Ascertaining relevant changes in visual data by interfacing AI reasoning and low-level visual information via temporally stable image segments.

Nataliya Shylo[1], Florentin Wörgötter[1], and Babette Dellen[2,3]

Abstract— **Action planning and robot control require logical operations to be performed on sensory information, i.e. images of the world as seen by a camera consisting of continuous values of pixels. Artificial intelligence (AI) planning algorithms however use symbolic descriptors such as objects and actions to define logic rules and future actions. The representational differences at these distinct processing levels have to be bridged in order to allow communication between both levels. In this paper, we suggest a novel framework for interfacing AI planning with low-level visual processing by transferring the visual data into a discrete symbolic representation of temporally stable image segments. At the AI planning level, action-relevant changes in the configuration of image segments are inferred from a set of experiments using the Group Method of Data Handling. We apply the method to a data set obtained by repeating an action in an abstract scenario for varying initial conditions, determining the success or failure of the action. From the set of experiments, joint representations of actions and objects are extracted, which capture the rules of the given scenario.**

I. INTRODUCTION

The visual scene presented to the camera of a robot while performing an action, i.e. manipulating objects in the scene, contains abundant information about the surrounding world, much of which is not relevant for understanding the consequences of the action. The extraction of action-relevant information from the visual scene is crucial for creating joint internal representations of actions and objects, i.e. object-action complexes (OACs), which are a prerequisite for the robot to interact with its environment in a meaningful way and to progressively accumulate world knowledge [1], [2].

During the course of the robot's exploration of a given scenario, symbolic instantiations of actions have to be compared with the visual input, i.e. continuous values of pixels, requiring an appropriate, condensed representation of the image sequence. There are four main requirements that need to be fulfilled by the visual descriptors: (i) the number of visual descriptors representing the scene should be small, since AI reasoning often requires computationally exhaustive combinatorial searches to be executed, (ii) the visual descriptors should be discrete in order to be compatible with functions at the action level, (iii) the visual descriptors should be traceable troughout the frames of the image sequence (temporal stability), and (iv) the visual descriptors should capture sufficient content of the scene, i.e. parts of objects. In our framework, appropriate image descriptors are obtained from an image segmentation algorithm which tracks segments from frame to frame [3], hence returning temporally stable discrete segment labels, which can be immediately utilized for AI planning. They represent large, connected image areas, which usually can be linked to (parts of) an object. These temporally stable segments provide the interface between the sensory level and the AI planning stage.

The process of pairing actions and symbolic visual descriptors requires relevant changes in the configuration of the image parts to be detected. By repeatedly performing a particular action, reoccurring chains of visual events can be derived from the experimental data. This task can be posed as an induction problem, i.e. we want to extract functions having dependencies between input and output data such that the functions represent actions while the variables of the function represent attributes of objects. Various techniques have been suggested for solving the induction problem (for a review see [4]- [5]), e.g. methods using multiple regression analysis [4], [6], case-based reasoning systems [7]–[9], decision trees [10], [11], algorithms of boundary combinatorial search [12], including the WizWhy by A. Meiden [13], [14], neuron models [15], [16], genetic algorithms [17]–[19], and evolution programming [5], [20], [21]. The group method of data handling [5], [20], [21], employed in this work, is an evolutionary algorithm which successively selects and tests models of functions according to a cross-validation criterion, thus implementing the scheme of mass selection. This method has advantages in case when rather complex objects have no definite theory because object knowledge is derived directly from data sampling.

The paper is structured as follows: In Section II, we introduce the algorithm consisting of the segmentation algorithm and the GMDH applied to the extracted segments. In Section III the results for an abstract scenario of "cup filling" are presented. A discussion of the results and an outlook are given in Section IV.

II. ALGORITHMIC FRAMEWORK

In the following, we will create a scenario in which an agent (or robot) repeatedly performs an action on objects which are connected with the action space through a stable set rules. Here, we choose a scenario dealing with the filling of cups. In this scenario, a cup object can be in two different

This work was not supported by any organization

[1]Bernstein Center for Computational Neuroscience in Göttingen, University of Göttingen, Bunsenstrasse 10, 37073 Göttingen, Germany {natalia,worgott}@nld.ds.mpg.de

[2]Bernstein Center for Computational Neuroscience in Göttingen, Max Planck Institute for Dynamics and Self-Organization, Bunsenstrasse 10, 37073 Göttingen, Germany bkdellen@bccn-goettingen.de

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

states “full” or “empty”. Being empty further implies that it can be filled via an action called “Filling”. After the action, the cup object is in the state “full”. If the cup however is already full, the action “Filling” will not lead to any change in the state of the cup. Hence, potentially meaningful actions (with respect to a particular object) are characterized by their property of inducing characteristic reproducible changes in the scene. The successful linking of the action “Filling” with an “empty” cup object defines an OAC, capturing one of the laws of the cup world. Note, we do not deal with a continuous domain in this example, since cups are considered to be either “full” or “empty”. Starting with a data set of various sequences monitoring the application of the action “Filling” on different initial configurations of cup objects and non-cup objects, we suggest the following algorithm for finding the relevant OACs:

1) Transfer the images of the experiments into a discrete representation of segment labels via an n-d segmentation algorithm [3]. A detailed description of the algorithm can be found in Section IIA.

2) Ascertain the relational position of the segment labels, e.g. relative distance of segments, and define corresponding relational discrete descriptors.

3) Changes in the relational positions of the segments from the start to end of the action provide a set of potential OACs.

4) AI reasoning (see Section IIB) validates or dismisses potential OACs based on statistical recurrence.

A. Creating symbolic temporally stable visual descriptors from image data

We employ the method of superparamagnetic clustering to find temporally stable image segments in the image sequence as seen by the robot [3]. In this method, image pixels are represented by a Potts model of spins, which can be in different, discrete states. Neighboring spins interact such that spins corresponding to pixels of similar gray values tend to be in the same spin state [22]–[27]. Segments are then defined as groups of correlated spins. To define this joint process of simultaneous segmentation, the spin dynamic is developed simultaneously in all frames, while spins in adjacent frames are allowed to interact with each other only if they belong to locally corresponding image points.

We further utilize a technique called energy-based cluster updating (ECU) to accelerate the equilibration of the spin system [26], [27]. The algorithm consists of the following steps:

1. Initialization: A spin value σ_i between 1 and q is assigned randomly to each spin i. Each spin represents a pixel of the image sequence.

2. Definition of neighborhood: Within a single frame (2D bonding), two spins i and k with coordinates (x_i, y_i, z_i) and (x_k, y_k, z_k), respectively, are neighbors if

$$
\begin{aligned}
|(x_i - x_k)| &\leq \varepsilon_{2D} &(1)\\
|(y_i - y_k)| &\leq \varepsilon_{2D} &(2)\\
z_i &= z_k \quad , &(3)
\end{aligned}
$$

where ε_{2D} is the 2D-interaction range of the spins. The coordinates x and y label the position within each image, while z labels the frame number.
Across frames (n-D bonding), two spins i and j are neighbors if

$$
\begin{aligned}
|(x_i + d^x_{ij} - x_j)| &\leq \varepsilon_{nD} &(4)\\
|(y_i + d^y_{ij} - y_j)| &\leq \varepsilon_{nD} &(5)\\
z_i &\neq z_j &(6)\\
a_{ij} &> \tau \quad , &(7)
\end{aligned}
$$

where ε_{nD} is the n-D interaction range. The values d^x_{ij} and d^y_{ij} are the shifts of the pixels between frames z_i and z_j along the axis x and y, respectively, obtained from the optic-flow map. The parameters a_{ij} are the respective amplitudes (or confidences), and τ is a threshold, removing all local correspondences having a small amplitude. However, since the images in the examples given in this paper are changing only little from frame to frame, we will use a zero-flow approximation of the optic-flow field in order to simplify the computation.

3. Computing 2D-bond probabilities: If two spins i and k are neighbors in 2D and are in the same spin state $\sigma_i = \sigma_k$, then a bond between the two spins is created with a probability

$$P^{2D}_{ik} = 1 - \exp(-0.5 J_{ik}/T) \quad , \tag{8}$$

where $J_{ik} = 1 - |g_i - g_k|/\bar{\triangle}$ is the interaction strength of the spins and the parameter T represents a system temperature. The function

$$\bar{\triangle} = \sum_{<ik>_{2D}} |g_i - g_k| / \sum_{<ik>_{2D}} 1 \tag{9}$$

computes the averaged gray-level distance of all 2D neighbors $< ik >_{2D}$, where g_i and g_k are the gray values of pixel i and k, respectively. The function $\bar{\triangle}$ is constant for a given set of parameters and gray values. Negative probabilities are set to zero. This step is identical to previous algorithms of superparamagnetic clustering [26], [27]. It allows spins within each frame to interact and form clusters. If using colored images, the gray values g_i have to be replaced by a color vector $\mathbf{g}_i$, and gray-level differences are replaced by the absolute differences between color vectors $|\mathbf{g}_i - \mathbf{g}_j|$.

4. Computing n-D bond probabilities: If two spins i and j, which belong to different frames ($z_i \neq z_j$), are neighbors in n-D and are in the same spin state

$\sigma_i = \sigma_j$, then a bond between the two spins is created with a probability

$$P_{ij}^{nD} = a_{ij}[1 - \exp(-0.5 J_{ij}/T)] \quad , \qquad (10)$$

where

$$J_{ij} = 1 - |g_i - g_j|/\bar{\triangle} \qquad (11)$$

is the interaction strength of the spins, and a_{ij} is the amplitude (or confidence) that spin i and j are neighbors. The amplitude map A, containing the amplitude values a_{ij}, is provided by the stereo algorithm or optic-flow algorithm together with the respective disparity map D or optic-flow field $\mathbf{O}$. Negative probabilities are set to zero. This step is added to the ECU algorithm to allow spins to interact across frames, thus enabling the formation of n-D clusters.

5. Cluster identification: Spins, which are connected by bonds, define a cluster. A spin belonging to a cluster u has by definition no bond to a spin belonging to a different cluster v.

6. Cluster updating: We perform a Metropolis update [28] that updates all spins of each cluster simultaneously to a common new spin value. The new spin value for a cluster c is computed considering the energy gain obtained from a cluster update to a new spin value w_k. This is done by considering the interactions of all spins in the cluster c with those outside the cluster, assuming that all spins of the cluster are updated to the new spin value w_k, giving an energy

$$\begin{aligned} E(W_k^c) \quad = \quad & \sum_{i \in c} \big[K_i - \sum_{\substack{\langle ij \rangle_{2D} \\ c_k \neq c_j}} \eta J_{ij} \delta(\sigma_i - \sigma_j) \\ & - \sum_{\substack{\langle ij \rangle_{nD} \\ c_k \neq c_j}} \eta a_{ij} J_{ij} \delta(\sigma_i - \sigma_j) \big] \end{aligned} \qquad (12)$$

where $\langle ik \rangle_{2D}, c_k \neq c_j$ and $\langle ij \rangle_{nD}, c_k \neq c_j$ are the noncluster neighborhoods of spin i, and W_k^c symbolizes the respective spin configuration. The function

$$K_i = \sum_j \kappa \delta(\sigma_i - \sigma_j)/N \qquad (13)$$

is an optional global inhibitory term, ensuring that faraway segments get different spin values, where κ is a parameter and N is the total number of pixels of the image sequence. The parameter κ can be set to zero, since K_i does not have any influence in the clustering process itself. The constant η is chosen to be 0.5.
Similar to a Gibbs sampler, the selecting probability $P(W_k^c)$ for choosing the new spin value to be w_k is given by

$$P(W_k^c) = \exp(E(W_k^c)) / \sum_{l=1}^{q} \exp(E(W_l^c)) \quad . \qquad (14)$$

The ECU algorithm has been shown to preserve the concept of detailed balance, and is thus equivalent to standard Metropolis-based simulations of spin systems from a theoretical point of view [26].

7. Iteration: The new spin states are returned to step 3 of the algorithm, and steps 3-7 are repeated, until the total number of clusters stabilizes.

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, the consecutive pairs are connected by identifying the identical segments in the overlapping images. This strategy is used in order to be able to handle long motion image sequences.

We apply the algorithm to a realistic image sequence, showing the filling of a cup (Fig. 1, left column). The respective segmentation results are shown in the middle column. Most of the segments can be tracked through the sequence. We represent the results as graphs, where the nodes are segment labels, plotted at the position of the segment center. Two nodes are connected by an edge if they touch each other. The resulting graph are rather complex. In order to test and validate the main idea of this paper, we therefore chose a simplified abstract example of this scenario (see Fig. 2). In the future however, we aim to apply our method to more realistic sequences.

B. AI reasoning

To extract the relevant objects and actions from the data set depicted in Fig. 2, we pose the task as an induction problem, where the actions are functions $f(\chi)$ which connect between the input data χ and the output data φ such that $\varphi = f(\chi)$. We solve the induction problem by applying the group method of data handling (GMDH) [5], [20], [21], which reproduces the evolutionary scheme of mass selection. The GMDH finds the relevant functional dependencies of a given data set. Initially, several candidate models, i.e. functions, are proposed. The method tests these models and selects the more interesting ones, which are then recombined to allow more complex combinations. The algorithm consists of the following steps:

1a. Representation of input data: For each experiment, labeled with index i, we have an input data vector $\chi_{\mathbf{i}} = (\chi_{i,1}, ..., \chi_{i,j}, ...)$. The data set contains $i = \overline{1, N}$ experiments and $j = \overline{1, M}$ attributes of objects, where $N \geq M$. The data set divides in two parts: N_A is used for learning, and N_B for evaluation of the created models and for decision making regarding stopping the selection process. In the specific example investigated in this paper, a data set is created for each image segment, either before or after the action. The input vector $\chi_{\mathbf{i}}$ for a segment h contains the relative distance of the center of segment h to the other segments, labeled j, before the application of the action. We omit the index h in the following for reasons of readability.

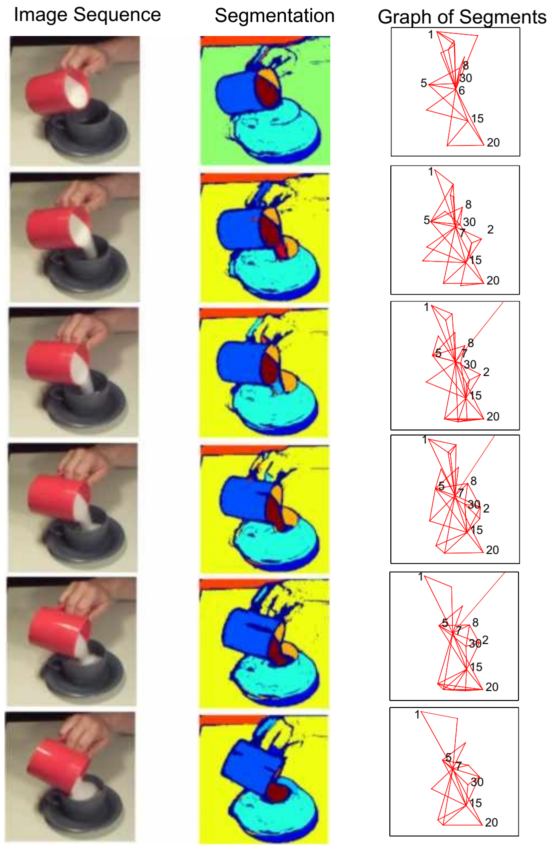

Fig. 1. Filling-a-cup real action sequence. In the left column, several frames of a motion sequence showing the process of filling a cup with sugar are presented. The middle column shows the respective segmentation results. In the future, we aim to apply our method to a set of experiments showing the filling of real cups, defining a data set similar to the one given for the abstract cup scenario. In the right column, the respective graph representations are shown. The nodes of the graphs are plotted at the center of the respective segment, together with its label. Two nodes are connected by an edge if they are neighbors, i.e. if their boundaries touch each other. For reasons of display, we omitted all labels which belong to temporally unstable segments.

1b. Representation of output data: For each experiment, labeled with index i, an output value φ_i is created from the output data by taking the mean of the output data vector $\varphi_\mathbf{i} = (\varphi_{i,1}, ..., \varphi_{i,j}, ...)$ such that

$$\varphi_i = \sum_j \varphi_{i,j}/M \quad , \tag{15}$$

where j is the object label, consistent with the input data. Hence, if $\chi_\mathbf{i}$ represents the values of the object attributes before the action, then $\varphi_\mathbf{i}$ represents the values of the object attributes after the action. The such constructed output data φ_i is used to find the function $f(\chi_\mathbf{i})$ for which $\varphi_i = f(\chi_\mathbf{i})$ is fulfilled best considering all experiments i. The function f then describes the action inducing the relevant changes in the data set.

2. Choosing the particular description of the candidate models, i.e. candidate functions: Almost all types of functions f(χ) can be theoretically expressed by Volterra functional series. Its discrete analogue is the Kolmogorov-Gabor polynomial:

$$\varphi = a_0 + \sum_{j=1}^{M} a_j\chi_j + \sum_{j=1}^{M}\sum_{k=1}^{M} a_{jk}\chi_j\chi_k + \sum_{j=1}^{M}\sum_{k=1}^{M}\sum_{l=1}^{M} a_{jkl}\chi_j\chi_k\chi_l \quad , \tag{16}$$

where $(\chi_1, \chi_2, ..., \chi_M)$ are taken from the input data, and $\mathbf{a} = (a_1, a_2, ..., a_M)$ is the vector of coefficients or weights.

In the following, we choose a multilayered algorithm, thus the iteration rule (particular description) remains the same for all series. A linear particular description of the form

$$\begin{aligned} \varphi_1^1 &= a_1\chi_1 + a_2\chi_2, \\ \varphi_2^1 &= a_1\chi_1 + a_3\chi_3, \\ &\vdots \\ \varphi_r^1 &= a_t\chi_t + a_l\chi_l, \\ &\vdots \\ \varphi_s^1 &= a_{M-1}\chi_{M-1} + a_M\chi_M, \end{aligned} \tag{17}$$

is used for the first iteration, containing $s = M^2$ candidate functions labeled r. The upper index represents the iteration number u, here $u = 1$. In the second iteration ($u = 2$), we get

$$\begin{aligned} \varphi_1^2 &= b_1\varphi_1^1 + b_2\varphi_2^1, \\ \varphi_2^2 &= b_1\varphi_1^1 + b_3\varphi_3^1, \\ &\vdots \\ \varphi_r^2 &= b_t\varphi_t^1 + b_l\varphi_l^1, \\ &\vdots \\ \varphi_p^2 &= b_{s-1}\varphi_{s-1}^1 + b_s\varphi_s^1 \quad , \end{aligned} \tag{18}$$

with $p = s^2$ and so on in the following iterations

3. Estimation of coefficients: At each iteration, the coefficients of the candidate functions are computed using a least-squares method

$$\sigma = \sum_{i=1}^{N} (\varphi_i - \sum_{j=1}^{M} \chi_{i,j} a_j)^2 \rightarrow min \quad , \tag{19}$$

taking all experiments into account. Thus, the initial data transforms to the quadratic array of normal equations, which are solved using the Gauss method.

4. Estimation of regularity: At each iteration, we find the candidate function for which the regularity minimizes

$$PRR(r) = 1/N \sum_{i=1}^{N} (\varphi_i^u - \varphi_i(N_b))^2 \rightarrow min \quad , \tag{20}$$

where φ_i^u is the model output data at the respective iteration step, and $\varphi(N_B)$ is the output taken from the test data.

5. Stopping of selection: The candidate functions are passed to the next iteration step as long as the regularity measure decreases. Practically it is recommended to stop the iteration already when the regularity is decreasing too slowly. Here, we stop the iteration if the function

$$
\begin{aligned}
E = & (PRR_{t-2}(r) - PRR_{t-1}(r)) \\
& - (PRR_{t-1}(r) - PRR_t(r))/ \\
& (PRR_{t-2}(r) - PRR_{t-1}(r)) \\
& + (PRR_{t-1}(r) - PRR_t(r)) \quad , \qquad (21)
\end{aligned}
$$

is smaller than a given threshold. Then, the candidate function with the smallest regularity measure is selected.

III. Results

We approach the task of finding the only existing OACs of our cup world by creating a simplified abstract scenario, in which paper shapes represent objects in a scene (see Fig. 2A). The coloring of objects and the overall color intensities have been varied from frame to frame using an image manipulation program to simulate more realistic conditions, providing additional challenges to the segmentation algorithm. The large blue and red oval shapes represent cup objects, while the black circle represents another object, here a liquid, e.g. coffee, which can be filled into the cup objects. If the liquid object is in the center of a cup object, the cup is full. If there is no liquid object close to the center of cup, the cup is considered empty. We simulate the action of "Filling" by placing the liquid into the center of a cup. The filling of the red cup object can be observed along the consecutive frames of the sequence shown in Fig. 2A. The color images are processed using the segmentation algorithm described in Section 2A (see Fig. 2B). The segment labels are color coded. Temporally stable segments can be tracked from frame-to-frame, hence, changes in the configuration of segments before and after the action can be determined. Here, frame 1 shows the configuration of the segments before the action, and frame 8 shows the configuration of segments after the action. The respective graph representations are depicted in Fig. 1C. The nodes of the graphs are plotted at the center of the respective segment, together with its label. Two nodes are connected by an edge if they are neighbors, i.e. if their boundaries touch each other.

In Fig. 3, the segment configuration before and after the action are shown for a total of eight experiments. From these experiments, the OACs have to be extracted. Experiments 1, 2, 3, 5, 6, and 8 evidence the successfull application of the "Filling" action to a cup object. Other non-cup objects and non-liquid objects are occasionally visible in the scene, i.e. the red square in the experiment 6. All the experiments are linked by "reset" actions (images not shown) which allow the

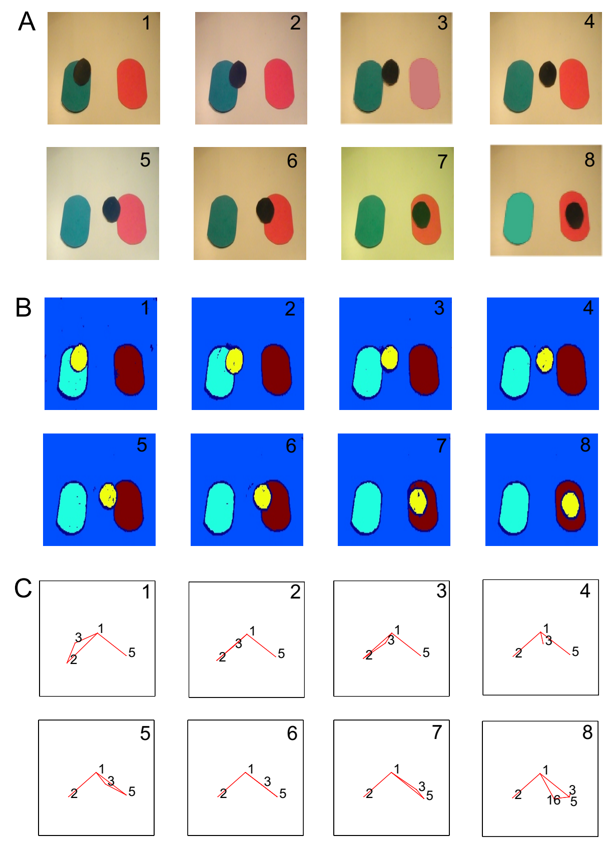

Fig. 2. Abstract example of a "Filling" action. A. The process of filling a cup is captured by a motion sequence containing eight color images. The coloring of objects and the overall color intensities have been varied from frame to frame using an image manipulation program to simulate more realistic conditions. The large oval blue and red paper shapes represent cups in this abstract scenario, while the black circle represents a liquid, e.g. coffee, which can filled into the cups. If a cup is filled with the liquid, the liquid object is placed close or at the center of the cup object, otherwise the cup is considered empty. B. Visual processing. Applying a segmentation algorithm [3] to the image sequence returns temporally stable image segments. The segment labels are color coded. The light blue and the dark red segment correspond to the blue cup and the red cup object, respectively, while the yellow segment represents the liquid object. C The respective graph representations are shown. The nodes of the graphs are plotted at the center of the respective segment, together with its label. Two nodes are connected by an edge if they are neighbors, i.e. if their boundaries touch each other. The background has the label 1 and is not considered further.

objects to be traced through the whole set of experiments. Thus, we can assign the same cluster label to the paper shapes in the images of the first sequence and of the last sequence.

To extract the relevant OAC of the cup scenario, we apply the group method of data handling (see Section IIB) to the data set shown in Fig. 3. For each experiment i and for each segment h, we compute the distance of the center of segment h to each other segment j before the action. These distance values d_{hj} define the input vector $\chi_\mathbf{i}$ to the GMDH, applied independently to each segment h. The output values φ_i are

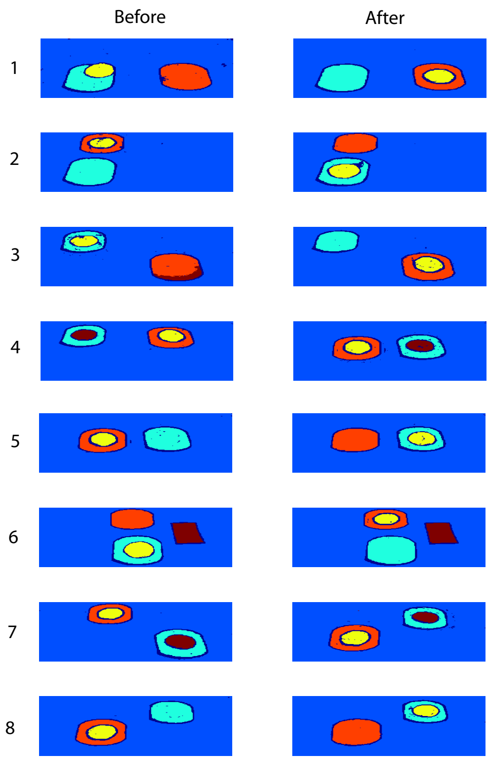

Fig. 3. Set of experiments in the cup scenario. The image segments before and after the application of the "Filling" action for different initial conditions are shown. The first experiment is identical to the example shown in Fig. 1. Experiments 1, 2, 3, 5, 6, and 8 show the successful application of the "Filling" action, i.e. after the action, one of the cups changed its state from empty to full. In experiment 4, both cups are already filled before application of the action, hence, applying the "Filling" action does not induce any relevant changes in the scene. Sometimes other objects, which are not relevant for the particular action, e.g. the rectangular shape in experiment 6.

constructed by computing the mean of the segment distances to segment h after the action. Hence, the task of finding the relevant action from the data set can be posed as an induction problem, i.e. finding the function f which fulfills $\varphi_i = f(\chi_i)$ best, considering all experiments i.

Before applying the GMDH, the input data is normalized to obtain scale invariance

$$\widetilde{d}_{hj} = \frac{d_{hj}}{2/N^2 \sum_h \sum_{j>h} d_{hj}} \quad , \tag{22}$$

where d_{hj} is the distance between segments h and j, and N is the total number of segments.

In Fig. 4, a generalized graph representation of the segments before and after the action is shown for illustration, containing the relevant changes induced by the "Filling" action. The nodes, representing the segments, are plotted with respect to their relative position. Here, only the edges connecting nodes 2,3, and 4 with all other nodes are plotted. Neighborhood information is not explicitly used in this example, since the GMDH uses the distances between segment

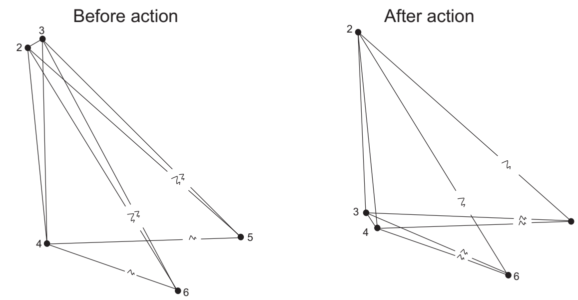

Fig. 4. Generalized graph representation of the segments before and after the action. Here, only the edges connecting nodes 2,3, and 4 with all other nodes are plotted. The edges connecting to segments far away of the scene are indicated by an arc symbol. For reasons of proper display these nodes could not be plotted at their true positions. Before the action, node 3 is situated close to node 2 and at distance to node 4. However, after the action, the node 3 is situated close to node 4 and at distance to node 2, constituting a relevant change. Node 5 represents another liquid filled in the red cup which only appears in experiment 4. Node 6 represents another object which has however no influence on the action.

centers. Before the action, node 3 (liquid) is situated close to node 2 (blue cup) and at distance to node 4 (red cup). However, after the action, node 3 has moved close to node 4 and at distance to node 2, constituting a relevant change in the scene, caused by our abstract "Filling" action.

The GMDH is applied two times to each segment. First, the segment configuration before the action is used to predict the segment configuration after the action, which we call the forward process. Then, the segments after the action are used to predict the segment configuration before the action, which we call the inverse process. Through this, relevant causes in the segment configuration before and after the action can be extracted.

From the functions describing the forward and inverse processes, we consider only the object attributes which have non-zero weight coefficients. The relevant values of the object attributes are computed by taking the mean over all experiments. Mean object attributes together with the associated action define the rule. Applying this method to the blue cup (segment $h = 2$) returns the function

$$\varphi = -898.8\chi_3 + 1503.2\chi_4 + 0.2\chi_5 \tag{23}$$

for the forward process, and the function

$$\varphi = 2988.8\chi_3 + 944.7\chi_4 + 0.4\chi_5 \tag{24}$$

for the inverse process. The respective rule of the blue cup is

$$\begin{array}{rl} \text{if} & \chi_3 = 0.1 \;\&\; \chi_4 = 2.2 \;\&\; \chi_5 = 9000.2 \\ \text{then} & \text{action = FILLING} \\ \text{result} & \chi_3 = 1.9 \;\&\; \chi_4 = 2.2 \;\&\; \chi_5 = 9000. \end{array} \tag{25}$$

For the liquid (segment $h = 3$) we obtain the function

$$\varphi = -188.5\chi_2 + 1768.2\chi_4 + 0.1\chi_6 \tag{26}$$

for the forward process, and the function

$$\varphi = 1426.4\chi_2 + 178\chi_4 + 0.2\chi_6 \tag{27}$$

for the inverse process. The resulting rule for the liquid is

$$\begin{array}{cl} \text{if} & \chi_2 = 0.1 \,\&\, \chi_4 = 2.1 \,\&\, \chi_6 = 9500 \\ \text{then} & \text{action = FILLING} \\ \text{result} & \chi_2 = 2 \,\&\, \chi_4 = 2.3 \,\&\, \chi_6 = 9500. \end{array} \tag{28}$$

For the red cup (segment $h = 4$) we obtain the function

$$\varphi = 1393.5\chi_2 + 0.1\chi_3 + 0.2\chi_5 \tag{29}$$

for the forward process and the function

$$y = 1332.6\chi_2 + 123.6\chi_3 + 0.2\chi_5 \tag{30}$$

for the inverse process. The resulting rule of the red cup is

$$\begin{array}{cl} \text{if} & \chi_2 = 2.2 \,\&\, \chi_3 = 2.2 \,\&\, \chi_5 = 9000 \\ \text{then} & \text{action = FILLING} \\ \text{result} & \chi_2 = 2.2 \,\&\, \chi_3 = 0.2 \,\&\, \chi_5 = 9000 \quad . \end{array} \tag{31}$$

The set of rules computed for the different segments represents an OAC of the cup scenario.

In Figs. 5-7 the extracted robot rules, presented as graphs, with respect to segment $h = 2$ (the blue cup), $h = 3$ (the liquid), and $h = 4$ (the red cup) are presented. The relevant edges for initiating the "Filling" action and the relevant resulting edges are plotted in red. In Fig. 5, the rule with respect to segment $h = 2$ are shown. A short edge between the blue cup and the liquid initiates the "Filling" action. As a result the liquid is situated far away from the blue cup. In Fig. 6, the rule with respect to segment $h = 3$, the liquid, is shown. A small distance between the liquid and the blue cup and a large distance between the liquid and the red cup initiates the "Filling" action, which causes distances between the liquid and the blue cup to increase largely and the distance between the liquid and the red cup to decrease, symbolizing the filling of the red cup. In Fig. 7, the rule extracted for segment $h = 4$, the red cup, is shown. The action has been initiated through the large distance between the liquid and the red cup. As a result, the liquid is situated close to the red cup.

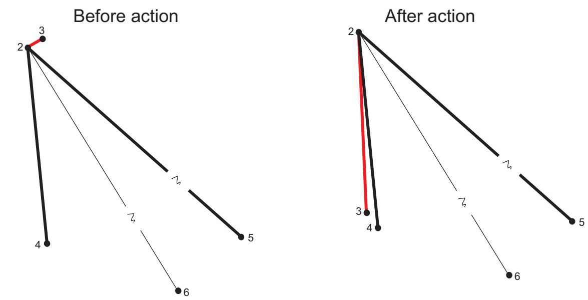

Fig. 5. Robot rule for the blue cup (segment 2). The most important action-relevant edges are depicted in red.

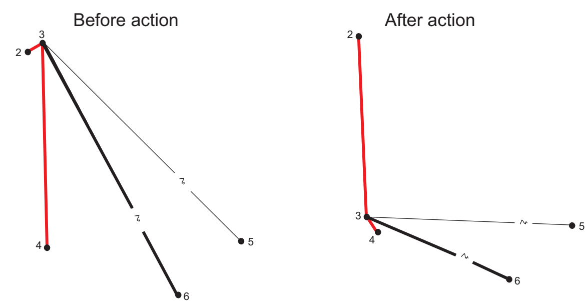

Fig. 6. Graph representation of the robot rule obtained for the liquid (segment 3). The most important action-relevant edges are depicted in red.

IV. DISCUSSION

We proposed an algorithm for the computation of OACs which applies AI reasoning to temporally stable image segments to ascertain change in visual data. From a set of experiments, relational attributes of segments could be associated with a particular action, here the "Filling" of a cup in an abstract scenario in which paper shapes represent "cups" and "liquids". Segment tracking results obtained for complex image sequences however suggest that the proposed method generalizes to more realistic scenarios. However, segment tracking through n-d segmentation might fail in some cases due to light reflexions or other changes in the images. Other techniques or heuristics will have to be employed to bridge these gaps, since the GMDH used in the reasoning process requires a temporally stable labelling of objects. However, it remains an open question whether the segment representation is descriptive enough for scenes containing complex objects. In real-world scenarios, simple segment relations, such as the distance between segment centers used in this work, might not suffice to capture all action-relevant object properties. In this case, higher-level features would have to be included in the scene description. In our future research, we aim to provide answers to these questions using more realistic scenarios and robot experiments.

We further wish to generalize the method such that the relations between all segments can be used in the GMDH simultaneously. As it its, the algorithm computes rules given the configuration of a particular segment to all the other segments.

The GMDH further relies on statistical recurrence requiring the repetitive execution of an action. Such a strategy is not always very efficient. Instead, we would like to augment the GMDH by drawing conclusions at an early stage, so that actions can be applied more efficiently during the exploratory phase.

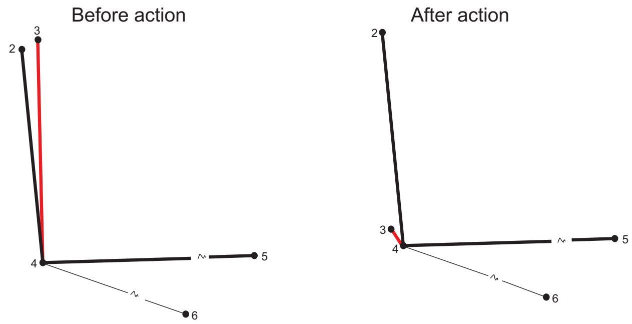

Fig. 7. Graph representation of the robot rule obtained for the yellow cup (segment 4). The most important action-relevant edges are depicted in red.

V. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support from the EU Project Drivsco under Contract No. 016276-2, the EU Project PACO-PLUS under Contract No. 027657, and the BMBF funded BCCN Göttingen.

REFERENCES

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

[2] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krüger, and F. Wörgötter, "Object action complexes as an interface for planning and robot control," *IEEE RAS Int. Conf. Humanoid Robots (Genova)*, pp. Dec. 4–6, 2006, 2006.

[3] B. K. Dellen and F. Wörgötter, "Extraction of region correspondences via an n-d conjoint spin relaxation process driving synchronous segmentation of image sequences," *Submitted*, 2008.

[4] J. W. Osborne, "Prediction in Multiple Regression," World Wide Web, urlhttp://www.pareonline.net/getvn.asp?v=7&n=2, 2000.

[5] F.Lemke and J. Muller, "Self-organizing data mining for a portfolio trading system," *Journal of Computational Intelligence in Finance*, vol. 5, no. 3, pp. 12–26, May-June 1997.

[6] W. Raaymakers and A. Weijters, "Using regression models and neural network models for makespan estimation in batch processing," in *Proceedings of the 12th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC00)*, Kaatsheuvel, The Netherlands, November 2000, pp. 141–148.

[7] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, March 1994.

[8] D. Aha, "The Omnipresence of Case-Based Reasoning in Science and Application," *Knowledge-Based Systems*, vol. 11, no. 5-6, pp. 261–273, November 1998.

[9] A. Stahl and T. Gabel, "Optimizing similarity assessment in case-based reasoning," in *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI-06)*, Boston, USA, Juli 2006, pp. 1667–1670.

[10] S. Russel and P. Norvig, *Artificial Intelligence : a modern approach*. Prentice Hall, 2003.

[11] M. Chen, "On the evaluation of attribute information for mining classification rules," in *Tools with Artificial Intelligence. Proceedings of the 10th IEEE International Conference*, Taiwan, November 1998, pp. 130–137.

[12] M. Bongard, *Pattern Recognition*. Spartan Books, 1970.

[13] "www.wizsoft.com," World Wide Web, http://www.wizsoft.com.

[14] S. Barai, "Data mining application in transportation engineering," *TRANSPORT*, vol. 18, no. 5, pp. 216–2233, 2003.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[16] P. J. G. Lisboa, B. Edisbury, and A. Vellido, *Business Applications of Neural Networks*. World Scientific, 2000.

[17] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT press, 1996.

[18] M. L. Raymer, W. F. Punch, E. D. Goodman, and L. A. Kuhn, "Genetic programming for improved data mining: An application to the biochemistry of protein interactions," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, Stanford, USA, July 1996, pp. 375–380.

[19] L. Jourdan, C. Dhaenens, and E.-G. Talbi, "A genetic algorithm for feature selection in data-mining for genetics," in *4th Metaheuristics International Conference*, Porto, Portugal, July 2001, pp. 29–34.

[20] H. Madala and A. Ivakhnenko, *Inductive Learning Algorithms for Complex Systems Modeling*. CRC Press, 1994.

[21] J. HowlandIII and M. Voss, "Natural gas prediction using the group method of data handling," in *Artificial Intelligence and Soft Computing (ASC 2003)*, Banff, Canada, July 2003. [Online]. Available: citeseer.ist.psu.edu/article/howland03natural.html

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

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

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

[25] U. Wolff, "Collective monte carlo updating for spin systems," *Physical Review Letters*, vol. 62, pp. 361–364, 1989.

[26] 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.

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

[28] N. Metropolis, A. W. Rosenbluth, A. H. T. M. N. Rosenbluth, and E. Teller, "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1091, 1953.