Semantic analysis of image sequences using computer vision methods

Dissertation

ZUR ERLANGUNG DES MATHEMATISCH-NATURWISSENSCHAFTLICHEN DOKTORGRADES
"DOKTOR RERUM NATURALIUM" DER GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

vorgelegt von
Eren Erdal Aksoy
aus Istanbul

Göttingen 2012

Referentin/Referent: Prof. Dr. Florentin Wörgötter
Koreferentin/Koreferent: Prof. Dr. Wolfgang May
Tag der mündlichen Prüfung: 18/07/2012

Abstract

Observing, learning, and imitating human skills are intriguing topics in cognitive robotics. The main problem in the imitation learning paradigm is the policy development. Policy can be defined as a mapping from an agent's current world state to actions. Thus, understanding and performing an observed human skill for a cognitive agent depends heavily upon the learned policy. So far, naive policies that use object and hand models with trajectory information have commonly been developed to encode and imitate various types of human manipulations. These approaches, on the one hand, can not be general enough since models are not learned by the agent itself but rather are provided by the designer in advance. It is also not sufficient to imitate complicated manipulations at the trajectory-level since even the same observed manipulation can have high variations in trajectories from demonstration to demonstration.

Nevertheless, humans have the capability of recognizing and imitating observed manipulations without any problem. In humans, the chain of perception, learning, and imitation of manipulations is developed in conjunction with the interpretation of the manipulated objects. To compose a human-like perception-action chain the cognitive agent needs a generic policy that can extract manipulation primitives as well as the essential (invariant) relations between objects and manipulation actions.

In this thesis, we introduce a novel concept, the so-called "Semantic Event Chain" (SEC), that derives the semantic essence and the invariant spatiotemporal relations of objects and actions to acquire a perception-action chain. We show that SECs are compact and generic encoding schemes for recognizing, learning, and executing human manipulations by relating them with manipulated objects. SECs basically make use of image sequences converted into uniquely trackable segments. The framework first interprets the scene as undirected and unweighted graphs, nodes and edges of which represent image segments and their spatial relations (e.g. touching or not-touching), respectively. Graphs hence become semantic representation of segments, i.e. objects (including hand) presented in the scene, in the space-time domain. The proposed framework then discretizes the entire graph sequence by extracting only main graphs each of which represents essential primitives of the manipulation. All extracted main graphs form the core skeleton of the SEC which is a sequence table, where the columns and rows correspond to main graphs and spatial relational changes between each object pair in the scene, respectively. SECs consequently extract only the naked spatiotemporal patterns which are basically "essence of an action" and are invariant to the followed trajectory, manipulation speed, or relative object poses.

In the perception phase, SECs let a cognitive agent not only recognize and classify different observed manipulations but also categorize the manipulated objects considering their roles exhibited in the manipulations. This process is accomplished by comparing both spatial and temporal features of event chains of given manipulations. By extracting only repetitive relational sequences, hence those which are commonly observed in the demonstrated training set, the agent can further learn an archetypical SEC model for each manipulation type. The learning process of SEC models is also enriched by recording additional decisive information such as relative coordinate frames and motion start and endpoints.

The perception-action cycle is finally completed by using the learned SEC model with the additional decisive information to derive high-level rules that the agent can use to execute a similar manipulation regardless of the type and configuration of objects presented in a new scene.

The main advantage of this framework is that SECs encode a manipulation in a highly invariant and abstract way independent from object poses, perspectives, and trajectories which can be interchanged to a very large degree. In this sense, SECs reduce the problem of action representation to the analysis of small, scaled matrices. With this the agent also gains the possibility of assessing the consequences of its own manipulation by simply comparing the obtained SEC with the learned one. Furthermore, the SEC is a unified bottom-up approach that combines actions and objects based on the temporal sequence of spatial relations between tracked image segments, which for a given manipulation remains "essentially" the same. Hence, different from model-based policy designs our system operates on spatiotemporal object relations without making assumptions about the structure of objects and actions. In this sense, the framework presented in this thesis is model-free.

To our knowledge, this is one of the first approaches which reaches an abstract symbolic rule-like representation (manipulation primitives) for manipulation recognition, learning, and execution while being fully grounded in the signal (image segments) domain.

Contents

Title Page i
Abstract iii
Table of Contents v
Citations to Related Publications vii
Acknowledgments ix
Dedication x

1 Introduction **1**
1.1 Action Understanding 2
1.2 Affordances and Object-Action Complexes 3
1.3 Semantic Event Chains 4
1.4 The State of the Art 6
  1.4.1 Scene Graphs 6
  1.4.2 Recognition of Manipulation Actions 7
  1.4.3 Recognition of Human Motion Patterns 8
  1.4.4 Object Recognition and the Role of Context 9
  1.4.5 Execution of Manipulation Actions 10
1.5 Outline and Contributions 10

2 Methods for Analyzing of Object-Action Relations **13**
2.1 Introduction 13
2.2 Preprocessing, Segmentation & Tracking (Step 1) 15
2.3 Relational Scene Graphs (Step 2) 18
2.4 Main Graphs (Step 3) 18
2.5 Semantic Event Chains (SECs) (Step 4) 20
2.6 Similarity Measure (Step 5) 20
  2.6.1 Spatial Similarity Analysis 21
  2.6.2 Temporal Similarity Analysis 24
2.7 Action Classification (Step 6A) 27
2.8 Segment Categorization (Step 6B) 28
2.9 Learning Algorithm (Step 7) 30
2.10 Manipulation Execution with SECs (Step 8) 34
  2.10.1 Segmentation and SEC-generation 34
  2.10.2 Defining Temporal Anchor Points 36
  2.10.3 Defining Spatial Anchor Points 38
  2.10.4 Planning 39
  2.10.5 Execution 42
  2.10.6 Simulation Results 43
2.11 Discussion 44

3 Statistics on Semantic Event Chains **47**
3.1 Analysis of the Similarity Measure 47
3.1.1 Effects on Action Classification 49
3.1.2 Effects on Object Categorization 50
3.1.3 Effects on Learning 53
3.2 Comparison of Classification Algorithms 55
3.2.1 Feed-Forward Backpropagation Neural Networks 55
3.2.2 Classification Results 57
3.3 Discussion 61

4 Applications **63**
4.1 Case Study I: Learning and Replaying an Action Sequence 63
4.2 Case Study II: Semantic Observation and Execution of Manipulations 67
4.2.1 Scenario 1: LABEX 67
4.2.2 Scenario 2: AUTAS 77
4.3 Case Study III: Plant Growth Measurement System 80
4.3.1 Tracking of Plant Graphs 80
4.3.2 Leaf Modeling 81
4.3.3 Extracting Measurement Parameters from Graphs 83
4.4 Discussion 86

5 Conclusion and Outlook **89**
5.1 Related Approaches 91
5.2 Features and Problems of the SEC Framework 92

A Appendix **103**
A.1 Defining Segment Relations in a Fast and Efficient Way 104
A.2 The GraphML File Format 107
A.3 Index of Multimedia Extensions 108

B Curriculum Vitae **109**

Citations to Related Publications

Chapter 2 has appeared in the following three papers:

> Aksoy, E.E. and Dellen, B. and Tamosiunaite, M. and Wörgötter, F. Execution of a dual-object (pushing) action with semantic event chains. *IEEE-RAS International Conference on Humanoid Robots (Humanoids), 576-583, 2011.*

> Aksoy, E.E., Abramov, A., Dörr, J., Ning, K., Dellen, B., and Wörgötter, F. Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research (IJRR), 30(10):1229-1249, 2011.*

> Aksoy, E.E., Abramov, A., Wörgötter, F., and Dellen, B. Categorizing object-action relations from semantic scene graphs. *IEEE International Conference on Robotics and Automation (ICRA), 398-405, 2010.*

Case study I given in Chapter 4 was published in Aksoy et al. (2011a). Main concepts introduced in *Case study II and III* are parts of European projects IntellAct (`http://www.intellact.eu`) and GARNICS (`http://www.garnics.eu`), respectively. *Case study III* was also introduced as a part of a patent (Wörgötter et al., 2012a).

Acknowledgments

First of all, I would like to thank my supervisors Prof. Dr. Florentin Wörgötter and Dr. Babette Dellen for guiding me through my work by sharing their experiences with me and for countless hours of fruitful discussions without which this work would not have been successful. This work has been done in collaboration with Alexey Abramov and Johannes Dörr, so I am very thankful for their efforts, too.

Secondly, I would like to thank all my colleagues and friends for their direct and/or indirect input to my work and a having great time together. Many thanks go to Dr. Tomas Kulvicius, Dr. Minija Tamosiunaite, Dr. Christoph Kolodziejski, Dr. Poramate Manoonpong, Dr. Irene Markelic, Dr. Alejandro Agostini, Christian Tetzlaff, Martin Biehl, Faramarz Faghihi, Jeremie Papon, Harm-Friedrich Steinmetz, Liu Guo Liang, KeJun Ning, Jan-Matthias Braun, Markus Schöler, and Ursula Hahn-Wörgötter.

A special thanks goes to my parents without whom I would not have achieved all that in my life what I have now. And finally, I would like to thank Sibel Aktay for the patience, understanding, support and being always by my side no matter what. Thank you very much indeed!

Dedicated to my parents.

1

Introduction

One of the central goals in cognitive robotics is to recognize, learn, and imitate human behaviors without human intervention. It has, however, long been addressed that raw observation and naive copying are insufficient to execute an action by a robot. It is because individual manipulations, even when "doing the same thing", can take vastly different forms just due to changes in posture, in the followed trajectory, and/or differences in the general (visual) context surrounding the manipulated objects. Nevertheless, humans have no problem recognizing variations in manipulations and even executing them under different circumstances. For instance, in human perception, it is, in a general sense, the same *picking up* action, no matter whether the left or right hand is following a circular or linear trajectory to pick up an apple or an orange from a plate. The human perception system essentially captures and interprets action invariants, essential primitives, by relating actions with objects. The difficulty in cognitive robotics is the question of how to define a unified framework that perceives and represents action primitives in conjunction with objects. This is required to make robots learn novel actions and execute them even with different objects and/or under different circumstances.

In order to accelerate the cognitive development process in robotics, we need to first understand what an action physically means, how it is retrieved, represented, and executed in the human perception-action system. In this thesis, we use different terms such as *action/motor primitives*, *action*, and *manipulation/manipulation action*. *Action* or *motor primitives* are the smallest components of *actions*. Different sequences of primitives introduce different types of *actions*. From a robotics standpoint, *motor primitives* generally stand for basic motor control commands to produce *actions* by robots. The term *action* is rather a general description for any type of motor behavior like "walking", "playing", or "cutting". In the context of this thesis, the more specific terms such as *manipulation* or *manipulation action* are used as subsets of the term *action*. To phrase it another way, they are more specific definitions in the sense of describing *actions* in which objects are basically manipulated by a manipulator. As an example, the *action* "pushing" can be referred to as a *manipulation* or *manipulation action* because an object, e.g., a box or a plate, needs to be manipulated (pushed) by a manipulator, e.g., a hand.

1.1 Action Understanding

The first step towards understanding actions starts with the primitives. The main idea behind action primitives is that the problem of action representation can be reduced into small scaled primitive sequences, compositions of which can then be used to execute the action. However, the extraction of action primitives is non-trivial and this problem is often addressed in the context of imitation learning. According to Meltzoff and his colleagues (Meltzoff, 2002; Meltzoff and Moore, 1997; Rao et al., 2004), infants follow four different phases to explore motor primitives and imitate actions. In the early phase, so-called "body babbling", newborns start with a random trial-and-error method to learn what specific muscle movements yield particular body configurations. This is a crucial step towards exploring "motor primitives" in infants. In the later phase, 12- to 21-day-old infants distinguish and imitate different facial and manual gestures even without getting any feedback from the experimenter (Meltzoff and Moore, 1977). In the next phase, 14 month old infants continue with imitating actions on objects that they have not seen before (Meltzoff, 1988). In the highest phase of the imitation learning, 18 month old infants infer the attempted goal and intention from unsuccessfully demonstrated actions and imitate them successfully.

Those experiments suggest that infants start creating their internal perception-action mechanisms by exploring their motor primitives and mapping the observed actions to their own motor primitives. Initially, the mapping phase could be even without having the same intention or action goal compared to the imitated one. In the later phases, the internal perception-action mechanism is continuously being developed with understanding not only actions but also relations between actions and objects. In the higher phases of the cognitive development, there is a generalization stage in which actions can be imitated in different forms and under different unseen conditions.

Action understanding and execution have a tight bond to neurophysiology as well. Studies on mirror neurons (Rizzolatti et al., 2001; Rizzolatti and Craighero, 2004) highlight the underlying neural mechanism of action understanding and execution. Rizzolatti et al. (2001) advocate the "direct-matching hypothesis" which claims that the understanding of actions in humans occurs as the visual representation of the observed action is mapped onto the motor representation developed for the same action in the nervous system. Such direct matching between action observation and execution is supported by mirror neurons which are a type of visuomotor neurons and were first discovered in the ventral premotor cortex of monkeys (Rizzolatti et al., 2001). Mirror neurons tend to fire while the monkey is either performing an object-directed action or observing another monkey or a human doing a similar action. Nevertheless, mirror neurons respond neither to only objects nor to actions that are mimed without objects. In the experiments of Umiltà et al. (2001), a monkey was first shown a fully visible grasping action and then a piece of food that was hidden behind a screen. It was recorded that more than half of the mirror neurons fired while the monkey was observing only the beginning, but not the crucial hand-object interaction happening behind the screen, of a new grasping or a holding action. This strongly indicates that the monkey can understand and infer the intended action goal in spite of having incomplete visual information.

Consequently, strong mirror neuron activity during the perception and execution of actions is an important finding showing that action understanding and imitation share a common neural layer in which action primitives are encoded. The visual information basically triggers neural activities but can also be replaced with other cues such as sound information. Although no electrophysiological studies about experimenting mirror neuron activities in humans were presented, there are many neurophysiological and brain-imaging evidences indicating that a more developed mirror neuron system exists also in humans (Rizzolatti and Craighero, 2004).

1.2 Affordances and Object-Action Complexes

While observing other individuals manipulating objects, we retrieve information not only about the performed actions but also about the manipulated objects. In this sense, object affordances play an important role in action recognition and planning during the cognitive development. Assume an environment with a cup and a ball placed on a table. To be able to perform a simple "filling" action, the cognitive agent has to choose the cup but not the ball since balls have no hollow structure and thus are not fillable. This example explicitly emphasizes that each object has a certain set of affordances which suggests specific actions. Object affordances can basically be defined as a set of expected behaviors from an object based on its visual characteristics (e.g., shape and surface structure) without involving any model-based recognition step. Therefore, any hollow shaped solid object, even if it has not been seen before, can be used for the "filling" action since its affordances (being hollowed and solid) suggest this action. The affordance principle was first introduced by Gibson (1979), and especially in the recent years has had increasing influence in robotics (Hart and Grupen, 2009; Montesano et al., 2008; Ridge et al., 2009).

Wörgötter et al. (2009); Krüger et al. (2011) extended the idea of the affordance principle (Gibson, 1979) by introducing the concept of Object-Action Complexes (OACs), 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 a human-like description by which an object is identified considering both its (visual) properties and the actions that have been performed with it. In the OAC concept, the performed actions are attached to the objects as attributes. If we come back to the "filling" example above, since any kind of cylindrical, hollow object could be used for filling, the action-type "filling" creates the object-type "container". However, when a sample "container" (e.g., a cup or glass) is turned upside down, it can not be filled at all and therefore, exhibits totally new affordances which can be classified as "pedestal". Now the former container has become a pedestal on which we can put something. While physically the same thing, a "pedestal" is a different object type altogether. Considering the perception-action loop, this example indicates the significant role of object affordances in action planning. As affordances suggest cognitive agents how to manipulate arbitrary unseen objects, the agent can, the other way around, extract required object features for performing any desired action.

1.3 Semantic Event Chains

To acquire a generalized human-like perception-action mechanism, the cognitive agent requires a manipulation representation system that can extract manipulation primitives and invariant relations between objects and manipulations. To arrive at such a representation is a very difficult problem and commonly one uses models of objects (and hands) and trajectories to encode a manipulation (see next section for literature discussion). These approaches, however, are not general enough because models are almost always given by the designer and not learned by the agent itself. Furthermore, it is so far unknown how to solve the variability problem of manipulations in a model-based way, since there is no definition for the correct model (or model-class) to combine objects and manipulations together in all vastly differing manipulations.

In this thesis, we introduce the so-called "Semantic Event Chain" (SEC) which is a novel, compact, and generic encoding scheme for manipulations. SECs can be used to allow a cognitive agent to classify different manipulations by observation and to categorize the manipulated objects based on their roles exhibited in the manipulation. Furthermore, the agent can learn an archetypical SEC model in an unsupervised way by observing demonstrated manipulations. In the process of learning, the SEC model is enhanced with additional decisive information to make the robot able to execute the manipulation with objects in different position and orientation. The main advantage of this framework is that SECs link the signal domain (observed image sequences) to a symbolic rule-like domain (manipulation primitives) by encoding a manipulation in a highly invariant way, where, for a given manipulation, objects, poses, perspectives, and trajectories can be interchanged to a very large degree. Thus, SECs provide one possible, quite efficient way to perform manipulation recognition and to execute a learned manipulation model. To our knowledge, this is one of the first approaches which reaches an abstract symbolic representation for manipulation recognition, learning, and execution while being fully grounded in the signal domain.

SECs are created based on a bottom-up approach in which raw image sequences are first segmented to track objects and then represented by scene graphs that store spatial segment relations (e.g., touching or not-touching) in the temporal domain. Scene graphs are invariant to object and hand locations, trajectories, and poses in 3D. By using an exact graph matching technique main graphs of the manipulation are extracted. The sequence of these main graphs describes all structural changes (manipulation primitives) in the scene. All those primitives are then encoded by the SEC which is a sequence-table, the rows and columns of which represent spatial and temporal anchor points, respectively. Comparison of the extracted SECs in the spatiotemporal domain leads to both manipulation classification and object categorization. The learning phase is then concluded by extracting common temporal and spatial anchor points observed in all SECs of demonstrated type-similar manipulation samples. Since a raw SEC representation is in an abstract form, stripped from all pose and trajectory information, it is initially impossible to execute the learned SEC model of a manipulation. Thus, to perform an accurate manipulation, SECs store additional decisive information, for example relative coordinate frames and information about motion start and endpoints, in the process of learning the SEC model. But, because the SEC provides a temporal sequence of rules, we have well defined temporal anchor points when we have to store the additionally required trajectory information. Finally, the learned SEC model and the additional information are used to extract rules and anchor points in order to let the robot execute a similar manipulation regardless of the initial state of the scene.

Consequently, this thesis presents a unified perception-action framework, motivated by the mirror neuron system, by which a sequence of action primitives can be recognized, learned, and executed considering the same symbolic SEC representation. The agent can decide by self-observation whether or not an executed manipulation sequence is correct. Thus, SECs give the machine a basic tool by which it can assess the consequence of a manipulation step directly linking the symbolic planning domain to the signal domain (image) addressing the difficult cause-effect problem of how to observe and verify self-induced changes. In the same manner, the agent can imitate the manipulation even when the object locations and camera perspective are altered. Last but not least, SECs are essentially related to the affordance principle and the OAC concept since manipulated objects can be categorized by considering their common roles in manipulations. Thus, the agent can suggest what kind of manipulations are more likely to be performed with a given object as the agent is able to create a link between objects and manipulations.

The proposed framework relies on a front-end algorithm which allows for the continuous tracking of scene segments. SECs are based on the sequence of neighborhood relations between those segments, which for a given manipulation is "essentially" the same. Hence, different from feature-based (or model-based) approaches our system operates on spatiotemporal object relations without presupposing assumptions about the structure of object and manipulation. Thus, the framework presented in this thesis is model-free. This leads to a high degree of invariance against position and orientation, but we need to make sure that segment tracking is stable, which is not in the core of this thesis and currently achieved by several means described elsewhere Abramov et al. (2010, 2012).

1.4 The State of the Art

To date, there exists no common framework for both learning the semantics of manipulation actions in conjunction with the manipulated objects and executing the manipulation after learning it from demonstrations. Different approaches have been presented but rather for vision-based recognition of manipulations and human-motion patterns, and non-visual recognition of other types of activities (Modayil et al., 2008; Liao et al., 2005; Hongeng, 2004). In the literature, there exist many works on vision-based object recognition (Mundy, 2006; Lowe, 2004) and manipulation execution (Fitzpatrick et al., 2003; Omrcen et al., 2009) which are related to this proposed work. The latter will not be discussed any further, because vision is the focus of the work presented in this thesis. In the following, short summaries of previous achievements obtained in these areas are given.

1.4.1 Scene Graphs

Graphs have been commonly used in scene analyses. Badler (1975) introduced the first approach about the directed scene graphs in which each node identifies one object. Edges hold spatial information (e.g., LEFT-OF, IN-FRONT-OF, etc.) between objects. Based on object movement (trajectory) information events are defined to represent actions. The main drawback of this approach is that continuous perception of actions is ignored and is substituted instead by the idealized hand-made image sequences.

Sridhar et al. (2010) represented a whole video sequence by an activity graph with different levels each of which represents qualitative spatial and temporal relations between objects involved in activities. Frequent subgraphs of the activity graph define events, i.e. significant activities, which are classified by a level-wise graph mining procedure. In addition to this, a Hidden Markov Model (HMM) is used to improve calculations of the qualitative spatial relations from noisy video inputs. Since the complete video sequence is represented by a single graph, the approach leads to complex and large graphs which need to be decomposed separately.

Brendel and Todorovic (2011) analyzed human activity videos with weighted and directed scene graphs nodes of which are homogeneous video subvolumes in the space-time domain. Graph edges have three components: hierarchical, temporal, and spatial subvolume relations. Although the proposed approach basically learns weighted least-squares graphs that model the respective activity for monitoring, the framework does not include any further process regarding the manipulated objects as well as execution of the learned models.

Wen-Jing and Tong (2000) introduced a sub-scene graph matching method just for object recognition, combining it with a Hopfield neural network to get local matches between graphs. A scene is first represented by polygons that indicate the outer 2D object boundaries. Vertices and sides of the polygon construct graph nodes and edges with additional information like local properties. Such a scene graph is partitioned into subgraphs each of which is then compared with a model object graph by using neural nets. To compute the final match of the complete scene graph additional statistics are applied which combine matching results between subgraphs and models. In their work, however, a separate neural network is required for each subgraph and high computational time is needed to compute subgraph matches.

1.4.2 Recognition of Manipulation Actions

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

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

Kjellström et al. (2008) segmented hand and objects from the video and then defined hand/object features (shape-based) and manipulation features, providing a sequence of interrelated manipulations and object features. Semantic manipulation-object dependencies, e.g. drink/glass, are then extracted using conditional random fields (CRFs) and connected hierarchical CRFs. Hand/manipulator and the manipulated object together define the manipulation, and for this reason the recognition process simultaneously involves both hand/manipulator and objects (Vicente et al., 2007; Kjellström et al., 2008). In Vicente et al. (2007), manipulations are represented as sequences of motion primitives. Here, five different manipulations of different levels of complexity were investigated. The process is modeled using a combination of discriminative support vector machines and generative HMMs. HMMs have also been used by Ogawara et al. (2002) to extract manipulation primitives by learning several HMMs and then clustering these HMMs such that each cluster represents one primitive. Raamana et al. (2007) recognized simple object manipulations such as pointing, rotating and grasping in a table-top scenario using HMMs and selected the best features for recognition automatically. These works demonstrate that HMMs are a useful tool if the manipulation primitives are hidden in the sensory feature set provided to solve the recognition tasks. Usually this the case if low-level features are used instead of higher-level "object" like entities. However, in our case, manipulations are represented by chained relations between image segments (see Chapter 2), which directly represent manipulation primitives, and as such they can be compared, grouped, and superimposed without having to assume a hidden model. This holds at least for the manipulation examples considered in this thesis.

1.4.3 Recognition of Human Motion Patterns

Recognition of human motion has received much attention in recent years and many contributions exist, but are often unrelated to manipulation recognition (Laptev and Perez, 2007; Niebles et al., 2008; Dee et al., 2009; Hakeem and Shah, 2005; Calinon and Billard, 2004, 2005, 2007; Maurer et al., 2005; Gilbert et al., 2011; Junejo et al., 2011). Much work has been done by the group of Aude Billard (Calinon and Billard, 2004, 2005, 2007; Maurer et al., 2005) addressing the aspect of gesture recognition. Naturally a strong focus lies here on finding a way to describe complete trajectories and different methods (including Principal Component Analysis, Independent Component Analysis, HMM and Hopfield nets) have been used in different combinations to address this problem and also to deal with the question of sequence learning (Maurer et al., 2005). In Laptev and Perez (2007) spatiotemporal volumes of optical flow are used to classify human motion patterns. In Niebles et al. (2008) human actions are learned in an unsupervised way by using spatiotemporal "words" that represent space-time interest points. Dee et al. (2009) segment images into regions of similar motion structure and learn pair wise spatial relations between motion regions, roughly corresponding to semantic relations such as "above", "below", and "overlapping". By combining these learned spatial relations with the segmentations learned from data, a compact representation can be provided for each video, representing a motion-based model of the scene, which allows classifying videos containing different kinds of motion patterns, e.g. indoor scenarios with moving people, roads, squares or plazas. In Hakeem and Shah (2005) events involving multiple agents are detected and learned considering temporally correlated sub-events. In Gilbert et al. (2011) simple 2D corners are grouped in both the spatial and the temporal domains, using a hierarchical process at each stage and the most descriptive features are then learned by using data mining. This way, fast and accurate action recognition in video sequences is achieved in real time. Junejo et al. (2011) propose a self similarity-based descriptor that can be used for recognizing human actions under different views. A set of feature is first extracted and then euclidean distances between extracted features for all frame pairs are stored in a Self Similarity Matrix (SSM) which is invariant to view changes.

1.4.4 Object Recognition and the Role of Context

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

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

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

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

1.4.5 Execution of Manipulation Actions

In this thesis, we explain how to execute a manipulation from the learned SEC model by the example of a pushing action (see Chapter 2). Therefore, we here address works that highlight learning and execution of pushing actions. In the literature many works focus more on the (mechanical) aspects of controllability and planning of stable pushing actions (Lynch and Mason, 1995; Li and Payandeh, 2007). Such aspects are not in the core of this thesis.

In Fitzpatrick et al. (2003) the authors showed how an agent can learn simple pushing actions on a toy object and then execute them as goal-directed behaviors. During the training phase, the time evolution of the initial hand position and the direction of object displacement at the moment of contact were continuously recorded. As will be shown in chapter 2, this is to some degree similar to our approach. In each trial the robot learns to map from the initial hand position to the direction of object movement. However, the robot had only four possible initial positions which restricts the flexibility of manipulations in the execution phase of the learned maps. The high number of required trials (approximately 70) is another unrealistic drawback of this work.

In a different study (Omrcen et al., 2009) the problem of learning a general pushing rule has been addressed. The rule represents the relationship between the point and angle of push on the object's boundary and the observed object motion right after the pushing action. In the learning case the robot experimented with different pushing actions on different objects at different positions. The normalized retinal images of the experimental data served as input to a neural network to predict the object velocity in all directions. However, the input images had to be down-sampled to 20x15 pixels which causes much information loss. Moreover, in the testing case the robot has to drive an optimization process, the computational complexity of which is relatively high.

In Salganicoff et al. (1993), the authors described an on-line learning method for pushing an object to a desired (image) position. The system used past pushing operations to estimate future pushing actions. The main handicap of their approach is that the object is connected to the robot with a rotational point contact.

1.5 Outline and Contributions

This thesis divides into three main parts. The first part highlights the main algorithms used for classifying manipulations, categorizing manipulated objects, learning archetypal SEC models, and executing manipulations from the learned SEC models. In the second part, we evaluate the statistical robustness of the proposed algorithms, whereas the last part addresses different experiments to which those core algorithms are applied.

The contribution of each chapter can be summarized as follows:

- **Chapter 2** was published in Aksoy et al. (2010, 2011a,b) and introduces scene graphs and semantics event chains to analyze the object-action relations in image sequences. This chapter proposes novel approaches to extract and encode the semantics of manipulation actions in conjunction with manipulated objects for the issues of monitoring, learning, and execution.

- **Chapter 3** evaluates the robustness of the proposed semantic methods with synthetic data and compares it with neural networks. All provided experiments in this chapter serve to test action classification, object categorization, and learning phases in the face of different types and degrees of noise.

- **Chapter 4** provides three different application studies each of which benefits from different aspects of the scene graphs and semantic event chains. Furthermore, this chapter discusses the crucial problems observed in real experiments by comparing with the ones driven from simulated environments. The first application study given in this chapter was published in Aksoy et al. (2011a). Main concepts introduced in the second and third application studies are parts of European projects IntellAct and GARNICS, respectively. The last study was also introduced as a part of a patent (Wörgötter et al., 2012a).

Finally, in Chapter 5 the thesis is concluded by comparing the SEC framework with other approaches and by discussing the limitations. We also present an outlook for future investigations.

2

Methods for Analyzing of Object-Action Relations

In cognitive robotics, recognition and execution of a manipulation after learning from demonstration is one of the most intriguing and still unsolved problems. In this chapter, we introduce the so-called "Semantic Event Chain" (SEC) as a novel and generic scheme for manipulations. SECs basically encode object-action relations in the spatiotemporal domain for further semantic analyses. In the next sections, we provide a comprehensive description of the core methods used for recognizing, learning, and executing manipulations with SECs. Parts of this work were also published in Aksoy et al. (2010, 2011a,b).

2.1 Introduction

We mainly aim at defining a generic method for manipulations that can be used to allow an agent to learn by observation not only to distinguish between different manipulations but also to classify the observed objects and to execute manipulations. In this sense, we implemented an approach that gives the agent a basic tool by which it can assess the consequence of a manipulation step by directly linking the symbolic planning domain to the signal domain (image) addressing the difficult cause-effect problem of how to observe and verify self-induced changes in the scene.

We start with providing an overview of different algorithmic steps of our approach (see Fig. 2.1 and 2.2). Fig. 2.1 shows a processing example of a manipulation resulting in its semantic event chain representation. We first extract all frames from the manipulation movie (Fig. 2.1 (a)). Frames (Fig. 2.1 (b)) are then segmented (Fig. 2.1 (c)) by superparamagnetic clustering in a spin-lattice model (Dellen et al., 2009; Abramov et al., 2010), which allows for consistent marker-less tracking (Fig. 2.1 (e)) of the individual segments due to spin-linking across images using optic-flow information. The scene is then represented by undirected and un-weighted graphs (Fig. 2.1 (d)), the nodes and edges of which represent segments and their neighborhood relations, respectively. Graphs can change by continuous distortions (lengthening or shortening

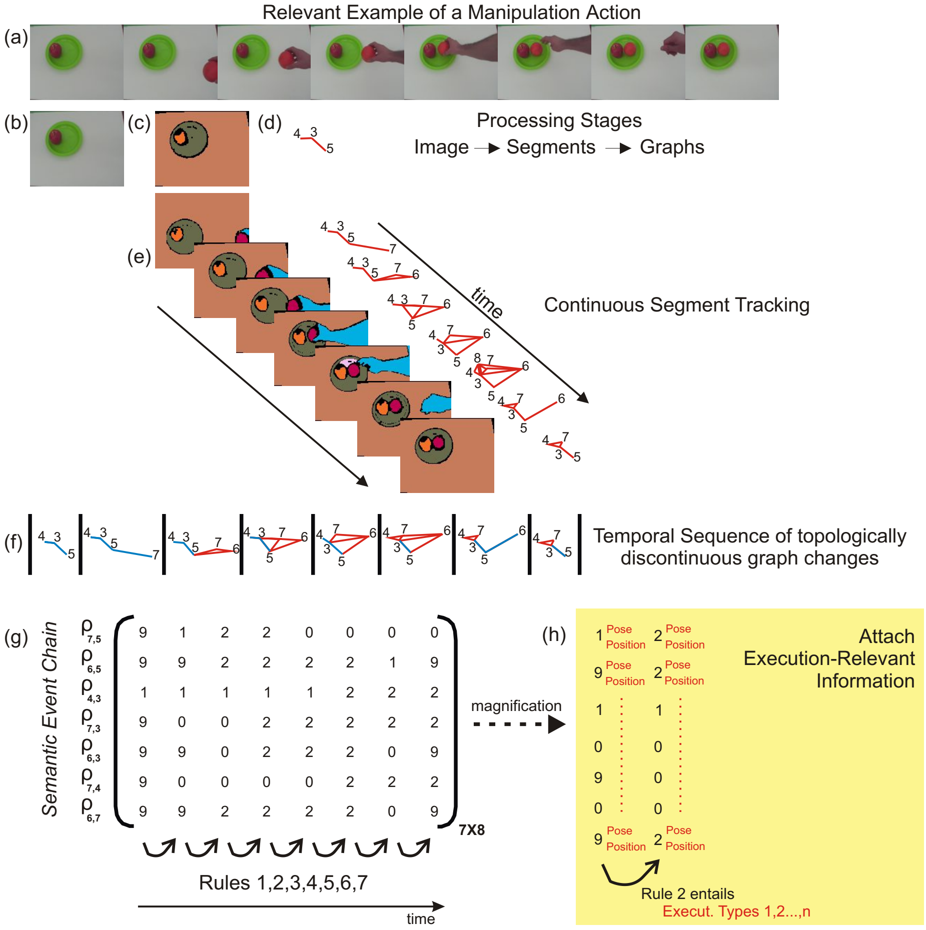


Figure 2.1: *Processing example and semantic event chain representation.* **(a)** *Frames from a movie recorded during a manipulation. All frames* **(b)** *are segmented* **(c)** *by superparamagnetic clustering in a spin-lattice model (Dellen et al., 2009), which also allows for consistent marker-less tracking* **(e)** *of the individual segments. From the image segments, graphs are constructed* **(d)** *where graph nodes represent the segments' centers and graph edges encode whether or not two segments touch each other. Then we encode a manipulation by storing only main graphs between which a topological change has taken place* **(f)**. *Such a change happens whenever an edge or a node has been newly formed or has been deleted. This type of representation is then given by the semantic event chain* **(g)**, *which is a sequence-table, where each entry encodes the spatial relations between each segment pair* $\rho_{i,j}$ *counting graph edges (*2 *means that segments touch (denoted by red edges),* 1 *means that they overlap (denoted by blue edges),* 0 *means that there is no edge between two segments, and absence of a previously existing segment yields* 9*).* **(h)** *Each column of SECs represent a temporal rule which are then enriched with pose and trajectory (position) information for the execution process of manipulations.*

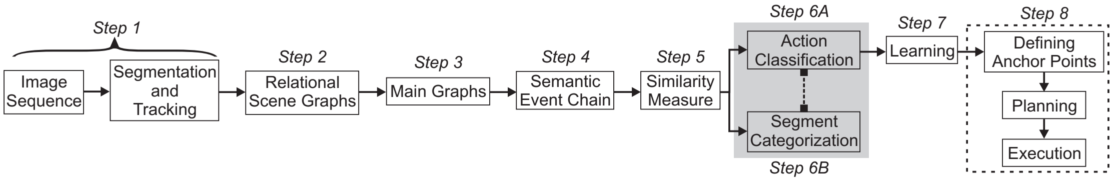


Figure 2.2: *Block diagram of the algorithm*

of edges) or, more importantly, through discontinuous changes (nodes or edges can appear or disappear). Such a discontinuous change represents a natural breaking point: All graphs before are topologically identical and so are those after the breaking point. Hence, we can apply an exact graph-matching method at each breaking point and extract the corresponding topological main graph. The sequence of these main graphs (Fig. 2.1 (f)) thus represents all structural changes in the scene. This type of representation is then encoded by the semantic event chain (Fig. 2.1 (g)), which is essentially a symbolic representation encoding the manipulation by a temporal sequence of rules (motor primitives). Each entry of SECs encodes spatial segment relations where 0 means that there is no edge between two segments, corresponding to two spatially separated segments, 1 means that one segment overlaps with the other completely, and 2 represents segments that touch each other. A special case exists when segment has disappeared, which is denoted by 9. Note that the complete image sequence, which has here roughly 100 frames, is represented by an event chain with a size of only 7×8. The above described steps (1-4) are also presented in Fig. 2.2, showing the block diagram of the complete algorithm. The following steps (5-7) utilize the SEC to compute similarity values between videos showing manipulations (step 5), to perform action classification (step 6A) and conjointly performed segment categorization (step 6B). The approach learns from demonstration an archetypal event chain (model-SEC) consisting only of consistently repeated rows (spatial relations) and columns (motor primitives) (step 7). Finally, the observed manipulation is executed after defining temporal and spatial anchor points and planning phase (step 8) on the learned SEC model which can basically be enriched with required pose and trajectory information at each temporal rule (Fig. 2.1 (h)). In the following, we describe all different algorithmic steps in detail.

2.2 Preprocessing, Segmentation & Tracking (Step 1)

Manipulation movies are recorded in indoor environments with limited context. All movies used in this study can be found at `www.dpi.physik.uni-goettingen.de/~eaksoye/movies.html` (See Appendix A.3). Typical examples are shown in Fig. 2.4.

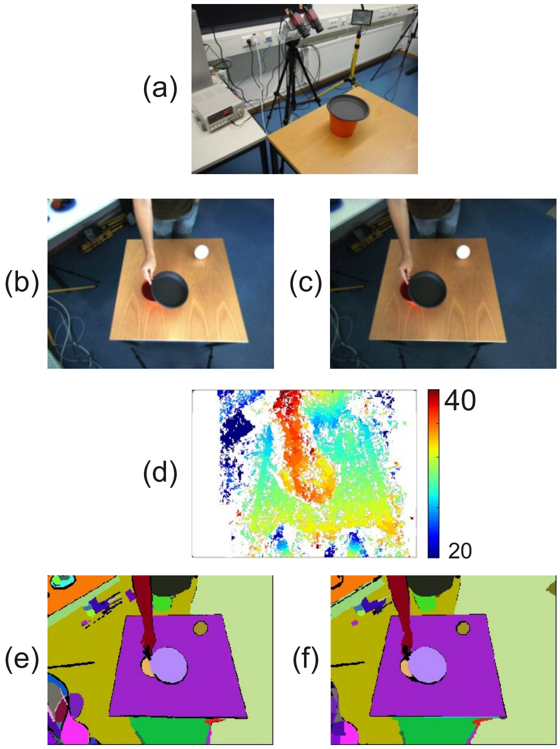


Figure 2.3: *Schematic of recording and visual preprocessing. (a) Stereo camera setup. (b,c) Original example frames from the left and right image sequences. (d) Sparse phase-based disparity map. (e,f) Extracted segments for the left and right image.*

We use a stereoscopic camera setup using AVT Marlin F080C CCD firewire cameras and lenses with variable focal length of 2.7-13.5mm (see Fig. 2.3(a)). Distance to the manipulation scene is about $1.0 - 1.5\ m$. Images are rectified (see Fig. 2.3(b-c)), stereo and optic-flow information is extracted by different standard algorithms (Pauwels and Van Hulle, 2008; Sabatini et al., 2007). An example of a resulting sparse phase-based disparity map is shown in Fig. 2.3(d). For step 1 (Fig. 2.2), we use an image-segmentation method, developed by us earlier, in which segments are obtained and tracked by a 3D linking process (see Fig. 2.3(e-f)). The method has been been described in detail elsewhere (Shylo et al., 2009; Dellen et al., 2009; Dellen and Wörgötter, 2009; Abramov et al., 2010). It is mainly implemented on GPUs and operates close-to real-time at about 23 fps at a resolution of 256 × 320 pixels. For reasons of brevity details are omitted here. The main result from these steps is that we receive consistently tracked image segments, the fate of which can be used to encode a manipulation as described next.

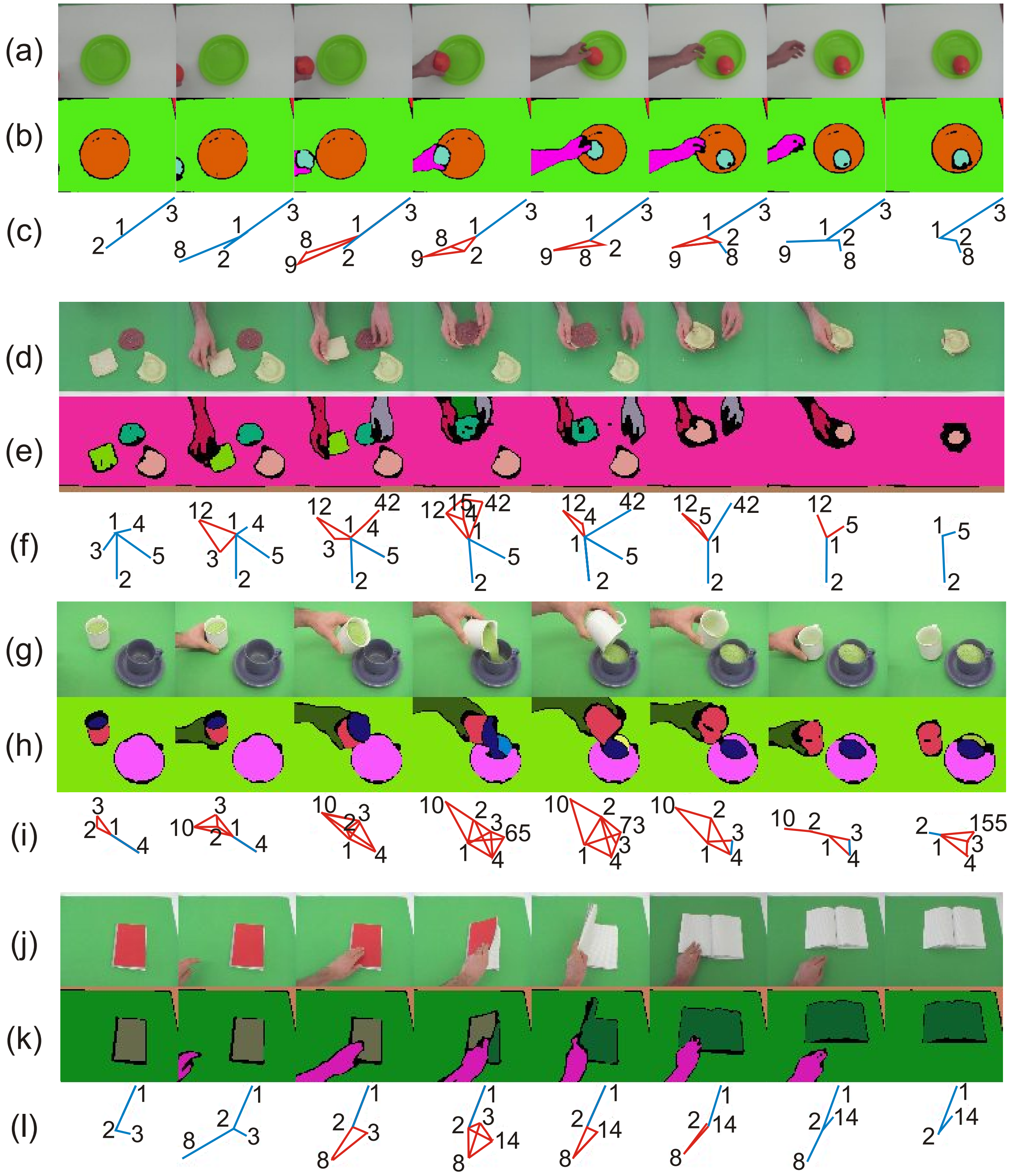


Figure 2.4: *Four different real action types. (a),(d),(g),(j) Original images, (b),(e),(h),(k) corresponding image segments, and (c),(f),(i),(l) scene graphs from the following manipulations: Moving Object, Making Sandwich, Filling Liquid, and Opening Book. In blue and red are indicated Overlapping and Touching relations.*

2.3 Relational Scene Graphs (Step 2)

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

As mentioned in the algorithmic overview above, we define four relations between segments: *Touching=2*, *Overlapping=1*, *Non-touching=0*, and *Absent=9*, which refers to an image segment that is not observed in the scene. We redefined standard concepts used in the field of topology (e.g. hole, neighbor, etc.) on purpose to make the terminology more appropriate in the context of manipulation recognition. Terms such as overlapping and touching are directly referring to primitive manipulations. Whenever necessary, we use 3D-information from our stereo setup to disambiguate perspective effects, which would lead to false relations when using only 2D.

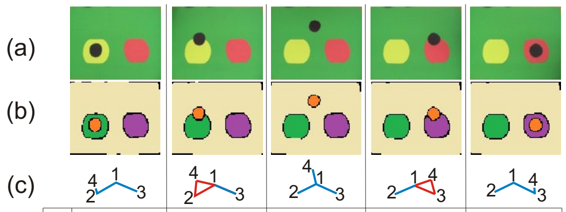
Given that image segments often have strangely-shaped as well as noisy borders, the correct assignment of these relations is non-trivial and we had to design a fast and efficient special algorithm for this. As this is not in the core of this thesis, details are provided only in Appendix A.1. This algorithm gives us the required spatial relations (e.g. *Touching*, *Overlapping*,etc.). The spatial relations can also be calculated by simply counting the number of edges at each graph node as described in Aksoy et al. (2010)

Once the image sequence has been segmented and spatial relations have been extracted, we represent the scene by undirected and unweighted labeled graphs. The graph nodes are the segment labels and plotted at the center of each segment. Nodes are then connected by an edge if segment relations are either *Touching* or *Overlapping*.

Fig. 2.4 shows original frames and corresponding segments and their scene graphs from four different real action types: *Moving Object*, *Making Sandwich*, *Filling Liquid*, and *Opening Book*. In the *Moving Object* action a hand is putting an orange on a plate while moving the plate together with the orange (Fig. 2.4 (a-c)). The *Making Sandwich* action represents a scenario in which two hands are putting pieces of bread, salami, and cheese on top of each other (Fig. 2.4 (d-f)). In the *Filling Liquid* action a cup is being filled with liquid from another cup (Fig. 2.4 (g-i)). The *Opening Book* action describes a scenario in which a hand is opening a book (Fig. 2.4 (j-l)).

2.4 Main Graphs (Step 3)

In order to clarify the remainder of the algorithm better, we use simple, artificial scenes to describe steps 3 to 6 of Fig. 2.2. Real scenes will be referred later. Fig. 2.5 (a-b) depict original frames and their corresponding segments of an artificial *Moving Object* action (sample action 1) in which a black round object is moving from a yellow vessel into a red vessel.



(d)

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

ξ_o^1

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

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

Figure 2.5: *Simple example of the Moving Object action (sample action 1). (a) Original images. (b) Corresponding image segments. (c) Semantic scene graphs. In blue and red are indicated Overlapping and Touching relations. (d) Original semantic event chain (ξ_o^1). (e) Derivative of the semantic event chain (ξ_d^1). (f) Compressed semantic event chain (ξ_c^1).*

Scene graphs, such as those depicted in Fig. 2.4, represent spatial relations between nodes. Unless spatial relations change, the scene graphs remain topologically the same. The only changes in the graph structures are the node positions or the edge lengths depending on the object trajectory and speed. Consequently, any change in the spatial relation between nodes corresponds to a change in the main structure of the scene graphs. Therefore, those changes in the graphs can be employed to define manipulation primitives. Considering this fact, we apply an exact graph-matching method in order to extract the main graphs by computing the eigenvalues and eigenvectors of the adjacency matrices of the graphs (Sumsi, 2008). A change in the eigenvalues or eigenvectors then corresponds to a structural change of the graph. The whole image sequence of the artificial *Moving Object* action has 92 frames, however, after extracting the main graphs, only 5 frames are left, each defining a single manipulation primitive (Fig. 2.5 (c)).

2.5 Semantic Event Chains (SECs) (Step 4)

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

$$\rho_{\text{total}} = n(n-1)/2 \quad , \tag{2.1}$$

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

Since any change in the spatial relations represents an event that defines an action, we refer to this table as *original semantic event chain* (ξ_o). Fig. 2.5 (d) shows it for the artificial action explained above.

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

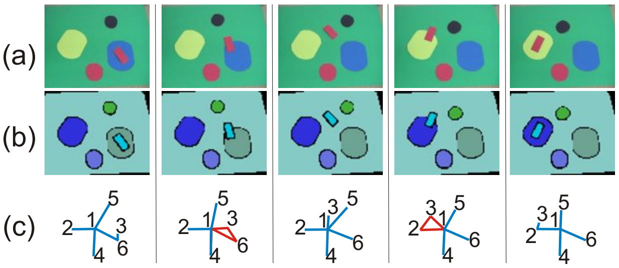
To achieve this, we first perform two data-compression steps. In general, it suffices to only encode the transitions from one state (one column) in the original chain (ξ_o) to another (next column). Therefore, we can perform a derivative-like operation on ξ_o and represent the result by ξ_d to simplify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

For this we scan each row of ξ_o from left to right and substitute "changes" by combining their numerical values into a two-digit number. For example a change from Overlapping to Touching, hence from 1 to 2, is now encoded by 12. When nothing has changed a double digit, like 11, occurs. Rows where nothing ever happens (e.g. row $\rho_{3,2}$ in Fig. 2.5 (d)) are immediately removed since they do not define any event. The resulting representation (ξ_d) is, thus, a mild, loss-less compression of the original one. It is a preprocessing step and is required for the second compression step. Fig. 2.5 (e) shows ξ_d^1 for the sample *Moving Object* action.

Then, in a second compression step all double-digits (00, 11, 22, and 99) are removed leading to ξ_c. This representation has lost all temporal information and is used for the spatial-relational analysis only. ξ_c^1 of the artificial action is given in Fig. 2.5 (f). The original chain (ξ_o) will then be used for the temporal analysis.

2.6 Similarity Measure (Step 5)

Next we will discuss how to calculate the similarity of two actions. Essential this comes down to sub-string search algorithms in the spatial as well as the temporal domain. In the spatial domain we are searching for the correspondences between rows of two compressed event chains to reduce the combinatorics (see section 2.6.1). Then



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

ξ_o^2 ξ_d^2 ξ_c^2

(d) (e) (f)

Figure 2.6: *Different version of the simple Moving Object action (sample action 2). (a) Original images. (b) Respective image segments. (c) Semantic scene graphs. In blue and red are indicated Overlapping and Touching relations. (d) Original semantic event chain (ξ_o^2). (e) Derivative of the semantic event chain (ξ_d^2). (f) Compressed semantic event chain (ξ_c^2).*

in the temporal domain the order of columns is examined to get the final recognition result (see section 2.6.2).

To explain this we created one more sample for the artificial *Moving Object* action. Fig. 2.6 depicts the main graphs with respective image segments of sample action 2 in which a red rectangular object is moving from a blue vessel into a yellow vessel following a different trajectory with different speed as compared to the first sample. Moreover, the scene contains two more objects which are either stationary (red round object) or moving randomly (black round object). Following the same procedure, the *event chain* ξ_o^2 and their compressed versions (ξ_d^2 and ξ_c^2) for the second sample are calculated and given in Fig. 2.6 (d-f). Note that even though the second sample contains more objects, the dimensions of the different chains are accidentally the same. This is of no importance as the sub-string search described next does not rely on dimensions, allowing to compare arbitrarily long action sequences.

2.6.1 Spatial Similarity Analysis

The goal of this subsection is to provide the first of two subsequent analysis steps, required to obtain a final measure of similarity between two event chains. The first step is based on a spatial analysis comparing the rows of compressed event chains (ξ_c^1

and ξ_c^2) accounting for a possibly shuffling of rows in different versions of the same manipulations. This way the number of possible relations is reduced before we can finally, in the second step, find the true similarity measures.

Let ξ_c^1 and ξ_c^2 be the sets of rows for the two manipulations, written as a matrix (e.g. Fig. 2.5 (f) and 2.6 (f)):

$$\xi_c^1 = \begin{bmatrix} r^1_{1,1} & r^1_{1,2} & \cdots & \cdots & r^1_{1,\gamma^1_1} & & \\ r^1_{2,1} & r^1_{2,2} & \cdots & r^1_{2,\gamma^1_2} & & & \\ \vdots & \vdots & \ddots & \vdots & & & \\ r^1_{m,1} & r^1_{m,2} & \cdots & \cdots & \cdots & r^1_{m,\gamma^1_m} \end{bmatrix},$$

and

$$\xi_c^2 = \begin{bmatrix} r^2_{1,1} & r^2_{1,2} & \cdots & \cdots & \cdots & \cdots & r^2_{1,\gamma^2_1} \\ r^2_{2,1} & r^2_{2,2} & \cdots & \cdots & r^2_{2,\gamma^2_2} & & \\ \vdots & \vdots & \ddots & \vdots & & & \\ r^2_{k,1} & r^2_{k,2} & \cdots & \cdots & \cdots & r^2_{k,\gamma^2_k} & \end{bmatrix},$$

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

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

The lengths of the rows are usually different and given by indices γ.

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

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

$$f_t = \frac{100}{\gamma_b^2} \sum_{i=1}^{\gamma_a^1} \delta_i \quad , \tag{2.2}$$

where γ_b^2 is the normalization factor and i is the row index and with

$$\delta_i = \begin{cases} 1 & \text{if } r^1_{a,i} = r^2_{b,i+t-1} \\ 0 & \text{else} \end{cases} \quad , \tag{2.3}$$

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

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

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

For the case $\gamma_a^1 > \gamma_b^2$, a symmetrical procedure is performed by interchanging all indices of Eqs. (2.2), (2.3) above.

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

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

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

$$\psi_{spatial} = \frac{1}{m}\sum_{i=1}^{m} \max_j(M_{i,j}), \quad j \in [1, \cdots, k], \tag{2.4}$$

if

$$\max_j(M_{i,j}) = \max_t(M_{t,j}), \quad t \in [1, \cdots, m] \quad . \tag{2.5}$$

Note that Eq. (2.5) above makes the similarity measurement symmetrical.

The complete similarity matrix ($\zeta_{spatial}$) between the artificial *Moving Object* samples (ξ_c^1 and ξ_c^2) is given in Table 2.1. Visual inspection of ξ_c^1 (Fig. 2.5 (f)) and ξ_c^2 (Fig. 2.6 (f)) immediately confirms these similarity values. We find 100% similarity ($\psi_{spatial}$) between both artificial “manipulations”. One can see that Eq. 2.4 can still lead to multiple assignments of permutations with the same maximal similarity. This is resolved by the temporal similarity measurement stage following below. In more realistic scenes 100% is, of course, often not reached (see Fig. 2.7 below) and one needs to define a threshold above which one would consider two similarity values as equal.

However, we also observe that there are several 100% matches between rows in these examples. As a consequence, for the second example two permutations $\xi_c^{2,1}$ and $\xi_c^{2,2}$ exist with equal row-matching probability as given in Table 2.2. Note, for real scenes after thresholding, even more permutations might exist. Hence, the analysis in not yet complete.

2.6.2 Temporal Similarity Analysis

In the now following second step we can use the time sequence, encoded in the order of events in the event chains to find the best matching permutation and thereby arrive at the final result. To this end will now use temporal information, hence the original chain (ξ_o), to find the truly matching permutation. Note that the derivative like term (ξ_d) can also be used for temporal analysis as presented in Aksoy et al. (2011a), however, the real similarity value can not be reached since it is slightly compressed.

Thus, we compare both permutations $\xi_o^{2_p}$, $p = 1, 2$ of the second event chain, shown in Tab. 2.2 (a-b), with the first one ξ_o^1. In such cases where ξ_o^1 has more rows than $\xi_o^{2_p}$, hence rows which have no correspondences, $\xi_o^{2_p}$ will be filled with dummy rows that have no possible similarities.

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

$$\xi_o^1 = \begin{bmatrix} e^1_{1,1} & e^1_{1,2} & \cdots & e^1_{1,u} \\ e^1_{2,1} & e^1_{2,2} & \cdots & e^1_{2,u} \\ \vdots & \vdots & \ddots & \vdots \\ e^1_{q,1} & e^1_{q,2} & \cdots & e^1_{q,u} \end{bmatrix},$$

and

$$\xi_o^{2_p} = \begin{bmatrix} e^{2_p}_{1,1} & e^{2_p}_{1,2} & \cdots & e^{2_p}_{1,v} \\ e^{2_p}_{2,1} & e^{2_p}_{2,2} & \cdots & e^{2_p}_{2,v} \\ \vdots & \vdots & \ddots & \vdots \\ e^{2_p}_{q,1} & e^{2_p}_{q,2} & \cdots & e^{2_p}_{q,v} \end{bmatrix},$$

where $\xi_o^{2_p}$ is a permutation of ξ_o^2 and $e^1_{i,j}$ and $e^{2_p}_{i,j}$ represent the possible relations between any segment pair

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

Table 2.1: *Similarity table ($\zeta_{spatial}$). Similarity values between the rows of ξ_c^1 and ξ_c^2 the artificial Moving Object samples.*

$$\begin{array}{ccc} \xi_c^1 & & \xi_c^{2_1} \\ \rho_{2,1} & \Leftarrow 100\% \Rightarrow & \rho_{2,1} \\ \rho_{3,1} & \Leftarrow 100\% \Rightarrow & \rho_{6,1} \\ \rho_{4,1} & \Leftarrow 100\% \Rightarrow & \rho_{3,1} \\ \rho_{4,2} & \Leftarrow 100\% \Rightarrow & \rho_{3,6} \\ \rho_{4,3} & \Leftarrow 100\% \Rightarrow & \rho_{3,2} \end{array} \qquad \begin{array}{c} \xi_o^{2_1} \\ \begin{bmatrix} 1 & 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 0 & 2 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \end{bmatrix} \end{array}$$

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

$$\begin{array}{ccc} \xi_c^1 & & \xi_c^{2_2} \\ \rho_{2,1} & \Leftarrow 100\% \Rightarrow & \rho_{6,1} \\ \rho_{3,1} & \Leftarrow 100\% \Rightarrow & \rho_{2,1} \\ \rho_{4,1} & \Leftarrow 100\% \Rightarrow & \rho_{3,1} \\ \rho_{4,2} & \Leftarrow 100\% \Rightarrow & \rho_{3,6} \\ \rho_{4,3} & \Leftarrow 100\% \Rightarrow & \rho_{3,2} \end{array} \qquad \begin{array}{c} \xi_o^{2_2} \\ \begin{bmatrix} 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 0 & 2 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \end{bmatrix} \end{array}$$

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

Table 2.2: *Permutations* $\xi_o^{2,p}$.

$$e_{i,j} \in \{0, 1, 2, 9\} \ . \tag{2.6}$$

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

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

$$\theta_{a,b} = \frac{100}{q} \sum_{j=1}^{q} \delta_j \quad , \tag{2.7}$$

where j is the column index and with

$$\delta_j = \begin{cases} 1 & \text{if } e_{j,a}^1 = e_{j,b}^{2_p} \\ 0 & \text{else} \end{cases} \quad , \tag{2.8}$$

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

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

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

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

$$\psi_{temporal}^p = \frac{1}{u}\sum_{i=1}^{u} \max_j(\theta_{i,j}), \quad j \in [1, \cdots, v] \tag{2.9}$$

For each permutation $\xi_o^{2_p}$ given in Tab. 2.2 a $\psi_{temporal}^p$ value is calculated by using Eqs. (2.7), (2.8), and (2.9), yielding $\psi_{temporal}^1 = 84\%$ and $\psi_{temporal}^2 = 100\%$. Note that we use Longest Common Subsequence (LCS) in order to guarantee that the order of columns is the same. LCS is generally used to find the longest sequence observed in both input sample sequences. Columns of event chains are used as sequences for this task. Since the number of sequences is constant, the problem is solvable in polynomial time by dynamic programming. Consequently, our two sample actions have 100% similarity and permutation $\xi_o^{2_2}$ represents the final row correspondences to the first action. The best matching permutation is further used for categorizing objects as described in sub-section 2.8.

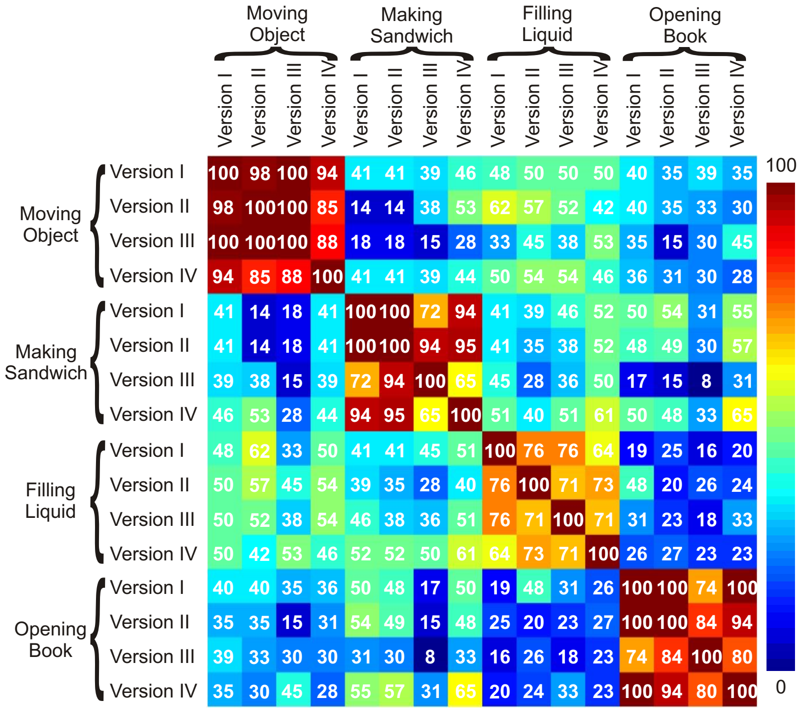
As mentioned above, in real scenes often 100% are not reached and we call two actions "type-similar" as soon as their final similarity value exceeds a certain threshold. We would also like to point out that the final spatial and temporal similarity values are not necessarily identical. Action classification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

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

2.7 Action Classification (Step 6A)

We applied our framework to four different real action types: *Moving Object*, *Making Sandwich*, *Filling Liquid*, and *Opening Book* (see Fig. 2.4). For each of these actions, we recorded four movies with highly different trajectories, speeds, hand positions, and object shapes. These examples were introduced to show that really different instantiations of a manipulation will still be recognized as belonging to the same type.

Event chains of each real test data are compared with each other by using the similarity measurement algorithm explained in Step 5. We used γ_a^1 as a normalization factor in Eq. (2.2) and thresholded the maximum similarity in Eq. (2.4) in order to increase the number of permutations, thus, accuracy. The resulting final maximal similarity values are given in Fig. 2.7. Each test data has high similarity with the other versions of its type-similar action (see close to diagonal) and almost always, the similarity between type-similar actions is much bigger than the similarity between non-type-similar actions. The minimum similarity value for type-similar actions is measured as 64% between the first and forth versions of *Filling Liquid*, but the similarity between *Filling Liquid* and non-type-similar actions is far less. Setting a threshold at 64% would, across all examples, lead to zero false negatives and to one false positives (opening book version IV and making sandwich version IV), which



		Moving Object				Making Sandwich				Filling Liquid				Opening Book			
		Version I	Version II	Version III	Version IV	Version I	Version II	Version III	Version IV	Version I	Version II	Version III	Version IV	Version I	Version II	Version III	Version IV
Moving Object	Version I	100	98	100	94	41	41	39	46	48	50	50	50	40	35	39	35
	Version II	98	100	100	85	14	14	38	53	62	57	52	42	40	35	33	30
	Version III	100	100	100	88	18	18	15	28	33	45	38	53	35	15	30	45
	Version IV	94	85	88	100	41	41	39	44	50	54	54	46	36	31	30	28
Making Sandwich	Version I	41	14	18	41	100	100	72	94	41	39	46	52	50	54	31	55
	Version II	41	14	18	41	100	100	94	95	41	35	38	52	48	49	30	57
	Version III	39	38	15	39	72	94	100	65	45	28	36	50	17	15	8	31
	Version IV	46	53	28	44	94	95	65	100	51	40	51	61	50	48	33	65
Filling Liquid	Version I	48	62	33	50	41	41	45	51	100	76	76	64	19	25	16	20
	Version II	50	57	45	54	39	35	28	40	76	100	71	73	48	20	26	24
	Version III	50	52	38	54	46	38	36	51	76	71	100	71	31	23	18	33
	Version IV	50	42	53	46	52	52	50	61	64	73	71	100	26	27	23	23
Opening Book	Version I	40	40	35	36	50	48	17	50	19	48	31	26	100	100	74	100
	Version II	35	35	15	31	54	49	15	48	25	20	23	27	100	100	84	94
	Version III	39	33	30	30	31	30	8	33	16	26	18	23	74	84	100	80
	Version IV	35	30	45	28	55	57	31	65	20	24	33	23	100	94	80	100

Figure 2.7: *Similarity values between event chains of the real test data set.*

would be confused with other manipulations. This is interesting as opening or closing a book can indeed look very similar to the making sandwich (picking up a book-cover and closing it indeed looks very similar from picking up an object and putting on another object). In the making-sandwich action many subcomponents exist where objects are being moved. Thus, in general the manipulation analysis shown in Fig. 2.7 corresponds very well to our human understanding of action (sub-)components!

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

2.8 Segment Categorization (Step $6B$)

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

We will explain this using again the two artificial examples (Fig. 2.5 and Fig. 2.6). The row similarity values of the second permutation given in Table 2.2 (b) represent the correspondences between manipulated nodes in ξ^1 and ξ^2.

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

This can be achieved in a fully unsupervised way by a simple counting procedure. We first analyze which node number in ξ^1 is repeating in conjunction with which node number in ξ^2 in Table 2.2. We start with node number 1 in ξ^1, which occurs in relations $\rho_{2,1}, \rho_{3,1}$ and $\rho_{4,1}$. Its corresponding best matching relations are given by:

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

While node 1 is repeating three times in ξ^1 (left side), the same node number 1 in ξ^2 (right side) 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 examples, ξ^1 and ξ^2, had the same roles. In fact, both graph nodes represent the green background in both actions.

We continue the node relation analysis with node number 2 in ξ^1, and obtain

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

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 placed and from which they then move away.

For the case of node number 3 in ξ^1 we obtain

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

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

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

Node number 4 in ξ^1 and node number 3 in ξ^2 are both 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 the same highest value more than once, e.g. having two times 100% similarity values in the same row of the similarity matrix, segment categorization might lead to ambiguous results, i.e. one segment would correspond to two different segment in the other manipulation. This sort of conflicts can be solved by taking the second highest values in the similarity matrix and calculating the node-similarity again. This way we always achieve unique segment categorization results.

We applied this categorization algorithm to our four different real manipulation scenes (Fig. 2.4). The results showed that the manipulated segments in each action type can be categorized according to their roles in the actions. Fig. 2.8 illustrates the categorization results, e.g. the *Moving Object* actions include three different segment groups here named by their object-names for simplicity (apples or oranges, plates, and hands) each of which performed different roles. In the *Filling Liquid* action the hands are grouped correctly despite having different poses. Note that for the sake of simplicity the backgrounds are ignored in Fig. 2.8 although they are also detected

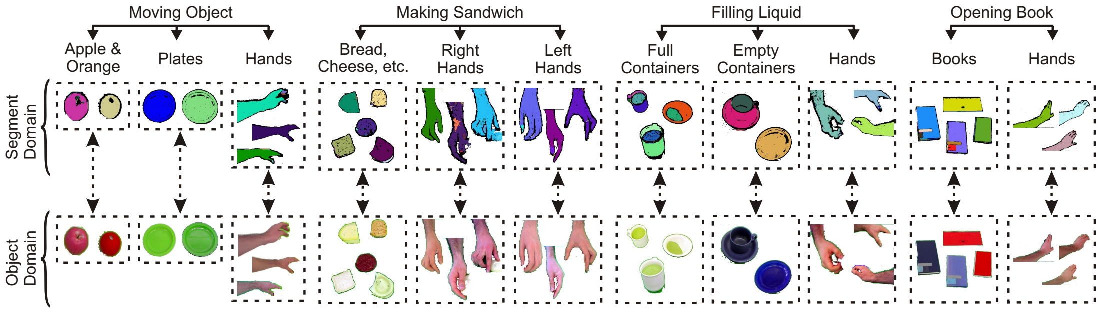


Figure 2.8: *Segment categorization results. In each action type, the manipulated segments can be classified and grouped based on their action roles. Note, classification happens at the level of segments or segment groups. A link to the object-domain (indicated by the dashed arrows) could be introduced by including explicit object models, but this is not part of this study.*

and grouped correctly.

While we are here strictly at the level of segments it is evident (albeit non-trivial!) that this unsupervised categorization process could be coupled to object models, thus, providing access to object categorization, too. It is interesting to remark that in this case any object-like entity will be classified strictly in the context of the observed manipulation. Thus, steps 6A and 6B are tightly linked as depicted by the gray box in Fig.2.2. For example a “cup-being-filled” would be grouped with other objects-being-filled. The same cup, when occurring in an action of “cup-used-as-pedestal” (where the cup is first turned upside down and then something is put on top), would be classified together with other objects-used-as-pedestals. This relates to the cognitive concept of affordances (Gibson, 1977) and will be discussed later in more detail.

2.9 Learning Algorithm (Step 7)

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

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

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

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

Note, online learning could suffer from bad first examples with which all next following manipulations would be classified. There are obvious work-arounds, for example cross-comparing the manipulations with each other. Ultimately, batch-mode learning is more useful. For this one would first record scenes from many manipulations, then perform clustering of the similarity matrix (e.g. Fig. 2.10) after which learning can be done for each cluster in the same way as described above.

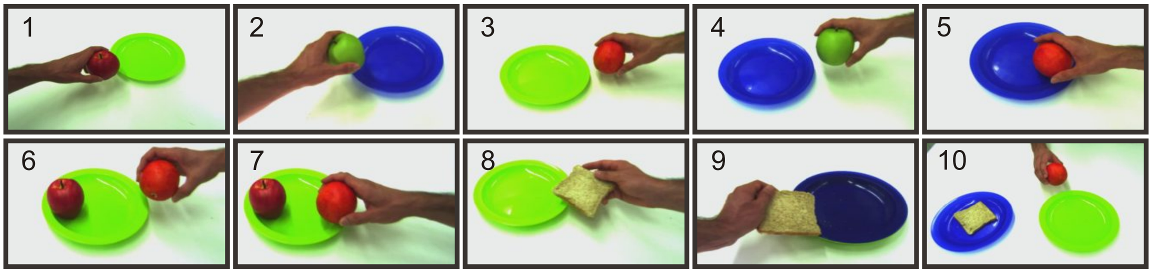


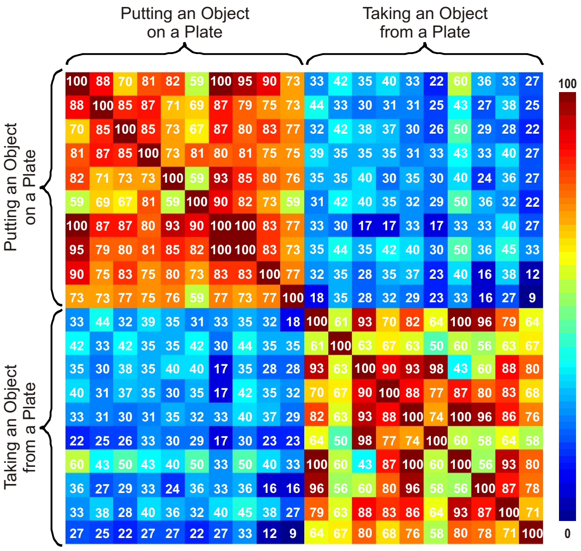
Figure 2.9: *Sample frames from* 10 *different versions of a "Putting an object on a plate" action. In this action type a hand is appearing in the scene, putting different kinds of objects (e.g. apple, orange, a piece of bread, etc.) on a plate following different trajectories with different speeds, and then leaving the scene.*

We applied the learning framework in batch-mode to two different manipulation types: *"Putting an object on a plate"* and *"Taking an object from a plate"* each of which has 10 different versions with strongly different trajectories, speeds, hand positions, and objects (see Fig. 2.9 to get an impression of the level of difference).

Unsupervised classification of the similarity matrix (see Fig. 2.10) is used to classify those 20 versions. Note, some times high similarity values (around 50%) are observed between non-type-similar actions. The reason is that except for the sequencing, which is inverted for "putting" versus "taking", primitives of both action types necessarily look similar. Differences are big enough, though, such that unsupervised classification will still lead to completely correct classification. Due to noisy segmentation low similarity values (around 40%) are also observed between type-similar actions. However, such noisy outcomes do not affect the classification phase.

Next, a SEC model is learned for each manipulation class by searching for the similar common rows and columns observed in all 10 different versions as explained above. Fig. 2.10 (b-c) show the learned SEC models for both action types with corresponding row (ω_i^r) and column (ω_i^c) weight values. To prove the accuracy of the learned SEC models we prepared 5 test movies which all contain *both* action types – putting and taking –, but performed in different temporal order (or sometimes with two hands at the same time!). Fig. 2.11 shows some sample frames from each of the test movies.

Fig. 2.12 depicts the similarity results between two learned models and all 25 movies, 20 of which are the training data and the remaining 5 are unknown test data. Similarity is measured as described in Step 5. In red and blue are indicated the similarities for a given movie with the *"Putting an object on a plate"* and *"Taking an object from a plate"* models, respectively. For the first 10 training data the learned model of *"Putting an object on a plate"* has higher a similarity, whereas the model of *"Taking an object from a plate"* has a lower one (Fig. 2.12, green area). It is the



		Putting an Object on a Plate										Taking an Object from a Plate								
Putting an Object on a Plate	100	88	70	81	82	59	100	95	90	73	33	42	35	40	33	22	60	36	33	27
	88	100	85	87	71	69	87	79	75	73	44	33	30	31	31	25	43	27	38	25
	70	85	100	85	73	67	87	80	83	77	32	42	38	37	30	26	50	29	28	22
	81	87	85	100	73	81	80	81	75	75	39	35	35	35	31	33	43	33	40	27
	82	71	73	73	100	59	93	85	80	76	35	35	40	30	35	30	40	24	36	27
	59	69	67	81	59	100	90	82	73	59	31	42	40	35	32	29	50	36	32	22
	100	87	87	80	93	90	100	100	83	77	33	30	17	17	33	17	33	33	40	27
	95	79	80	81	85	82	100	100	83	73	35	44	35	42	40	30	50	36	45	33
	90	75	83	75	80	73	83	83	100	77	32	35	28	35	37	23	40	16	38	12
	73	73	77	75	76	59	77	73	77	100	18	35	28	32	29	23	33	16	27	9
Taking an Object from a Plate	33	44	32	39	35	31	33	35	32	18	100	61	93	70	82	64	100	96	79	64
	42	33	42	35	35	42	30	44	35	35	61	100	63	67	63	50	60	56	63	67
	35	30	38	35	40	40	17	35	28	28	93	63	100	90	93	98	43	60	88	80
	40	31	37	35	30	35	17	42	35	32	70	67	90	100	88	77	87	80	83	68
	33	31	30	31	35	32	33	40	37	29	82	63	93	88	100	74	100	96	86	76
	22	25	26	33	30	29	17	30	23	23	64	50	98	77	74	100	60	58	64	58
	60	43	50	43	40	50	33	50	40	33	100	60	43	87	100	60	100	56	93	80
	36	27	29	33	24	36	33	36	16	16	96	56	60	80	96	58	56	100	87	78
	33	38	28	40	36	32	40	45	38	27	79	63	88	83	86	64	93	87	100	71
	27	25	22	27	27	22	27	33	12	9	64	67	80	68	76	58	80	78	71	100

(a)

(b)

$$\begin{array}{ccccccccccc}
\omega^c & 0.983 & 0.933 & 0.717 & 0.833 & 0.683 & 0.899 & 0.733 & & & \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & & \\
\rho_{2,1} & 2 & 2 & 2 & 1 & 1 & 2 & 2 & \rightarrow & \omega^r_{2,1} & 0.95 \\
\rho_{3,1} & 9 & 1 & 1 & 1 & 0 & 0 & 0 & \rightarrow & \omega^r_{3,1} & 0.51 \\
\rho_{5,1} & 9 & 9 & 1 & 1 & 1 & 1 & 9 & \rightarrow & \omega^r_{5,1} & 0.9 \\
\rho_{3,2} & 9 & 0 & 0 & 1 & 1 & 2 & 2 & \rightarrow & \omega^r_{3,2} & 0.892 \\
\rho_{5,2} & 9 & 9 & 0 & 0 & 1 & 0 & 9 & \rightarrow & \omega^r_{5,2} & 0.8 \\
\rho_{5,3} & 9 & 9 & 1 & 1 & 1 & 0 & 9 & \rightarrow & \omega^r_{5,3} & 0.934
\end{array}$$

(c)

$$\begin{array}{ccccccccccc}
\omega^c & 0.767 & 0.883 & 0.799 & 0.784 & 0.915 & 0.916 & 0.883 & & & \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & & \\
\rho_{2,1} & 2 & 2 & 1 & 1 & 1 & 2 & 2 & \rightarrow & \omega^r_{2,1} & 0.642 \\
\rho_{6,1} & 0 & 0 & 0 & 0 & 1 & 1 & 9 & \rightarrow & \omega^r_{6,1} & 0.733 \\
\rho_{7,1} & 9 & 1 & 1 & 1 & 1 & 1 & 9 & \rightarrow & \omega^r_{7,1} & 0.95 \\
\rho_{6,2} & 2 & 2 & 1 & 1 & 1 & 0 & 9 & \rightarrow & \omega^r_{6,2} & 0.792 \\
\rho_{7,2} & 9 & 0 & 1 & 0 & 0 & 0 & 9 & \rightarrow & \omega^r_{7,2} & 0.8 \\
\rho_{7,6} & 9 & 0 & 1 & 1 & 1 & 1 & 9 & \rightarrow & \omega^r_{7,6} & 0.907
\end{array}$$

Figure 2.10: *(a) Similarity values between event chains of "Putting an object on a plate" and "Taking an object from a plate" actions. (b) The learned SEC model for the action type "Putting an object on a plate" with corresponding row (ω_i^r) and column (ω_i^c) weight values. These weight vectors are just for illustration since different weight values might be observed for different action types due to degree of noise in the event chains. (c) Same for "Taking an object from a plate".*

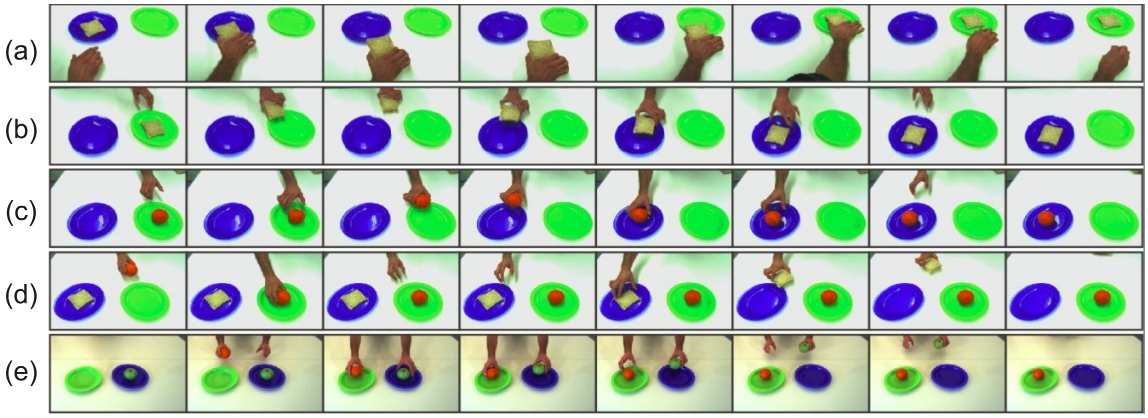


Figure 2.11: *Sample frames from* 5 *different mixed actions in which both manipulation types "Putting an object on a plate" and "Taking an object from a plate" are performed in different orders. (a) A hand is first taking a piece of bread from a plate and then putting it on a different plate. (b) Another piece of bread is moved from one plate to another with a different trajectory. (c) A hand is replacing an orange. (d) A hand is first putting an orange on a plate and then taking a piece of bread from another plate. (e) A hand is putting an orange on a plate and in the mean time the other hand is simultaneously taking an apple from the second plate.*

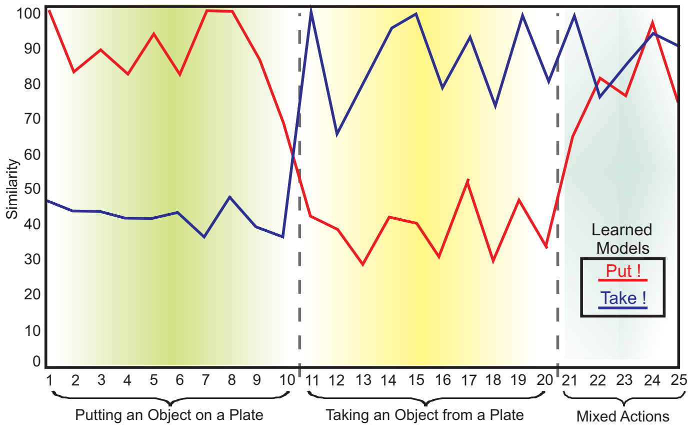


Figure 2.12: *Similarity results between the two learned modes and all* 25 *movies. In red and blue are indicated the similarities for a given movie with the "Putting an object on a plate" and "Taking an object from a plate" models, respectively. First* 20 *data are the training data and represent different versions of the "Putting an object on a plate" and "Taking an object from a plate" actions, respectively. The last* 5 *data represent the mixed actions used for testing the learned models.*

other way around for the next 10 training data (Fig. 2.12 yellow area). However, for the last 5 test data, in which both manipulation types are performed in different orders both learned models have high similarity. (Fig. 2.12 blue area). When doing time-slicing (data not shown) one sees that the similarity in the last 5 data for either manipulation increases together with the completion of the respective manipulation. Thus, one after the other in the first 4 movies and simultaneously in the last one, where both actions are performed simultaneously.

2.10 Manipulation Execution with SECs (Step 8)

In this last step of the whole framework, we will address the problem of executing a manipulation starting from a SEC. Clearly, the naked SECs explained so far, do not contain information regarding pose, trajectory, and object recognition without which a meaningful execution can not be realized. Therefore, in the process of learning a manipulation model (step 7) additional information must be stored, for example the start and endpoint of movement trajectories. As the SEC provides a temporal sequence of rules, we have well defined temporal anchor points when we have to store the additionally required trajectory information.

Explanation of this section will mainly be based on a pushing action which is performed by means of SECs and additional stored information. For this purpose, we used the Webots software that simulates a 6 DOF Neuronics Katana robot arm. The experiment consists of two phases: Learning and execution (steps 7-8 in Fig. 2.2). In the first phase we perform manipulation demonstration which is repeated using different setups. For this we already use the robot simulation and program it by hard-coding to push a red box to a green box on a table until they touch each other and then the robot is going to a home position. Note that it is not required that the red and green boxes should touch each other in some exact configuration just to ignore the pose estimation problem which is not in the focus of this thesis.

From these demonstrations a SEC-model is then learned as explained in the previous section. During learning also additional decisive information, for example relative coordinate frames and information about motion start and endpoints, is recorded. In the second phase (execution), we use the SEC and the additional information to let the robot execute a similar pushing action regardless of the initial state of the table.

2.10.1 Segmentation and SEC-generation

Fig. 2.13 and Fig. 2.14 show a processing example of a manipulation resulting in its semantic event chain representation. We first extract all frames from the manipulation movie (Fig. 2.13 (a)) with corresponding depth maps from a range finder (Fig. 2.13 (b)). Frames are then segmented by a simple HSV color-based segmentation algorithm, which allows for consistent marker-less tracking of each object

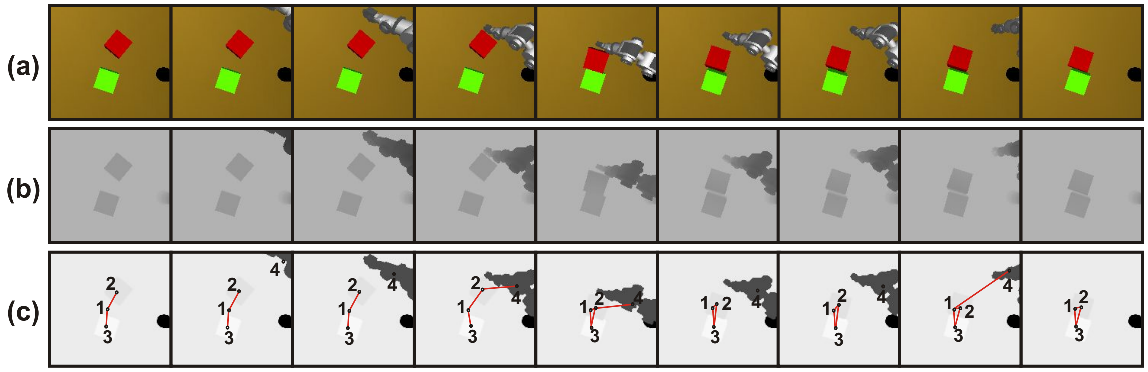


Figure 2.13: *Pushing action.* **(a)** *Original images from a movie recorded during the action.* **(b)** *Corresponding depth map from a range finder.* **(c)** *Corresponding HSV color-based segmented images with extracted 3D scene graphs (see steps (1-2) in Fig. 2.2). Note that each object is represented by a unique segment label (e.g.* 1, 2, 3, *and* 4 *that represent table, red box, green box, and robot arm, respectively). Graph nodes represent the segments' centers and graph edges encode whether or not two segments touch each other in 3D.*

(Fig. 2.13 (c)). Note that each object is represented by a unique segment label (e.g. 1, 2, 3, and 4). Once segments are calculated we drive a simple color-based object recognition algorithm to replace those unique labels with object names (e.g. *table*, *red box*, *green box*, and tip of *robot arm* instead of 1, 2, 3, and 4, respectively). After this step the agent knows which segment corresponds to which object. The scene is then represented by semantic graphs (Fig. 2.13 (c)). Nodes represent object center points and edges between nodes exist whenever two objects touch each other in 3D. As explained in step 3 we apply an exact graph matching method to extract the corresponding topological main graphs. The sequence of these main graphs represents all structural changes (manipulation primitives) in the scene. The movie frames that hold such changes are called *"Key Frames"*. Fig. 2.14 (a-c) show the *"Key Frames"* with corresponding depth map, segments, and main graphs for the action in Fig. 2.13. The final semantic event chain representation is depicted in (Fig. 2.14 (d)). In SECs continuous time is now replaced by time-chunks where the same main graph persists until in the next chunk a new one appears. Each row in a SEC represents the temporally changing spatial relations between one pair of objects in the scene, for example the first row in (Fig. 2.14 (d)) shows the relation between the red box and green box. Note that in this section we will omit the spatial relation *overlapping* since it is also different interpretation of the relation *touching* in 3D world. Therefore, from now on we will use three spatial relations; *no connection* (i.e. *no touching*), *touching*, and *absence*), each will be represented by letters (e.g. N, T, and A) instead of decimal numbers (e.g. 0, 2, and 9) for the sake of simplicity.

Consequently, the complete image sequence, which has here roughly 320 frames,

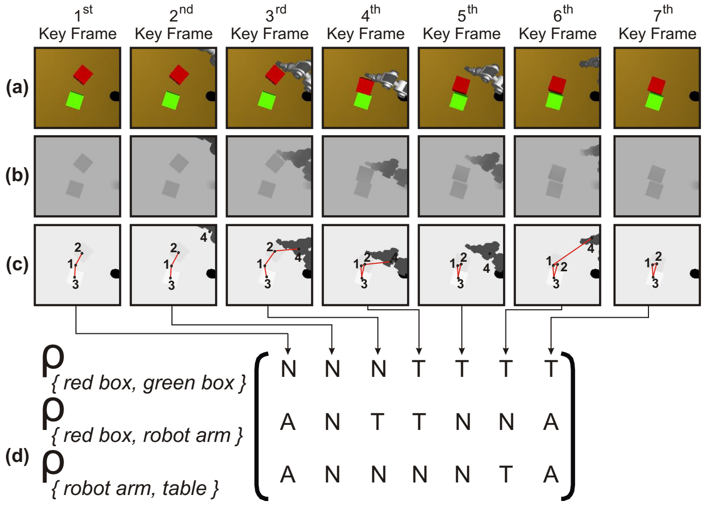


Figure 2.14: *Semantic event chain representation.* **(a)** *Original "Key Frames".* **(b)** *Corresponding depth maps.* **(c)** *Corresponding HSV color-based segmented images with extracted main graphs (see step 3 in Fig. 2.2).* **(d)** *Corresponding semantic event chain (see step 4 in Fig. 2.2), which is a sequence-table, where each entry encodes the spatial relations between each segment pair* $\rho_{i,j}$ *at each main graph. There are three possible spatial relations defined between segments: no connection (N), touching (T), and absence (A). N means that there is no edge between two segments, corresponding to two spatially separated segments, T represents segments that touch each other, and absence of a previously existing segment yields A.*

is represented by an event chain with a size of only 3×7. Note that several spatial relations, for example between *green box and robot arm*, are not included in this SEC since they do not contain any N-T or T-N transitions, which are decisive for a manipulation. Thus, we can always ignore such rows in the semantic event chain since they do not describe any manipulation relevant event. The SEC is also converted into a machine readable format to be able to encode additional object, pose, and trajectory information. Please see the Appendix A.2 for a machine readable (GraphML) format of the SEC.

These aspects are represented by steps (1-4) in Fig. 2.2, showing the block diagram of the complete algorithm.

2.10.2 Defining Temporal Anchor Points

For learning we record the same pushing action from 10 different perspectives by changing the camera positions and extract the corresponding SECs. Fig. 2.15 shows the same, specific moment of the manipulation for each of the 10 different manipulation instances which allows us to get an impression of the level of perspective

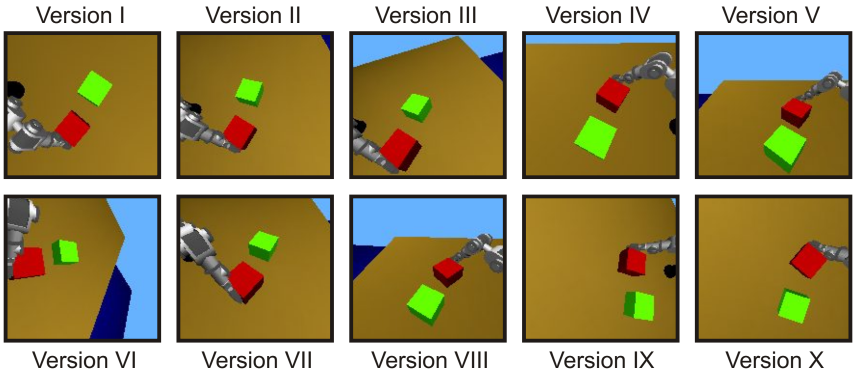
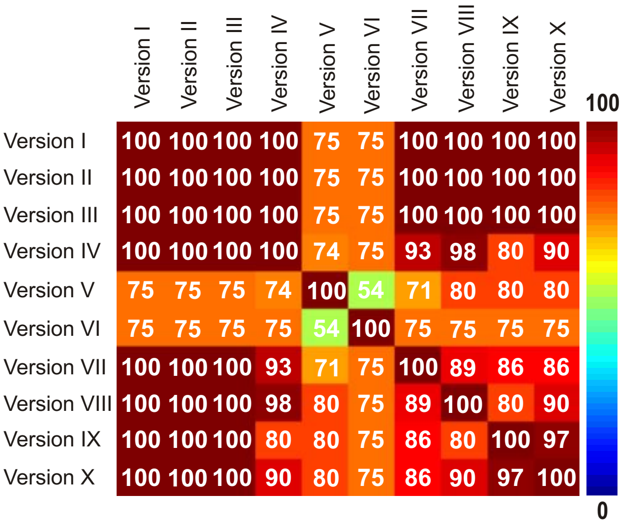


Figure 2.15: *Differences between the same moment of the pushing manipulation in all 10 different versions.*

difference. All movies used in this study can be found at `http://www.dpi.physik.uni-goettingen.de/~eaksoye/movies.html` (See Appendix A.3). As described in step 5, we then calculate the pairwise percent-similarity between the manipulations to show that indeed a high mutual similarity exists between those 10 repetitions. Fig. 2.16 depicts the confusion matrix where one outlier with only 54% similarity occurred due to some error in the image segmentation.

Having assured that the individual SECs represent indeed the same manipulation we are allowed to perform a weighted average and extract all re-occurring rows and columns in the ten SECs as explained in step 7. The resulting SEC is shown in Fig. 2.17. Weights ω represent the normalized occurrence frequency of a given row or column and are sometimes less than 1.0 due to the situation that not all rows



	Version I	Version II	Version III	Version IV	Version V	Version VI	Version VII	Version VIII	Version IX	Version X
Version I	100	100	100	100	75	75	100	100	100	100
Version II	100	100	100	100	75	75	100	100	100	100
Version III	100	100	100	100	75	75	100	100	100	100
Version IV	100	100	100	100	74	75	93	98	80	90
Version V	75	75	75	74	100	54	71	80	80	80
Version VI	75	75	75	75	54	100	75	75	75	75
Version VII	100	100	100	93	71	75	100	89	86	86
Version VIII	100	100	100	98	80	75	89	100	80	90
Version IX	100	100	100	80	80	75	86	80	100	97
Version X	100	100	100	90	80	75	86	90	97	100

Figure 2.16: *Similarity values between 10 different versions of the pushing action.*

and columns are present all the time in the individual SECs. This is also visible by comparing model (Fig. 2.17) with the single SEC in Fig. 2.14. For example, the third row and sixth column of the single SEC in Fig. 2.14 are detected as noisy data and thresholded in the model SEC in (Fig. 2.17) since they are not observed in other demonstrated versions. Thus, the model represents the archetype-SEC for this particular pushing action.

At this stage it is important to note that the start points of each temporal chunk, given by the time moments of the different columns in the model-SEC, represent *temporal anchor points* (Key Frames of the movie sequence). The SECs, thus, solve a difficult chunking problem in a natural way: By these anchor points the different motor primitives needed for a manipulation are already defined. Furthermore, these moments are also decisive for defining the spatial anchor points at the objects, needed to define actually execute an action.

2.10.3 Defining Spatial Anchor Points

The temporal anchors tell us "what happened when?", but they do not yet answer the question "how did it happen?". In the most general case, we would also have to know, (1) which objects are moved, (2) how the final spatial configuration (relative poses) of the objects looks like, and how the movement trajectories are shaped requiring (3) start- and endpoints and (4) trajectory shapes.

We will in the following section show that an analysis at the temporal anchor points, hence of the Key Movie Frames, suffices to extract components 1-3: 1) objects involved, 2) required poses, and 3) movement start and end-points, which define motion segments. Only when wanting information about (4) the complete movement trajectory we need to analyze also movie frames *between* the key frames.

To keep the algorithm general we assume that only one prime mover exists (usually the robot arm), bimanual manipulations need a different treatment. As the robot arm can only do "one thing at a time" we can in general state that for all possible manipulations the manipulation is started when the robot arm produces the first

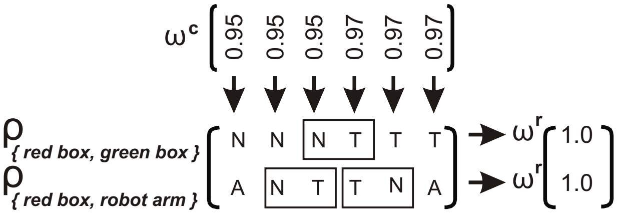


Figure 2.17: *The learned SEC model for the pushing action with corresponding normalized row* (ω_i^r) *and column* (ω_i^c) *weight values. Boxes indicate important transitions during this manipulation.*

N-T (non-touching to touching) transition in the SEC and that it ends when the arm produces a T-N transition. Thus, we first have to find the prime mover by analyzing the image segments. Furthermore, it is evident that all other existing N-T transitions are decisive for the manipulation. Hence, we need to analyze those – one after the other – too. Somewhat more unusual, we will also make use of the fact that *continuous contact* of prime mover with another object effectively makes the other object a secondary mover allowing us to use vector addition in task space for defining the complete motion path. For example, as long as the robot arm remains in contact with the red object (unchanging T relation), we can neglect the robot arm and just consider the newly resulting spatial relations of red object with other objects (here the green object) as spatial anchors[1]. Note, this reflects also the aspect of a robot using a tool. As long as the machine holds the tool, the robot's body is essentially extended and the tool defines now the end-effector of the machine Wörgötter et al. (2009).

2.10.4 Planning

In the planning phase we first analyze the 10 demonstration pushes to see how the movement segments actually looked like. By D1-D8 we denote in the following those constraints which are used to actually define the movement segments for execution. As we do not need poses, we did not implement any pose estimation steps.

To find the prime mover, we take the first N-T transition ($N_{2,2}-T_{2,3}$) in the model-SEC (Fig. 2.17). Here we use conventional row,column indices just for making it easier to find the entries in the SEC. Now we subtract the image segment configuration at key frame at $N_{2,2}$ from that at $T_{2,3}$, leading to a difference image only at the robot-arm image segment as the red box has not yet moved.

D1: Thus, we obtain as prime mover the "robot arm segment".

Then we consider the actual start points of the movement (red points in Fig. 2.18), which are widely distributed. The average is given by the starred red point.

D2: Hence there are essentially no constraints on the starting point S_1 of the complete sequence.

Next, we record at key frame at $T_{2,3}$ the coordinates of the red object at the touching point (see green points in Fig. 2.18).

D3: This defines the endpoint E_1 for this specific motion segment V_1.

---

[1] There might be some complicated manipulation actions where the arm (or hand) touches (or picks up) a second object before releasing the first. In this case, the argument about secondary mover would not hold. But such manipulations are uncommon and even for a human quite difficult. Hence, we do not consider them.

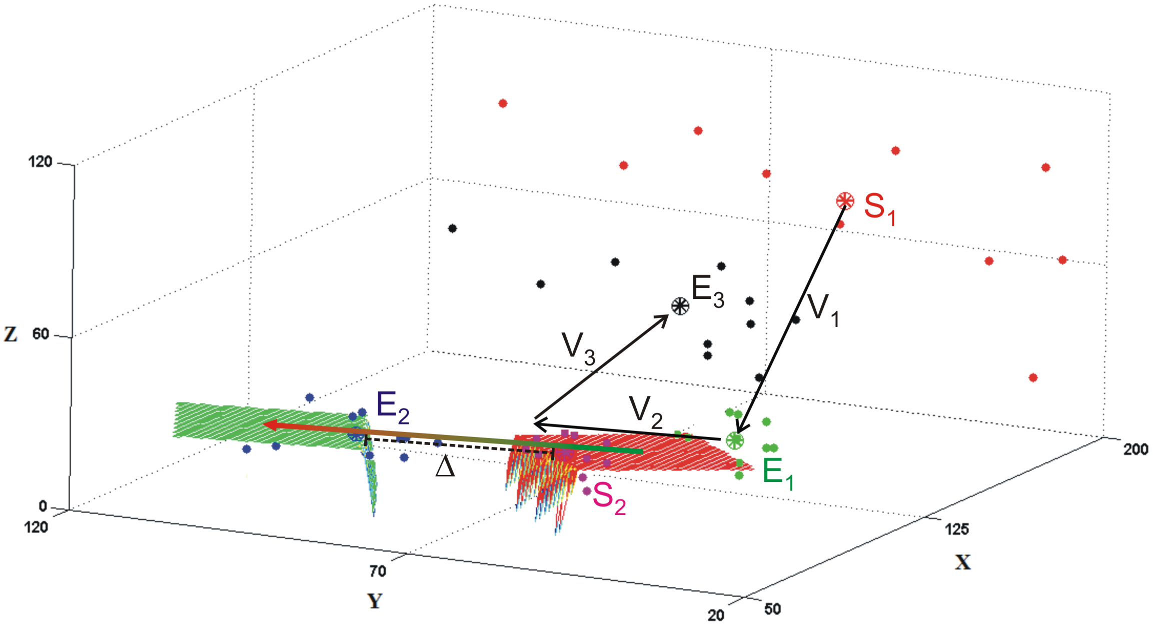


Figure 2.18: *Start (S_1, S_2) and end (E_1, E_2, E_3) point distribution as provided by the training dataset. Starred points are the average locations. Robot manipulator travels along the path given by vectors V_1, V_2, V_3. The coordinate origin is at the center of the red object. The motion vector for pushing is depicted by the color-changing vector that connects red with green object. Distance Δ is defines as $|E_2 - S_2|$ and vector V_2 has length Δ.*

We need to make sure that execution can cope with all kinds of different spatial configurations of robot arm and object. This requires defining a coordinate system which allows for such a generalization. To this end we use as the coordinate origin the segment center of the first touched object. This definition holds true for all conceivable basic single- or dual-object manipulation actions as *relations* between objects are decisive for the manipulation(s). Hence, we can always fix the origin on the first one touched and define coordinates relative to this[2], where we use any generic cartesian coordinate system just keeping it fixed for the remainder of the process.

D4: Thus, the center of the first touched object defines the coordinate origin.

The second found N-T transition concerns the red and the green object ($N_{1,3} - T_{1,4}$). As there is no change between the relation of robot arm and red object (relation

---

[2] Most basic, uni-manual manipulations are performed either at one object, leading to some configuration change at the object, or at two objects, where the first touched object is combined with the second one. Other manipulations, where more objects are directly involved are very rare (e.g. grasping two objects keeping both in the hand and combining them with a third one) or they can be considered as a chain of single- or dual-object manipulations.

remains $T_{2,4}$) we have indeed found a "secondary" mover (the red box) and a second touched object (the green box).

The segment center of the green object (the second touched object) defines together with the coordinate origin (center of red object) the so-called *dual object connection vector* (short: connection vector). Also this definition holds for all dual-object manipulations where a first object is supposed to make contact with a second object. In all these cases the first object must travel along a path (vector) that connects it to the second one. Clearly, in many dual-object manipulations additional difficult pose-constraints may arise, but the general connection vector will remain the same.

D5: Thus, the connection vector is spanned between the centers of the first and second touched object. Movement segment V_2 should follow the direction of this vector.

Now we need to define the path length. So far, the definitions do not require any prior knowledge about the actual action to be performed. They hold for pushing as well as, for example, for pick-and-place actions. The fact that we want to perform a pushing action only comes in now: Similar to above, we record for key frame at $T_{1,4}$ the coordinates of the red *and* green objects at their touching point. They are shown back-projected onto the start frame in Fig. 2.18 (pink and blue). One can see that for a push, start and endpoints E_1, S_2, E_2 of the motion segments are roughly aligned with the connection vector (see Fig. 2.19 A and Fig. 2.18).

D6: *Points E_1, S_2, E_2 can be computed from the 3D-coordinates representing the intersection of the connection vector with the edges of the objects (Fig. 2.19 B).*

D7: From this, we also note that the distance $\Delta = |E_2 - S_2|$ defines the length of the second motion segment V_2. It's direction is given by the connection vector.

The core of the manipulation ends at the $T_{2,4} - N_{2,5}$ transition of robot arm with red object. The final homing motion of the robot arm, which follows thereafter, is not relevant for the manipulation and can be performed in any possible way. We

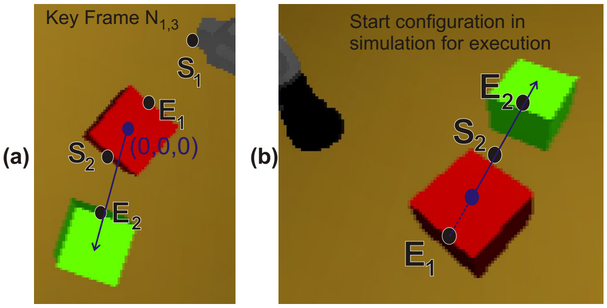


Figure 2.19: *Start and end points as given by demonstration (A), and as calculated for execution (B).*

look at the now following $N_{2,5} - A_{2,6}$ transition at the prime mover and plot the end points E_3 from $A_{2,6}$ (black points in Fig. 2.18) producing a set of actually observed final endpoints of the robot arm, which are also widely distributed.

D8: Any endpoint for the motion can be used as long as the robot arm withdraws from the red object in a collision-free way.

2.10.5 Execution

Once the planning phase is realized, the robot starts the actual execution process. For this, the model-SEC is used and every transition is treated like a rule. Constraints D1-D8 are attached to the transition rules as defined above.

For example the first N-T transition corresponds to a rule that demands that some movement by the prime mover should take place such that at the end the robot arm touches the red box and so on.

Thus, a new visual scene is presented to the system and segmented as usual. Robot arm, red box, and green box are recognized by their color or by any other object recognition algorithm. The model-SEC is split into its rules and the *actual* movement sequence is prepared by calculating the movement segments relative to the target objects. Start point S_1 is given by the momentary location of the robot arm. Again we use as coordinate origin the center of the red object and the connection vector points to the center of the green object. This origin- and vector-definition holds for all dual-object manipulations. For the specific purpose of pushing, and as explained above, the respective start and endpoints E_1, S_2, E_2 are now computed from the 3D-coordinates representing the cross-section of the connection vector with the edges of the objects (Fig. 2.19 B). The movement amplitude for the second N-T transition is in the same way given by $\Delta = |E_2 - S_2|$. We note that the touching point of the second object can be a bit over-estimated by this procedure if the diagonal of the object is aligned with the connection vector. In this case we will get a bit of a "push-second-object-away" when executing the action. This shows that pose estimation will at some point have to be added, too. Movement from S_1 to E_1 is defined by any collision-free trajectory all other motion segments are straight. The last motion segment is a homing movement to any desired endpoint (E_3).

Without having to explain the details, execution now proceeds by following the N-T or T-N transitions from the model-SEC using conventional inverse kinematics for the Katana arm by vector addition of all motion segments until the sequence of motion segments has been consumed.

It is important to note the the robot has now immediately also a means to check whether the outcome of its actions are correct. After each movement of the arm, the machine needs to check the resulting relational changes between the image segments. These should match the changes in the model-SEC (see Chapter 4 for an example).

2.10.6 Simulation Results

We let the agent realize the pushing action considering the learned model-SEC, and the motion segments as defined above. Fig. 2.20 (a-c) show how the *robot arm* pushes the *red box* to the *green box* even if the object locations are different. In Fig. 2.20 (d-e) we used a red sphere and a bigger red box as pushable objects. In such cases the *robot arm* can still execute the manipulation. In Fig. 2.20 (f) a red cone and one more blue sphere were used. Finally, the robot arm chose and pushed the red conic to the bigger green box. All those simulation results, with corresponding depth, segment, and graph representations, can be found at `http://www.dpi.physik.uni-goettingen.de/~eaksoye/movies.html` (See Appendix A.3). Finally we performed a self-check and Fig. 2.21 shows the SEC obtained from the last execution example in Fig. 2.20, which is very similar to the model-SEC by which the robot can accept its own execution as correct. This is true for all executed examples.

These results show that with such a semantic representation the agent can learn and imitate a pushing manipulation independent of object shapes and positions.

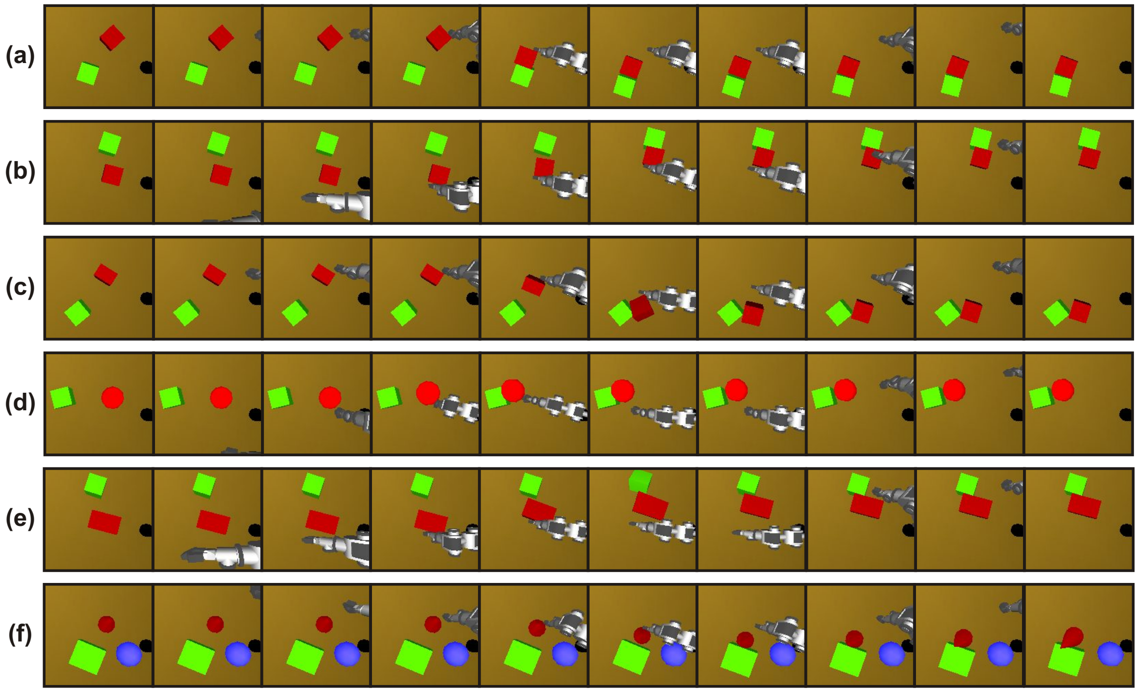


Figure 2.20: *Execution results. (a-c) Robot arm pushes the red box to the green box even if the object locations are different. (d-e) A red sphere and a bigger red box are used as pushable objects. (f) A red cone is used as a pushable object and one more blue sphere is added in the scene.*

$$\begin{array}{l}\rho_{\{\textit{red box, green box}\}} \\ \rho_{\{\textit{red box, robot arm}\}} \\ \rho_{\{\textit{robot arm, table}\}}\end{array}\left(\begin{array}{ccccccc} N & N & N & T & T & T & T \\ A & N & T & T & N & N & A \\ A & N & N & N & N & T & A \end{array}\right)$$

Figure 2.21: *SEC obtained from execution of the last example in Fig. 2.20.*

2.11 Discussion

In this chapter, we have introduced a novel representation for manipulations, called the semantic event chain, which focuses on the relations between objects (including hands) in a scene. The representation generates column vectors in a matrix where every transition between neighboring vectors can be interpreted as an action rule (key frame), which defines which object relations have changed in the scene. Hence, event chains reach a rather high level of abstraction, but on the other hand, they remain tightly linked to the images from which they originate, because they rely on continuously tracked segments. We have devised simple algorithms based on sub-string comparisons and counting procedures by which event chains can be compared and actions as well as segments can be classified in an unsupervised way. No prior object models are required in this approach and learning of archetypal event chains (model-SECs) from demonstrations relies only on weight upgrade of consistently repeating rows (spatial relations) and columns (repeating rules). Apart from the demonstration no other supervision is needed in this step, hence SECs are learned in a model-free way. The learned SECs are further enriched by determining the movement segments by which the manipulation can be executed regardless of the configuration of the objects in a scene. Execution phase can then follow the enriched SEC rules and the robot can test its own success by checking the SEC, which results from execution, against the model-SEC.

The work presented in this chapter is, to our knowledge, the first approach which reaches an abstract symbolic representation for manipulation recognition, learning, and execution while being fully grounded in the signal domain. One of the advantages comes with SECs is that each column of a SEC can be interpreted as a state that defines motor primitives. Therefore, we can conclude that SECs do not include any hidden state which is the most important difference from other conventional probabilistic methods such as hidden Markov models.

We also emphasize that no object recognition was used in our approach. Graph nodes, in other words image segments, are invariant to object type and appearance. SECs relate the objects only with performed actions. As a consequence of that, objects can be categorized based on their roles in actions. For instance, the agent can relate a pen with both writing and piercing actions and thus can further use the pen not only to write something but also to make a hole in a paper due to its sharp tip. This is an important achievement in the sense of object affordances first defined by Gibson (1977).

The procedure of encoding only the relational changes without depending on the segment shape makes SECs invariant to perspective changes as long as the visual entities (i.e. image segments) are visible. The confusion matrix given in Fig. 2.16 supports this claim since it shows high similarity values between perspectively different versions of the same pushing action. It is also important to note that such visual entities can be detected by any kind of continuous tracking system. This makes SECs compatible also with other front-ends.

We are aware of the fact that the algorithm uses some simplifications and thus has some drawbacks such as our approach highly depends on conventional computer vision techniques. Therefore, any failure in the tracking procedure would harm our approach. Other than that, objects can be over-segmented which causes many subgraphs that have to be analyzed separately. However, this cannot be solved in the low level signal domain since we have no object model-based assumption that can be used as ground truth for segmentation. An extra high level reasoning method needs to be used.

Moreover, during the execution phase, no object dynamics or pose estimations are considered. Due to this fact, in some cases it was observed that the object could not be pushed in the desired direction, because of wrong object and/or gripper poses and the frictional restrictions both on the background and object surface. However, such additional required information can also be attached to SECs during demonstrations.

All this notwithstanding, our proposed execution procedure can be macronized for all *dual-object manipulations*. In a very abbreviated form, the instructions for such a macro would read:

1. Identify prime mover.
2. Identify first touched object by first N-T transition and set coordinate origin.
3. Define first motion segment.
4. (Extract relative poses between prime mover and first object, if needed).
5. Identify second touched object and fix connection vector and coordinate system.
6. Define second motion segment for second N-T transition relative to this coordinate system. (For pushing do this by cross-sectioning with object borders).
7. (Extract relative poses between objects involved, if needed).
8. Define third motion segment (home).

Such a macro can be enriched by adding pose information from a pose estimation algorithm where required. This would be needed for a pick&place manipulation (which is also a dual object manipulation), where the final resulting relative pose of the two combined objects is most of the time important. Aspects of grasping an object (e.g. grasp preparation and the performing of a grasp) are not considered at all in this framework. Grasping is a very difficult technical problem but for manipulation actions it takes usually just a preparatory role. We do not wish to downgrade the importance of this role but the actual outcome of the manipulation is in most cases only in a secondary way affected by the way an object is grasped. Clearly, if the grasp is totally unsuitable, a pick&place action will fail. But these considerations must be taken into account before the first N-T transition in the SEC and are not part of this paper.

3

Statistics on Semantic Event Chains

In real experiments we observed that SECs can contain not only noisy indexes but also extra noisy rows and/or columns due to noisy segmentation. *Case study I and II* given in chapter 4 provide some examples for real noisy SECs. Therefore, the algorithms used for analyzing SECs have to be robust against noise. In this chapter, we discuss some statistical results on the robustness of such algorithms. The chapter contains two sections: First, we test how stable the proposed algorithms used for action classification, object categorization, and learning of an archetypal SEC model are, and then, we compare the robustness of the action classification algorithm with neural networks in the second section. For this purpose, we create a seed SEC with a certain size, which is then altered by adding noisy rows and columns and/or by changing the indexes with a noisy one. By increasing the noise level in the seed, the stability of the proposed algorithms are examined for different SEC sizes.

3.1 Analysis of the Similarity Measure

In chapter 2, we have shown that the similarity measure leads to action classification, object categorization, and also learning of a SEC model (see step 5 in Fig. 2.2). Therefore, the step of measuring the similarity plays the most crucial role in the whole framework. In the first section of this chapter, we address the question of how the similarity measure behaves when the degree of noise in SECs increases. Furthermore, we analyze the effects of such behaviors on the action classification, object categorization, and learning algorithms, separately.

To produce more data for statistics, we first create a seed SEC that encodes a manipulation. Fig. 3.1 shows a sample seed SEC that holds spatial relations between a hand, a table, and a box. For each element of the seed, we define a probability value (p) which represents how likely the seed element will be changed to a dissimilar one in order to introduce noise. The probability entries for the sample seed are shown in Fig. 3.1. Such probability values are also defined for each row and column to introduce additional noisy rows and columns as observed in real scenarios. Note that we let the system add maximally one noisy row/column between each existing row/column.

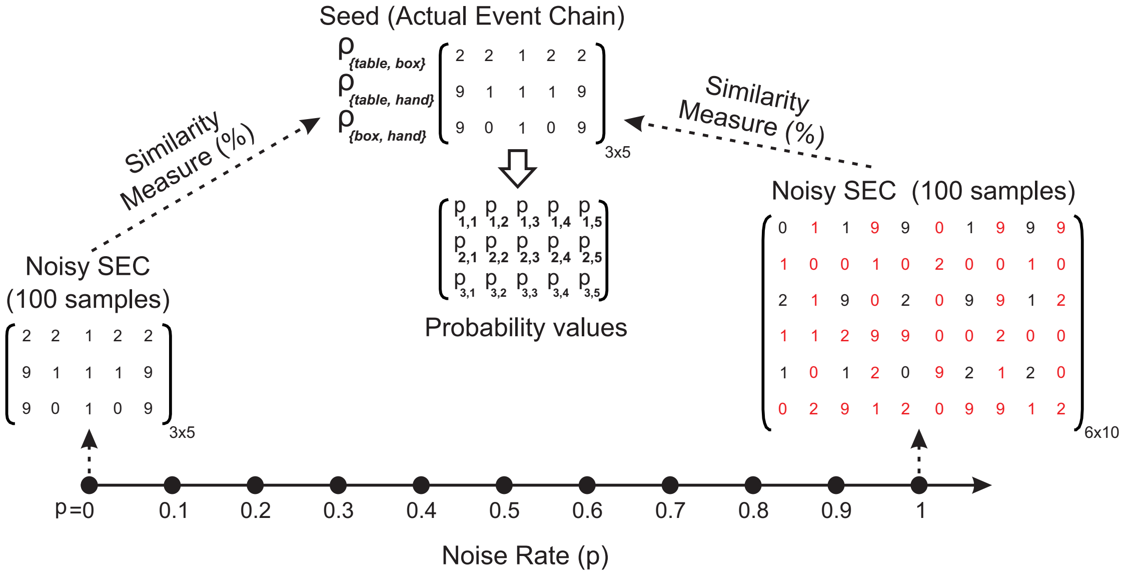


Figure 3.1: *Producing noisy data for statistical analysis. For each element of the seed a probability value (p) is defined. Such probability values are also used for introducing additional noisy rows and columns. The p value is then varied from* 0 *to* 1 *with the step of* 0.1*. At each noise level* 100 *SEC samples are produced, which are then compared with the seed.*

The p value is then varied from 0 to 1 with a step of 0.1. Fig. 3.1 depicts how the noisy SECs look like at different noise levels. As expected, when p equals to 0, the noisy SEC and the seed are identical. However, at the highest noise level ($p = 1$) all elements of the seed are flipped and new noisy rows and columns (shown in red) are added. At each noise level, 100 SEC samples are produced, each of which is then compared with the seed by using the method given in step 5 in chapter 2.

In Fig. 3.2, the red curve shows the mean values with standard error means of all 100 similarity measures, each between the seed and one noisy sample, for the case when we both flip the seed indexes and add noisy rows and columns to the seed given in Fig. 3.1. It is obvious that the slope of the curve is changing dramatically around $p = 0.5$ after which the similarity measure swings around 30%. The blue curve in Fig. 3.2 indicates the mean similarity values for the case when we add only noisy rows and columns without flipping the original SEC indexes. In this case, the results are much more promising as the mean similarity value is still around 70% even at noise level 0.8. Such high similarity values can be observed when p is only 0.2 in the red curve. Those curves prove that the noisy rows and columns do not affect the similarity algorithm significantly as long as the original SEC indexes remain the same. Once the SEC indexes are flipped the similarity measure drops dramatically. This is an important feature showing the importance of the original SEC indexes for

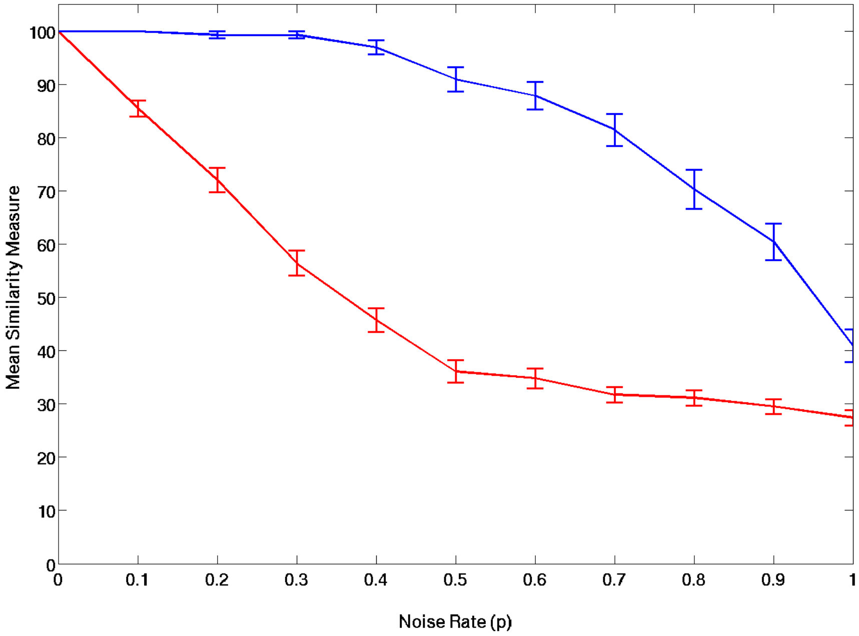


Figure 3.2: *Mean values of all* 100 *similarity measures, each for one sample, at different noise levels. The red curve is for the cases when we both flip the seed indexes and add noisy rows and columns to the seed given in Fig. 3.1. The blue one is for the case when we add only noisy rows and columns to the same seed. The vertical bars show the standard error mean.*

the similarity measurement algorithm.

It is important to note that the similarity measure never drops to 0 in both cases. The reason is that, as the main algorithm given in step 5 in chapter 2 is searching for the correspondences between rows of two SECs, because the rows of two type-similar SECs can be shuffled, some incorrect matchings with low similarity values can incidentally be calculated.

Fig. 3.2 illustrates behaviors of the similarity measures of a 3×5 seed (see Fig. 3.1) for two different noisy cases. Now, we would like to analyze the effects of such behaviors on the action classification, object categorization, and learning issues in detail. For this purpose, we created four different seeds with different sizes: 4×6, 5×7, 6×8, and 8×8. Fig. 3.3 shows all those SEC seeds to get an impression of the level of difference. For each seed, we produced 100 noisy samples at different noisy levels by following the method illustrated in Fig. 3.1.

3.1.1 Effects on Action Classification

In step $6A$ in chapter 2, we have explained how the similarity measures between SECs can be used for classifying actions. Fig. 2.7 in chapter 2 depicted that the minimum

Seed I

$$\begin{array}{c} \rho_{1,2} \\ \rho_{2,3} \\ \rho_{2,4} \\ \rho_{3,4} \end{array} \left(\begin{array}{cccccc} 2 & 2 & 1 & 1 & 2 & 2 \\ 9 & 0 & 1 & 2 & 2 & 2 \\ 9 & 0 & 1 & 1 & 0 & 9 \\ 9 & 1 & 1 & 0 & 9 & 9 \end{array}\right)_{4x6}$$

Seed II

$$\begin{array}{c} \rho_{1,2} \\ \rho_{1,3} \\ \rho_{1,5} \\ \rho_{2,3} \\ \rho_{2,5} \end{array} \left(\begin{array}{ccccccc} 2 & 2 & 2 & 1 & 1 & 2 & 2 \\ 9 & 1 & 1 & 0 & 0 & 0 & 0 \\ 9 & 9 & 1 & 1 & 1 & 1 & 9 \\ 9 & 0 & 0 & 1 & 2 & 2 & 2 \\ 9 & 9 & 0 & 1 & 1 & 0 & 9 \end{array}\right)_{5x7}$$

Seed III

$$\begin{array}{c} \rho_{1,2} \\ \rho_{1,3} \\ \rho_{1,5} \\ \rho_{2,3} \\ \rho_{2,5} \\ \rho_{3,5} \end{array} \left(\begin{array}{cccccccc} 2 & 2 & 2 & 1 & 1 & 2 & 2 & 2 \\ 9 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 9 & 9 & 1 & 1 & 1 & 1 & 1 & 9 \\ 9 & 0 & 0 & 1 & 2 & 2 & 2 & 2 \\ 9 & 9 & 0 & 1 & 1 & 1 & 0 & 9 \\ 9 & 9 & 1 & 0 & 0 & 0 & 0 & 9 \end{array}\right)_{6x8}$$

Seed IV

$$\begin{array}{c} \rho_{1,2} \\ \rho_{1,3} \\ \rho_{1,4} \\ \rho_{2,3} \\ \rho_{2,4} \\ \rho_{3,4} \\ \rho_{2,5} \\ \rho_{3,5} \end{array} \left(\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 9 & 1 & 1 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 2 & 1 & 1 & 2 & 2 \\ 9 & 2 & 1 & 0 & 0 & 1 & 2 & 9 \\ 9 & 1 & 1 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 1 & 9 & 9 \\ 9 & 0 & 2 & 2 & 1 & 1 & 0 & 9 \end{array}\right)_{8x8}$$

Figure 3.3: *Four different seeds with different sizes:* 4×6*,* 5×7*,* 6×8*, and* 8×8*. For each seed* 100 *noisy samples are created at each noisy level by following the method illustrated in Fig. 3.1.*

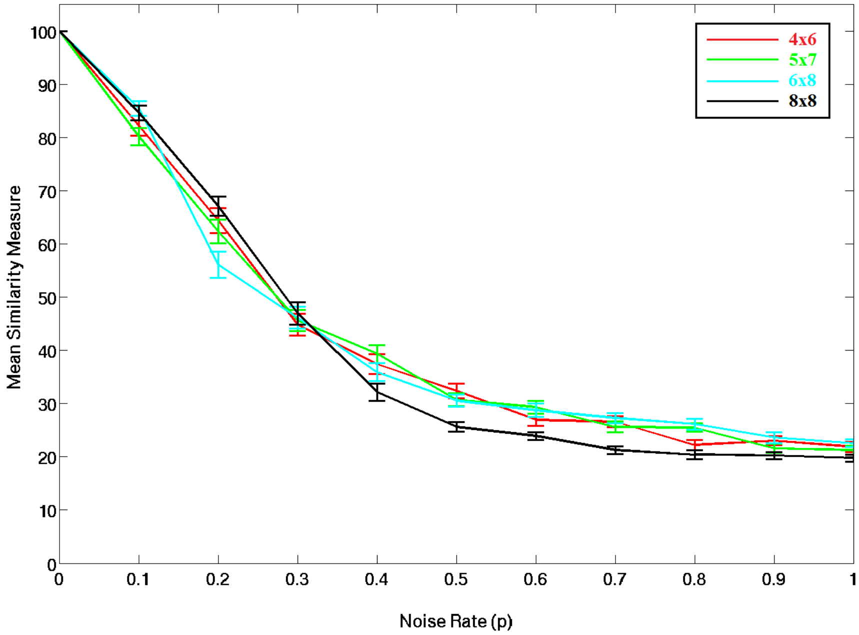
similarity value for type-similar actions was 64% for all four real action scenarios: *Moving Object*, *Making Sandwich*, *Filling Liquid*, and *Opening Book*. Therefore, we concluded that setting a threshold at 64% would be enough to distinguish action classes. Considering this result, for further statistical analysis we can make an assumption claims that similarity between type-similar actions should be above 64% for a correct classification.

Fig. 3.4(a) illustrates the mean similarity values with standard error means between the four seeds defined in Fig. 3.3 and their noisy samples for the case when we both flip the seed indexes and add more rows and columns to the seeds. The first impression the figure conveys is that the similarity measure is invariant to SEC size, since all four curves are exhibiting similar behaviors. This figure also demonstrates that, according to the assumption above, classification above a noise rate of 0.2 can not be achieved successfully due to low similarity values.

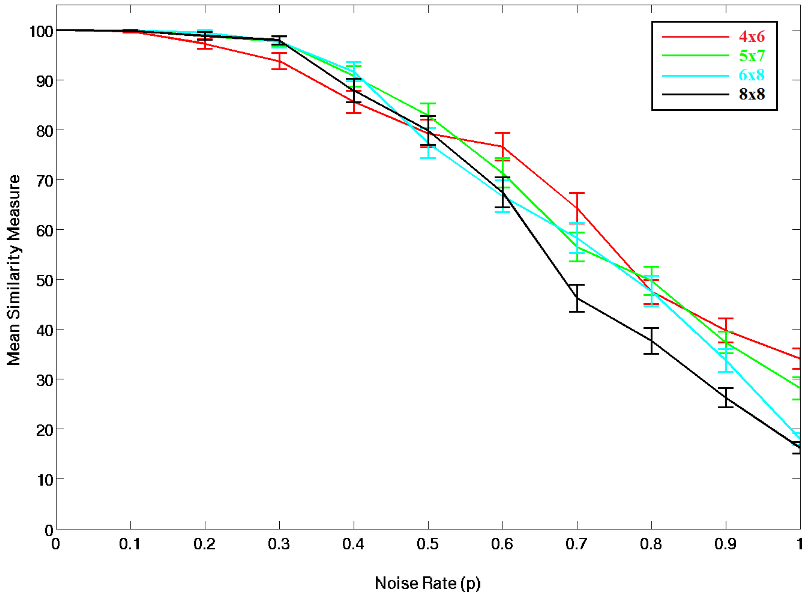
Fig. 3.4(b) indicates the mean similarity values between the same four seeds and their noisy samples, but for the case when we add only noisy rows and columns without flipping the original seed indexes. In such a case, classification is still applicable around noise rate 0.6, which is much better than the previous case. One reason of such high difference is that any change in the original SEC elements is interpreted as being a different action representation, thus, compared to the size the original SEC elements are more crucial in the process of similarity measurement. Another reason is that noisy rows are eliminated once the correspondences between the shuffled rows are calculated.

3.1.2 Effects on Object Categorization

In chapter 2, we have also discussed that in conjunction with action classification the manipulated objects can be categorized based on their performed roles. Fig. 2.8 in chapter 2 illustrated how such a categorization looks like for four real scenarios:

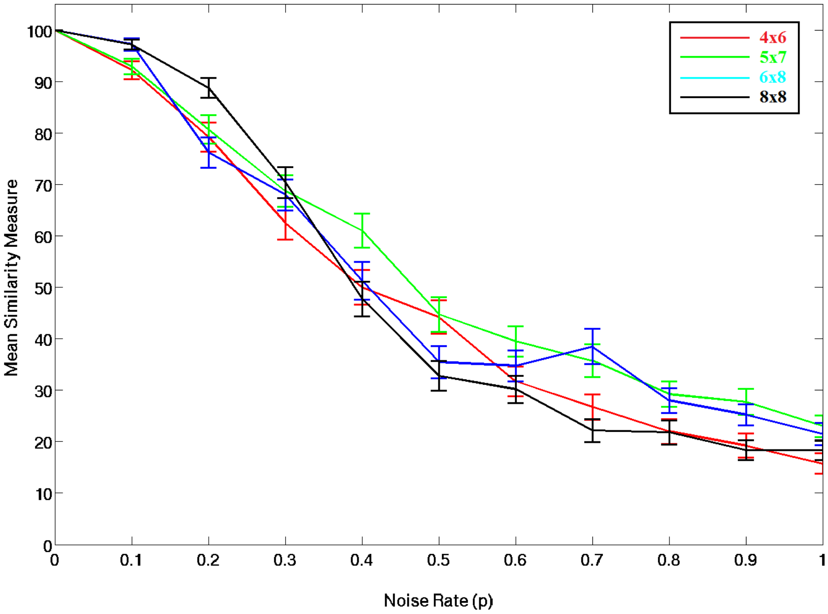


(a) *For the case when we both change the original seed indexes and add noisy rows and columns to the seeds.*

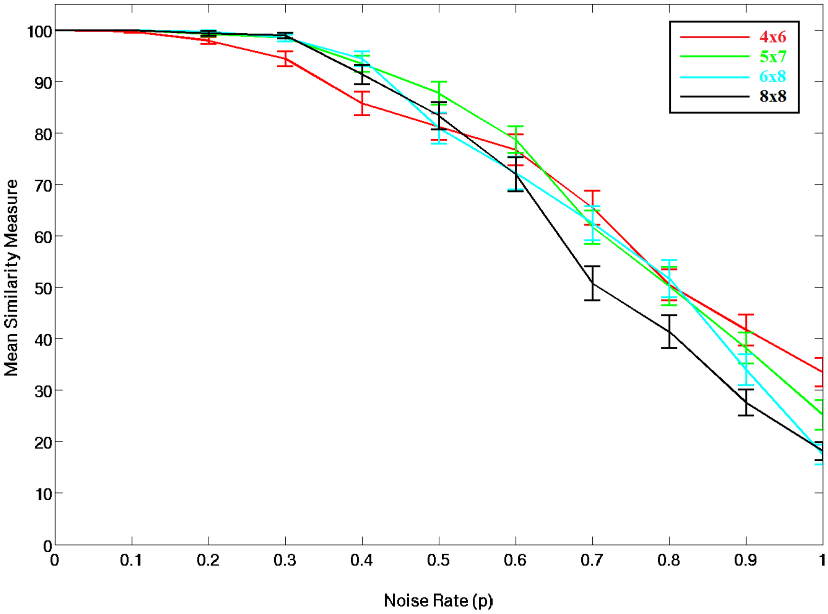


(b) *For the case when we add only noisy rows and columns without changing the original seed indexes.*

Figure 3.4: *Similarity behavior of four SEC seeds at different noise rates. The vertical bars show the standard error mean.*



(a) *For the case when we both change the original seed indexes and add noisy rows and columns to the seeds.*



(b) *For the case when we add only noisy rows and columns without changing the original seed indexes.*

Figure 3.5: *The mean value of the matched object numbers between the four SEC seeds and their noisy samples. The vertical bars show the standard error mean.*

Moving Object, *Making Sandwich*, *Filling Liquid*, and *Opening Book*.

Now, we would like to analyze how the object categorization results are changing when we compare the noisy event chains with the original seeds. Fig. 3.5(a) depicts the mean value of the matched object numbers between the seeds given in Fig. 3.3 and their noisy samples for the case when we both flip the seed indexes and add noisy rows and columns to the seeds. For example, at noise level 0.2 approximately 80% of the manipulated objects are correctly categorized, which means that one out of every 5 objects mismatched. This mismatching rate becomes more important when we note that seeds in Fig. 3.3 have maximally 5 objects. This result actually supports the assumption made for action classification above, such that once the similarity result between actions drops to less than 64% (as observed at $p = 0.2$ in Fig. 3.4(a)) actions can not be distinguished correctly as the manipulated objects are also starting to be mismatched at the same noise rate ($p = 0.2$). Consequently, the noise rate 0.2 becomes a critical point for both action classification and object categorization. Moreover, this figure proves that sizes of seeds are not effective on object categorization as well, since all curves in the figure exhibit similar behaviors.

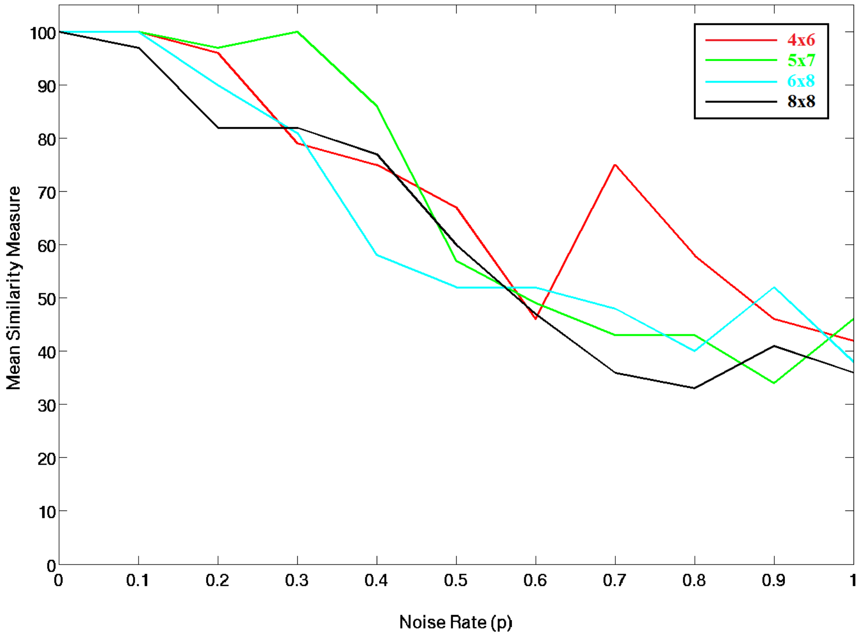
Fig. 3.5(b) shows mean values of object matching results for the case when we add only noisy rows and columns without flipping the original seed indexes. Here, the results are better as observed in the action classification case, because the same object matching rate (80%) occurs around the noise rate 0.6.

3.1.3 Effects on Learning

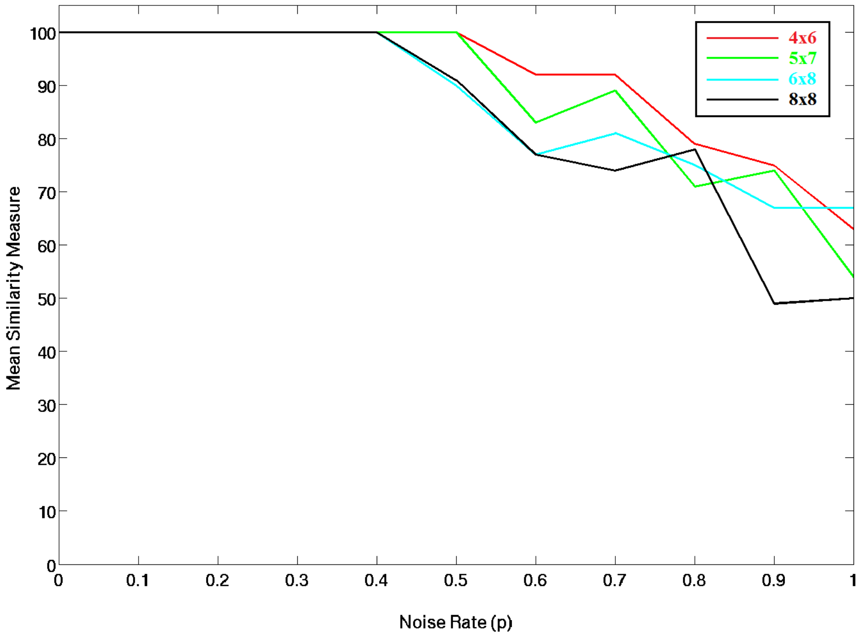
Another claim given in chapter 2 is that an archetypal SEC model can be learned by considering repetitive rows and columns observed in the training data (see step 7 in chapter 2). It is now important to analyze how stable the learning algorithm is while the noise rate is increasing. For this purpose, we used all 100 noisy event chain samples, produced for each seed given in Fig. 3.3, as a training data set to render a model SEC. Since each seed is the ground truth of its own model, we then compared the learned model with its respective seed to calculate the deviation.

Fig. 3.6(a) shows similarity results between learned models and seeds for the noisy case when we both flip the seed indexes and add more rows and columns to the seeds. Due to being ground truth, 100% similarity is expected between seeds and models. However, such high similarity measures can not be observed as shown in Fig. 3.6(a). In the case of adding only noisy rows and columns without flipping the original seed indexes, it is observed that the models can be correctly learned until the noise rate 0.4 as depicted in Fig. 3.6(b). As those two figures emphasize, the learning method is more affected by flipped SEC indexes than by adding only noisy rows and columns.

In Fig. 3.7(a-b), three SEC models with corresponding row and column weight values, learned at different noise levels ($p = 0.1$, $p = 0.2$, and $p = 0.4$), are shown for two noisy cases. In the first case (see Fig. 3.7(a)), as the weight values are dramatically decreasing some incorrect rows and columns are included, whereas in the second case



(a) *For the case when we both change the original seed indexes and add noisy rows and columns to the seeds.*



(b) *For the case when we add only noisy rows and columns without changing the original seed indexes.*

Figure 3.6: *Similarity measures between learned SEC models and their original seeds.*

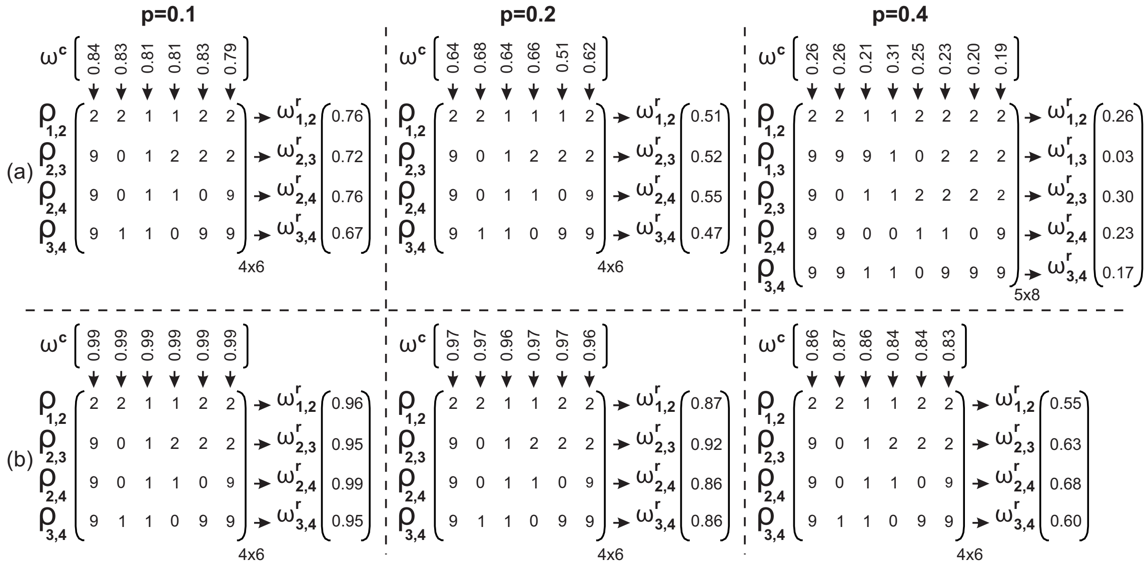

p=0.1
p=0.2
p=0.4
(a)
ω^c [0.84 0.83 0.81 0.81 0.83 0.79]
ρ1,2 [2 2 1 1 2 2] → ω^r 1,2 0.76
ρ2,3 [9 0 1 2 2 2] → ω^r 2,3 0.72
ρ2,4 [9 0 1 1 0 9] → ω^r 2,4 0.76
ρ3,4 [9 1 1 0 9 9] → ω^r 3,4 0.67
4x6
ω^c [0.64 0.68 0.64 0.66 0.51 0.62]
ρ1,2 [2 2 1 1 1 2] → ω^r 1,2 0.51
ρ2,3 [9 0 1 2 2 2] → ω^r 2,3 0.52
ρ2,4 [9 0 1 1 0 9] → ω^r 2,4 0.55
ρ3,4 [9 1 1 0 9 9] → ω^r 3,4 0.47
4x6
ω^c [0.26 0.26 0.21 0.31 0.25 0.23 0.20 0.19]
ρ1,2 [2 2 1 1 2 2 2 2] → ω^r 1,2 0.26
ρ1,3 [9 9 9 1 0 2 2 2] → ω^r 1,3 0.03
ρ2,3 [9 0 1 1 2 2 2 2] → ω^r 2,3 0.30
ρ2,4 [9 9 0 0 1 1 0 9] → ω^r 2,4 0.23
ρ3,4 [9 9 1 1 0 9 9 9] → ω^r 3,4 0.17
5x8
(b)
ω^c [0.99 0.99 0.99 0.99 0.99 0.99]
ρ1,2 [2 2 1 1 2 2] → ω^r 1,2 0.96
ρ2,3 [9 0 1 2 2 2] → ω^r 2,3 0.95
ρ2,4 [9 0 1 1 0 9] → ω^r 2,4 0.99
ρ3,4 [9 1 1 0 9 9] → ω^r 3,4 0.95
4x6
ω^c [0.97 0.97 0.96 0.97 0.97 0.96]
ρ1,2 [2 2 1 1 2 2] → ω^r 1,2 0.87
ρ2,3 [9 0 1 2 2 2] → ω^r 2,3 0.92
ρ2,4 [9 0 1 1 0 9] → ω^r 2,4 0.86
ρ3,4 [9 1 1 0 9 9] → ω^r 3,4 0.86
4x6
ω^c [0.86 0.87 0.86 0.84 0.84 0.83]
ρ1,2 [2 2 1 1 2 2] → ω^r 1,2 0.55
ρ2,3 [9 0 1 2 2 2] → ω^r 2,3 0.63
ρ2,4 [9 0 1 1 0 9] → ω^r 2,4 0.68
ρ3,4 [9 1 1 0 9 9] → ω^r 3,4 0.60
4x6

Figure 3.7: *Learned SEC models with corresponding row and column weight values at three different noise levels:* $p = 0.1$, $p = 0.2$, *and* $p = 0.4$. *(a) For the case when we both flip the seed indexes and add noisy rows and columns to the seeds. (b) In the case of adding only noisy rows and columns without changing the original seed indexes.*

(see Fig. 3.7(b)) only weight values are slightly decreasing.

3.2 Comparison of Classification Algorithms

In the second section of this chapter, our intent is to focus on the action classification algorithm and compare it with feed-forward backpropagation neural networks. We start with a brief summary of neural networks and afterwards compare the classification approaches.

3.2.1 Feed-Forward Backpropagation Neural Networks

A neural network is a combination of many artificial neurons each of which is a mathematical model of a biological neuron. Neural networks generally have multiple layers: input, hidden, and output layers. Dashed box in Fig. 3.8 shows how such a multi-layer network looks like. Neurons in each layer are basically connected with the one in the subsequent layer by a specific weight value which is used to calculate the output of the network. In feed-forward networks, information is essentially flowing forward from the input layer to the output layer without forming any loop. Such networks use backpropagation (Rojas, 1996) as a supervised learning technique in which some sets of inputs with desired outputs are fed to the network for training.

The backpropagation method first calculates an error value by comparing the network output with the desired output values, and then propagates the error back to the network for updating the weights during the training phase. The network is mainly trained with an iterative process until the error converges to a small value after which the network is assumed to be learned. Each network iteration with the entire training data set is called an epoch, and in each epoch the weights are adjusted with a variable, so-called the learning rate, which defines how the new acquired information will be combined with the previous one. Low learning rate, e.g. 0, would lead to ignoring the new information and, thus, network learns nothing. Higher learning rates let the network take the new acquired information into account. A comprehensive survey on feed-forward backpropagation neural networks can be found in Nelson and Illingworth (1991); Mitchell (1997).

After this brief introduction, we can now explain how to combine SECs with neural networks for the classification issue. We are, here, interested in the robustness of the action classification algorithm, and therefore, to have more test data we create four different 6 × 6 SEC seeds each representing one action type. As explained in section 3.1, for each seed we provide 100 noisy samples at different noise levels. However, in this case we do not add any extra noisy rows or columns, but rather flip the seed indexes. This is because the neural networks rely on a fixed-size feature vector as an input. For efficient classification we create a multi-layer neural network with two hidden layers, each with five neurons, and one output layer with four neurons, each corresponding to one class type. Fig. 3.8 illustrates the complete neural network and also mapping of a sample SEC matrix into a fixed-size feature vector.

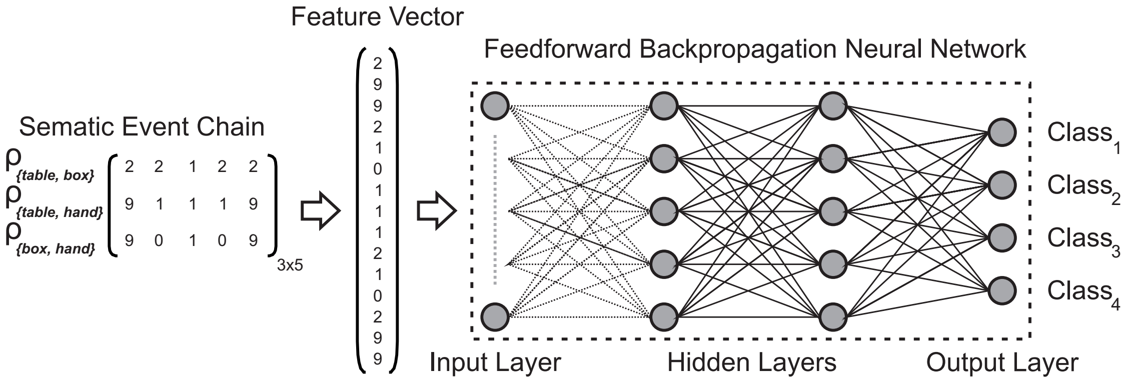


Figure 3.8: *Feed-forward backpropagation neural network with two hidden layers. The output layer has four neurons, each corresponding to one class type. SECs are converted to a form of a vector form which is fed to the network as an input.*

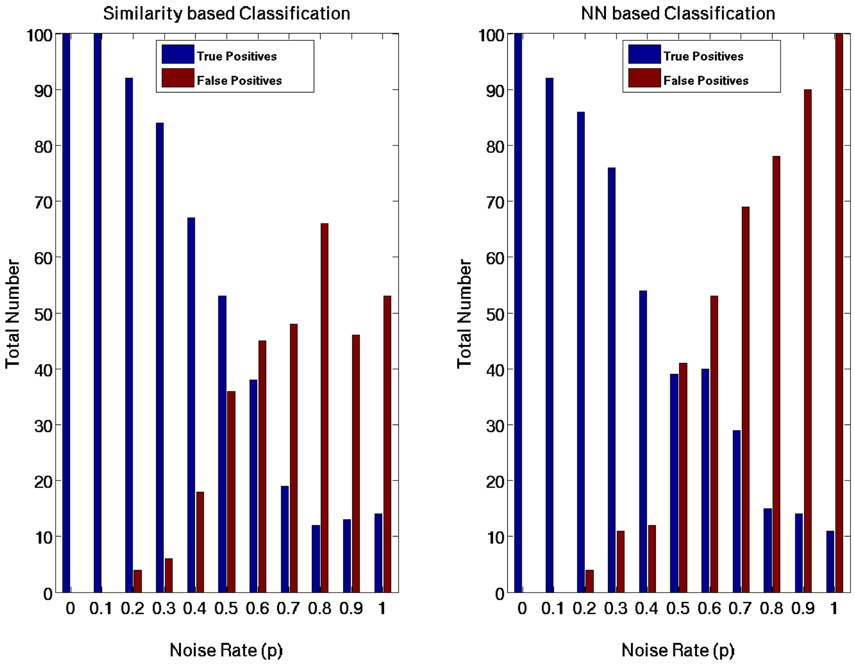
3.2.2 Classification Results

The reliability of the classification algorithm is examined with those four action seeds and with their noisy samples. The neural network depicted in Fig. 3.8 is trained with 1000 epochs at the learning rate 0.5 with 400 noisy samples (100 for each seed), created at the noise rate 0.1, by taking the desired output values into account. The testing phase of the network is then realized with new 400 noisy samples produced at different noise levels that vary from 0 to 1 with the step of 0.1. To correctly verify the network output, a coding technique is used to convert the output to a binary form by using the closest Euclidean distance. For instance, the network output with the values of $\{0.1, 0.2, 0.8, 0.5\}$ is encoded as $\{0, 0, 1, 0\}$.

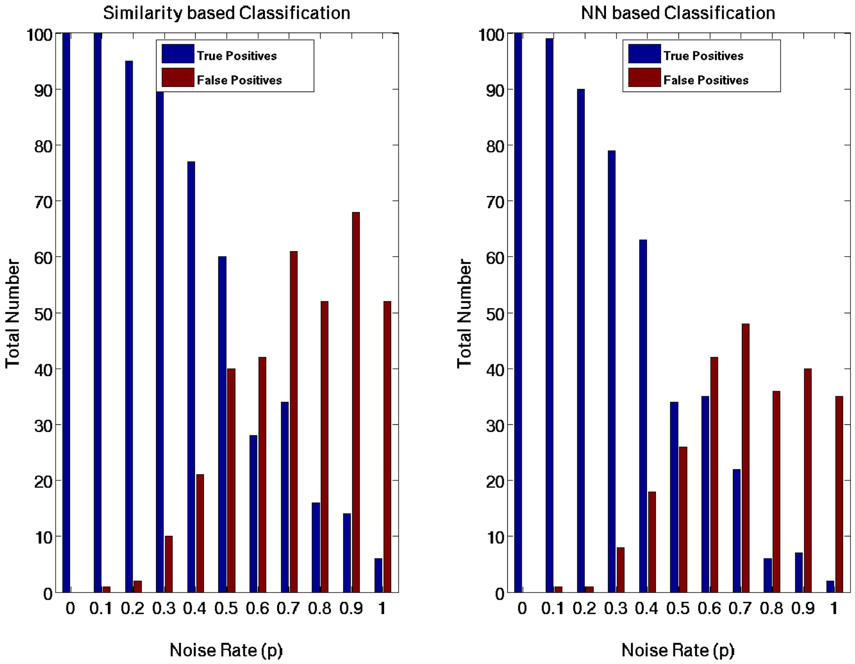
The same training and testing data are used to evaluate the similarity-based classification algorithm explained in step $6A$ in chapter 2. In the training phase, a SEC model for each seed is learned, which is then compared with the noisy test samples to measure the similarities at each noise rate. The same coding technique is also used here to convert the final similarity measures to a binary output form.

Fig. 3.9 (a-d) indicate the final comparison of the similarity-based and neural-network-based classification algorithms for all four action seeds. In blue and red bars are indicated the total numbers of true positive (TP) and false positive (FP) calculated at different noise rates. In binary classification, TP indicates the number of relevant samples correctly retrieved as positive and FP stands for the number of irrelevant samples wrongly retrieved as positive. As depicted on the left in Fig. 3.9 (a) for the first seed the number of TP is still above 50% at the noise rate 0.5 in similarity-based classification. However, in the case of neural-network-based classification, the number of TP is slightly less than the one in similarity-based classification, and even less than the number of its own FP at the same noise rate 0.5, as shown on the right in Fig. 3.9 (a). When we take a look at the other three action classes, we observe that the number of TP is always above 80% even at the noise rate 0.3, and the rate of TP versus FP is getting lower only after the noise rate 0.6 in the case of similarity-based classification (see left side in Fig. 3.9 (b-d)). In neural network results, such high number of TP (80%) is mostly not observed after the noise rate 0.2, and in some cases the number of FP exceeds the number of TP at the noise rate 0.5 (see right side in Fig. 3.9 (b-d)).

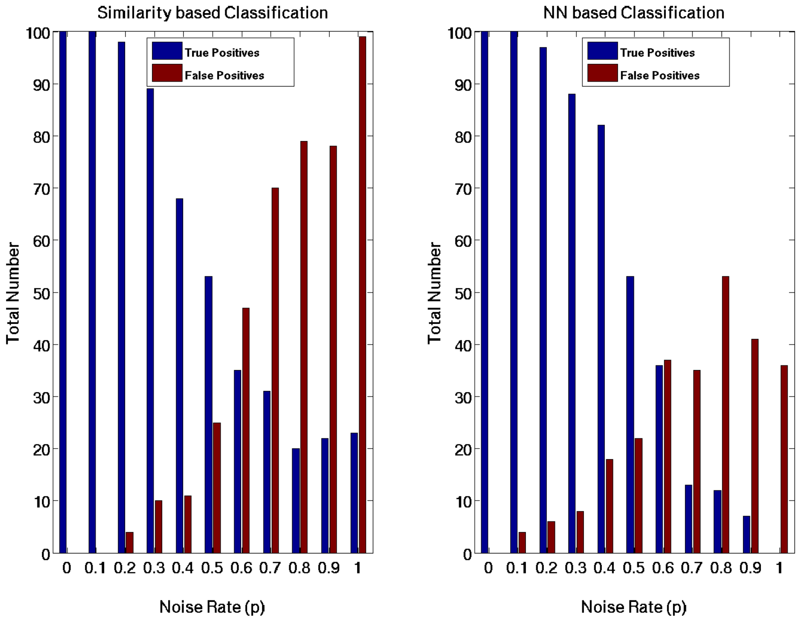
To correctly measure the performance of classification algorithms, we calculate the $Precision$ and $Recall$ values which are two popular metrics in machine learning. $Precision$ signifies the proportion of correctly predicted relevant samples (TP) to the total number of all retrieved samples. On the other hand, $recall$ indicates the rate of correctly retrieved relevant samples (TP) to the total number of all relevant items. $Precision$ versus $recall$ (PR) graphs finally represent retrieval effectiveness of classification algorithms. In the PR graphs, the main goal is to be at the upper right corner, such that the precision value is higher than the recall value. Further information on information retrieval and PR graphs can be found in Witten and



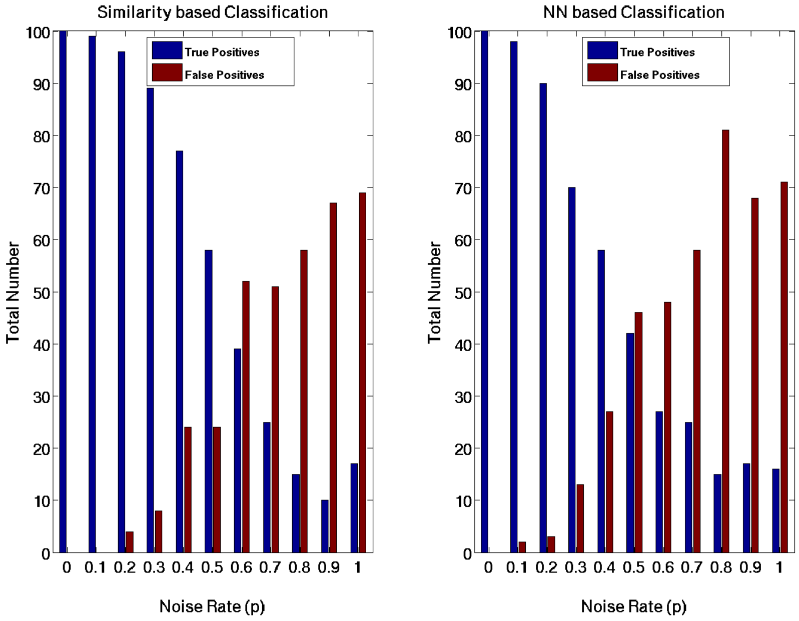
(a) *Classification results for the first SEC seed.*



(b) *Classification results for the second SEC seed.*



(c) *Classification results for the third SEC seed.*



(d) *Classification results for the fourth SEC seed.*

Figure 3.9: *Comparison of the similarity-based and neural-network-based classification algorithms for all fourth SEC seeds. In blue and red bars are indicated the total numbers of true positive (TP) and false positive (FP) calculated at different noise rates that vary from* 0 *to* 1 *with the step of* 0.1.

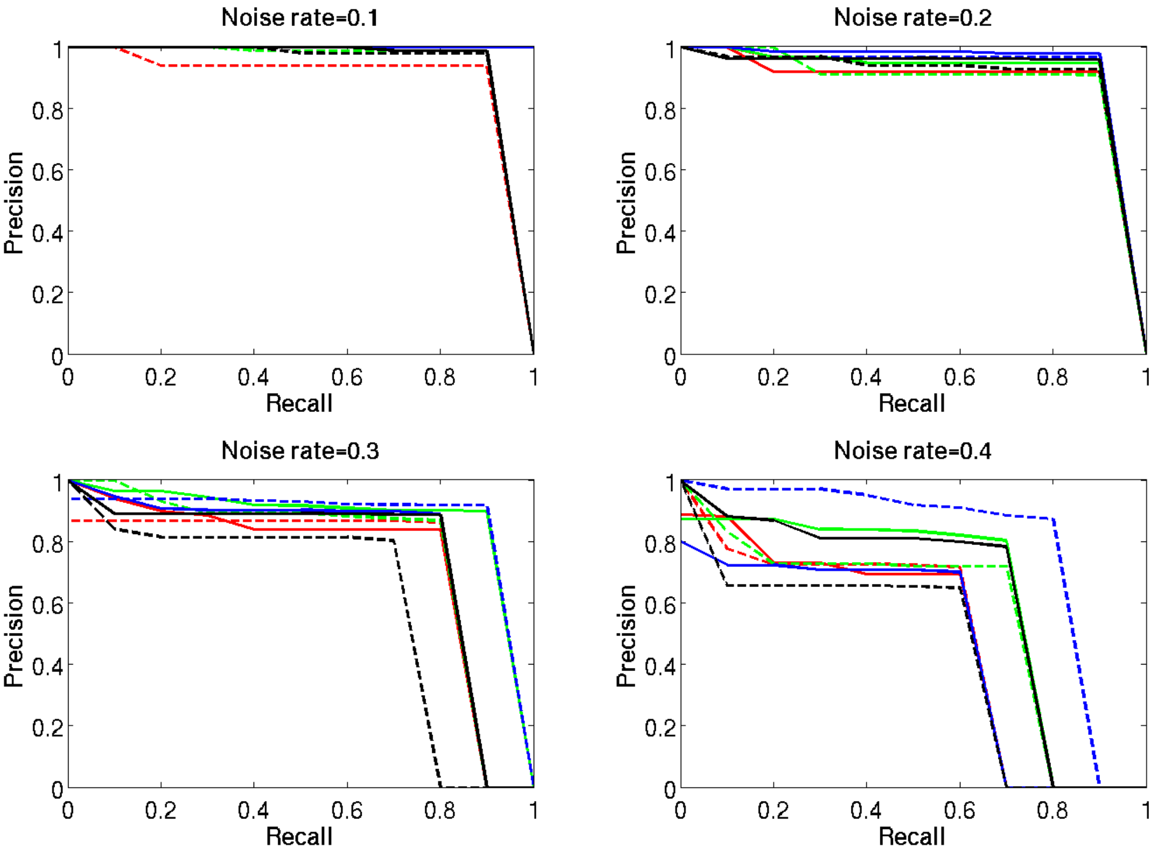


Figure 3.10: *Precision versus Recall curves at the noise rates* 0.1*,* 0.2*,* 0.3*, and* 0.4*. Red, green, blue, and black colors stand for the class types* 1*,* 2*,* 3*, and* 4*, respectively. Solid and dashed lines are for the similarity-based and neural-network-based classification algorithms, respectively.*

Frank (2005); Davis and Goadrich (2006).

Fig. 3.10 depicts some sample PR curves of both similarity-based and neural-network-based classifiers at noise rates 0.1, 0.2, 0.3, and 0.4. Each color (red, green, blue, and black) here represents one of four class types, and solid and dashed lines are for the similarity-based and neural-network-based algorithms, respectively. As explained above an ideal PR curve should be at the upper right corner and this is observed only at the noise rate 0.1 just in similarity-based method (solid lines). Once the noise rate increases, curves of both methods deviate to the left bottom as the number of FPs increases. Hence, after the noise rate 0.2 we do not generally observe any significant difference between curves except for some class types (e.g. blue dashed line deviates less).

Consequently, our experimental results show that there is no significant difference between two classification algorithms. Although one algorithm performs slightly better for an action type at a specific noise rate, it is not consistently the same for all other action types at other noise rates.

3.3 Discussion

In this chapter, some statistical results on the robustness of our proposed algorithms for action classification, object categorization, and learning were highlighted. For this, a seed SEC with a certain size was created and then was altered in two different ways: (1) by adding noisy rows and columns and (2) by replacing original seed indexes with a noisy one. As the noise level in the seed was increasing, we tested how the similarity measure changed, how the classification, categorization, and learning issues were affected, and especially how stable the classification algorithm was compared to neural networks.

The first outcome of the experiments was that the mean similarity measure never dropped to zero even all seed elements were flipped. This is because the similarity algorithm searches for the maximum correspondences between each row of SECs. Furthermore, experiments showed that the proposed classification and categorization algorithms were invariant to the size of event chains. The main argument for this conclusion is that the curves of seeds with different sizes exhibited quite similar behaviors as depicted in Fig. 3.4 and also in Fig. 3.5. These figures as well as Fig. 3.6 also proved that adding only noisy rows and columns had no dramatic affect on the proposed algorithms as long as the original SEC elements remain the same. The reason is that noisy rows and columns are automatically eliminated while the best matching permutation between the shuffled rows is being calculated as explained in step 5 in chapter 2.

The experimental results further showed that at the noise rate 0.2 approximately 20% of objects were mismatched as the similarity value between type-similar actions was getting less than roughly 64%. This statistical result is actually consistent with the one observed in four real action scenarios explained in step $6A$ in chapter 2.

Finally, and most importantly, we compared the proposed similarity-based action classification algorithm with a feed-forward backpropagation neural network. The most important drawback of using neural networks is that they need fixed-size feature vectors as inputs. However, this makes the whole approach unrealistic since SECs can contain noisy rows and columns and can also be in different sizes even for different versions of the same actions as observed in most of real scenarios. On the other hand, statistical results proved that similarity-based classifier had the advantage of being able to cope with noisy rows and columns. Another downside of using such neural-network-based approaches is the training phase for which a huge data set with desired outputs is needed although they are quite fast and efficient. It is also important to note that outputs of neural networks highly depend on the training phase. If networks are not properly trained, results are bad. However, we have already shown that our proposed learning algorithm can render a model SEC even from 10 training samples by considering repetitive rows and columns as explained in step 7 in chapter 2. Thus, from the viewpoint of memory usage, using similarity-based classification algorithm is more efficient.

Last but not least, in Fig. 3.9 (a-d) we analyzed the total number of TP and FP in both classification methods at different noise rates. As an important finding, we observed that the similarity-based classifier performed slightly better, because even around the noise rate 0.3, which means 30% of all entries in the whole event chain are flipped (which is quite a substantial deterioration of the original event chain!), the number of TP is approximately 80%. Performance measures (PR curves) of those two classifiers also emphasized that there was no significant difference between two approaches as both curves deviates almost by the same amount to the left bottom while the noise rate was increasing.

Consequently, statistical results are indicating that our proposed similarity-based classification algorithm is as robust as neural networks. In addition to this, we can categorize objects in conjunction with action classification. This issue would need an additional processing step in neural-network-based approaches, because correspondences between objects are ignored while mapping SECs to feature vectors. The possibility of executing actions directly from the learned SEC model can also be counted as another advantage of using our proposed algorithms.

4

Applications

So far, we introduced a general concept for recognizing, learning, and executing manipulation actions by means of SECs and statistically analyzed robustness of the whole algorithm. In this chapter, we highlight some basic application areas of semantic graphs and event chains in the context of manipulation and scene analyses in cognitive robotics. We, here, basically provide three different applications each of which benefits from different aspects of the semantic graph representation. In the first application *(Case study I)* we emphasize that an agent can recognize (in)correctness of its own actions by trial and error even though the scene is rearranged with different objects compared to the learned one. In *Case study II* we focus on how SECs of long and complex chained manipulations structurally vary from one version to another in a simulated environment to address the most crucial problems observed in real-world patterns. The last application, *Case study III*, introduces a slightly new concept for gardening with a cognitive system that analyzes plant development parameters by means of enhanced scene graphs.

Note that *Case study I* given in this chapter was published in Aksoy et al. (2011a). Main concepts introduced in *Case study II and III* are parts of European projects IntellAct (`http://www.intellact.eu`) and GARNICS (`http://www.garnics.eu`), respectively. *Case study III* was also introduced as a part of a patent (Wörgötter et al., 2012a).

4.1 Case Study I: Learning and Replaying an Action Sequence

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

In the following we show results of a system that allows learning the rules of an action sequence without explicit supervision and then executing actions in a scenario self-assessing "action-effects". Both processes rely on the event chains and the agent can without any pre-defined rule set learn the sequence and then assess the (in)correctness of its actions just by comparing the resulting chains. Condensation into event chains thus helps solving the grounding- as well as the frame problem.

Our robot system is quite simple, consisting of a 3 DOF arm with magnetic gripper (Neurorobotics, Sussex). Thus, we used "pushing" as well as "pick-and-place" as action repertoire. To generate trajectories we used predefined dynamic movement primitives (Ijspeert et al., 2002; Kulvicius et al., 2012) and trajectory start- and end-points (for touching) were visually pre-defined and transferred onto the robot via a standard inverse kinematics procedure (no servoeing). Motion generation and control are not in the focus of this study, therefore we kept this simple here (for an advanced treatment of these aspects see Kulvicius et al. (2012)). Objects for pick-and-place were magnetic.

The desired action sequence was first demonstrated by a human. Fig. 4.1 (a-b) (blue frame) show sample frames of the action sequence in which a hand is "pushing" a lid off a container and then "picking-and-placing" a ball inside. We assume that the event chains of this action sequence is learned by our system as explained in 2.9 in chapter 2. It can be broken into two sub-chains and the final result is shown in Fig 4.2 (a,b).

In the next step we confront the robot with a scene, provide it with a possible set of motion-trajectory start points, and let the robot randomly try out pushing and pick-and-place actions. Fig. 4.1 (c-f)(red frame) show a subset of the different types of actions the robot has tried out (many more were performed but cannot be shown here). The blue tip of the robot arm is visible in the images. Note, objects are usually different from the ones used by the human. In Fig. 4.1 (c) the robot is only pushing a lid but does not continue with pick&place. In (d) a black ball is pushed. Fig. 4.1 (e) shows how the robot picks up a ball and then drops it on the table. Panel (f) represents an action where the robot is taking the ball from a container and places it on the table. All these examples do not (or only incompletely in (c)) reproduce the observed action sequence. Fig 4.1 (g-h) (green frame) show the correct action sequence which at some point was also executed by the robot. All movies used in this experiment can be found at `www.dpi.physik.uni-goettingen.de/~eaksoye/movies.html` (See Appendix A.3).

Corresponding event chains of all those action sequences are given in Fig 4.2. Due to different noise sources (in tracking, segmentation or depth information) the sizes of individual event chains can vary considerably. Still, as discussed in chapters 2 and 3, individual chains contain the relevant information, which is not harmed by noise-induced rows and columns. As a consequence, even very different looking event chains

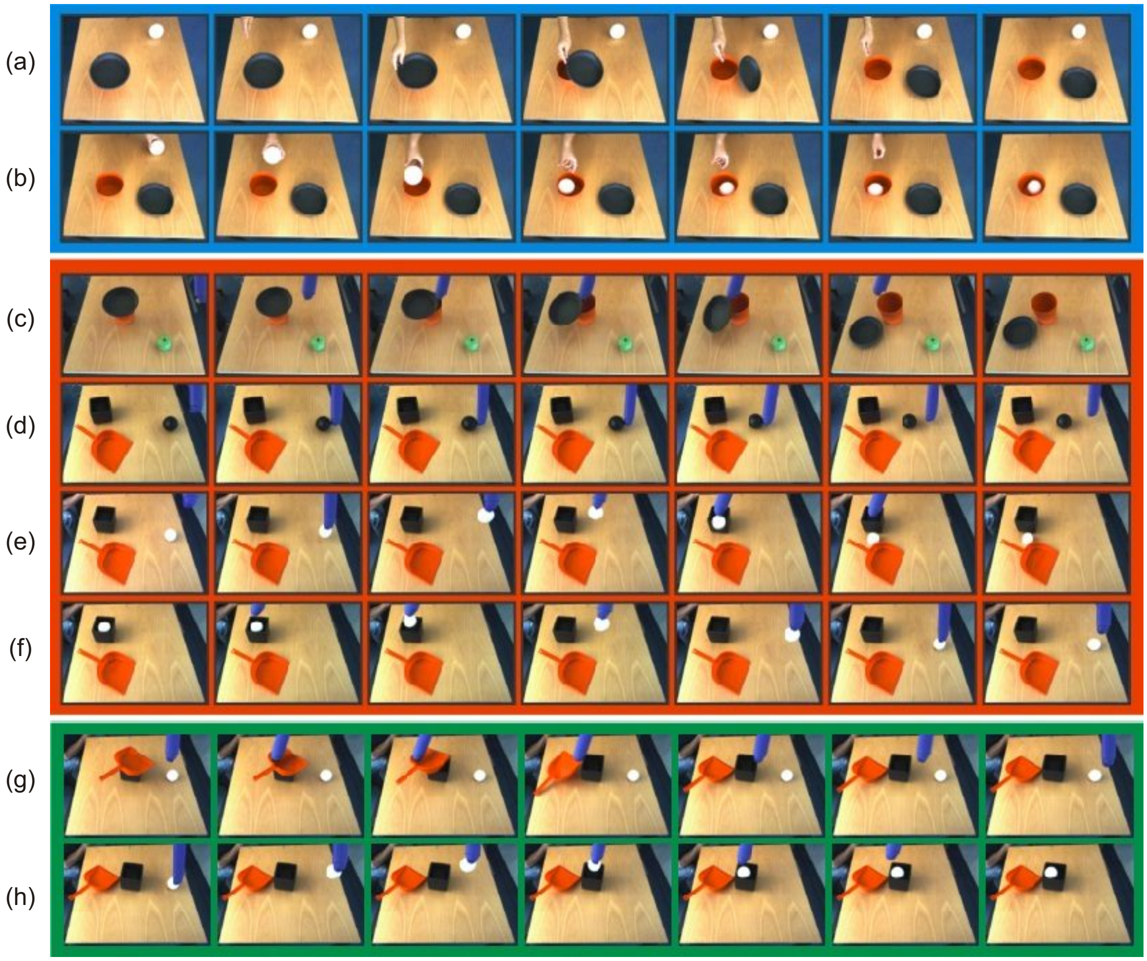
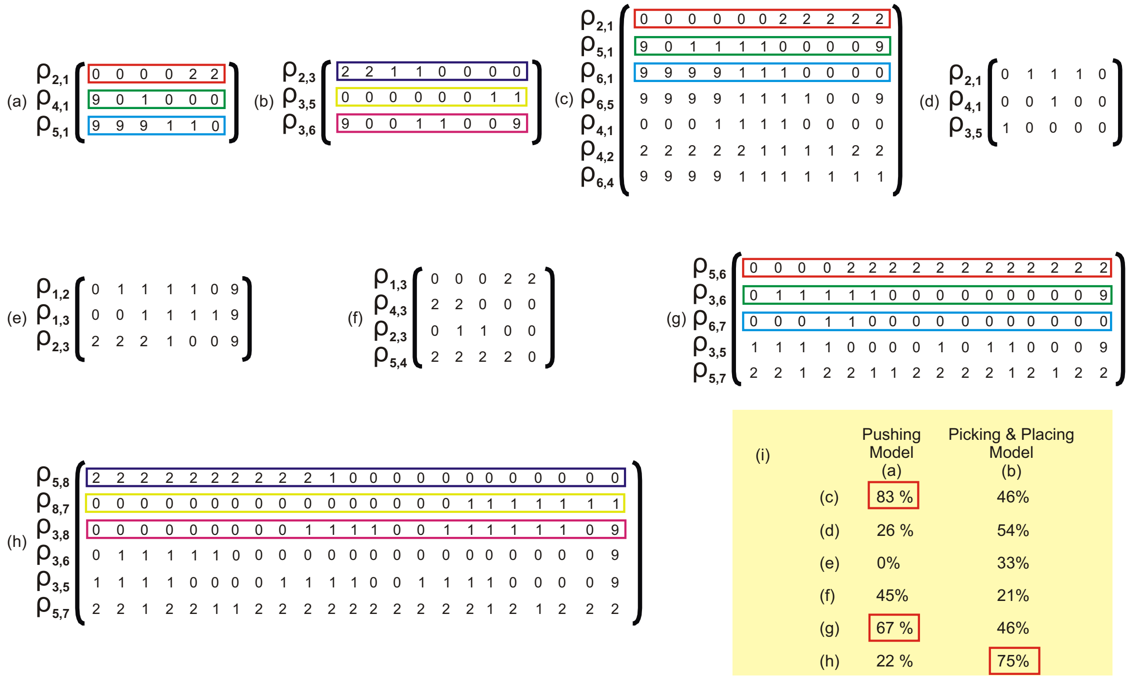


Figure 4.1: *Action sequence of (a) pushing a lid off a container and then (b) putting a ball inside demonstrated by a human (blue frame). Different types of robot actions (red frame). (c) pushing a lid, (d) pushing a ball, (e) lifting a ball and dropping it on the table, (f) taking the ball from a container and putting it on the table. The green frame shows a robot action sequence similar to the one performed by the human, in which (g) a lid is first pushed off and then (h) a ball is placed inside a container.*

can be robustly compared to the learned models (a,b) using the described similarity algorithm in chapter 2. Figure labels (a-h) in Fig. 4.1 correspond to those in Fig. 4.2. Colored boxes in Fig. 4.2 show rows with high similarities. This occurs for panel (c) and (g), which are similar to (a), as well as for (h), which is similar to (b). A similarity table is shown in Fig. 4.2 (i). It shows that manipulation (c) is similar to the learned pushing model (a). The same is true for manipulation (g), which both are above 60% similarity. Only manipulation (h) is similar to the pick and place-inside model (b) with 75% similarity. Sequence (g-h) of both manipulations following each



(a)
$$\begin{array}{cc}\begin{array}{c}\rho_{2,1}\\ \rho_{4,1}\\ \rho_{5,1}\end{array} & \begin{pmatrix} 0&0&0&0&2&2\\ 9&0&1&0&0&0\\ 9&9&9&1&1&0 \end{pmatrix}\end{array}$$

(b)
$$\begin{array}{cc}\begin{array}{c}\rho_{2,3}\\ \rho_{3,5}\\ \rho_{3,6}\end{array} & \begin{pmatrix} 2&2&1&1&0&0&0&0\\ 0&0&0&0&0&0&1&1\\ 9&0&0&1&1&0&0&9 \end{pmatrix}\end{array}$$

(c)
$$\begin{array}{cc}\begin{array}{c}\rho_{2,1}\\ \rho_{5,1}\\ \rho_{6,1}\\ \rho_{6,5}\\ \rho_{4,1}\\ \rho_{4,2}\\ \rho_{6,4}\end{array} & \begin{pmatrix} 0&0&0&0&0&0&2&2&2&2&2\\ 9&0&1&1&1&1&0&0&0&0&9\\ 9&9&9&9&1&1&1&0&0&0&0\\ 9&9&9&9&1&1&1&1&0&0&9\\ 0&0&0&1&1&1&1&0&0&0&0\\ 2&2&2&2&2&1&1&1&1&2&2\\ 9&9&9&9&1&1&1&1&1&1&1 \end{pmatrix}\end{array}$$

(d)
$$\begin{array}{cc}\begin{array}{c}\rho_{2,1}\\ \rho_{4,1}\\ \rho_{3,5}\end{array} & \begin{pmatrix} 0&1&1&1&0\\ 0&0&1&0&0\\ 1&0&0&0&0 \end{pmatrix}\end{array}$$

(e)
$$\begin{array}{cc}\begin{array}{c}\rho_{1,2}\\ \rho_{1,3}\\ \rho_{2,3}\end{array} & \begin{pmatrix} 0&1&1&1&1&0&9\\ 0&0&1&1&1&1&9\\ 2&2&2&1&0&0&9 \end{pmatrix}\end{array}$$

(f)
$$\begin{array}{cc}\begin{array}{c}\rho_{1,3}\\ \rho_{4,3}\\ \rho_{2,3}\\ \rho_{5,4}\end{array} & \begin{pmatrix} 0&0&0&2&2\\ 2&2&0&0&0\\ 0&1&1&0&0\\ 2&2&2&2&0 \end{pmatrix}\end{array}$$

(g)
$$\begin{array}{cc}\begin{array}{c}\rho_{5,6}\\ \rho_{3,6}\\ \rho_{6,7}\\ \rho_{3,5}\\ \rho_{5,7}\end{array} & \begin{pmatrix} 0&0&0&0&2&2&2&2&2&2&2&2&2&2&2&2\\ 0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&9\\ 0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\ 1&1&1&1&0&0&0&0&1&0&1&1&0&0&0&9\\ 2&2&1&2&2&1&1&2&2&2&2&1&2&1&2&2 \end{pmatrix}\end{array}$$

(h)
$$\begin{array}{cc}\begin{array}{c}\rho_{5,8}\\ \rho_{8,7}\\ \rho_{3,8}\\ \rho_{3,6}\\ \rho_{3,5}\\ \rho_{5,7}\end{array} & \begin{pmatrix} 2&2&2&2&2&2&2&2&2&2&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&1&1&1&1&1&1&0&9\\ 0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&9\\ 1&1&1&1&0&0&0&0&1&1&1&1&0&0&1&1&1&1&0&0&0&0&9\\ 2&2&1&2&2&1&1&2&2&2&2&2&2&2&2&2&2&1&2&1&2&2&2 \end{pmatrix}\end{array}$$

(i)

	Pushing Model (a)	Picking & Placing Model (b)
(c)	83 %	46%
(d)	26 %	54%
(e)	0%	33%
(f)	45%	21%
(g)	67 %	46%
(h)	22 %	75%

Figure 4.2: *Corresponding event chains of human demonstrated actions and different types of robot actions. The numbers 0, 1, 2, and 9 in SECs stand for Non-touching, Overlapping, Touching, and Absence, respectively. Labels (a-h) correspond to the manipulations shown in Fig 4.1 (a-h). (a,b) Event chain model extracted from human demonstration of (a) "pushing" as well as (b) "pick-and-place-inside". (c-f) Event chains corresponding to the wrong or incomplete actions in Fig 4.1 (c-f, red frame). (g,h) Event chains corresponding to the correct sequence in Fig 4.1 (g,h, green frame). (i) Similarity table between all actions the robot has tried (c-h) and the learned models (a,b) demonstrated by the human.*

other is, thus, correctly recognized as being the one that reproduces the complete learned model (a-b).

Consequently, this case study is indeed a simple set of examples, however, it demonstrates that by using SECs recognition of manipulations is possible for a robot. The main achievement, we believe, lies here in the very high level of abstraction, which allows the robot to recognize (in)correctness of its actions without any supervision even when objects and their arrangements are very different in the different scenes. Moreover, the robot can successfully derive the correspondences between manipulated objects used in different scenes by considering only their roles as explained in chapter 2. For instance, the red container and black lid used in human demonstration (see Fig. 4.1 (a-b) (blue frame)) correspond to the black container and red lid that the robot tried with (see Fig 4.1 (g-h) (green frame)).

4.2 Case Study II: Semantic Observation and Execution of Manipulations

An intelligent way of imitating human behaviors requires parsing human action primitives and transferring them to a robot in a semantic manner. Therefore, the agent needs to be controlled from the signal (image) level to the semantic level by considering object-action relations that potentially change during manipulations. Our intent, in this section, is to address the problem of semantic decomposing of human actions in conjunction with manipulated objects to be able to reproduce actions with robots. This case study is implemented as a part of the European project IntellAct (`http://www.intellact.eu`) which mainly concentrates on two types of application scenarios (so-called LABEX (Laboratory Experiment) and AUTAS (Automated Assembly)) in which actions need to be monitored and reproduced by a virtual-reality-enhanced simulation as well as by physical robots. The main reason of using the virtual simulation is that scene graphs can be acquired with less noise compared to those from real segmented scenes. Hence, simulated results can be used both to derive more accurate learning policy and to address the most crucial vision-based problems observed in real-world patterns. The first proposed scenario, i.e. LABEX, is more about monitoring long and complex human manipulations for correctness, however, the scenario AUTAS is more about teaching a cognitive robot how to assemble a benchmark in an industrial context. In both scenarios SECs are used to bridge the gap between signal and semantic levels. Note that all simulation experiments given in this section are implemented with “projective virtual reality” provided by the Institute for Man-Machine Interaction from the RWTH-Aachen (`http://www.mmi.rwth-aachen.de`).

4.2.1 Scenario 1: LABEX

In this scenario, we realize a scientific experiment in a laboratory environment. The laboratory setup is chosen as having two cupboards (*cupboard_left* and *cupboard_right*) with doors opening in different directions and two trays (*tray_1* and *tray_2*) to carry two different experiment containers (*container_1* and *container_2*). The LABEX scenario contains in total 13 different objects (including the hand, door handles, etc.) and has three different versions in order to analyze structural deviations in a semantic manner.

We start with virtual simulation of the scenario. First, simulated scenes are represented by graphs; nodes and edges of which represent objects and their 3D spatial relations. The main difference between simulation and real experimental procedures is that segmentation of a scene is not required in the simulation since each object is constructed by a predefined model. Thus, we can directly exclude common segmentation-based vision problems. Moreover, in simulations, spatial relations between objects are calculated by taking bounding boxes of each object into account. For instance, a graph edge appears in the case of having intersecting bounding boxes. Fig. 4.3 shows some sample simulation frames with bounding boxes (depicted in yellow) of different objects.

In all three LABEX scenarios, a hand is basically taking out trays with objects from one cupboard and putting them back in another cupboard after replacing objects on trays. However, for each replacement, cupboard doors need to be opened and closed and trays need to be supported on a global pedestal to replace objects. In each version of the scenario, actions are repeated in different orders. Fig. 4.4 depicts some sample simulated frames with corresponding semantic graphs from the first version of the scenario. Complete list of objects with their corresponding graph node numbers is as follows:

Graph node 1 : *Hand* ,
Graph node 2 : *Tray_1* ,
Graph node 3 : *Pedestal* ,
Graph node 4 : *Door_left* ,
Graph node 5 : *Handle_left* ,
Graph node 6 : *Cupboard_left* ,
Graph node 7 : *Door_right* ,
Graph node 8 : *Handle_right* ,
Graph node 9 : *Cupboard_right* ,
Graph node 10 : *Container_1* ,
Graph node 11 : *Container_2* ,
Graph node 12 : *Tray_2* ,
Graph node 13 : *Table* .

Main action steps followed in this first version can be summarized as follows:

1. Open the door of the left cupboard *(Open_left_door)*
2. Take out the first tray and support it on a global pedestal *(Take_out_tray_1)*
3. Close the door of the left cupboard *(Close_left_door)*
4. Remove the first container *(Remove_container_1)*
5. Open the door of the left cupboard *(Open_left_door)*
6. Insert the first tray back into the left cupboard *(Insert_tray_1)*
7. Close the door of the left cupboard *(Close_left_door)*

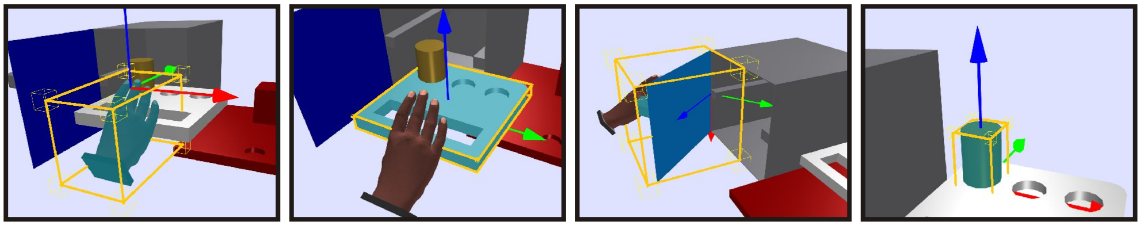
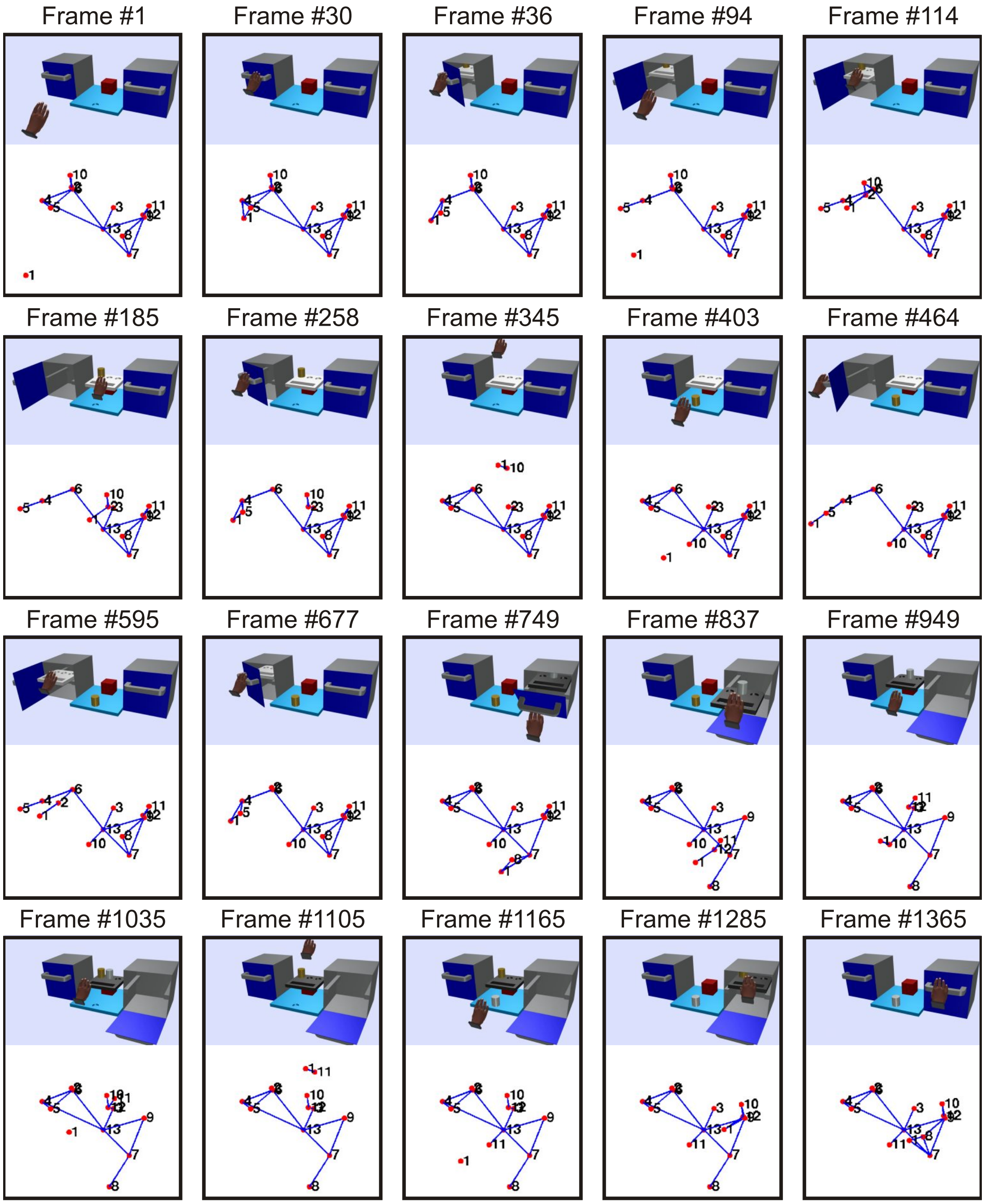


Figure 4.3: *Sample simulation frames from the LABEX scenario. Yellow lines represent the bounding boxes of surrounded objects.*

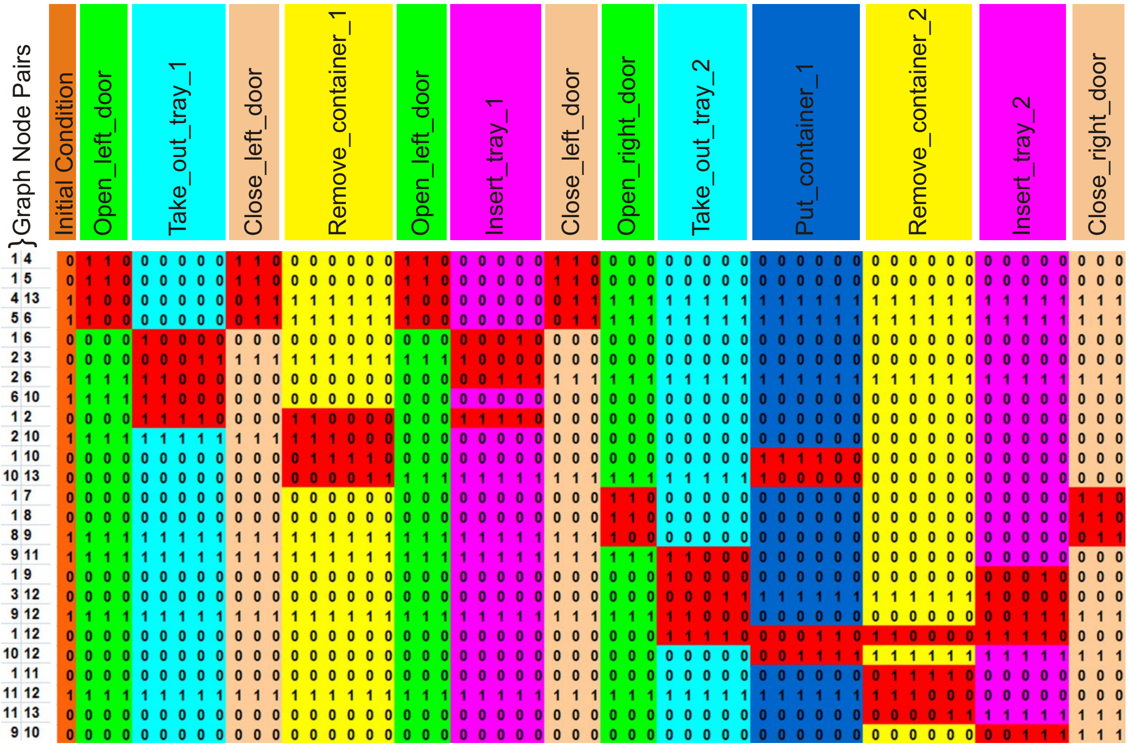


Frame #1
Frame #30
Frame #36
Frame #94
Frame #114
Frame #185
Frame #258
Frame #345
Frame #403
Frame #464
Frame #595
Frame #677
Frame #749
Frame #837
Frame #949
Frame #1035
Frame #1105
Frame #1165
Frame #1285
Frame #1365

Figure 4.4: *Sample simulated frames with corresponding semantic graphs from the first version of the LABEX scenario. Although the scenario has in total* 13 *main steps (see text), the hand basically repeats* 6 *different actions: Open, Take_out, Close, Remove, Insert, Put. Graph nodes from* 1 *to* 13 *correspond to Hand, Tray_1, Pedestal, Door_left, Handle_left, Cupboard_left, Door_right, Handle_right, Cupboard_right, Container_1, Container_2, Tray_2, and Table, in order.*

8. Open the door of the right cupboard *(Open_right_door)*
9. Take out the second tray and support it on a global pedestal *(Take_out_tray_2)*
10. Put the first container on the second tray *(Put_container_1)*
11. Remove the second container *(Remove_container_2)*
12. Insert the second tray back into the right cupboard *(Insert_tray_2)*
13. Close the door of the right cupboard *(Close_right_door)*

Despite of having in total 13 main steps, the hand basically repeats 6 different actions: *Open*, *Take_out*, *Close*, *Remove*, *Insert*, and *Put*. Fig. 4.5 illustrates how those 6 action types, shown in different colors, are embedded in the complete SEC of the scenario. Although the scenario has 1391 frames, it is now encoded by a 25 × 57 matrix, rows and columns of which represent pairwise object relations (e.g. 1 and 0 stand for Touching and Non-touching) and motor primitives, respectively.



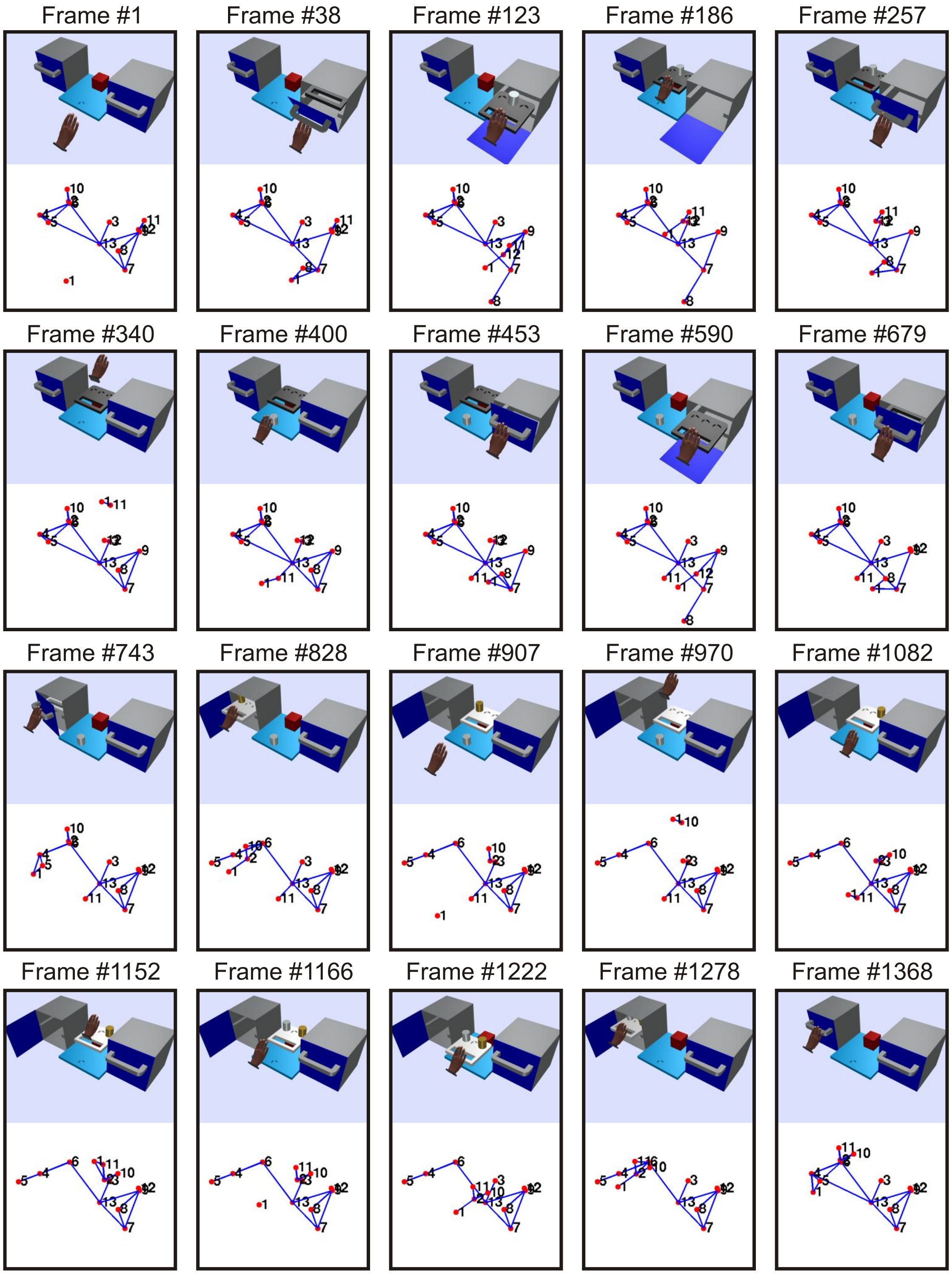
Graph Node Pairs	Initial Condition	Open_left_door			Take_out_tray_1					Close_left_door			Remove_container_1						Open_left_door			Insert_tray_1					Close_left_door			Open_right_door			Take_out_tray_2					Put_container_1						Remove_container_2						Insert_tray_2					Close_right_door		
1 4	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 5	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 13	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5 6	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1 6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 6	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6 10	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
1 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
8 9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
9 11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3 12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
9 12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1 12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0
10 12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
11 12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
11 13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
9 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 4.5: *Semantic event chain of the first version of the LABEX scenario. Each action type (Open, Take_out, Close, Remove, Insert, and Put) is depicted by a different color. The rows are manually rearranged, and relational changes between graph nodes are highlighted in red.* 1 *and* 0 *stand for Touching and Non-touching, respectively.*

In the SEC, each action type is actually encoded by a certain set of relational changes which are highlighted in red. For the sake of simplicity, we manually rearrange the rows, hence, it can easily be seen which objects, i.e. graph nodes, play roles in which action types. For instance, during the first action *Open_left_door*, depicted in green color, only graph node pairs 1-4, 1-5, 4-13, and 5-6 exhibit relational changes as indicated in red, and all other pairs remain the same. The *Open_left_door* action, i.e. first four columns of the SEC, corresponds to the first four graphs shown in Fig. 4.4. In the SEC, the same relational changes, however, between different object pairs (1-7, 1-8, and 8-9), are observed in the case of performing the action *Open_right_door*. This is actually evidence of the fact that in a certain action always the same relational changes are observed even though different objects are manipulated in different time points. The same outcome is observed in other actions (*Take_out*, *Close*, *Remove*, and *Insert*) as well. Note that as observed in the comparison of *Open_left_door* with *Open_right_door* or of *Insert_tray_1* with *Insert_tray_2*, some versions of an action type might naturally include more relational changes due to involving more objects.

Now, we repeat the same actions in a different order and also add a new action type to create a new version of the scenario LABEX. The main aim here is to point out any structural change in the SEC in the case of changing action orders and of including new action types. Fig. 4.6 shows some sample frames with respective scene graphs from the second LABEX version. Main action steps of this second version is as follows:

1. Open the door of the right cupboard *(Open_right_door)*
2. Take out the second tray and support it on a global pedestal *(Take_out_tray_2)*
3. Close the door of the right cupboard *(Close_right_door)*
4. Remove the second container *(Remove_container_2)*
5. Open the door of the right cupboard *(Open_right_door)*
6. Insert the second tray back into the right cupboard *(Insert_tray_2)*
7. Close the door of the right cupboard *(Close_right_door)*
8. Open the door of the left cupboard *(Open_left_door)*
9. Take out the first tray and support it on a global pedestal *(Take_out_tray_1)*
10. Displace the first container *(Displace_container_1)*
11. Put the second container on the first tray *(Put_container_2)*
12. Insert the first tray back into the left cupboard *(Insert_tray_1)*
13. Close the door of the left cupboard *(Close_left_door)*



Frame #1 Frame #38 Frame #123 Frame #186 Frame #257

Frame #340 Frame #400 Frame #453 Frame #590 Frame #679

Frame #743 Frame #828 Frame #907 Frame #970 Frame #1082

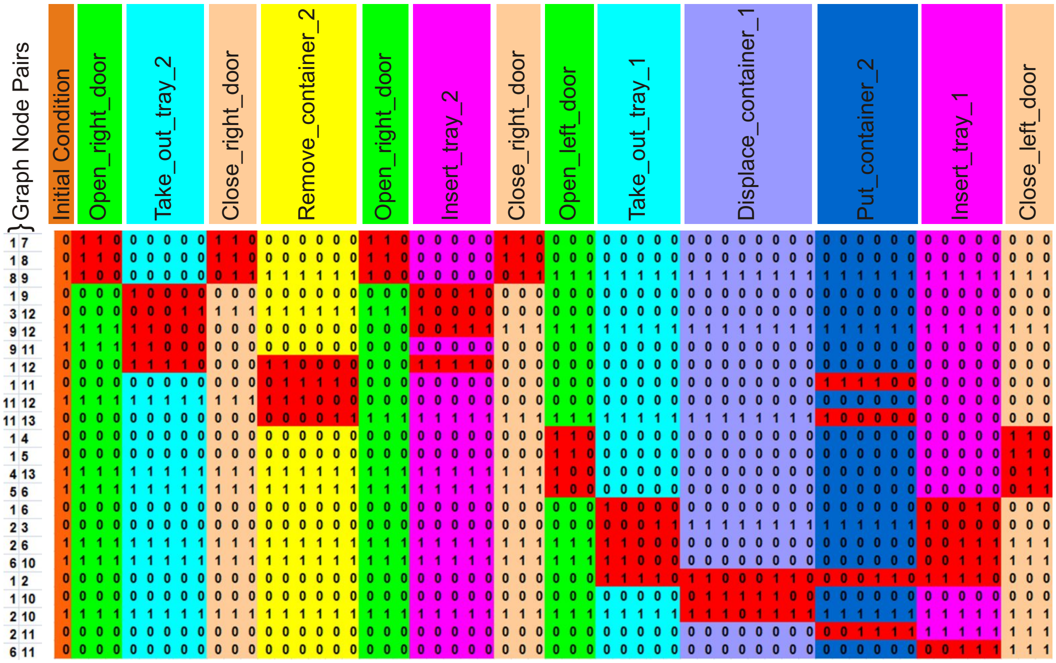
Frame #1152 Frame #1166 Frame #1222 Frame #1278 Frame #1368

Figure 4.6: *Sample frames with corresponding scene graphs from the second version of the LABEX scenario.*

When we compare the two versions, it is obvious that the hand, here, repeats the action sequence starting from the right cupboard and ending with the left cupboard. Moreover, instead of the action *Remove_container_2* performed in the first version, the hand executes a new action tagged as *Displace_container_1*. The SEC of the new version, given in Fig. 4.7, finally exhibits the same structure. The main difference is, as expected, the order of the graph node pairs since the action sequence starts the other way around. The other difference is the new action type, i.e. *Displace_container_1*, which appears in a new blue block in the SEC. This outcome concludes that SECs of long action sequences remain structurally the same even if the sequence order varies.

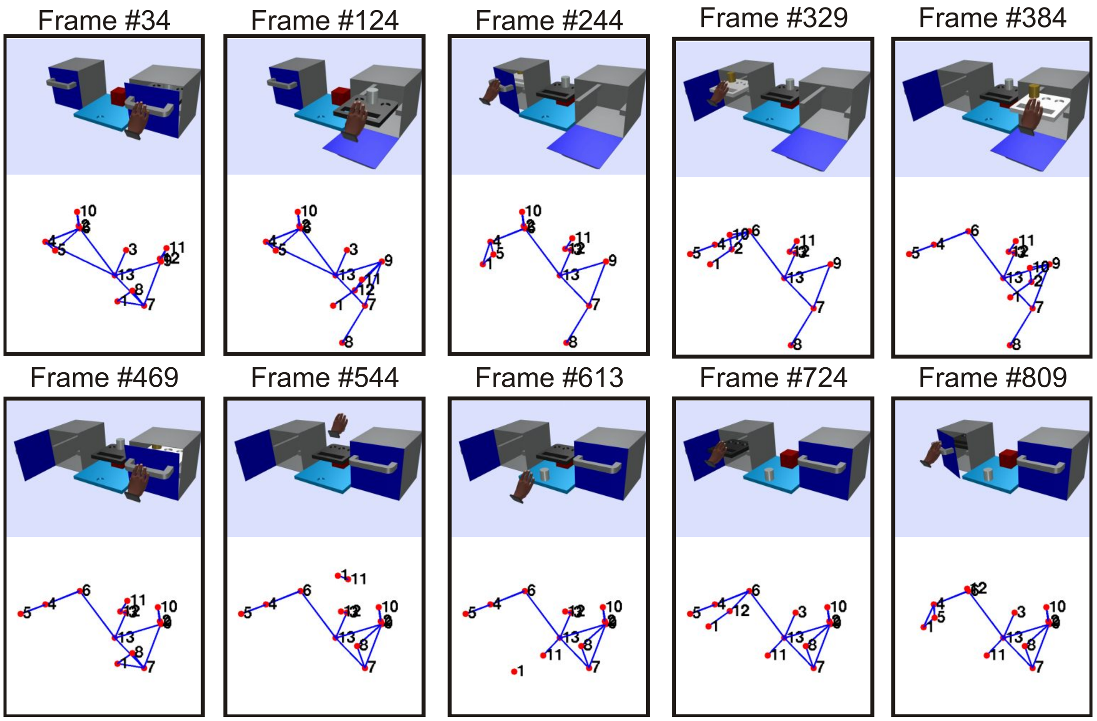
In the third and last version we slightly change the sequence in a way that the order is more shuffled and another new action, tagged as *Replace_tray_1*, is added. Fig. 4.8 (a-b) illustrate some sample frames with respective scene graphs and SEC from the last version. The SEC, here, encodes each action set in the same way as observed in other versions. The main differences are again the sequence order and existence of the new action.

So far, we have analyzed the scenario in the virtual reality (VR) domain. Next, we would like to address basic problems which might occur in real laboratory exper-

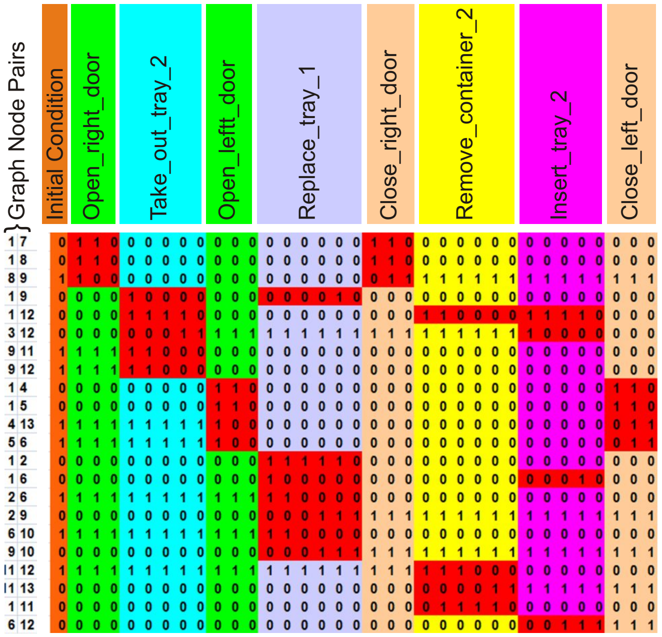


Graph Node Pairs	Initial Condition	Open_right_door	Take_out_tray_2	Close_right_door	Remove_container_2	Open_right_door	Insert_tray_2	Close_right_door	Open_left_door	Take_out_tray_1	Displace_container_1	Put_container_2	Insert_tray_1	Close_left_door
1 7	0	1 1 0	0 0 0 0 0	1 1 0	0 0 0 0 0 0	1 1 0	0 0 0 0 0	1 1 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
1 8	0	1 1 0	0 0 0 0 0	1 1 0	0 0 0 0 0 0	1 1 0	0 0 0 0 0	1 1 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
8 9	1	1 0 0	0 0 0 0 0	0 1 1	1 1 1 1 1 1	1 0 0	0 0 0 0 0	0 1 1	1 1 1	1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1	1 1 1 1 1	1 1 1
1 9	0	0 0 0	1 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 1 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
3 12	0	0 0 0	0 0 0 1 1	1 1 1	1 1 1 1 1 1	1 1 1	1 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
9 12	1	1 1 1	1 1 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 1 1 1	1 1 1	1 1 1	1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1	1 1 1 1 1	1 1 1
9 11	1	1 1 1	1 1 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
1 12	0	0 0 0	1 1 1 1 0	0 0 0	1 1 0 0 0 0	0 0 0	1 1 1 1 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
1 11	0	0 0 0	0 0 0 0 0	0 0 0	0 1 1 1 1 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	1 1 1 1 0 0	0 0 0 0 0	0 0 0
11 12	1	1 1 1	1 1 1 1 1	1 1 1	1 1 1 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
11 13	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 1 1	1 1 1	1 1 1 1 1	1 1 1	1 1 1	1 1 1 1 1	1 1 1 1 1 1 1 1	1 0 0 0 0 0	0 0 0 0 0	0 0 0
1 4	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	1 1 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	1 1 0
1 5	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	1 1 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	1 1 0
4 13	1	1 1 1	1 1 1 1 1	1 1 1	1 1 1 1 1 1	1 1 1	1 1 1 1 1	1 1 1	1 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 1 1
5 6	1	1 1 1	1 1 1 1 1	1 1 1	1 1 1 1 1 1	1 1 1	1 1 1 1 1	1 1 1	1 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0	0 1 1
1 6	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	1 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 1 0	0 0 0
2 3	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1	1 0 0 0 0	0 0 0
2 6	1	1 1 1	1 1 1 1 1	1 1 1	1 1 1 1 1 1	1 1 1	1 1 1 1 1	1 1 1	1 1 1	1 1 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 1 1 1	1 1 1
6 10	1	1 1 1	1 1 1 1 1	1 1 1	1 1 1 1 1 1	1 1 1	1 1 1 1 1	1 1 1	1 1 1	1 1 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 1 1 1	1 1 1
1 2	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	1 1 1 1 0	1 1 0 0 0 1 1 0	0 0 0 1 1 0	1 1 1 1 0	0 0 0
1 10	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 1 1 1 1 1 0 0	0 0 0 0 0 0	0 0 0 0 0	0 0 0
2 10	1	1 1 1	1 1 1 1 1	1 1 1	1 1 1 1 1 1	1 1 1	1 1 1 1 1	1 1 1	1 1 1	1 1 1 1 1	1 1 1 0 1 1 1 1	1 1 1 1 1 1	1 1 1 1 1	1 1 1
2 11	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 1 1 1 1	1 1 1 1 1	1 1 1
6 11	0	0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 0	0 0 0	0 0 0 0 0	0 0 0	0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0 1 1 1	1 1 1

Figure 4.7: *Semantic event chain of the second version of the LABEX scenario. A new action type tagged as Displace_container_1 is added. The SEC exhibits the same structure with the first version. The rows are manually rearranged, and relational changes between graph nodes are highlighted in red.* 1 *and* 0 *stand for Touching and Non-touching, respectively.*



(a) *Sample frames with corresponding scene graphs.*



Graph Node Pairs		Initial Condition	Open_right_door			Take_out_tray_2					Open_leftt_door			Replace_tray_1						Close_right_door			Remove_container_2						Insert_tray_2					Close_left_door		
1	7	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	8	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	9	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	12	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0
3	12	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
9	11	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	12	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	4	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
1	5	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
4	13	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
5	6	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
11	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
6	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

(b) *Semantic event chain. Action orders are shuffled and a new action type tagged as Replace_tray_1 is added.*

Figure 4.8: *Some sample frames with corresponding scene graphs and SEC of the third version of the LABEX scenario.*

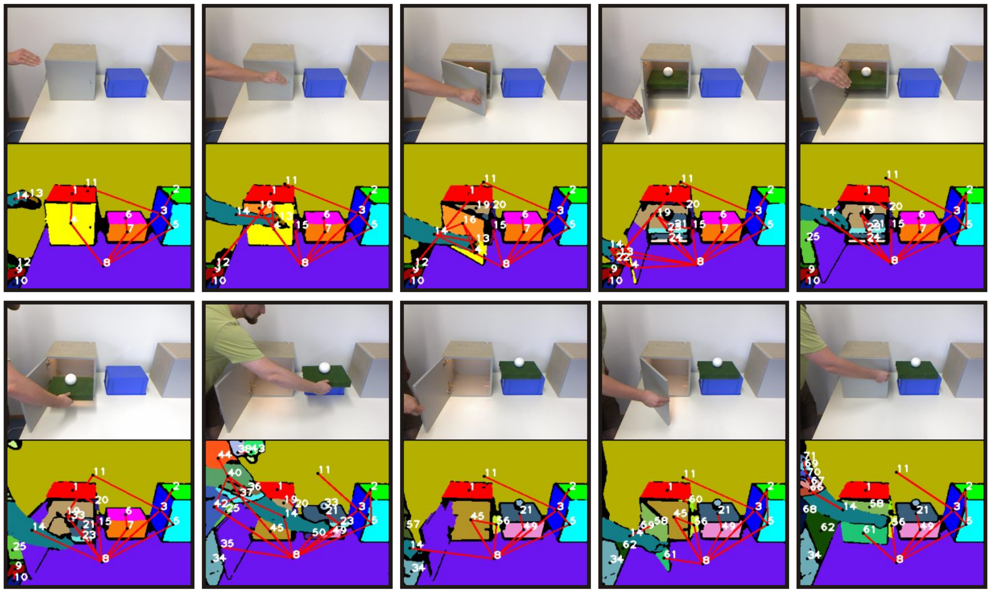
iments. We can essentially assume that the VR data can be used as a ground truth for the real data since simulation results are noiseless. Thus, we can easily see how the semantic graph approach deviates due to problems in early vision.

Fig. 4.9 (a-b) show sample original frames with respective segments and scene graphs from two different real LABEX experiments. In the first one, a hand is opening a cupboard door, extracting a tray with an object, and then closing the door. In the second one, the action sequence is repeated the other way around by starting from the cupboard on the right. In the semantic graphs it is obviously seen that the scene is now represented by more graph nodes which lead to dramatic changes in the graph structure. Potential reasons for such structural changes can be summarized under three different branches: (1) Domain fragmentation, (2) Object recognition, and (3) Occlusion.

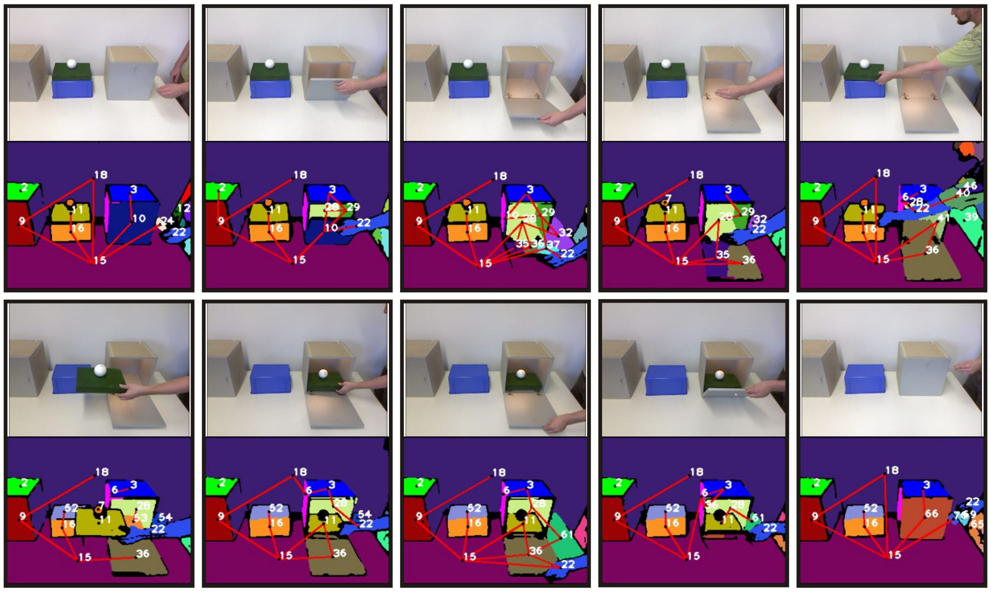
At the early stage of our vision system, we segment images based on super-paramagnetic clustering of pixels considering color and depth cues (Papon et al., 2012a; Abramov et al., 2012). In such clustering-based approaches large regions might be represented by at least two different segments due to high attractive forces between segments. Those artifacts are defined as domain fragmentation (Abramov et al., 2012; Eckes and Vorbrüggen, 1996). It is also non-trivial to detect new upcoming objects in early vision. Such new approaching objects are generally represented by many segments as well. For instance, as shown in Fig. 4.9 (a-b) cupboard doors and the cupboards' insides, while doors are being opened, are represented by at least two different graph nodes due to these two reasons. Our recent tests show that applying a longer relaxation process during the segmentation phase might overcome such problems, however, the system then requires faster hardware to process the movies still in real-time. Alternatively, a model-based object recognition approach could be involved at this stage in order to detect sub-graphs (segments) and to merge them. This process can highly simplify the scene for further calculations of respective SECs. It is, here, important to note that so far the proposed algorithm did not make use of any object recognition approach. Therefore, it requires high-level reasoning to re-form segments.

Partial or full occlusion is another common non-trivial problem in video segmentation. Occlusion is the case when an object becomes (partially) invisible due to being covered by another object. For instance, as shown in Fig. 4.9 (a), due to occlusion, tracking of the blue global pedestal fails while the hand is putting the tray on top. Most segmentation algorithms indeed suffer from occlusion problems since they make use of only color and depth cues without any predictions for the next time point. Our vision system is now being improved by means of particle filters which predict each segment position even under occlusion and reinitialize the segmentation procedure with those predictions (Papon et al., 2012b).

Note that all those proposed solutions for the respective problems are works in progress. However, we can conclude that due to those three main reasons scene graphs structurally deviate compared to the ones observed in the VR domain.



(a) *First version. Cupboards are represented by at least two different graph nodes. Due to occlusion, tracking of the blue global pedestal fails while the hand is putting the tray on top.*



(b) *Second version. Cupboard doors and cupboards insides are represented by at least two different graph nodes.*

Figure 4.9: *Sample original frames with respective segments and scene graphs from two different real LABEX experiments. Scene graphs include many sub-graphs and are more complicated. Potential reasons are: (1) Domain fragmentation, (2) Object recognition, and (3) Occlusion.*

4.2.2 Scenario 2: AUTAS

In the second scenario, we aim at teaching a cognitive robot how to assemble a benchmark in an industrial context. For this purpose we use a standardized assembly task, the so-called "Cranfield Benchmark" (Collins et al., 1984), on which the robot first learns how to assemble different parts in a correct order and then executes the complete manipulation sequence.

Fig. 4.10 depicts some sample steps from the assembly procedure of the Cranfield benchmark. The complete assembly set includes two big face-plates, painted in green and blue, and a black pendulum in between. A yellow shaft is additionally needed to support the face-plates. The red and brown pegs with different shapes and sizes are used to loosely connect all parts. Note that unlike the original benchmark, all parts are here colorized to increase the accuracy in the early vision phase.

Since the complete assembly sequence is quite long and consists of similar repetitive sub-actions (e.g. hold, lift, and place), we can downgrade the whole task to a fundamental action unit, a so-called "Peg in Hole" (PiH) action in which a hand is basically holding a peg and then placing it in a hole. It is important to note that although the PiH action seems to be simple, it is getting complicated due to pose estimation. Unless the peg is held in a certain orientation with respect to the hole, the action will not be executed successfully. Fig. 4.11 (a) illustrates the simulated version of the PiH action with corresponding graphs. The respective SEC of this version is given in Fig. 4.12 (a).

Different human demonstrations of the PiH action are shown in Fig. 4.11 (b-d). Due to the same three problems, discussed in the scenario LABEX in the previous subsection, segmented images include noisy elements which lead to extra nodes and/or edges in the scene graphs. This side effect can obviously be observed as extra rows and/or columns in the corresponding SECs given in Fig. 4.12 (b-d). Despite of having such noisy cases, the essential SEC structure given in Fig. 4.12 (a) can still be observed in all human demonstrations as illustrated with colored frames in Fig. 4.12 (b-d). Note that similarity values between all human demonstrations (Fig. 4.12 (b-d)) and

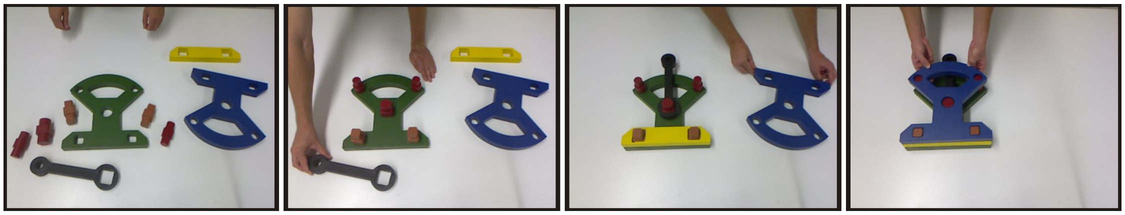
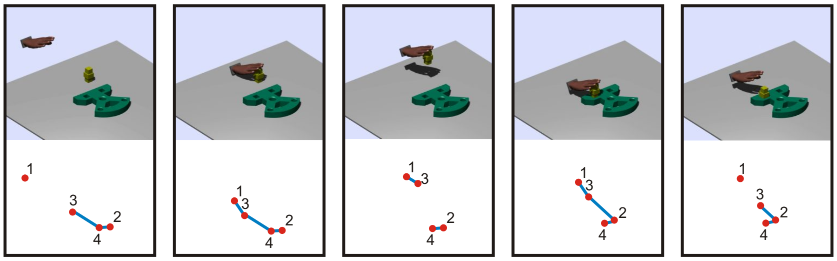
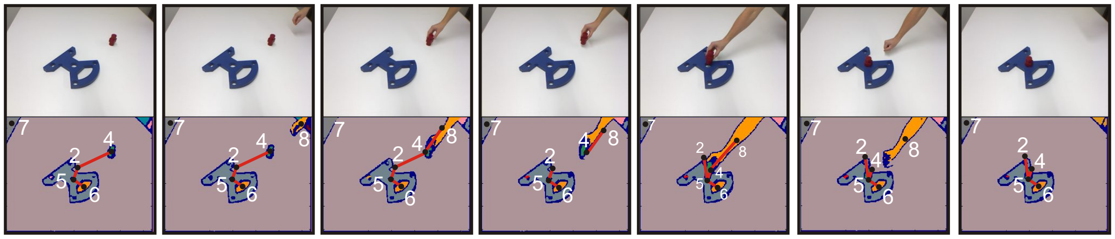


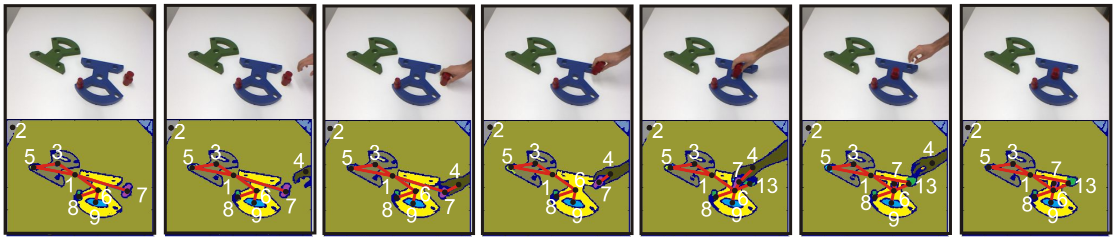
Figure 4.10: *The Cranfield Benchmark (Collins et al., 1984). The complete assembly set includes two big face-plates, painted in green and blue, and a black pendulum in between. A yellow shaft is additionally needed to support the face-plates. The red and brown pegs are used to loosely connect all parts.*



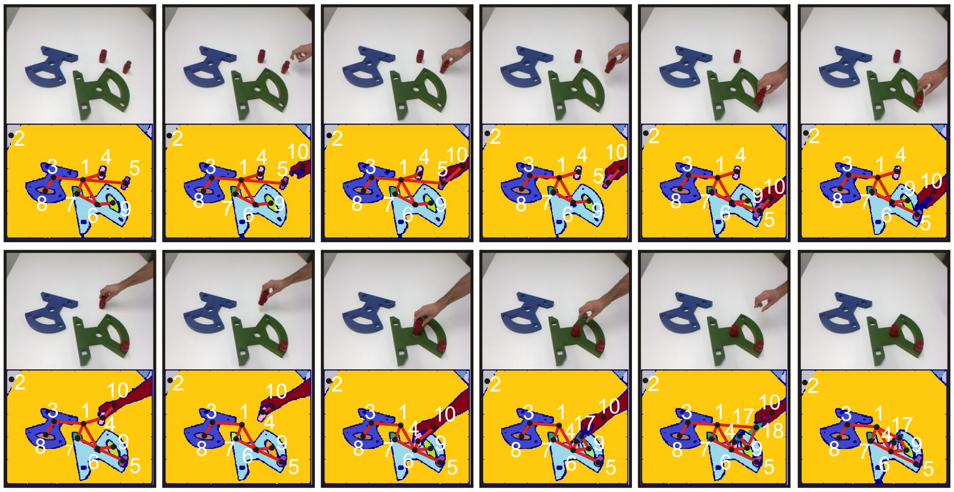
(a) *Virtual simulation data with respective scene graphs.*



(b) *A human demonstration with respective segments and scene graphs.*



(c) *A different human demonstration with respective segments and scene graphs.*



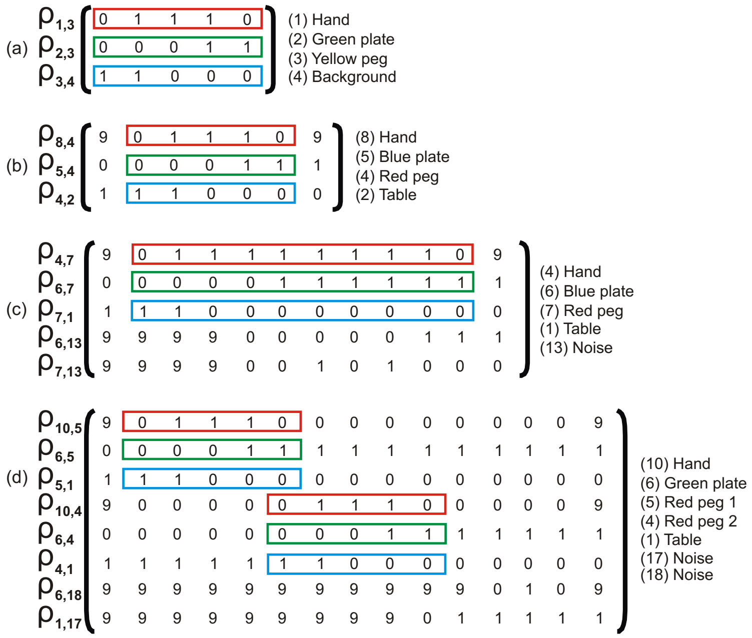
(d) *Two sequential human demonstrations with respective segments and scene graphs.*

Figure 4.11: *Virtual simulation and human demonstrations of the PiH action.*

the simulated one (Fig. 4.12 (a)) are measured as 100%.

In this scenario, it is aimed at learning a representative SEC model of the PiH action from demonstrations to let a robot execute the action directly from the SEC model. However, it is evident that the naked SEC representation is not enough for the execution phase since it includes information neither on objects nor of their poses. Therefore, an extra mid-level reasoning mechanism is needed right after the early vision stage. This issue is currently work in progress and can be solved by means of object modeling. Particularly, segments will be used to calculate the best object models only at decisive temporal anchor points (SEC columns), so that computationally expensive object recognition and pose estimation units can be realized efficiently. Such additional model-based information will then be attached to the graph nodes for the execution issue.

Consequently, those two scenarios explained in this case study clarify decomposition of long and complex actions with SECs by considering main perception problems in the early vision phase. It is also emphasized that some parts of the execution issue are works in progress.



(a)

$$\begin{array}{c} \rho_{1,3} \\ \rho_{2,3} \\ \rho_{3,4} \end{array} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

(1) Hand
(2) Green plate
(3) Yellow peg
(4) Background

(b)

$$\begin{array}{c} \rho_{8,4} \\ \rho_{5,4} \\ \rho_{4,2} \end{array} \begin{bmatrix} 9 & 0 & 1 & 1 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(8) Hand
(5) Blue plate
(4) Red peg
(2) Table

(c)

$$\begin{array}{c} \rho_{4,7} \\ \rho_{6,7} \\ \rho_{7,1} \\ \rho_{6,13} \\ \rho_{7,13} \end{array} \begin{bmatrix} 9 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 9 & 9 & 9 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 9 & 9 & 9 & 9 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

(4) Hand
(6) Blue plate
(7) Red peg
(1) Table
(13) Noise

(d)

$$\begin{array}{c} \rho_{10,5} \\ \rho_{6,5} \\ \rho_{5,1} \\ \rho_{10,4} \\ \rho_{6,4} \\ \rho_{4,1} \\ \rho_{6,18} \\ \rho_{1,17} \end{array} \begin{bmatrix} 9 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 0 & 1 & 0 & 9 \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(10) Hand
(6) Green plate
(5) Red peg 1
(4) Red peg 2
(1) Table
(17) Noise
(18) Noise

Figure 4.12: *Corresponding SECs of virtual simulation and human demonstrated PiH actions. Labels (a-d) correspond to PiH actions shown in Fig. 4.11 (a-d). Despite of having noisy row/columns, (a) the essential SEC structure calculated from VR data can now be observed in (b-d) all human demonstrations as illustrated with colored frames. On the right of each SEC graph node and object correspondences are given. Note that similarity values between all human demonstrations (b-d) and the simulated one (a) are measured as* 100%.

4.3 Case Study III: Plant Growth Measurement System

Scene graphs, explained so far, essentially encode object action relations in a more abstract way. This last case study highlights how an agent can enhance scene graphs in a way that each graph node becomes more descriptive for the perception phase. In this sense, we introduce a cognitive gardening system that analyzes development parameters of plants by means of enhanced semantic scene graphs.

Plants are living organisms and exhibiting slow behavioral changes over time. The complexity of a plant structure increases proportionally during development even in the case of restricted parametric treatments (e.g., changing only watering and lighting conditions) and any change in the parametric space has a delayed effect on the plant growth. In this case study, we address the problem of sensing and controlling plant growth parameters by ways of semantic scene graphs for a cognitive robot-plant interaction. For this purpose, semantic plant graphs are enriched with a simple leaf model which shows behavioral changes like in size, color and mobility with respect to the applied treatments. Such changes are used to monitor the plant behavior and to learn the most appropriate action sequence (e.g, first water then apply more light) for an optimal plant growth. This case study is a part of the European project GARNICS (`http://www.garnics.eu`) and also described as a part of a patent (Wörgötter et al., 2012a). Note that all plant experiments including image acquisition steps given in this section are provided by the Forschungszentrum Juelich GmbH (`http://fz-juelich.de`).

4.3.1 Tracking of Plant Graphs

Our main intent is to get the control over the plant development parameters, such as grow rate, in order to minimize and/or optimize water, nutrients, and light requirements. Size and color distribution of plant leaves are important cues to monitor the lack of such requirements. However, in the case of occlusion it is non-trivial to track all leaves separately and estimate the actual plant grow rate. This is because all leaves have almost the same color distribution and are weakly textured, except for the vein structure. Fig. 4.14 (a-b) show a sample stereo image of a tobacco plant, leaves of which are weakly textured, partly occluded, and have the same color distribution.

To track each leaf separately and drive accurate plant development parameters we apply our segmentation algorithm and enrich the 3D graphs with ellipse mod-

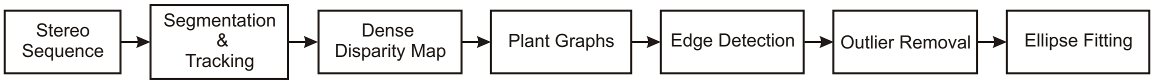
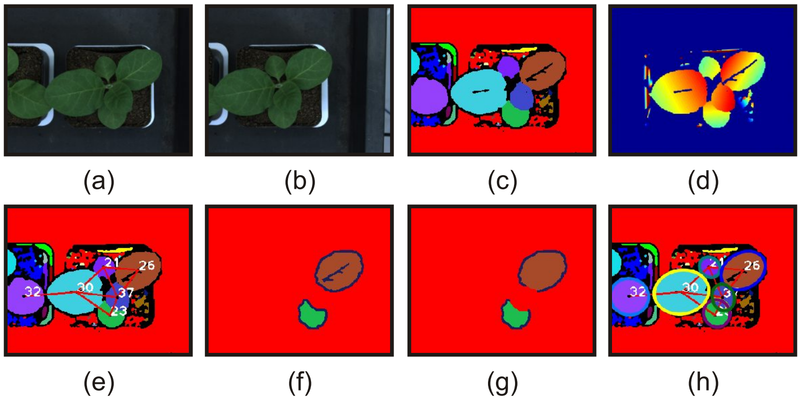


Figure 4.13: *Block diagram of the algorithm*



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

Figure 4.14: *Leaf modeling for a tobacco plant. (a) Original left image. (b) Original right image. (c) Extracted segments for the left image. (d) The dense disparity map obtained for extracted stereo segments. The disparity values are color-coded from blue (small) to red (large). (e) Extracted plant graph. (f) Extracted edge points for the graph nodes 23 and 26. (g) Edge points after removing the occluded ones and outliers. (h) Fitted ellipses and graphs.*

els by taking the dense disparity maps driven from stereo segments into account. Fig. 4.13 shows the block diagram of the complete modeling approach. Fig. 4.14 (c-e) depict how the corresponding segments, dense disparity map, and graph of the sample tobacco image look like. Dense disparity map is the representation of pixel correspondences required for the depth perception. Once graphs are extracted, we fit a simple ellipse model to each graph node. For this purpose, we first retrieve only the extreme outer edges of each segment and then ignore those that are occluded and outliers. Occlusion information of an edge point is easily calculated by comparing its dense disparity value with the ones belonging to the neighboring pixels. Outliers are detected by considering the history information of ellipses. Fig. 4.14 (f-g) illustrate the detected edge points as well as the smoothed ones after removing the occluded points and outliers for the graph nodes 23 and 26. Finally, ellipses are fitted to those edge point clouds with the Least Square technique. Fig. 4.14 (h) shows how ellipses for all graph nodes are fitted.

4.3.2 Leaf Modeling

As tobacco plant leaves used in our experiments are more like ellipsoidal, we represent each leaf with a simple ellipse model. In the regression process, we fit an ellipse to N edge points $(x_i, y_i),\ i \in [1, \cdots, N]$. A general ellipse equation is

$$\left(\tfrac{(x_i - x_c)\cos\theta + (y_i - y_c)\sin\theta}{w}\right)^2 + \left(\tfrac{(x_i - x_c)\sin\theta - (y_i - y_c)\cos\theta}{h}\right)^2 = 1 \quad , \tag{4.1}$$

where x_c and y_c are the center coordinates of the ellipse, θ is the tilt angle, and w and h are the lengths of the minor and major semiaxes (width and height) of the ellipse in the x and y directions, respectively. The fitting error of each point can be calculated as

$$\epsilon_{(x_i,y_i)} = 1 - \left(\frac{(x_i-x_c)\cos\theta+(y_i-y_c)\sin\theta}{w}\right)^2 - \left(\frac{(x_i-x_c)\sin\theta-(y_i-y_c)\cos\theta}{h}\right)^2 \quad . \tag{4.2}$$

In the Least Square technique, to calculate ellipse parameters x_c, y_c, θ, w, and h we minimize the total fitting error as

$$\epsilon_{total} = \sum_{i=1}^{N} [\epsilon_{(x_i,y_i)}]^2 \quad . \tag{4.3}$$

Although the least squares is a fast and commonly used technique, it is not resistant to outliers, i.e. noisy edge points. As an alternative, iterative-based ellipse fitting algorithm, like random sample consensus (RANSAC) (Fischler and Bolles, 1981), can also be used since it detects the outliers. However, RANSAC is a computationally expensive and non-deterministic approach, as it can converge to an optimum solution with a certain probability which increases proportionally to the number of iterations. Therefore, in addition to the least squares we apply a cheap preprocessing step which takes the previous history information of ellipses into account to detect outliers. In the preprocessing step, for instance at frame number t, the ellipse model E_t^j of a graph node j is computed by first calculating the absolute fitting error $|\epsilon_{(x_i,y_i)}|$ of each edge point (x_i, y_i) according to the previously calculated ellipse model E_{t-1}^j. The points with high fitting error, compared to a predefined threshold value, are counted as outliers. The Least Square technique is finally applied only to inliers, thus, the total fitting error (ϵ_{total}) is much more minimized. Note that for the new appearing segments the preprocessing step is ignored as there is no history data.

The preprocessing step is playing a more important role in the case of merged leaf segments. Since our segmentation algorithm is based on color cues and all leaves have almost the same color distribution, segments of neighbor leaves are more likely to be merged. This is a crucial drawback of such a color-based segmentation algorithm as the graph tracking phase can consequently fail. Fig. 4.15 (a-b) show a plant image sequence, segments of which are merged and as a consequence graph node number 29 is wrongly tracked. However, in the preprocessing step edges of the new appearing leaf are detected as outliers, and as the number of those outliers increases continuously from frame to frame, we fit a new ellipse to outliers. Fig. 4.15 (c) depicts the final fitted ellipses for the merged leaf segments.

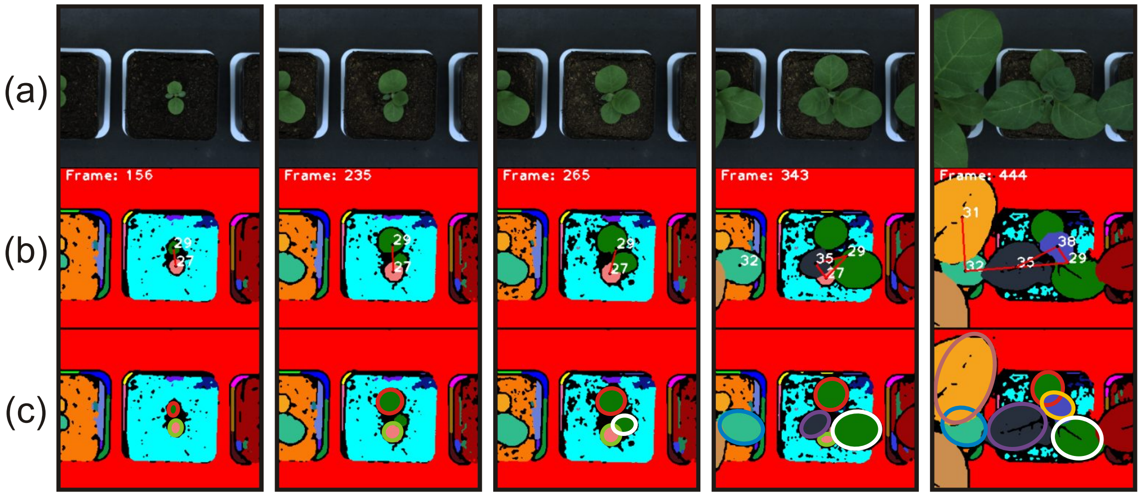


Figure 4.15: *Plant sequence with merged leaf segments. (a) Original image sequence. (b) Extracted segments with corresponding graphs. Two different leaves are represented by the same segment given in green, thus, tracking of the graph node number 29 fails. (c) Fitted ellipses. In the preprocessing step edges of the new appearing leaf are detected as outliers and as the number of outliers increase from frame to frame a new ellipse shown in white is fitted to outliers.*

4.3.3 Extracting Measurement Parameters from Graphs

We applied the proposed graph tracking and leaf modeling algorithm to a tobacco plant sequence which is first a seedling, then growing into a mature plant, and finally wilting due to lack of water. Fig. 4.16 shows original plant images and corresponding segments together with extracted graphs and fitted ellipses. Graphs are continuously tracked and enriched with ellipse parameters which are changing proportionally to the leaves.

Now, in the perception phase we are able to measure plant development parameters like size, color, and mobility directly from graphs. Fig. 4.17 illustrates changes in the leaf size and mobility for the graph nodes 21, 23, 26, and 37 of the plant sequence given in Fig. 4.16. In red and green are indicated the h (height) and w (width) parameters of the fitted ellipses. In blue we plot the final fitting error (ϵ_{final}) which correlates the total fitting error (ϵ_{total}) with size parameters as by:

$$\epsilon_{final} = \epsilon_{total}(\frac{h}{w})(h-w) \quad . \tag{4.4}$$

This is an important correlation to be considered as having less total fitting error (ϵ_{total}) does not guarantee that the leaf is growing in a proper way. The relation between h and w values of the fitted ellipse has to be constant as observed in healthy

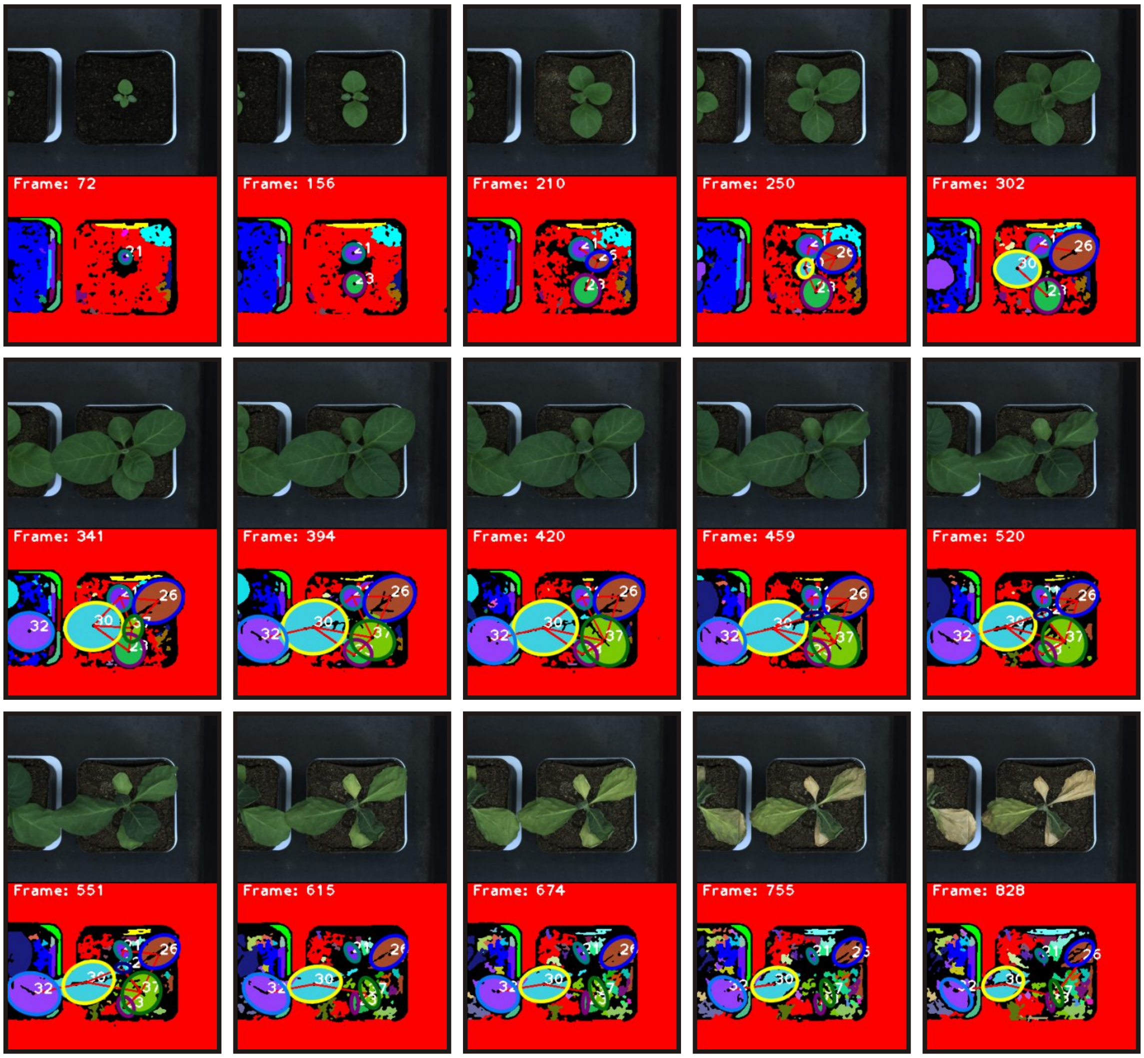
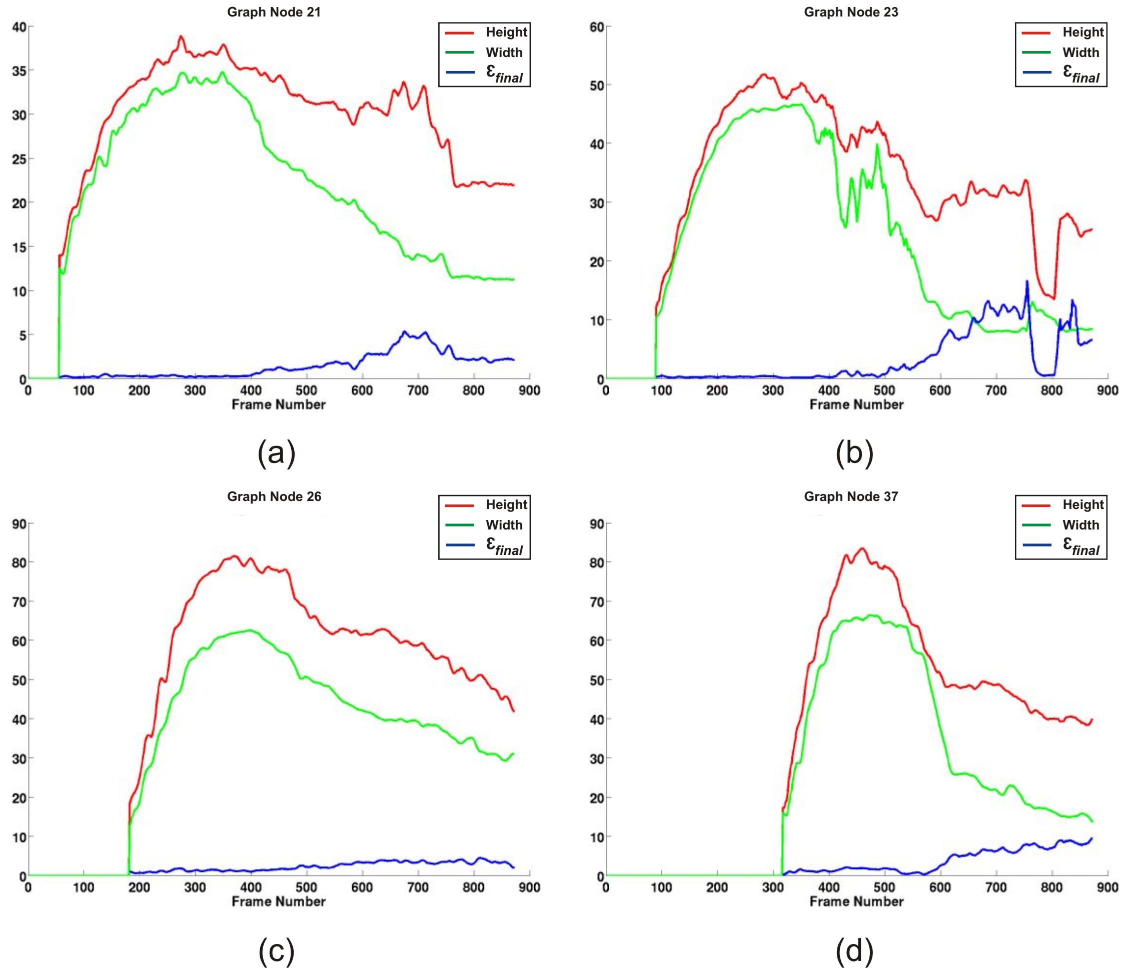


Figure 4.16: *A tobacco plant sequence and corresponding segments together with extracted graphs and fitted ellipses. Graphs are continuously tracked and ellipse parameters are changing proportionally to the leaves.*

plants. As shown in Fig. 4.17, the ϵ_{final} value is quite low while the leaves are growing healthily. After the frame number 600, due to lack of water the leaves start wilting. Therefore, ellipses are elongated in the major axis plane and ϵ_{final} is getting higher. Note that the reason of having less increment of ϵ_{final} in the graph node 26 (see Fig. 4.17 (c)) is that the leaf size is not changing so dramatically as observed for the others.

Not only the size but also the color distribution of the leaves is an important parameter. We calculate the color histogram of each leaf over H (Hue) and S (Saturation) channels in HSV color space and attach this to the respective graph node.



(a) (b)

(c) (d)

Figure 4.17: *Leaf mobilities for the graph nodes* 21, 23, 26, *and* 37 *of the plant sequence given in Fig. 4.16. In red and green are indicated the height* (h) *and width* (w) *parameters of the fitted ellipses. Final fitting error* (ϵ_{final}) *is given in blue.*

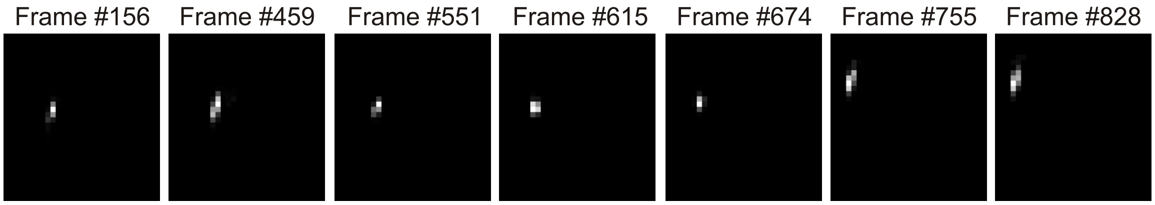


Figure 4.18: *The 2D* 30×32 *Hue-Saturation color histograms of graph node* 21 *at different frames. Around frame number* 600 *the color histogram starts to change gradually and after frame number* 700 *there is a certain amount of variation. Original images in Fig. 4.16 explicitly illustrate the color variation.*

Fig. 4.18 depicts HS color histograms of graph node 21 at different frames to give an impression of the differences in the color domain. Channel H is quantized into 30 bins and channel S into 32 bins. It is obviously seen that around frame number 600 the color histogram starts to change gradually and after frame number 700 there is a certain amount of variation. Original images in Fig. 4.16 explicitly illustrate this color variation.

Consequently, semantic scene graphs can be enriched by attaching more information, like model parameters, color histograms, or even leaf poses given by dense maps. This is an important step since measured parameters from the plant graphs will serve as inputs to the perception phase in which rules for an optimal plant growth can be extracted and learned. This issue is currently work in progress. Model parameters are also useful to rectify segmentation failures as observed in the merged leaf segments. In this sense, an additional feedback mechanism which sends a control signal from scene graphs to the segmentation unit is needed, hence, segmentation of the next frame can be initialized with the model parameters. Implementation of such a feedback loop is also work in progress.

4.4 Discussion

In this chapter, we have mainly concentrated on three different application areas of semantic graphs and event chains in the sense of manipulation and scene analyses. Each application example emphasized different aspects of the semantic graph representation in cognitive robotics.

In the very first application, a robot was introduced with a manipulation sequence, first "pushing" and then "picking-and-placing", SECs of which were stored as learned models. The robot then tried out several manipulations in various orders even on different objects and judged (in)correctness of its own actions just by comparing the resulting event chains. We have seen that, because of having noisy sources like segmentation or depth maps, extracted event chains had different sizes. However, even in such cases the main similarity algorithm had a robust performance. This essentially proves that the SECs can be used for unsupervised monitoring of actions as well as categorization of manipulated objects.

In the second application, we basically highlighted early vision problems observed in the semantic decomposing of long chained manipulations by taking simulation results as reference. In the process of simulated manipulations, no structural variation was observed even though sequence orders were being varied. This proves the stability of SECs. In the semantic analyses of human demonstrated manipulations different segmentation-based problems were addressed. Accuracy of segmentation algorithms can not be evaluated since there is no ground truth for the final segments. As a consequence of that, objects can in the end be over-segmented, such as cupboards and global pedestal which were represented by at least two different graph nodes as shown in Fig. 4.9 (a-b). In order to detect such subgraphs and to merge them for further SEC calculation, a model-based object recognition approach can be used.

The last application scenario made use of such a model-based approach for plant images. A simple ellipse model was calculated for each leaf segment and stored at the respective graph nodes. As an important finding we have observed that even such a simple model was enough to rectify segmentation failures, e.g. the merging problem. Therefore, advanced object modeling and graph enrichment with those models can be used in the semantic analysis of human demonstrated manipulations as well. It is important to note that an additional feedback mechanism is needed to initialize segments of the next frame with the model parameters. Then, any observed error can not propagate from frame to frame.

5

Conclusion and Outlook

All previous chapters, explained so far, include their own "Discussion" sections in which main findings with corresponding advantages and drawbacks are discussed. In this chapter, we will first provide a brief summary of each chapter and then conclude the thesis by comparing our framework with other approaches and presenting an outlook for future investigations.

In this thesis, we presented a novel perception-action framework, the so-called "Semantic Event Chain" (SEC), which bridges the signal (image sequences) to symbol (manipulation primitives) gap by considering the spatiotemporal relations between objects (including hands) in a scene. SECs make a cognitive agent capable of recognizing manipulations *without* requiring prior object knowledge and to categorize manipulated objects solely based on their exhibited roles in a manipulation. Furthermore, using this the agent becomes capable of executing a manipulation directly from an archetypical SEC model which is learned from demonstrated manipulation samples and enriched with additional decisive information. To our knowledge, this is one of the first works in which the extraction of object-manipulation relations have become possible in a model-free way by arriving at a very high symbolic representational level while being fully grounded in the signal domain.

In chapter 2, the complete framework was introduced comprehensively by explaining algorithmic steps illustrated in Fig. 2.2. In the very first step, an acquired image sequence was segmented into unique regions to allow stable tracking during the whole manipulation. However, we emphasized that the realization of this step is not in the core of this thesis since it was mainly achieved by several means described elsewhere (Abramov et al., 2012). Once the segments were extracted, in step 2, they were replaced by scene graphs, nodes and edges of which are segments and their spatial relations such as touching or overlapping. The next step computed the main graphs from the complete graph sequence by taking the eigenvalues of graphs into account. Each main graph basically represents a decisive relational change between segments, i.e. a manipulation primitive. SECs were then created in a matrix form at the fourth step by storing spatial relations between each graph node at each main graph. Hence, SECs reach a high-level symbolic representation of the manipulation while still being tightly linked to the signal level, i.e. image segments from which they originate. In step number 5, we introduced a two-step similarity measure algorithm based on sub-string comparisons and counting procedures in which first the spatial relations (rows) and then temporal relations (columns) of SECs are compared for both classifying manipulations and categorizing manipulated objects in an unsupervised way. In step $6A$ and $6B$, we applied the proposed algorithm to four different real manipulation sequences of scenes containing limited context. Each manipulation type had four different versions which differed in trajectories, speeds, hand positions, and object shapes. The experimental results in Fig. 2.7 and Fig. 2.8 showed that the agent can classify all these manipulation types by measuring the degree of similarity between the manipulation sequences and it can also categorize the participating manipulated objects according to their roles in the manipulations. The next step highlighted the learning mechanism in which all re-occurring rows and columns in the demonstrated SEC samples are extracted. The learning phase was evaluated with two types of manipulations and their mixed versions. The results in Fig. 2.12 indicated that the learned SEC models can be detected in longer chained manipulations even when manipulations are performed concurrently or sequentially. The step number 8 in chapter 2 concluded the framework by introducing the execution phase. We showed how to enrich SECs by storing additional decisive information such as relative coordinate frames and motion start and end points to generate manipulations directly from the learned SECs. Manipulation experiments in Fig. 2.20 showed that regardless of object shapes and positions the cognitive agent can generate the learned manipulation and simultaneously evaluate its own execution success by comparing the learned SEC model with the one resulting from its own execution.

In chapter 3, we presented statistical experiments that serve to test the robustness of action classification, object categorization, and learning phases in the face of different types and levels of noise. All synthetic data produced in the third chapter included not only noisy indexes but also noisy rows/columns which were sampled from a uniform distribution in [0,1]. As an important finding, we observed that once the original event chain remained the same. Adding noisy rows/columns did not change the similarity measure dramatically. Furthermore, the classification algorithm was compared with feed-forward backpropagation neural networks and we concluded that the proposed similarity-based classification algorithm is as robust as neural nets in the process of manipulation classification. Besides, we underlined that SECs are not compatible with neural network-like algorithms which require fixed-size feature vectors. It is because rows of SECs are not always in a certain order, i.e. they can be shuffled, and also sizes of SECs are not predictable and can vary from one manipulation to another. As a consequence of that, while SECs are being converted to a fixed-size feature vector, object information is certainly lost and therefore, neural nets can not succeed well with object categorization. Last but not least, statistical experiments indicated that our unified manipulation classification and object categorization algorithms are invariant to the size of SECs.

Chapter 4 highlighted three different application areas of scene graphs and SECs in the field of cognitive robotics. In the first application study, we demonstrated the feasibility of the perception approach through experiments with a robotic arm. By observation, the robot extracted the SEC of a human manipulation, and then reproduced the associated manipulation type in a new scenario via repeated experimentation. At each try, the robot evaluated the accuracy of its own action by comparing event chains extracted from human demonstrations with the ones obtained from its own execution. Hence, we showed how a cognitive agent can decide (in)correctness of its own action by trial and error while manipulating different objects in a different scene context. In the second application, we further analyzed the semantic perception of long and complex chained manipulations both in Virtual Reality (VR) and real domains. We basically addressed the early vision problems (e.g. domain fragmentation, object recognition, and occlusion) which yield noisy graphs and thus noisy event chains in real experiments. By taking VR simulations as ground-truth we concluded that event chains of real experiments have structural variations due to such early vision problems and this can be overcome by making use of model-based object recognition algorithms. The last application scenario studied a cognitive gardener which analyzes plant developments by means of enriched scene graphs. We showed how plant graph nodes can be enriched with simple leaf models and additional visual information like color histograms. We used leaf models not only to control plant growth parameters but also to cope with low-level early vision problems, e.g. merged leaf segments as depicted in Fig. 4.15.

5.1 Related Approaches

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

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

Furthermore, compared to our work, execution of the perceived manipulations is not covered in their framework.

Our approach is different from the one presented by Kjellström et al. (2011). In our method, event chains are already highly invariant with respect to viewpoint changes and object features because the relations between segments are of qualitative character - e.g., touching is already a semantic description -, while in the work of Kjellström et al. (2011) semantics only emerge at the last stage of the modeling process. In their approach, manipulations are classified according to the segmented hand poses and velocities. Manipulated objects are then searched for in the neighborhood of the hand with a histogram of gradients and categorized with support vector machines. The correlation between manipulation and object features is extracted with factorial conditional random fields which do not model the data generation process. Although their framework to a certain extent improves classification of both objects and manipulations with contextual dependencies, they cannot generate manipulations after learning from demonstrations since manipulation primitives are not extracted. Another limiting factor is that the framework needs fully labeled training data.

Scene graphs and event chains, introduced first in this thesis, have also been interpreted and extended in different contexts (Luo et al., 2011; Griffith et al., 2011). In Luo et al. (2011), the authors built a kernel-based vectorial representation of event chains, which makes SECs more compatible with machine learning techniques. They basically used a different segmentation-based tracking technique to produce scene graphs and event chains. Spatial relations (rows) of the extracted event chains were then converted into string kernels for the issue of action classification. The kernel-based approach indicated improvements in the manipulation classification phase, however, object categorization and execution were not discussed in their framework. Griffith et al. (2011) used scene graphs to analyze co-movement relationships between the robot arm (manipulator) and (manipulated) objects. The graph nodes represent the tracked features of manipulator and manipulated objects. Edges are created when manipulator and manipulated object perform the same movements. As the arm manipulates objects, the graph structure changes with the movement patterns of the tracked features. A statistical test is then used to determine when to delete or insert graph edges. Finally, structural changes in the graph sequences are used as signatures of manipulations to categorize objects. Using only two types of objects (containers and non-containers) and working in off-line mode are the main drawbacks of this algorithm.

5.2 Features and Problems of the SEC Framework

In this thesis, we do not perform any object recognition in the classical sense (see section 1.4.4). Image segments used for finding the SECs are “beneath” the object level, which means that an object may be composed of several segments. Nevertheless, during the manipulation recognition procedure, image segments emerge naturally in conjunction with their associated action, providing a means to extract "action-relevant" objects from the scene by recognizing the respective actions in the SECs. This result is congruent with psychophysical evidence that humans recognize objects more easily if they are embedded in a consistent action context (Helbig et al., 2010). The SEC approach has the advantage that it is highly invariant to the object's appearance and only takes into account the functionality of the object with respect to a given set of actions (see Fig. 2.8). However, the rich information provided by the object's appearance in the image is ignored and thus the algorithm does not allow recognizing objects without providing any action context.

To our knowledge, 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) (see section 1.2). In our framework, objects are being categorized always in the context of the performed manipulation and the rule-character of the event chains was used to let an agent assess the success of its own actions. These properties are closely related to the OAC concept. Thus, a complete semantic event chain (together with its actions and objects) represents a chain of OACs and can be understood as a category which groups objects and actions into the cognitive concept of a manipulation. Thus, the here suggested framework provides, to our knowledge, the first entry point to a grounded, agent-learnable cognitive categorization process of rather high complexity. In addition, it provides a link to the symbolic, language domain because of its rule-like character.

As SECs extract certain relations between object and manipulations, the agent can simply suggest and execute different combinations by changing object or manipulation types, for example, using a glass instead of a cup for the drinking action. We are currently experimenting with such scenarios in which agent is supposed to interchange objects and manipulations.

Our perception-action framework heavily relies on the "segment permanence" (i.e. reliable tracking) which is performed by advanced computer vision methods and we are aware that failures in the computer vision can harm our approach. Clearly, on the computer vision side improvements can be made to better assure this, but this is not in the core of this thesis.

It is important to note that in the case of heavily textured objects, feature binding based on color alone as employed in the segmentation framework will lead to a large number of segments, i.e. objects will be highly fragmented. This may cause problems to the tracking procedures (matching complexity) and thus will affect the quality of the SECs in an undesirable way. The main problem is that there is no comprehensive and common description to define "textures" or "objects", hence there is no ground truth for the final segmentation of objects unless making use of object models. Therefore, we need an additional high level reasoning mechanism to detect and classify segment-groups as objects. We are currently investigating potential solutions to this problem.

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

And last but not least, adults can robustly classify objects and actions using a very high degree of invariance and generalization. To reach such a high classification robustness in artificial systems we are currently creating a large ontology of manipulation actions by taking SECs as reference (Wörgötter et al., 2012b). This may help to understand how manipulation actions are fundamentally structured in the spatiotemporal domain.

Bibliography

Abramov, A., Aksoy, E. E., Dörr, J., Pauwels, K., Wörgötter, F., and Dellen, B. (2010). 3d semantic representation of actions from efficient stereo-image-sequence segmentation on GPUs. In *5th International Symposium 3D Data Processing, Visualization and Transmission*, pages 1–8.

Abramov, A., Pauwels, K., Papon, J., Wörgötter, F., and Dellen, B. (2012). Real-time segmentation of stereo videos on a portable system with a mobile GPU. *IEEE Transactions on Circuits and Systems for Video Technology (in press).*

Aksoy, E. E., Abramov, A., Dörr, J., Ning, K., Dellen, B., and Wörgötter, F. (2011a). Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research*, 30(10):1229–1249.

Aksoy, E. E., Abramov, A., Wörgötter, F., and Dellen, B. (2010). Categorizing object-action relations from semantic scene graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 398–405.

Aksoy, E. E., Dellen, B., Tamosiunaite, M., and Wörgötter, F. (2011b). Execution of a dual-object (pushing) action with semantic event chains. In *Proceedings of 11th IEEE-RAS International Conference on Humanoid Robots*, pages 576–583.

Badler, N. (1975). *Temporal Scene Analysis: Conceptual Descriptions of Object Movements*. PhD thesis, University of Toronto, Canada.

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

Brandes, U., Eiglsperger, M., Lerner, J., and Pich, C. (2010). Graph markup language (graphml).

Brendel, W. and Todorovic, S. (2011). Learning spatiotemporal graphs of human activities. In *IEEE International Conference on Computer Vision (ICCV)*, pages 778–785.

Calinon, S. and Billard, A. (2004). Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2769–2774.

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

Calinon, S. and Billard, A. (2007). Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pages 255–262, New York, NY, USA. ACM.

Collins, K., Palmer, A. J., and Rathmill, K. (1984). The development of a european benchmark for the comparison of assembly robot programming systems. In *Proceedings 1st Robotics Europe Conference*, pages 187–199.

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

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine learning*, pages 233–240. ACM.

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

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

Dellen, B. and Wörgötter, F. (2009). Disparity from stereo-segment silhouettes of weakly textured images. In *Proceedings of the British Machine Vision Conference*, pages 96.1–96.11.

Eckes, C. and Vorbrüggen, J. C. (1996). Combining data-driven and model-based cues for segmentation of video sequences. In *Proc. of World Congress on Neural Networks*, pages 868–875.

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

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fitzpatrick, P. M., Metta, G., Natale, L., Rao, A., and Sandini, G. (2003). Learning about objects through action - initial steps towards artificial cognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3140–3145.

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

Gibson, J. (1979). The ecological approach to visual perception. Boston. Houghton Mifflin.

Gilbert, A., Illingworth, J., and Bowden, R. (2011). Action recognition using mined hierarchical compound features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):883–897.

Griffith, S., Sukhoy, V., and Stoytchev, A. (2011). Using sequences of movement dependency graphs to form object categories. In *Humanoids*, pages 715–720.

Hakeem, A. and Shah, M. (2005). Multiple agent event detection and representation in videos. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, AAAI'05, pages 89–94.

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

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

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

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

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

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

Junejo, I., Dexter, E., Laptev, I., and Perez, P. (2011). View-independent action recognition from temporal self-similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):172–185.

Kjellström, H., Romero, J., and Kragić, D. (2011). Visual object-action recognition: Inferring object affordances from human demonstration. *Comput. Vis. Image Underst.*, 115(1):81–90.

Kjellström, H., Romero, J., Mercado, D. M., and Kragic, D. (2008). Simultaneous visual recognition of manipulation actions and manipulated objects. In *European Conference on Computer Vision*, pages 336–349.

Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., Ude, A., Asfour, T., Kraft, D., Omrčen, D., Agostini, A., and Dillmann, R. (2011). Object-action complexes: Grounded abstractions of sensorimotor processes. *Robotics and Autonomous Systems*, 59(10):740–757.

Kulvicius, T., Ning, K., Tamosiunaite, M., and Wrgtter, F. (2012). Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, 28(1):145–157.

Laptev, I. and Perez, P. (2007). Retrieving actions in movies. In *IEEE 11th International Conference on Computer Vision, (ICCV)*, pages 1–8.

Li, Q. and Payandeh, S. (2007). Manipulation of convex objects via two-agent point-contact push. *Int. J. Rob. Res.*, 26:377–403.

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

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

Luo, G., Bergström, N., Ek, C. H., and Kragic, D. (2011). Representing actions with kernels. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS)*, pages 2028–2035.

Lynch, K. and Mason, M. (1995). Stable pushing: Mechanics, controllability, and planning. In *Algorithmic Foundations of Robotics*, pages 239–262, Boston, MA.

Maurer, A., Hersch, M., and Billard, A. (2005). Extended hopfield network for sequence learning: Application to gesture recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Biological Inspirations - Volume Part I*, pages 493–498.

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

Meltzoff, A. N. (1988). Infant imitation after a 1-week delay: Long-term memory for novel acts and multiple stimuli. *Developmental Psychology*, 24(4):470–476.

Meltzoff, A. N. (2002). *Elements of a developmental theory of imitation*, pages 19–41. Cambridge University Press, Cambridge, MA, USA.

Meltzoff, A. N. and Moore, M. K. (1977). Imitation of facial and manual gestures by human neonates. *Science*, 198(4312):75–78.

Meltzoff, A. N. and Moore, M. K. (1997). Explaining facial imitation: a theoretical model. *Early Development and Parenting*, 6(34):179–192.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1st edition.

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

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26.

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

Mundy, J. L. (2006). Object recognition in the geometric era: A retrospective. In *Toward Category-Level Object Recognition*, pages 3–28. Springer.

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

Nelson, M. M. and Illingworth, W. (1991). *A practical guide to neural nets*. Texas Instruments.

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

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168.

Ogawara, K., Takamatsu, J., Kimura, H., and Katsushi, I. (2002). Modeling manipulation interactions by hidden markov models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1096–1101.

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

Omrcen, D., Asfour, C. B. T., Ude, A., and Dillmann, R. (2009). Autonomous acquisition of pushing actions to support object grasping with a humanoid robot. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 277–283, Paris, France.

Papon, J., Abramov, A., Aksoy, E. E., and Wörgötter, F. (2012a). A modular system architecture for online parallel vision pipelines. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 361–368.

Papon, J., Abramov, A., and Wörgötter, F. (2012b). Occlusion handling in video segmentation via predictive feedback (in review). In *12th European Conference on Computer Vision (ECCV).*

Pauwels, K. and Van Hulle, M. (2008). Realtime phase-based optical flow on the GPU. In *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Computer Vision on the GPU*, pages 1–8, Anchorage, Alaska.

Raamana, P. R., Grest, D., and Krüger, V. (2007). Human action recognition in table-top scenarios: an HMM-based analysis to optimize the performance. In *Proceedings of the 12th International Conference on Computer Analysis of Images and Patterns*, pages 101–108, Berlin, Heidelberg. Springer-Verlag.

Rao, R. P. N., Shon, A. P., and Meltzoff, A. N. (2004). *A Bayesian Model of Imitation in Infants and Robots*. Cambridge University Press.

Ridge, B., Skočaj, D., and Leonardis, A. (2009). Unsupervised learning of basic object affordances from object properties. In *Proceedings of the Fourteenth Computer Vision Winter Workshop (CVWW)*, pages 21–28, Eibiswald, Austria.

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

Rizzolatti, G., Fogassi, L., and Gallese, V. (2001). Neurophysiological mechanisms underlying the understanding and imitation of action. *Nature Reviews Neuroscience*, 2(9):661–670.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction*. Springer, 1 edition.

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

Salganicoff, M., Metta, G., Oddera, A., and Sandini, G. (1993). A vision-based learning method for pushing manipulation. In *AAAI Fall Symposium Series: Machine Learning in Vision: What Why and How?*

Shylo, N., Wörgötter, F., and Dellen, B. (2009). Ascertaining relevant changes in visual data by interfacing AI reasoning and low-level visual information via temporally stable image segments. In *Proceedings of the International Conference on Cognitive Systems*, pages 153–160.

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 1470–1477.

Sridhar, M., Cohn, A. G., and Hogg, D. C. (2010). Discovering an event taxonomy from video using qualitative spatio-temporal graphs. In *19th European Conference on Artificial Intelligence*, pages 1103–1104.

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

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

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

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

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

Umiltà, M. A., Kohler, E., Gallese, V., Fogassi, L., Fadiga, L., Keysers, C., and Rizzolatti, G. (2001). I know what you are doing: A neurophysiological study. *Neuron*, 31:155–165.

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

Wen-Jing, L. and Tong, L. (2000). Object recognition by sub-scene graph matching. In *Proceedings of the 20th IEEE International Conference on Robotics and Automation*, pages 1459–1464.

Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques.* The Morgan Kaufmann Series in Data Management Systems.

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

Wörgötter, F., Aksoy, E. E., Abramov, A., and Dellen, B. (2012a). Method and device for estimating the plant development parameters. *Patent, 3294-057 PCT-1, (pending).*

Wörgötter, F., Aksoy, E. E., Krüger, N., Piater, J., Ude, A., and Tamosiunaite, M. (2012b). A simple ontology of manipulation actions based on hand-object relations. *IEEE Transactions on Autonomous Mental Development, (submitted).*

A

Appendix

A.1 Defining Segment Relations in a Fast and Efficient Way

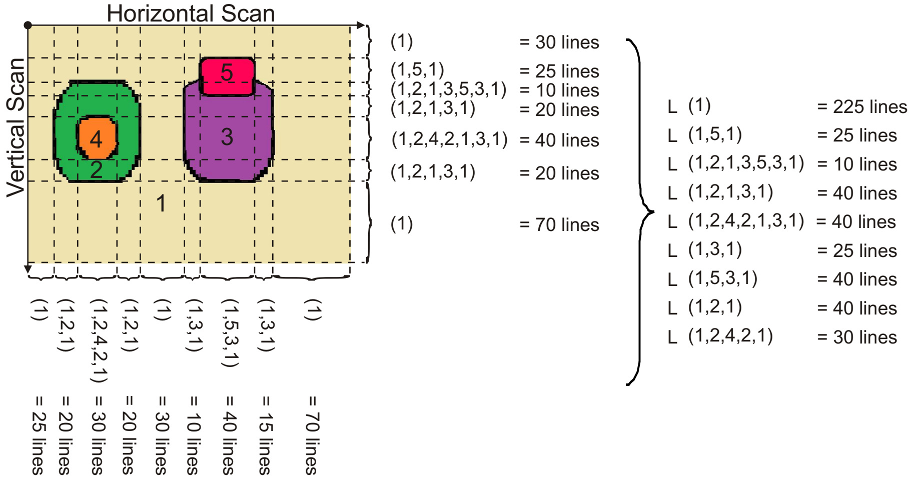


Figure A.1: *Calculating the spatial segment relations between background, two vessels, and two contents which are represented by segment number 1, 2, 3, 4, and 5, respectively.*

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

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

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

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

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

- **"Touching"**: Segments follow one right after the other in any sequence are touching, e.g. segments 5 and 3 are touching each other in such sequences $(..., 5, 3, ...)$ or $(..., 3, 5, ...)$.

- **“Overlapping”**: (i) If a segment is observed twice in a sequence, all segments in between are overlapping with it, e.g. in $(...,1,5,3,1,...)$ 5 and 3 are both overlapped (surrounded) by 1. (ii) However, the inner segments are not overlapping with each other, e.g. in $(...,1,5,3,1,...)$ 5 cannot overlap with 3 because it is not observed twice.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

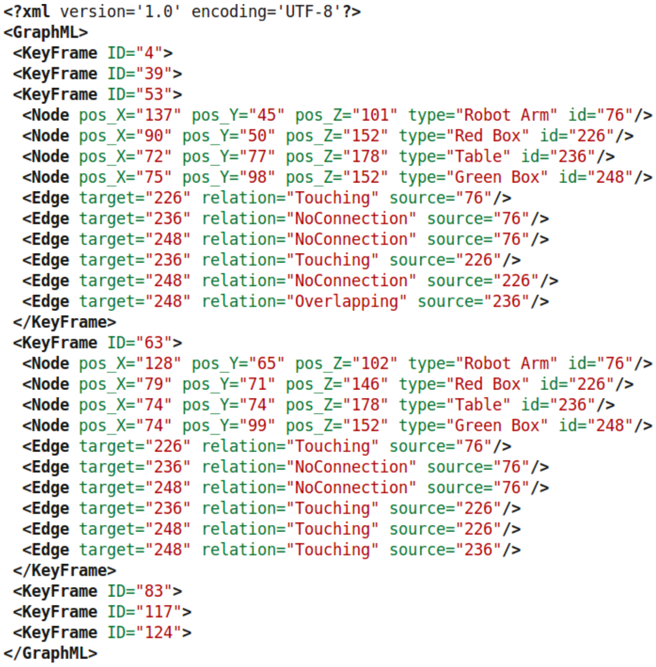
The main advantage of the proposed algorithm is that each step explained above can be calculated separately and hence can be parallelized.

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

A.2 The GraphML File Format

SECs are in the form of human interpretable tables, not a machine readable format, and they do not encode object, trajectory, and pose information. To address these aspects, we choose a different, second form of representation: XML-based GraphML file format (Brandes et al., 2010). The basis of the GraphML file is the SEC, where every key frame is encoded as one GraphML tag with more information. This representation is thus more detailed and difficult to read for a human. On the other hand, the GraphML code is machine compatible and can be used for automatic, computer-based manipulation processes such as recognition and execution. Essentially, it is identical to a SEC, just using a different type of encoding, but with the additional possibility of supplementing object, pose, and trajectory information.

Fig. A.2 shows the GraphML code of the action given in Fig. 2.13 and 2.14 in chapter 2, where a robot arm is pushing a red box to a green box on a table. Third and fourth "Key Frames", shown in detail in Fig. A.2, represent the instant when the red and green boxes are touching each other, thus, the relation between the boxes is changing from "NoConnection" to "Touching". Additional object and trajectory information (e.g. 3D segment positions) is attached in the GraphML code as node attributes. Note that pose information is not given in the GraphML code since it was not required in the pushing example.



```
<?xml version='1.0' encoding='UTF-8'?>
<GraphML>
 <KeyFrame ID="4">
 <KeyFrame ID="39">
 <KeyFrame ID="53">
  <Node pos_X="137" pos_Y="45" pos_Z="101" type="Robot Arm" id="76"/>
  <Node pos_X="90" pos_Y="50" pos_Z="152" type="Red Box" id="226"/>
  <Node pos_X="72" pos_Y="77" pos_Z="178" type="Table" id="236"/>
  <Node pos_X="75" pos_Y="98" pos_Z="152" type="Green Box" id="248"/>
  <Edge target="226" relation="Touching" source="76"/>
  <Edge target="236" relation="NoConnection" source="76"/>
  <Edge target="248" relation="NoConnection" source="76"/>
  <Edge target="236" relation="Touching" source="226"/>
  <Edge target="248" relation="NoConnection" source="226"/>
  <Edge target="248" relation="Overlapping" source="236"/>
 </KeyFrame>
 <KeyFrame ID="63">
  <Node pos_X="128" pos_Y="65" pos_Z="102" type="Robot Arm" id="76"/>
  <Node pos_X="79" pos_Y="71" pos_Z="146" type="Red Box" id="226"/>
  <Node pos_X="74" pos_Y="74" pos_Z="178" type="Table" id="236"/>
  <Node pos_X="74" pos_Y="99" pos_Z="152" type="Green Box" id="248"/>
  <Edge target="226" relation="Touching" source="76"/>
  <Edge target="236" relation="NoConnection" source="76"/>
  <Edge target="248" relation="NoConnection" source="76"/>
  <Edge target="236" relation="Touching" source="226"/>
  <Edge target="248" relation="Touching" source="226"/>
  <Edge target="248" relation="Touching" source="236"/>
 </KeyFrame>
 <KeyFrame ID="83">
 <KeyFrame ID="117">
 <KeyFrame ID="124">
</GraphML>
```

Figure A.2: *GraphML code of the action given in Fig. 2.13 and 2.14 in chapter 2, where a robot arm is pushing a red box to a green box on a table.*

A.3 Index of Multimedia Extensions

The multimedia extensions (see Table A.1) to this thesis are at: `http://www.dpi.physik.uni-goettingen.de/~eaksoye/movies.html`.

Extension	Type	Description
1	Video	Artificial Moving Object Actions
2	Video	Four Different Real Action Types
3	Video	Learning Data Set
4	Video	Case Study
5	Video	Pushing Action Training Data
6	Video	Pushing Action Test Data

Table A.1: Multimedia Extensions

B
Curriculum Vitae

EREN ERDAL AKSOY

Research Assistant, Bernstein Center for Computational Neuroscience (BCCN)
Georg-August-Universität Göttingen
III Physikalisches Institut - Biophysik
Friedrich-Hund Platz 1
37077 Göttingen

Date and place of birth:	11 July 1982 Kars, Turkey
Citizenship:	Republic of Turkey
E-mail:	eaksoye@physik3.gwdg.de

EDUCATION

2008 Jun – 2012 Jul	PhD Student at the BCCN Georg-August-Universität Göttingen, Germany
2005 Sep – 2008 Apr	M.Sc. in Mechatronic Engineering Siegen University Siegen, Germany
2000 Sep – 2005 Jun	B.Sc. in Electrical and Electronics Engineering Cukurova University Adana, Turkey

HONORS

- May, 2010 Nominated for Best Cognitive Robotics Paper Award at ICRA in Alaska, USA.

- May, 2009 3rd Best Student Talk, European Workshop on Advanced Predictive Sensor-Motor Control, Judokrante, Lithuania.

- Jan, 2007 Winner of the Mobile-robot Competition, Siegen University.
- Jun, 2005 Graduation from Cukurova University with the best GPA in the class and also the second best GPA overall in the department.

RESEARCH INTERESTS

Scene Graphs & Semantic Event Chains
Object Action Complexes
Learning by Observation
Computer Vision & Robotics

LIST OF PUBLICATIONS

Journal Papers

- Wörgötter, F., Aksoy, E.E., Krüger, N., Piater, J., Ude, A., and Tamosiunaite, M. A simple ontology of manipulation actions based on hand-object relations. *IEEE Transactions on Autonomous Mental Development, (submitted), 2012.*
- Aksoy, E.E., Abramov, A., Dörr, J., Ning, K., Dellen, B., and Wörgötter, F. Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research (IJRR), 30(10):1229-1249, 2011.*
- Dellen, B, Aksoy, E.E., and Wörgötter, F. Segment tracking via a spatiotemporal linking process including feedback stabilization in an n-D Lattice Model. *Sensors, 2009; 9(11):9355-9379.*

Conference Papers

- Papon, J., Abramov, A., Aksoy, E.E., and Wörgötter, F. A modular system architecture for online parallel vision pipelines. *IEEE Workshop on the Applications of Computer Vision (WACV), pages 361-368, 2012.*
- Aksoy, E.E., Dellen, B., Tamosiunaite, M., and Wörgötter, F. Execution of a dual-object (pushing) action with semantic event chains. *IEEE-RAS International Conference on Humanoid Robots (Humanoids), pages 576-583, 2011.*
- Aksoy, E.E., Abramov, A., Wörgötter, F., and Dellen, B. Categorizing object-action relations from semantic scene graphs. *IEEE International Conference on Robotics and Automation (ICRA), pages 398-405, 2010.*

- Abramov, A., Aksoy, E.E., Dörr, J., Wörgötter, F. and Dellen, B., 3D Semantic representation of actions from efficient stereo-image-sequence segmentation on GPUs. *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), pages 1-8, 2010.*

Conference Abstract

- Aksoy, E.E., Abramov, A., Dellen, B., and Wörgötter, F. Learning object-action relations from semantic scene graphs. *Frontiers in Computational Neuroscience. Conference Abstract: Bernstein Conference on Computational Neuroscience. doi: 10.3389/conf.neuro.10.2009.14.002.*

Patent

- Wörgötter, F., Aksoy, E.E., Abramov, A., Dellen, B. Method and device for estimating the plant development parameters. *Patent, 3294-057 PCT-1, (pending), 2012*