case. For the fourth version of *Making Sandwich* and the fourth version of *Moving Object* we receive a large similarity of 57%. This may happen in some cases when action primitives are quite similar and, in addition, noise in the data leads to a few spurious nodes and

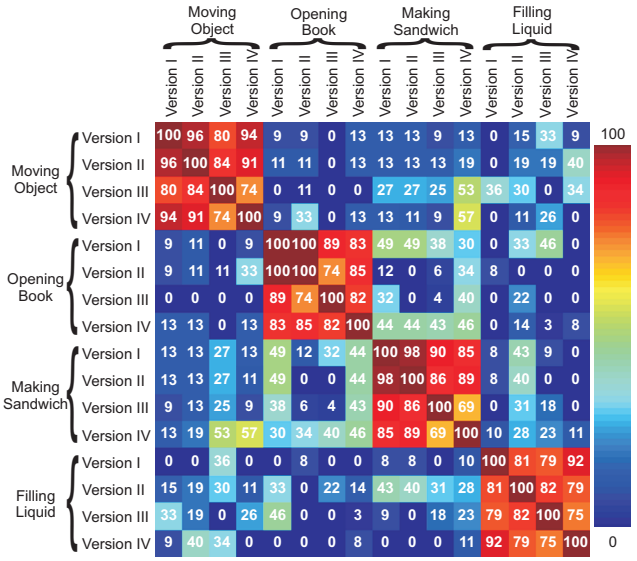


Fig. 8. Action-classification results. Similarity values between event tables of the real test data set.

false tracking. During tests with real images the accuracy of the whole algorithm is observed as 91%.

Moreover, the results showed that the manipulated objects in each action type can be categorized according to their roles in the actions. Fig. 9 illustrates the categorization results of objects that performed same roles in different versions of actions. As an example, the apple and orange are in the same group since they are being manipulated in the *Moving Object* action.

Notwithstanding some remaining problems, the results shown here clearly demonstrate that it is possible to classify objects and actions in scenes with limited context without prior (model) knowledge.

V. Extending the Algorithm

So far we showed how the algorithm works in 2D. Next we will discuss how to extend the algorithm to 3D in order to apply our framework to more complex and even longer scenes containing high-level context. To this end we have to introduce model free stereo-video segmentation and stereo segment tracking leading to 3D scene graphs [22]. As an example, Fig. 10(a) shows sample frames of disentangling a complete "making a breakfast" stereo sequence. In this scenario two arms are first taking a bread from a toaster, putting a piece of cheese on it, and then cutting off a slice of salami with a knife. After putting the salami on top of the cheese, the sandwich is being placed on a plate and the arms are leaving the scene. Respective image segments of the sample frames from the left stereo image sequences are given in Fig. 10(b). The corresponding dense disparity maps obtained for extracted stereo segments are shown in Fig. 10(c) [10]. Therein the low-confidence-value area of the table segment is depicted with a black color. Fig. 10(d) illustrates 3D semantic scene graphs of the selected frames. In 3D graphs, edges show that the segments are neighbors and their depth differences are less than a predefined threshold value. While the number of segments might much increase in more complex scenes, the number of *consistently changing* edges will remain small as real 3D changes of touching relations (valid derivatives) are rare. Thus we are currently improving the presented algorithm in two ways: 1) We are decomposing action sequences into action sub-sequences and analyze each of them separately and 2) We are also compressing the event tables by taking their derivatives. This way we are decreasing complexity and also computation time.

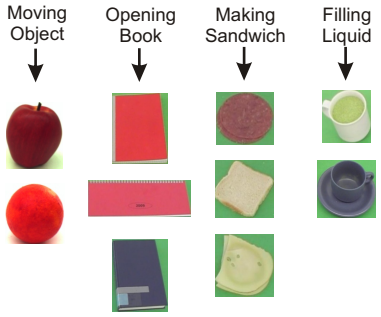


Fig. 9. Object categorization results. In each action type the manipulated objects can be detected based on their action roles.

Furthermore, determining similarity values between each action makes the whole system computational expensive, especially if the database contains a lot of training data. In order to avoid this problem, we plan to construct a template main-graph model for each kind of action. Template graph models can be constructed by considering the main graphs of a scenario which accurately represents the respective action type. Actions will then be classified by calculating the similarity values with those template models instead of with one another. In addition to this, we intend to let the agent learn the template main-graph models from a training data set. To achieve all this a parallel implementation of the framework on GPUs for real-time robotics applications is currently being implemented.

VI. Discussion

We presented a novel algorithm that represents a promising approach for recognizing actions *without* requiring prior object knowledge, and for categorizing objects solely based on their exhibited role within an action sequence. Our framework is mainly based on the analysis of object relations in the spatiotemporal domain. We are aware of the fact that "segment permanence" (i.e., reliable tracking) needs to be assured without which our method would fail. Clearly on the computer vision side improvements can be made to better assure this. This, however, is not the point of this paper. As far as we see it this is one of the first papers in which the categorization of object-action relations has become possible in a model free way. This procedure can thus be entirely based on the experimentation of the robot (here simulated by a human). Hence we arrive at a very high sub-symbolic representational level in a fully grounded way. From there on the grounded development and the learning of symbols (for example verbal utterances) which describe actions should be easier than before and this has been deemed as one of the major challenges in cognitive robotics. Furthermore, it should also be possible to "backwards unwrap" the learned event tables (the OACs) and this way *generate* an action. Obviously complex inverse kinematic (and dynamic) problems need to addressed to arrive at an actual movement sequence. However, the event graphs specify the fundamental "breaking points" whenever certain object relations change. Therefore movement segments between two such breaking points could be seen as motor primitives. The execution of such a primitive may then be optimized by whatever means but one always must assure that its starting point (prior) and its endpoint (posterior) corresponds to two subsequent entries in the event table.

The proposed algorithm has been applied to four different real action sequences of scenes containing limited context. Each action type had four different versions which differed in trajectories, speeds, hand positions, and object shapes. The experimental results showed that the agent can categorize all these action types by measuring the amount of similarity between action sequences and also categorize the participating

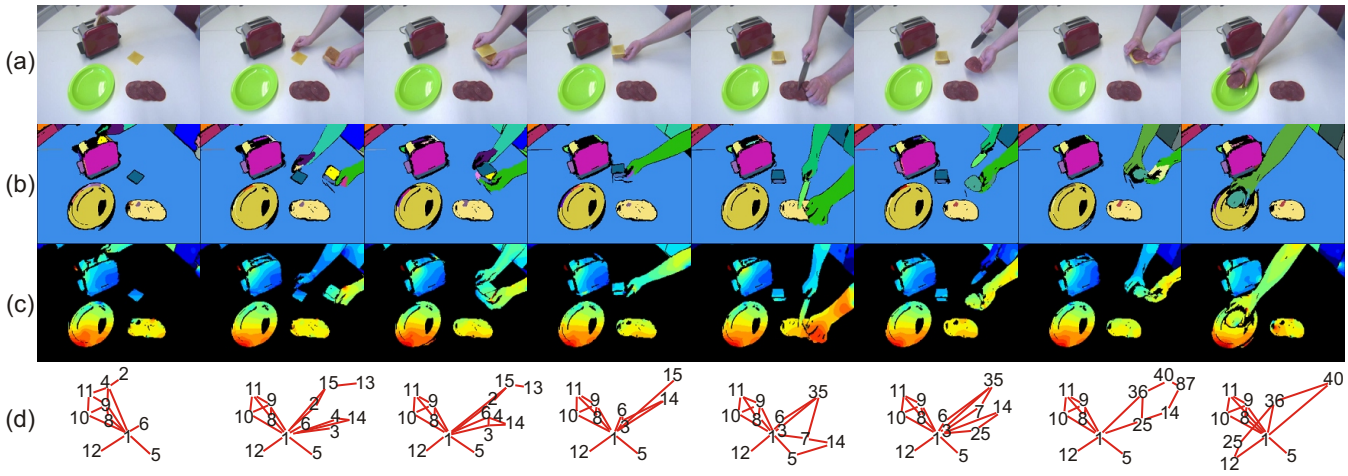


Fig. 10. A sample "making a breakfast" stereo sequence. (a) Original frames from the left image sequence. (b) Extracted segments for frames of the left sequence. (c) The dense disparity maps obtained for extracted stereo segments. The disparity values are color-coded from blue (small) to red (large). Areas of low confidence are colored black, i.e., the uniform and untextured area of the table, for which only poor disparity results could be obtained. (d) Final 3D semantic scene graphs.

manipulated objects according to their roles in the actions.

Several extensions of this algorithmic framework will be pursued in the future as we discussed above.

In summary, this study is one of the first to show that it is indeed possible to treat objects and actions as conjoint entities as suggested by the abstract idea of object-action complexes (OACs, [1], [2]). This is the first description of our new approach and the discussion above shows that it seems to have high potential. In general this contribution shows that this complex concept is algorithmically treatable and therefore we believe that the OAC indeed provides a promising approach for treating problems involving cause-effect relations in cognitive robotics.

References

[1] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr, "Cognitive agents - a procedural perspective relying on predictability of object-action complexes (oacs)," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 420–432, 2009.

[2] N. Krüger, J. Piater, C. Geib, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrcen, A. Agostini, and R. Dillmann, "Object-action complexes: Grounded abstractions of sensorimotor processes (submitted)," *Robotics and Autonomous Systems*, 2009.

[3] J. Gibson, "The ecological approach to visual perception." Boston: Houghton Mifflin, 1979.

[4] S. Hart and R. Grupen, "Intrinsically motivated affordance learning," in *Workshop on Approaches to Sensorimotor Learning on Humanoids, IEEE Conference on Robotics and Automation (ICRA)*, 2009.

[5] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2004.

[6] C. H. Andez, C. Hern, E. Esteban, and F. Schmitt, "Silhouette and stereo fusion for 3d object modeling," *Computer Vision and Image Understanding*, vol. 96, pp. 367–392, 2004.

[7] M. Tomono, "3d object modeling and segmentation based on edge-point matching with local descriptors," in *ISVC '08: Proceedings of the 4th International Symposium on Advances in Visual Computing*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 55–64.

[8] R. Opara and F. Wörgötter, "A fast and robust cluster update algorithm for image segmentation in spin-lattice models without annealing - visual latencies revisited," *Neural Computation*, vol. 10, pp. 1547–1566, 1998.

[9] N. Shylo, F. Wörgötter, and B. Dellen, "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)*, 2009.

[10] B. Dellen and F. Wörgötter, "Disparity from stereo-segment silhouettes of weakly textured images," in *Proceedings of the British Machine Vision Conference*, 2009.

[11] B. Dellen, E. E. Aksoy, and F. Wörgötter, "Segment tracking via a spatiotemporal linking process in an n-d lattice model," *Sensors*, vol. 9, no. 11, pp. 9355–9379, 2009.

[12] J. Modayil, T. Bai, and H. Kautz, "Improving the recognition of interleaved activities," in *Proceedings of the 10th international conference on Ubiquitous computing*, 2008, pp. 40–43.

[13] L. Liao, D. Fox, and H. Kautz, "Location-based activity recognition using relational markov networks," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005, pp. 773–778.

[14] S. Hongeng, "Unsupervised learning of multi-object event classes," in *Proc. 15th British Machine Vision Conference*, 2004, pp. 487–496.

[15] M. Sridhar, G. A. Cohn, and D. Hogg, "Learning functional object-categories from a relational spatio-temporal representation," in *Proc. 18th European Conference on Artificial Intelligence*, 2008.

[16] L. Wen-Jing and L. Tong, "Object recognition by sub-scene graph matching," in *ICRA '00: Proceedings of the 20th IEEE International Conference on Robotics and Automation*, 2000.

[17] R. B. Potts, "Some generalized order-disorder transformations," *Proc. Cambridge Philos. Soc.*, vol. 48, pp. 106–109, 1952.

[18] R. H. Swendsen and S. Wang, "Nonuniversal critical dynammics in monte carlo simulations," *Physical Review Letters*, vol. 76, no. 18, pp. 86–88, 1987.

[19] M. Blatt, S. Wiseman, and E. Domany, "Superparametric clustering of data," *Physical Review Letters*, vol. 76, no. 18, pp. 3251–3254, 1996.

[20] C. von Ferber and F. Wörgötter, "Cluster update algorithm and recognition," *Physical Review E*, vol. 62, pp. 1461–1664, 2000.

[21] M. F. Sumsi, "Theory and algorithms on the median graph. application to graph-based classification and clustering," Ph.D. dissertation, Universitat Autonoma de Barcelona, 2008.

[22] A. Abramov, E. E. Aksoy, B. Dellen, J. Doerr, and F. Wörgötter, "3d semantic representation of actions from efficient stereo-image-sequence segmentation on gpus (submitted)," in *3DPVT*, 2010.